TOWARDS DEEP LEARNING ACCELERATED SPARSE BAYESIAN FREQUENCY ESTIMATION

MIKA PERSSON

Master's thesis
2022:E68

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Abstract

The Discrete Fourier Transform is the simplest way to obtain the spectrum of a discrete complex signal. This thesis concerns the case when the signal is known to contain a small (unknown) number of frequencies, not limited to the discrete Fourier frequencies, embedded in complex Gaussian noise. A typical signal is generated from a digital radar and the frequency components stem from point scatters, typically targets. The task is to estimate the frequencies and their respective amplitudes. This is done in a hierarchical Bayesian framework known from literature. It allows frequencies to be off the Fourier grid. The thesis contains the derivation of involved distributions, complementing the literature, being one of the results.

The resulting algorithm is a so-called hybrid-Gibbs sampler utilising a mix of conjugate priors and Markov Chain Monte Carlo. It constitutes a triple loop and is computationally heavy. The innermost loop is a Metropolis-Hastings sampler that samples a type of generalised (univariate and conditional) von Mises distribution. The main task of this thesis is to investigate the possibility of replacing it with a deep generative model. This would yield a significant acceleration. The model used in the investigation is a Continuous Conditional Generative Adversarial Network (CCGAN). Such networks can be used to sample synthetic images from highly multidimensional and complex distributions. Encouraged by such results it is easy to think that training a CCGAN to sample a univariate conditional distribution is easy. Counter-intuitively the opposite seems to be true.

The first numerical result in the thesis is the successful reproduction of a result from literature, the sampling from a two-dimensional Gaussian distribution (constant covariance) conditioned on the mean being on the unit circle. This gives confidence in the implementation. In a second step the sampling of a univariate Gaussian distribution, conditioned on both mean and variance, is investigated. Performance is not satisfactory despite the simple nature of the problem. Learning von Mises type distributions, which are more complicated and also conditioned on high-dimensional data, yield, not surprisingly, even less good results with CCGAN than the univariate Gaussian case. Suggestions for further development of the final CCGAN are given with the hope of making the CCGAN useful in practical Bayesian inference.

Acknowledgements

Firstly, I would like to express my sincerest gratitude to my industrial supervisor Adam Andersson at SAAB Surveillance Gothenburg. Thank you for your time and advice. Your continuous support and invaluable insights are what made this thesis possible. I would further like to extend my gratitude to my manager Ulrika Svahn and the people at SAAB for warmly welcoming me to their workplace with interesting discussions and lunch walks. A sincere thank you to my academic supervisor, Professor Andreas Jakobsson at the Centre for Mathematical Sciences of Lund University, for answering my questions and providing knowledge even beyond the scope of this project. Lastly, I would like to thank Clea and Anders Persson for keeping me motivated and being my greatest supporters.

Mika Persson, Gothenburg, August 2022

Contents

Chapter 1

Introduction

1.1 Background

Any complex signal vector in N dimensions can be represented with N complex amplitudes corresponding to N fixed frequencies being referred to as the Fourier grid. The most effective way to compute the spectrum of the signal is to compute the amplitudes with the Fast Fourier Transform (FFT) [1]. But what if we know that the signal is generated by a small number of frequencies that do not necessarily coincide with the Fourier frequencies, and that we want to recover these frequencies and their complex amplitudes? Applying the discrete Fourier transform for this purpose has two problems. Firstly, the signal will be spread to many or all Fourier frequencies. Secondly, the frequencies will by definition be limited to the grid. One way to approach this problem is to densify the grid, making the problem under-determined.

By imposing sparsity on the problem, i.e., assuming that most frequencies are zero, theory from the field of compressed sensing guarantees the existence of a unique solution [2]. There are iterative methods for the approximation of it. But even if the grid is denser, in real applications, the signal components will unlikely be on the grid and signal amplitudes will be spread over the grid. Even with a very dense grid, there are both problems with the vectors corresponding to the frequencies being almost parallel, and with expensive computations. Another approach, which is taken in this thesis, is to consider a grid, dense or not, and introduce for every grid point an off-grid parameter that represents the offset from the grid point. While the original problem is linear, the off-grid problem is non-linear.

The thesis consists of two parts. The first part follows the work of Lasserre's article [3] closely, where a complex sparse signal embedded in Gaussian noise is modelled using hierarchical Bayesian methods. Some of the parameters in the hierarchy are the off-grid parameters previously mentioned, when estimating the frequencies of a signal. Bayesian inference in the form of a hybrid-Gibbs sampler is then used to estimate the target scene of the signal. However, the hybrid-Gibbs sampler is computationally heavy, consisting of a triple loop. The innermost loop is a Metropolis-Hastings algorithm for sampling from a dilated and truncated generalised von Mises distribution (dGvM$_{[a,b]}$). This naturally leads to the second part of this thesis, which is how the hybrid-Gibbs sampler may be accelerated by replacing the Metropolis-Hastings algorithm with a deep generative network for the sampling of this distribution.

One application of frequency estimation of a signal appears in the direction of arrival (DOA) problem. In short, the DOA problem is that of determining how the energy/power in the form of electromagnetic radiation is distributed in space, where the sources represent high energy locations. Considering a setup of M passive identical sensors uniformly placed along a line, a so called uniform linear array (ULA). Figure 1.1 illustrates the DOA scenario with a ULA and one source where the direction of arrival is denoted as the angle θ.

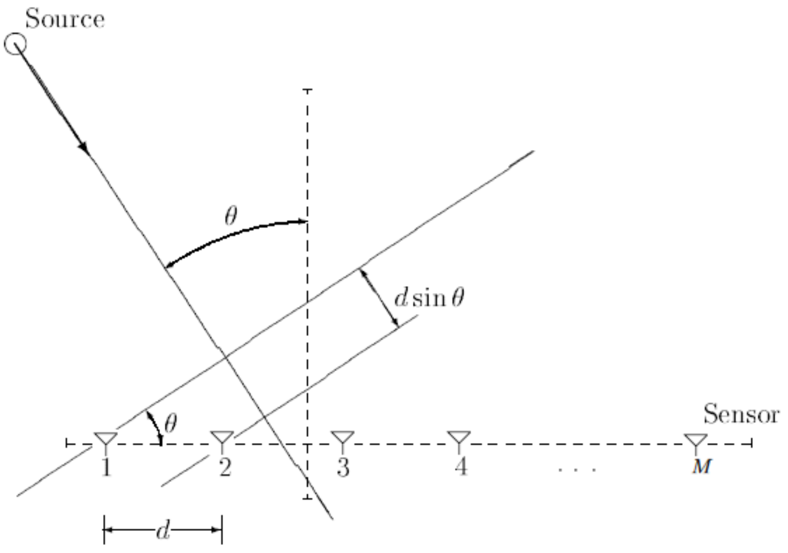


Figure 1.1: Setup of the direction of arrival problem with a uniform linear array of M sensors and one source. The goal is to determine the direction/angle of arrival θ and the power of the signal from the source.

1.2 Outlook

The theoretical background for hierarchical Bayesian modelling is given in Chapter 2. Readers familiar with Bayesian inference and Monte Carlo methods can skip this chapter, although the last two sections give a brief introduction to complex Gaussian random variables and the generalised von Mises distribution. Chapter 3 provides the theoretical background for constructing continuous conditional generative adversarial networks (CCGANs) with the purpose of modelling univariate conditional densities. In Chapter 4, the theory presented in Chapter 2 is applied for modelling a signal and estimating its target scene. Section 4.5 presents suggestions for how to accelerate the process of estimating the target scene. Chapter 5 describes the process of constructing a CCGAN used for this acceleration, by solving similar subtasks in order to gain a better understanding of where challenges may arise in constructing the final CCGAN, thus facilitating troubleshooting. Lastly, Chapter 6 concludes the results of the thesis and suggests future work for improving the performance of the CCGAN for simulating generalised von Mises distributions.

Chapter 2

Bayesian Inference using Markov Chain Monte Carlo

2.1 Bayesian inference

In general, reasoning and inference without complete information, where for example the observations - here suitably radar signals - contain noise, must incorporate a framework for analysing and quantifying uncertainty in order for one to make well-founded decisions. The mathematical framework for a part of this kind of reasoning is probability theory, which the reader is assumed to be acquainted with on a basic level. Other parts, like decision and utility theory, will be left out of this background. Although there are multiple interpretations of probability theory, the one of interest in this thesis is the Bayesian interpretation, which is also called the subjective or epistemic interpretation, for reasons that will become apparent.

In **Bayesian inference**, one often has access to the observation, effect or evidence of some unknown cause and would like to compute the probability of the cause, given the effect. This may appear like backward reasoning, but the applications are many. One field where this inference is applied is medical diagnosis, where one knows the probability of, for example, a patient having symptoms given that they have some disease but would like to know the probability of the patient having the disease given that the patient shows signs of the symptoms. The methodology for computing such probabilities is based on **Bayes' theorem**.

Considering two events C and E, interpreted as cause and effect, respectively, Bayes' theorem gives the probability of C given E as

$$P(C|E) = \frac{P(C \cap E)}{P(E)} = \frac{P(E|C)P(C)}{P(E)},$$

assuming that the probability of event E is nonzero, $P(E) > 0$. Bayes' theorem is also applicable for probability density functions (pdfs) concerning random variables. Given two random variables X and Y, the corresponding formula for pdfs is

$$f(X=x|Y=y)=\frac{f(X=x,Y=y)}{f(Y=y)}=\frac{f(Y=y|X=x)f(X=x)}{f(Y=y)}.$$

The factors in the formula are well known quantities. The factor $P(C)$ is the **a priori** probability, most often just referred to as the **prior**, which can be viewed as the statistician's subjective opinion or belief about the event C in the absence of any other information, hence the name "prior". The conditional probability $P(E|C)$ is called the **likelihood** and is the probability of observing E given C. In some sense this probability quantifies the causality of the phenomena observed. The probability $P(E)$ is termed the **marginal likelihood** or **model evidence** [4, p.485, 496]. The sought-after probability $P(C|E)$ is the **a posteriori** probability, most often just called the **posterior** probability. In light of new evidence or observations, the prior may be updated, taking the value of the previous posterior [5, p.35].

In practical Bayesian inference, the computations to retrieve the posterior probability can be cumbersome, often due to the complexity of marginal likelihood $f(Y)$. The marginal likelihood can be computed by marginalising or integrating over the model parameter space, which can be a complicated task since the integral may not have a closed form:

$$f(Y)=\int f(Y|X=x)f(X=x)\,\mathrm{d}x.$$

To overcome the problem of computing infeasible posteriors due to infeasible marginal likelihoods, approximative methods may be used to approximate the sought-after posterior or some expectation over it. One form of such methods are the computationally intensive sampling methods within the **Markov chain Monte Carlo** (MCMC) framework. In short, the MCMC methods seek to approximate the posterior by approximate samples from the posterior constructed by simulating a Markov chain. The details on how this is done are given in Sections 2.2 to 2.4.

After having obtained (an approximation of) the posterior distribution $f(X|Y)$, one can form estimators of distribution/model parameters X. One such estimator is the point estimate minimum mean square error defined as

$$\hat{x}_{\text{MMSE}}=\int xf(X=x|Y=y)\,\mathrm{d}x,$$

for some given observation $Y=y$, which is the posterior mean [3, equation (10a)].

One may understand why the Bayesian interpretation of probability is also referred to as the subjective interpretation, namely that the Bayesian framework incorporates the statistician's prior belief of the events into the probabilistic model. Because the prior must be chosen before the Bayesian model is complete, it is in fact a model choice problem. There is a trade-off between choosing a prior that makes the calculations mathematically tractable at the same time as the physical interpretation of the parameters on which one constructs the prior is preserved. It is possible to choose the prior in a way such that the posterior has the same form, belongs to the same family of distributions to be more exact, as the prior. The prior and posterior would then be **conjugate distributions** and the prior would be a **conjugate prior** to the likelihood. Using conjugate priors makes Bayesian inference easier through simpler mathematical manipulations [6, Section 1.2].

When parameters are dependent on each other on multiple levels, a hierarchical modelling approach can be taken. **Hierarchical Bayesian models** allow the prior parameters to be further modelled by some other parameters themselves, also known as **hyperparameters**. The joint probability of all the parameters should therefore in some sense reflect the dependencies between them [7, Chapter 5]. An example is in epidemiological modelling, where one examines how infectious diseases spread within and between countries based on the number of daily infected cases for each country. In this thesis, a signal $\boldsymbol{y}$ will be modelled using a hierarchical Bayesian model according to Figure 4.1 seen in Section 4.2.

As opposed to the frequentist interpretation of probability, also called the classical interpretation, the Bayesian approach may work better when no experimental data at all is available, which can be the case if one would like to model a phenomenon that seldom or never occurs. The frequentist view relies on the concept that a probability of an event can be defined by its relative frequency after many trials or occurrences [5, p.35-36]. As mentioned, this approach may be inadequate for modelling the probability that our sun will explode tomorrow since this event cannot be repeated numerous times in order to define the probability in the frequentist manner. Another difference between the two interpretations is that the frequentist framework mostly concerns point estimates of events or random variable probabilities, while the Bayesian approach lends a richer probabilistic analysis because complete distributions, in contrast to just point estimates, are obtained in the form of posteriors of the parameters of interest [5].

2.2 Markov chains

The necessary theory of Markov chains for understanding the MCMC methods in relation with the problem at hand will be given in this section.

A **Markov chain** is defined as a sequence of random variables (states) $X_0, X_1, \ldots, X_N$, such that for all $n = 0, 1, \ldots, N-1$, the variables are conditionally independent in accordance with the **Markov property**

$$f(X_{n+1} \in A | X_0, \ldots, X_n) = f(X_{n+1} \in A | X_n),$$

$$\begin{tikzcd} X_0 \arrow[r] & X_1 \arrow[r] & X_2 \arrow[r] & \cdots \arrow[r] & X_N \end{tikzcd}$$

Figure 2.1: Schematic illustration of a Markov chain

where the probability of the initial variable $f(X_0)$ and the conditional probabilities of subsequent variables, called **transition probability densities** or **transition kernel(s)**, are specified. The random variables take values in a state space $\mathcal{X}$ and a subset of it is denoted as $A \subset \mathcal{X}$. If the **state space** $\mathcal{X}$ is continuous, the transition probabilities are represented by a binary function called a transition kernel. In the discrete case, the transition probabilities are represented by a transition matrix where each element p_{ij} corresponds to the probability of going from the state in row i to column j. Intuitively, the Markov property says that the probability of the next variable only depends on the present variable and not the previous ones. A Markov chain can thus be represented as a directed graph in the form of a chain. Figure 2.1 illustrates this.

The probability of obtaining a specific realisation of a sequence $x_{0:N}$ can be written as

$$f(X_{0:n} = x_{0:N}) = f(X_0 = x_0) \prod_{n=1}^{N} q(X_n = x_n | X_{n-1} = x_{n-1}),$$

where $X_{0:n}$ denotes the Markov chain $X_0, X_1, \ldots, X_n$, and q denotes the transition probability density. The marginal probability of a specific element in the Markov chain at position n can be expressed recursively with respect to the marginal probability of the previous variables in the chain as

$$f(X_n) = \int_{\mathcal{X}} q(X_n | x_{n-1}) f(x_{n-1}) \,\mathrm{d}x_{n-1}.$$

A **stationary** Markov chain is a Markov chain satisfying

$$f(X_0 = x_0, X_1 = x_1, \ldots, X_k = x_k) = f(X_n = x_0, X_{n+1}, \ldots, X_{n+k} = x_k)$$

for all n and k. Expressed differently, a **stationary distribution** π is said to be stationary with respect to a Markov chain if

$$\pi(X') = \int_{\mathcal{X}} q(X'|x) \pi(x) \,\mathrm{d}x, \tag{2.1}$$

thus $X_0 \sim \pi$ implies that $X_n \sim \pi$ for all n. Equation (2.1) is called the **global balance equation**. One can ensure that a Markov chain has a specific stationary distribution π by constructing the transition probabilities in a way such that the **detailed balance equation** is satisfied:

$$\pi(x) q(x'|x) = \pi(x') q(x|x') \tag{2.2}$$

for all $x, x' \in \mathcal{X}$. Statistically, one cannot distinguish the direction between the two states x and x'. When a Markov chain satisfies this property, it is referred to as **reversible**. That stationarity is implied can be shown through straight forward calculations:

$$\int_{\mathcal{X}} \pi(x)q(x'|x)\,\mathrm{d}x = \int_{\mathcal{X}} \pi(x')q(x|x')\,\mathrm{d}x = \pi(x') \int_{\mathcal{X}} q(x|x')\,\mathrm{d}x = \pi(x'),$$

where one sees that the definition of stationarity in equation (2.1) is satisfied.

The final property of Markov chains to be introduced is of great importance for assuring that the constructed Markov chain converges towards a unique stationary distribution of interest and lies at the core of the MCMC methods. This leads us to the ergodicity property of a Markov chains. A Markov chain is said to be **ergodic** if there exists a finite integer k such that every state of the the Markov chain can be reached from every other state in exactly k steps. If a Markov chain is ergodic, then it has a unique and positive stationary distribution π such that

$$\lim_{N\to\infty} f(X_N = x) = \pi(x)$$

for all states x that a Markov chain can take. Important here is also that the stationary distribution is independent of the choice of initial distribution $f(X_0)$ [8, 9, Section 11.2.1 and Chapter 6, respectively]. Intuitively, ergodicity for Markov chains imply that every state in the state space can be reached independently of what state the chain starts in. The two examples given next may make the property of ergodicity clearer.

Assuming a and b are two states with the transition probabilities given on the edges on the directed graph given in Figure 2.2, the probability of going from state a to b is thus 0.4. An example of a realisation of a Markov chain on this setting of length three could be $(X_0 = a, X_1 = a, X_2 = b)$. For this Markov chain, one can observe that every state (a and b) can reach every other state in exactly $k = 2$ steps, hence it is ergodic.

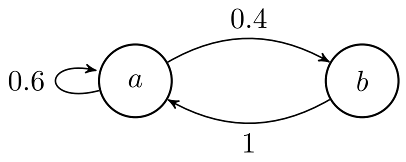


Figure 2.2: An ergodic Markov chain with two states a and b and given transition probabilities on the directed edges.

A second example is obtained by modifying the transition probabilities from the example given previously to the one seen in Figure 2.3. After inspecting this Markov chain, one can conclude that it is not ergodic, there

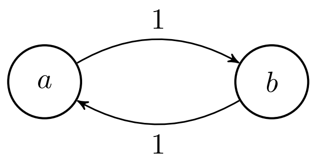


Figure 2.3: A non-ergodic Markov chain with two states a and b and given transition probabilities on the directed edges.

is not a finite integer k such that every state can reach every other state in exactly k steps.

As mentioned, the idea behind the MCMC methods in the given problem setting is to approximate the joint posterior of an observed signal $\boldsymbol{y}$, given some parameters introduced in a hierarchical Bayesian model, or some expectation over it by approximative samples from the posterior constructed by simulating a Markov chain. Using MCMC methods, the marginal likelihood does not have to be known in order to enable sampling from the joint posterior. A Markov chain has to be constructed in a way such that it is efficient to simulate from and has the joint posterior as its stationary distribution. These necessary properties are satisfied by the Markov chains generated by the MCMC methods.

2.3 The Metropolis-Hastings algorithm

There are two main flavors of MCMC methods: **Gibbs sampling** (algorithm) and the **Metropolis-Hastings** (MH) algorithm. Formally, the Gibbs sampler is in fact a special case of the MH algorithm [9, Section 10.1.1]. The MCMC methods are part of the general **Monte Carlo framework**, used for approximating expectations of the form

$$\mathbb{E}_f[\phi(X)] = \int_{\mathcal{X}} \phi(x) f(x)\,\mathrm{d}x, \tag{2.3}$$

which is the expectation of an objective function ϕ evaluated on the random variable X taking values in $\mathcal{X}$ with the pdf f. As an example, the random variable X could be the total time a person parks her car, Poisson distributed with probability mass function f and where the objective function is defined as $\phi(x) = 5 + 2x$ money units corresponding to the price of the parking of a car for x time units. The expectation of this objective function would then be the expected amount of money a person would have to pay for parking.

Monte Carlo integration is a method for evaluating expectations like the one in equation (2.3) benefiting from the **law of large numbers**. The method is as follows: sample elements $x_1, x_2, \ldots, x_N$ independently of each other from their common distribution and form the empirical average

$$\hat{\mathbb{E}}_f[\phi(X)] = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i).$$

By the law of large numbers, the empirical distribution converges towards the true expectation as $N \to \infty$, thus

$$\mathbb{E}_f[\phi(X)] \approx \hat{\mathbb{E}}_f[\phi(X)] = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i)$$

[9, p.83].

Approximating the expectation in equation (2.3) using Monte Carlo integration is possible through techniques that do not sample directly from the pdf f. Examples of such methods are **importance sampling** and MCMC methods. This comes in handy in Bayesian inference where the pdf f is a posterior where the normalising constant is unknown due to an intractable marginal likelihood.

Given an unnormalised distribution

$$f(X) = \frac{1}{Z}Q(X),$$

where the pdf $Q(X)$ is efficient to evaluate, but the normalising constant Z is intractable, the idea behind MCMC methods is to construct an ergodic Markov chain that is easy to simulate and has f as its stationary (target) distribution [9, p.268]. The Metropolis-Hastings algorithm does just this by generating a Markov chain through a special construction of the transition probability. The transition probability of a Markov chain generated by the MH algorithm is composed of two distributions, the **proposal** or **instrumental distribution** r, and the MH **acceptance probability** α. Given a sample in the Markov chain $x^{(t-1)}$, the next sample $x^{(t)}$ is obtained by first generating a proposed sample x^* for the next state by sampling from the proposal distribution as $x^* \sim r\big(x^*|x^{(t-1)}\big)$. The acceptance is then computed as

$$\begin{aligned} \alpha\big(x^*, x^{(t-1)}\big) &= \min\left(1, \frac{f(x^*)r\big(x^{(t-1)}|x^*\big)}{f\big(x^{(t-1)}\big)r\big(x^*|x^{(t-1)}\big)}\right) && \text{(2.4a)} \\ &= \min\left(1, \frac{Q(x^*)r\big(x^{(t-1)}|x^*\big)}{Q\big(x^{(t-1)}\big)r\big(x^*|x^{(t-1)}\big)}\right). && \text{(2.4b)} \end{aligned}$$

The proposed sample x^* is then taken as the next sample $x^{(t)}$ according to

$$x^{(t)} = \begin{cases} x^*, \text{ with probability } \alpha\big(x^*, x^{(t-1)}\big) \\ x^{(t-1)}, \text{ with probability } 1 - \alpha\big(x^*, x^{(t-1)}\big). \end{cases}$$

Important to note is that the fraction in the acceptance probability is independent of the normalising constant Z [9, Chapter 7.3.1]. The MH algorithm and different variations are presented in the Monte Carlo book [9, Chapter 7].

It can be proven that the transition probability for a Markov chain generated with the MH algorithm satisfies the detailed balance equation, hence the target distribution denoted as f is also the stationary distribution of the Markov chain. In order to show that detailed balance is satisfied, the transition probability for the MH generated Markov chain first must be known. Given an element $x^{(t-1)}$ in the chain, the conditional probability of the proposed element x^* when it is accepted is given by the product of the probability of the element first being proposed and the probability of this proposed element being accepted as

$$\mathbb{P}\Big(U \leq \alpha\big(x^*, x^{(t-1)}\big)\Big) r\big(x^* | x^{(t-1)}\big) = \alpha\big(x^*, x^{(t-1)}\big) r\big(x^* | x^{(t-1)}\big),$$

where U denotes a uniformly distributed stochastic variable on the unit interval. The probability of staying at the same element $x^{(t-1)}$, thus rejecting the proposed element x^*, is given by a point mass computed as

$$\begin{aligned}
&\int_{\mathcal{X}} \mathbb{P}\Big(U > \alpha\big(x^*, x^{(t-1)}\big)\Big) r\big(x^* | x^{(t-1)}\big) \, \mathrm{d}x^* \\
&\quad = \int_{\mathcal{X}} \Big(1 - \alpha\big(x^*, x^{(t-1)}\big)\Big) r\big(x^* | x^{(t-1)}\big) \, \mathrm{d}x^* \\
&\quad = 1 - \int_{\mathcal{X}} \alpha\big(x^*, x^{(t-1)}\big) r\big(x^* | x^{(t-1)}\big) \, \mathrm{d}x^* \\
&\quad = p_r\big(x^{(t-1)}\big).
\end{aligned}$$

The transition probability is thus composed as

$$q\big(x^{(t)} | x^{(t-1)}\big) = \alpha\big(x^{(t)}, x^{(t-1)}\big) r\big(x^{(t)} | x^{(t-1)}\big) + p_r\big(x^{(t-1)}\big) \delta_{x^{(t-1)}}\big(x^{(t)}\big),$$

where the Dirac delta function is defined as

$$\int_{-\infty}^{\infty} f(x) \delta_y(x) \, \mathrm{d}x = f(y),$$

where f is a continuous compactly supported function.

To show that the Markov chain generated by the MH algorithm has the target distribution as its stationary distribution, one must show it satisfies the global balance equation (2.1). In the proof below, one can think of the variables as $z = x^{(t)}$ and $x = x^{(t-1)}$:

$$\begin{aligned}
&\int_{\mathcal{X}} f(x)q(z|x)\,\mathrm{d}x \\
&\qquad = \int_{\mathcal{X}} f(x)\Big(\alpha(z,x)r(z|x) + p_r(x)\delta_x(z)\Big)\,\mathrm{d}x \\
&\qquad = \int_{\mathcal{X}} f(x)\alpha(z,x)r(z|x) + f(x)p_r(x)\delta_z(x)\,\mathrm{d}x,
\end{aligned}$$

inserting the definition of the acceptance probability lends

$$\begin{aligned}
&\int_{\mathcal{X}} f(x)q(z|x)\,\mathrm{d}x \\
&\quad = \int_{\mathcal{X}} f(x)\min\left(1, \frac{f(z)r(x|z)}{f(x)r(z|x))}\right) r(z|x)\,\mathrm{d}x + f(z)p_r(z) \\
&\quad = \int_{\mathcal{X}} \min\Big(f(x)r(z|x), f(z)r(x|z)\Big)\,\mathrm{d}x + f(z)p_r(z) \\
&\quad = \int_{\mathcal{X}} f(z)\min\left(\frac{f(x)r(z|x)}{f(z)r(x|z)}, 1\right) r(x|z)\,\mathrm{d}x + f(z)p_r(z) \\
&\quad = \int_{\mathcal{X}} f(z)\alpha(x,z)r(x|z)\,\mathrm{d}x + f(z)p_r(z) \\
&\quad = f(z)\int_{\mathcal{X}} \alpha(x,z)r(x|z)\,\mathrm{d}x + f(z)\left(1 - \int_{\mathcal{X}} \alpha(x,z)r(x|z), \mathrm{d}x\right) \\
&\quad = f(z)\left(1 + \int_{\mathcal{X}} \alpha(x,z)r(x|z)\,\mathrm{d}x - \int_{\mathcal{X}} \alpha(x,z)r(x|z), \mathrm{d}x\right) \\
&\quad = f(z),
\end{aligned}$$

which concludes the proof that the generated Markov chain from the MH algorithm has the target distribution as its stationary distribution. Moreover, under weak assumptions on the target and proposal distributions, the Markov chain generated by the MH algorithm is ergodic, implying that

$$\frac{1}{N}\sum_{k=1}^{N} \phi(x_k) \to \int_{\mathcal{X}} \phi(x)f(x)\,dx$$

as $N \to \infty$ [9, Section 7.3]. After some burn-in period N_{bi}, the samples generated are sampled as if they were sampled from the target distribution which we initially could not sample from directly due to its complexity. However, successive samples will be correlated, which actually is not a problem in practice [9, p.269]. To obtain independent samples one could subsample the returned samples from the MH algorithm [8, p.544].

The choice of proposal distribution has a substantial impact on how well the algorithm performs since the proposal determines how the state space is explored. Examining the acceptance probability in equation (2.4), one can get a better understanding of how the state space is explored. The ratio $f(x^*)/f\big(x^{(t-1)}\big)$ can be interpreted as high when the proposed element x^* is preferable than the current one, leading to a higher probability of accepting the proposal. As a consequence of this, the samples tend to stay around high-density regions of the target distribution. Examining the ration $r(x|z)/r(z|x)$ instead, this will be high when it is easy to go back from x^* to $x^{(t-1)}$.

2.4 Gibbs sampling

Say that one has a multivariate distribution f depending on p random variables, $\boldsymbol{\theta} = \big[\theta_1, \dots, \theta_p\big]$, that is difficult to sample from directly. If the conditional distributions of each one of the random variables are easy to sample from, the Gibbs sampler can aid in the sampling of the joint distribution. The joint pdf can be denoted as

$$f(\boldsymbol{\theta}) = f(\theta_1, \dots, \theta_p),$$

and the conditional (posterior) distribution of the random variable θ_k as

$$f_k(\theta_k|\boldsymbol{\theta}_{-k}),$$

where $\boldsymbol{\theta}_{-k}$ is the same as $\boldsymbol{\theta}$, but where the element at position k has been removed. Assuming that these are efficient to simulate from for all $k = 1, \dots, p$, the Gibbs sampler can be summarised as in the Monte Carlo book [9, Chapter 10].

The resulting Markov chain generated by the Gibbs sampler has the joint distribution $f(\boldsymbol{\theta})$ as its stationary distribution. The proof that the target distribution, $f(\boldsymbol{\theta})$, is the Gibbs generated Markov chain's stationary distribution is given for the two-dimensional state space case. The proof for higher dimensions is similar. Firstly, one can note that the transition kernel is

$$q\Big(\theta_1^{(t)}, \theta_2^{(t)}|\theta_1^{(t-1)}, \theta_2^{(t-1)}\Big) = f\Big(\theta_1^{(t)}|\theta_2^{(t-1)}\Big) f\Big(\theta_2^{(t)}|\theta_1^{(t)}\Big).$$

The global balance equation (2.1) can be shown to be satisfied as

$$\begin{aligned}
&\int\int f\left(\theta_1^{(t-1)},\theta_2^{(t-1)}\right) q\left(\theta_1^{(t)},\theta_2^{(t)}|\theta_1^{(t-1)},\theta_2^{(t-1)}\right) \mathrm{d}\theta_1^{(t-1)}\,\mathrm{d}\theta_2^{(t-1)} \\
&\quad = \int\int f\left(\theta_1^{(t-1)},\theta_2^{(t-1)}\right) f\left(\theta_1^{(t)}|\theta_2^{(t-1)}\right) f\left(\theta_2^{(t)}|\theta_1^{(t)}\right) \mathrm{d}\theta_1^{(t-1)}\,\mathrm{d}\theta_2^{(t-1)} \\
&\quad = \int f\left(\theta_2^{(t-1)}\right) f\left(\theta_1^{(t)}|\theta_2^{(t-1)}\right) f\left(\theta_2^{(t)}|\theta_1^{(t)}\right) \mathrm{d}\theta_2^{(t-1)} \\
&\quad = \int f\left(\theta_1^{(t)},\theta_2^{(t-1)}\right) f\left(\theta_2^{(t)}|\theta_1^{(t)}\right) \mathrm{d}\theta_2^{(t-1)} \\
&\quad = f\left(\theta_1^{(t)}\right) f\left(\theta_2^{(t)}|\theta_1^{(t)}\right) \\
&\quad = f\left(\theta_1^{(t)},\theta_2^{(t)}\right).
\end{aligned}$$

As for the MH algorithm, under weak assumptions, the Gibbs sampler can be shown to be (geometrically) ergodic, implying that

$$\frac{1}{N}\sum_{k=1}^{N}\phi(x_k) \to \int_{\mathcal{X}} \phi(x) f(x)\,dx$$

as $N \to \infty$ [9, Chapter. 10].

If some of the conditional posterior distributions used in the Gibbs sampler are difficult to sample from directly, the MH algorithm can be used to sample from these, leading to a **hybrid-Gibbs sampler**, also called Metropolis-within-Gibbs [9, p.393].

2.5 Complex random variables

Complex random variables are similar to real random variables, but with the difference that they map outcomes from some sample space into some complex space instead of some real space. As a complex number can be written on the form $z = a + jb$, where $a, b \in \mathbb{R}$ and j is the imaginary unit, a complex random variable can be considered as a pair of real random variables corresponding to the real and imaginary part of the variable, respectively. We denote a generic complex random variable as

$$Z = \mathrm{Re}(Z) + j\,\mathrm{Im}(Z) = A + jB,$$

where A and B are some real random variables [10, Chapter 17].

A complex random variable is distributed as a **univariate complex Gaussian** with mean zero and variance σ^2, $Z \sim \mathcal{CN}(0, \sigma^2)$, if the real and imaginary parts of Z are distributed according to

$$\mathrm{Re}(Z) \sim \frac{1}{\sqrt{2}}\mathcal{N}(0,\sigma^2) \qquad \text{and} \qquad \mathrm{Im}(Z) \sim \frac{1}{\sqrt{2}}\mathcal{N}(0,\sigma^2).$$

The scaling factor $1/\sqrt{2}$ ensures that the variance is correct. One can check that the complex random variable Z has the correct expectation, zero:

$$\begin{aligned}\mathbb{E}[Z] &= \mathbb{E}[\mathrm{Re}(Z) + j\,\mathrm{Im}(Z)]\\ &= \mathbb{E}[\mathrm{Re}(Z)] + j\,\mathbb{E}[\mathrm{Im}(Z)]\\ &= \frac{1}{\sqrt{2}}\cdot 0 + j\,\frac{1}{\sqrt{2}}\cdot 0\\ &= 0\end{aligned}$$

and the correct variance, σ^2:

$$\begin{aligned}\mathbb{V}[Z] &= \mathbb{E}\big[|Z|^2\big] - \big|\mathbb{E}[Z]\big|^2\\ &= \mathbb{E}\big[|Z|^2\big]\\ &= \mathbb{E}\big[\mathrm{Re}(Z)^2 + \mathrm{Im}(Z)^2\big]\\ &= \mathbb{E}\big[\mathrm{Re}(Z)^2\big] + \mathbb{E}\big[\mathrm{Im}(Z)^2\big]\\ &= \mathbb{V}[\mathrm{Re}(Z)] + \mathbb{E}[\mathrm{Re}(Z)]^2 + \mathbb{V}[\mathrm{Im}(Z)] + \mathbb{E}[\mathrm{Im}(Z)]^2\\ &= \frac{1}{2}\sigma^2 + 0 + \frac{1}{2}\sigma^2 + 0\\ &= \sigma^2.\end{aligned}$$

One can similarly show the same for a generic univariate complex Gaussian $Z \sim \mathcal{CN}\big(\mu, \sigma^2\big)$. The pdf of a complex Gaussian $Z \sim \mathcal{CN}\big(\mu, \sigma^2\big)$ is given by

$$f(z) = \frac{1}{\pi\sigma^2}\exp\bigg(-\frac{|z-\mu|^2}{\sigma^2}\bigg)$$

and the pdf of a generic complex multivariate Gaussian $\mathcal{CN}_M(\boldsymbol{\theta}, \boldsymbol{\Sigma})$ as

$$f(\boldsymbol{x}) = \frac{1}{\pi^M \det(\boldsymbol{\Sigma}^{-1})}\exp\Big(-(\boldsymbol{x}-\boldsymbol{\theta})^H\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\theta})\Big), \qquad (2.8)$$

where the superscript $(\cdot)^H$ denotes taking the Hermitian conjugate [11, p.26].

Simulating a univariate complex Gaussian with zero mean is summarised in Algorithm 2 in Appendix A. Sampling of a generic univariate complex Gaussian $Z \sim \mathcal{CN}\big(\mu, \sigma^2\big)$ is made in the same way as in the zero-mean case, except that the expectation $\mu \in \mathbb{C}$ is added to the sample from the zero mean complex Gaussian distribution.

A distribution that enforces sparsity, where the random variable is either zero or not, is the **Bernoulli complex Gaussian distribution**. This distribution is a mixture between a point mass at the value zero and a complex Gaussian if the variable is non-zero. A variable Z distributed according to this distribution is denoted as $Z \sim Ber\mathcal{CN}\big(w, \mu, \sigma^2\big)$, where the mixture pmf/pdf is

$$f(z) = (1-w)\delta(|z|) + w\frac{1}{\pi\sigma^2}\exp\bigg(-\frac{|z-\mu|^2}{\sigma^2}\bigg).$$

The weight parameter w is the probability that the variable takes a non-zero value. Sampling from a Bernoulli complex Gaussian is summarised in Algorithm 3 in Appendix A [12, Appendix 3].

2.6 The generalised von Mises distribution

One of the biggest tasks of this thesis is to use a generative model (CCGAN) to simulate a **dilated and truncated generalised von Mises distribution** denoted as $\text{dGvM}_{[a,b]}$, where $-\pi \leq a < b \leq \pi$ indicates the limits of the truncation. The $\text{dGvM}_{[a,b]}$ distribution can be seen as a generalisation of the **von Mises distribution** (vM), as indicated by its name.

The von Mises distribution $\text{vM}(\kappa, \phi)$ can be seen as a normal distribution "wrapped" around a circle with a concentration specified by $\kappa > 0$ and expected angle $\phi \in [-\pi, \pi]$. It is also called the circular normal distribution. The analogous normal distribution for $\text{vM}(\kappa, \phi)$ is $\mathcal{N}(\phi, 1/\kappa)$. The probability density function for the angle $\omega \in [-\pi, \pi]$ is given by

$$\text{vM}(\omega; \kappa, \phi) = \frac{\exp(\kappa \cos(\omega - \phi))}{2\pi I_0(\kappa)},$$

where I_0 is the modified Bessel function of order zero used to ensure that the density integrates to one [13]. Three examples of such pdfs can be seen in Figure 2.4.

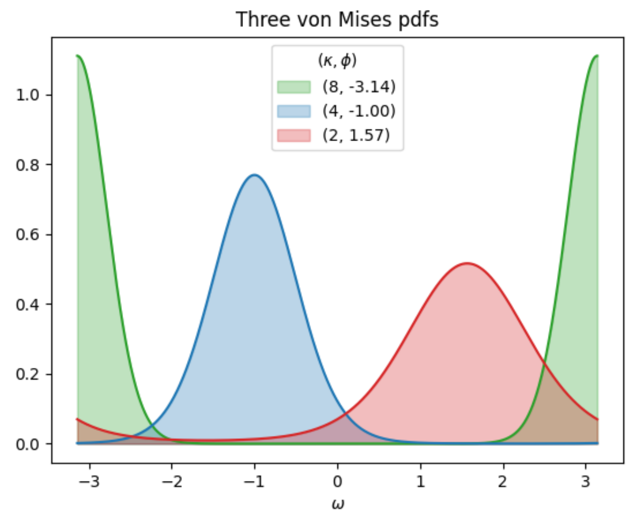


Figure 2.4: Three von Mises probability density functions specified by the parameters given in the figure. From the green and red plots one can observe that the density "wraps" around the interval on which ω is defined on.

To allow for asymmetry and multimodality as opposed to the von Mises distribution, its generalisation was introduced in the form of the **generalised von Mises distribution** $\mathrm{GvM}_M(\boldsymbol{\kappa}, \boldsymbol{\phi})$ (of order M). It is parametrised by $\boldsymbol{\kappa}$ with elements $\kappa_1, \kappa_2, \ldots, \kappa_M > 0$ and $\boldsymbol{\phi}$ with elements $\phi_1, \phi_2, \ldots, \phi_M \in [-\pi, \pi]$ and has the probability density function

$$\mathrm{GvM}_M(\omega; \boldsymbol{\kappa}, \boldsymbol{\phi}) \propto \exp\left(\sum_{m=1}^{M} \kappa_m \cos\left(\omega m - \phi_m\right)\right),$$

where the normalising constant is a series of Bessel functions which makes it complicated to sample from this distribution [14, Section 1]. The subscript $(\cdot)_M$ is omitted if it is clear what order the GvM distribution has. Three examples of such pdfs for $M = 2$ can be seen in Figure 2.5.

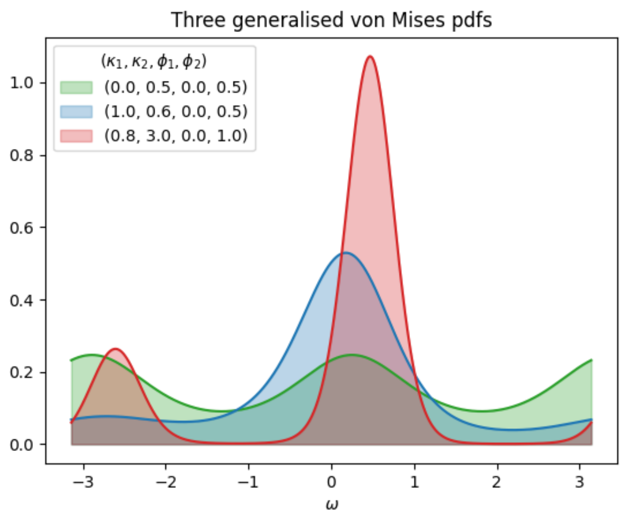


Figure 2.5: Three generalised von Mises probability density functions specified by the parameters given in the figure.

By scaling the angle variable ω in the GvM distribution with a factor $C > 0$, the number of modes of the density can be changed, resulting in a **dilated generalised von Mises distribution** (dGvM) with density:

$$\mathrm{dGvM}_M(\omega; \boldsymbol{\kappa}, \boldsymbol{\phi}) \propto \exp\left(\sum_{m=1}^{M} \kappa_m \cos\left(C\omega m - \phi_m\right)\right).$$

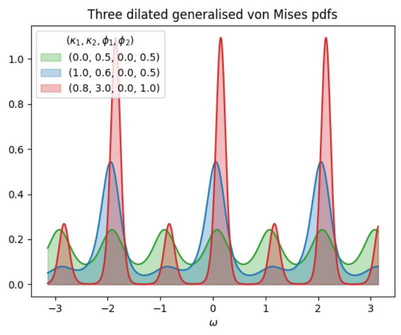
One finally obtains the already mentioned dilated and truncated generalised von Mises distribution by truncating the dGvM distribution on an interval $[a, b]$, where $-\pi \leq a < b \leq \pi$ as

$$\mathrm{dGvM}_{M,[a,b]}(\omega; \boldsymbol{\kappa}, \boldsymbol{\phi}) \propto \exp\left(\sum_{m=1}^{M} \kappa_m \cos\left(C\omega m - \phi_m\right)\right) \mathbb{1}_{[a,b]}(\omega).$$

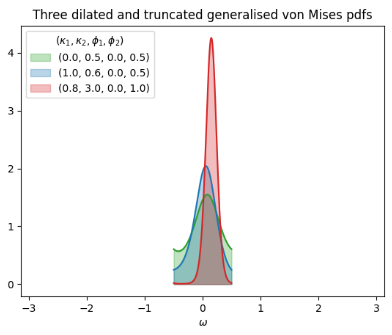
Here, $\mathbb{1}$ denoted the indicator function defined as

$$\mathbb{1}_A(x) = \begin{cases} 1, \text{ if } x \in A \\ 0, \text{ otherwise} \end{cases}$$

for some set A. Some illustrations of the densities of the dilated (and truncated) GvM distributions can be seen in Figure 2.6.



(a) Three examples of dilated generalised von Mises distributions with dilation factor π.



(b) Three examples of dilated and truncated ([-0.5,0.5]) generalised von Mises distributions with dilation factor π.

Figure 2.6: Visually comparing some densities between the dilated GvM and the dilated and truncated GvM pdfs.

In this thesis, methods like Monte Carlo (Sections 2.3 and 2.4) and generative models in the form of continuous conditional generative adversarial networks (Chapter 3) are used to sample from the dilated and truncated generalised von Mises distribution.

Chapter 3

Continuous Conditional Generative Adversarial Networks

A dataset can be seen as having been sampled from an unknown underlying data generating distribution. How a distribution like this can be modelled given a dataset, using CCGAN, is presented in this chapter. In contrast to discriminative models that model a conditional probability $\mathbb{P}(y|x)$ of a label y given a data point x, generative models instead model the conditional probability $\mathbb{P}(x|y)$ of an observable x given a label y [8, p.43].

Introduced year 2014 by Ian Goodfellow et. al., GAN is a framework for constructing a generative model by pitting two (neural network) models against each other, acting as adversaries. The formal definition of this game is defined in equations (3.3) and (3.4). Given a dataset, which can be seen as having been generated from some unknown data generating distribution p_{data}, one of the networks, G, is a generative model that approximates the sampling of p_{data}. The other network, D, is a discriminative model that distinguishes between whether a data point was drawn from the true data or sampled from the generator G [15, p.1].

Intuitively, one can understand the GAN framework by the analogy where the generator is an art forger with the goal of tricking a discriminator acting as a gallerist into falsely classifying a painting as authentic. Both parts start of not knowing what a real or fake painting looks like. As time passes, the discriminator gets better at deciding if a painting is authentic or fake, which the generator circumvents by finding new ways to make the counterfeited painting look more authentic. Ideally after a sufficiently long time, for the goal of the GAN framework, the generator creates paintings that are indistinguishable from the authentic ones. For this framework to work, the generator and discriminator have to improve at a similar pace. If the discriminator improves too fast in comparison with the generator, the generator will be overpowered by the discriminator in the sense that the generator will not comprehend how it should modify its generated samples (paintings) in accordance with the authentic ones. On the contrary, if the generator improves too fast, the discriminator will not be able distinguish between the authentic and fake samples.

There are several applications of GAN, most of which relate to image generation or modification. The StyleGAN architecture is used for image synthesis capable of generating photo realistic images of for instance people's faces, cars, and buildings [16]. The website [17] uses StyleGAN to generate images of persons that do not exist. Another application of GAN is image-to-image translation where a model is tasked to learn a mapping between an input image to an output image using a dataset of paired images. The CycleGAN is a GAN architecture that does this without having access to a dataset with paired images, but instead image collections with specific characteristics. An example from the CycleGAN paper [18] can be seen in Figure 3.1.

The application of GAN in this thesis will not be that of image generation, but instead of sampling from a univariate dilated and truncated generalised von Mises distributions introduced in Section 2.6, using a CCGAN model as described in Section 5.5. This problem is fundamentally different from the case of sampling images, since images are high-dimensional. Univariate distributions may be less complex than multivariate ones, however, due to a lesser complexity, the samples from univariate distributions are not as easily distinguishable from each other than those of multivariate ones.

As GANs consist of artificial neural networks (ANNs), a brief summary of the main concepts of ANNs is given in Section 3.1. Section 3.2 describes the formal game theoretical setup of the adversarial networks. It is equivalent with the training procedure of these networks. A modification in the GAN architecture to facilitate the generation of conditional samples that have a specific characteristic is introduced in Section 3.3. These **Conditional Generative Adversarial Networks** (CGAN) condition the inputs to each of the two networks (generator and discriminator) on a target label that can be seen as additional information for the networks. The case of target labels that belong to a finite set of values (which may have arisen from a multi-class classification problem) is introduced before moving on to the case where the conditioned values belong to a continuous set of values, for example the real line. This type of GAN will be referred to as **Continuous Conditional Generative Adversarial Networks** (CCGAN) and is presented in Section 3.3.1. Lastly, how the performance of GANs in this thesis are evaluated is presented in Section 3.4.

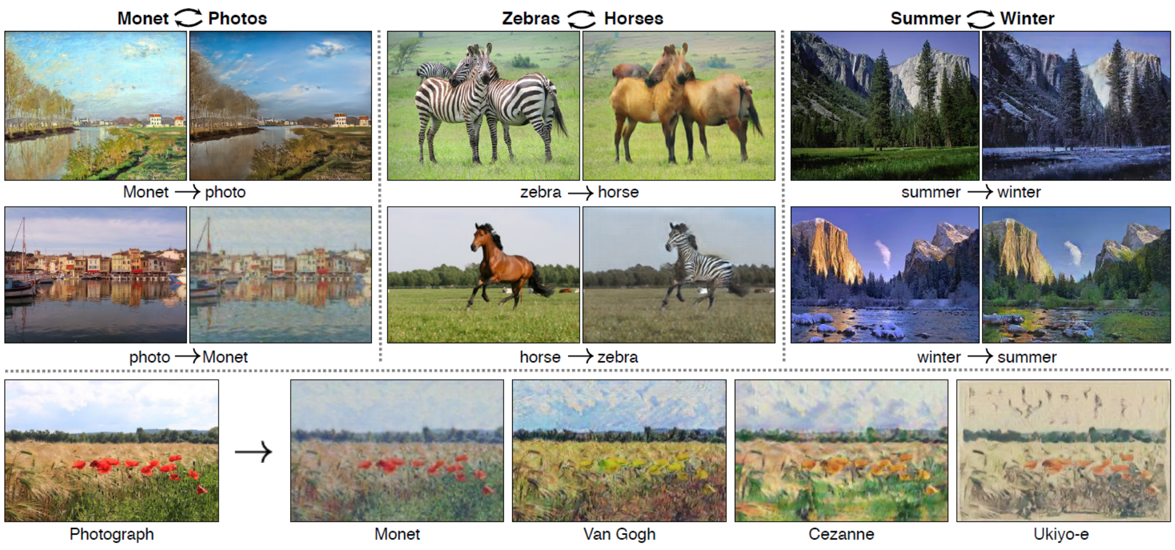


Figure 3.1: Illustration of CycleGAN image-to-image translation. Given two arbitrary collections of images with some specific characteristic each (for instance horse versus zebra images), the CycleGAN learns to translate an image from one of the collections to an image that attains the characteristic of the other collection.

3.1 Artificial Neural Networks

An **artificial neural network** (ANN) is a collection of computational nodes and edges connecting these. They can be used for modelling patterns in data, essentially function approximation, for solving classification and regression tasks in supervised learning, but also for modelling probability distributions. The **feedforward network** is a common ANN architecture that represents a composition of functions by adjacent layers, where each layer consists of a number of nodes, where each node in each layer is connected to every node in the subsequent layer. Assuming the network input and output dimensions are N and M, respectively, the feedforward network can be defined by a function

$$\mathbb{R}^N \ni \boldsymbol{x}_0 \mapsto f(\boldsymbol{x}_0; \boldsymbol{W}) \in \mathbb{R}^M,$$

where $\boldsymbol{x}_0$ is the network input, and $\boldsymbol{W}$ is the collection of weights parametrising the network. The general output of layer $\ell \in \{1, \dots, L\}$ of a feedforward network is

$$\boldsymbol{h}^{(\ell)}(\boldsymbol{h}^{(\ell-1)}) = \phi^{(\ell)}(\boldsymbol{W}^{(\ell)^T}\boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}),$$

where $\boldsymbol{h}^{(\ell-1)}$ is the output of the preceding layer, $\phi^{(\ell)}$ is the activation function in layer ℓ operating elementwise on the vector input, $\boldsymbol{W}^{(\ell)}$ is the weight matrix, $\boldsymbol{b}^{(\ell)}$ is the vector containing the biases for each node in the

ℓ:th layer, and

$$\begin{aligned} f(\boldsymbol{x}_0; \boldsymbol{W}) &= \boldsymbol{h}^{(L)} \\ \boldsymbol{h}^{(0)} &= \boldsymbol{x}_0. \end{aligned}$$

The sizes of the layers may vary, and the dimensions of the weight matrices and bias vectors are chosen accordingly. A common activation function for the hidden layers, namely neither the input nor output layer, is the **Rectified Linear Unit** (ReLU), defined as

$$\mathrm{ReLU}(x) = \max(0, x), \ x \in \mathbb{R}.$$

The **logistic sigmoid** is often used for binary classification problems as output activation function, given by

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

The output can be interpreted as the probability that the sigmoid input x belongs to one of the two classes in question [19, Chapter 6]. According to the **universal approximation theorem**, for any sufficiently regular function that is to be approximated, there exists, for an arbitrary tolerance, a feedforward network that approximates it with this tolerance. However, the theorem is nonconstructive [19, Section 6.4.1].

The task of training a network, given a dataset $\mathcal{D}$, is an optimisation problem where a **loss function** $\boldsymbol{W} \mapsto \mathcal{L}(\boldsymbol{W}; \mathcal{D})$, tailored to a specific problem, is to be minimised. The learning problem can thus be posed as finding the network weights satisfying the following:

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}; \mathcal{D}).$$

Optimising the loss function is synonymous with "training" the network. For classification and regression problems, the loss is usually chosen to be a cross-entropy or mean square error loss, respectively.

The loss is iteratively minimised by adjusting the network parameters in such a way as to descend the gradient of the loss function with respect to the parameters:

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta \nabla_{\boldsymbol{W}_t} \mathcal{L}(\boldsymbol{W}_t; \mathcal{D}),$$

where η is the learning rate. This is standard **Gradient Descent** (GD). To accelerate the optimisation and to avoid local optima, the dataset is batched, giving rise to the two GD methods **mini-batch gradient descent** and **stochastic gradient descent** depending on whether the batches have a larger cardinality than one or not, respectively. Due to the non-convexity of the problem, potential existence of plateaus and saddle points, and sparsity of the data features, several other GD methods have been suggested to alleviate these problems. A commonly used gradient descent algorithm that uses both an adaptive learning rate for each individual network parameter and momentum is **adaptive moment estimation** (ADAM). ADAM computes and keeps track of an exponential average of biased first moments (mean) m_{t+1} and second moments (variance) v_{t+1} of the gradients. The moments are updated as

$$\begin{aligned} m_{t+1} &= \beta_1 m_t + (1-\beta_1)\nabla_{\boldsymbol{W}_t}\mathcal{L}(\boldsymbol{W}_t;\mathcal{D}) \\ v_{t+1} &= \beta_2 v_t + (1-\beta_2)\big(\nabla_{\boldsymbol{W}_t}\mathcal{L}(\boldsymbol{W}_t;\mathcal{D})\big)^2, \end{aligned}$$

where β_1 and β_2 are hyperparameters usually set to 0.9 and 0.999 respectively. The moments are both initialised to zero. Calculating the unbiased moment estimates is done as

$$\begin{aligned} \hat{m}_{t+1} &= \frac{m_{t+1}}{1-\beta_1^t} \\ \hat{v}_{t+1} &= \frac{v_{t+1}}{1-\beta_2^t}. \end{aligned}$$

The ADAM parameter update rule is then defined as

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \frac{\eta}{\sqrt{\hat{v}_{t+1}}+\epsilon}\hat{m}_{t+1},$$

where ϵ is another hyperparameter used for numerical stabilisation whose default value usually is 10^{-8} [20, p.7-8].

Using the **backward propagation** algorithm, the gradients are computed by recursively applying the chain rule along each and every path of the network from the loss to each input and simultaneously caching the intermediate derivatives, essentially making backpropagation a dynamic programming algorithm with linear time complexity [19, Section 6.5].

To further accelerate the training process, but also stabilise it by diminishing **internal covariance shift**, the method of **batch normalisation** can be used. Internal covariance shift arises when the distribution of each layer's input changes due to the change of parameters of the previous layer. This makes the training more sensitive to the choice hyperparameters, for instance learning rate. Batch normalisation whitens the inputs to each layer; thus, the inputs are transformed to have zero mean and unit variance, reducing the effect of interval covariance shift [21].

3.2 The two-player minimax game

The probability distribution p_G of the generator approximates p_{data}. This is done by letting the input $\boldsymbol{z}$, called the **latent vector** or simply noise vector, to the generator be drawn from a distribution $p_{\boldsymbol{z}}$ that is efficient to sample from (usually from a unit uniform or standard normal distribution). The generator output is a value $\boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{W}_G)$, where $\boldsymbol{W}_G$ is the parameters of the generator. The space in which the latent vector $\boldsymbol{z}$ is drawn from is called the **latent space**. Given a sample $\boldsymbol{x}$ drawn from either p_{data} or p_G, the output $D(\boldsymbol{x}; \boldsymbol{W}_D)$ of the discriminator represents the probability that the input $\boldsymbol{x}$ came from the training data. The two networks can be described by the following mappings:

$$\begin{aligned} \mathbb{R}^N \ni \boldsymbol{z} &\mapsto \boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{W}_G) \in \mathbb{R}^M \\ \mathbb{R}^M \ni \boldsymbol{x} &\mapsto D(\boldsymbol{x}; \boldsymbol{W}_D) \in \mathbb{R}, \end{aligned}$$

where N is the dimension of the latent space and M is the dimension of the samples/discriminator inputs. Figure 3.2 illustrates how the networks are setup. From now on, the network parameters $\boldsymbol{W}_G$ and $\boldsymbol{W}_D$ will be omitted to increase readability.

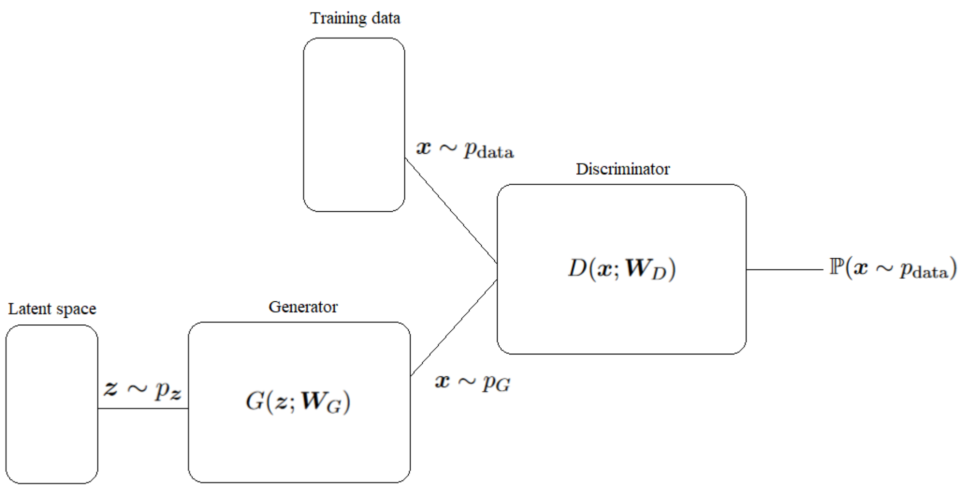

Training data
$\boldsymbol{x} \sim p_{\text{data}}$
Discriminator
$D(\boldsymbol{x}; \boldsymbol{W}_D)$
$\mathbb{P}(\boldsymbol{x} \sim p_{\text{data}})$
Latent space
Generator
$\boldsymbol{z} \sim p_{\boldsymbol{z}}$
$G(\boldsymbol{z}; \boldsymbol{W}_G)$
$\boldsymbol{x} \sim p_G$

Figure 3.2: Schematic illustration of the GAN setup.

The discriminator is a binary classifier, outputting the probability that the input $\boldsymbol{x}$ is authentic. Training of the discriminator is therefore done in such a way as to maximise the probability that the discriminator assigns the correct target label (authentic or fake) for the inputs drawn from both the training data and fake data sampled from the generator. The loss function for the discriminator, L_D, to be minimised is therefore the binary cross-entropy loss defined as

$$L_D = -\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log(D(\boldsymbol{x}))] - \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))], \tag{3.1}$$

where the expectation is split up into two terms since the mini-batch the discriminator will train on consists of two parts: a mini-batch drawn from the GAN training data and a mini-batch of samples sampled from the generator. Each data point in the authentic mini-batch is assigned a target label of value 1, while the fake samples are assigned value 0. Since the training data is not labeled, except implicitly while training the discriminator, the GAN framework falls under the unsupervised learning category. Observing the discriminator loss, one notes that the value of the loss is smaller when the discriminator correctly classifies an authentic input $\boldsymbol{x}$ ($D(\boldsymbol{x}) \to 1 \implies \log(D(\boldsymbol{x})) \to 0$) and larger when it misclassifies an authentic input ($D(\boldsymbol{x}) \to 0 \implies \log(D(\boldsymbol{x})) \to -\infty$). A similar observation can be made by examining the term $\log(1 - D(G(\boldsymbol{z})))$ in the loss function, where a discriminator that distinguishes between authentic and fake samples has a smaller loss.

As formerly stated, the goal of the generator is to trick the discriminator into believing that the samples generated from the generator are authentic. A suitable loss function L_G for the generator is therefore

$$L_G = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]. \tag{3.2}$$

The mini-batch that the generator receives is simply a mini-batch of latent vectors sampled from the latent space as $\boldsymbol{z} \sim p_{\boldsymbol{z}}$. One notes that a generator that is better at fooling the discriminator, thus making the discriminator believe the target label of the sample $G(\boldsymbol{z})$ is 1, gives a lower generator loss ($D(G(\boldsymbol{z})) \to 1 \implies \log(1 - D(G(\boldsymbol{z}))) \to -\infty$).

Because of the dependence of the generator loss on the discriminator loss, the two losses of the networks can be combined to form the following two-player minimax game:

$$\min_G \max_D V(G, D), \tag{3.3}$$

with value function $V(G, D)$ defined as

$$V(G, D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log(D(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]. \tag{3.4}$$

The optimisation of the value function eventually leads to the obtaining of the optimal generator parameters

$$\boldsymbol{W}_G^* = \arg \min_G \max_D V(G, D),$$

after which the generator should be able to sample data points that are indistinguishable from the authentic ones.

As shown in [15, Proposition 1], for a fixed generator G, the optimal discriminator D is

$$D_G^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}.$$

The value function in equation (3.4) can thus be reformulated as

$$\begin{aligned}
V(G, D_G^*) &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log(D_G^*(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] && \text{(3.5a)}\\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log(D_G^*(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{x}\sim p_G}[\log(1 - D_G^*(\boldsymbol{x}))] && \text{(3.5b)}\\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}\right] && \text{(3.5c)}\\
&\quad + \mathbb{E}_{\boldsymbol{x}\sim p_G}\left[\log\frac{p_G(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}\right]. && \text{(3.5d)}
\end{aligned}$$

Using the definition of the **Kullback-Leibler divergence** (KL divergence) [22, equation (1)]:

$$KL(p\,\|q) = \int_{\boldsymbol{x}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\, d\boldsymbol{x}$$

which measures how close two probability distributions p and q defined over the same support (part of domains with non-zero probability) are, the last equality in equation (3.5) can be rewritten as

$$V(G, D_G^*) = KL(p_{\text{data}} \| p_{\text{data}} + p_G) + KL(p_G \| p_{\text{data}} + p_G).$$

A further rewrite can be made by using the **Jensen-Shannon divergence** (JS divergence) [23, equation (10)] which is a symmetrisation of the KL divergence defined as:

$$JS(p\,\|q) = \frac{1}{2}\left(KL\Big(p\,\Big\|\frac{p+q}{2}\Big) + KL\Big(q\,\Big\|\frac{p+q}{2}\Big)\right).$$

Using this definition, one can check that the value function can be rewritten as

$$V(G, D_G^*) = -\log(4) + 2 \cdot JS(p_{\text{data}}\,\|p_G).$$

The JS divergence is non-negative and is zero when the distributions p_{data} and p_G are the same. It follows that the global minimum of the value function obtains its minimum at $-\log(4)$ when $p_{\text{data}} = p_G$. The generator thus seeks to minimise the JS divergence between its represented distribution p_G and the data generating distribution p_{data}. Research has been done on whether there are other loss functions the adversarial networks may use for training, resulting in for example Wasserstein GAN (WGAN) [24] and WGAN with gradient penalty (WGAN-GP) [25]. The presented GAN is commonly referred to as the Vanilla GAN.

In practice, a modified generator loss is used to improve the training:

$$L_G = -\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(D(G(\boldsymbol{z}))].$$

Rather than minimising the log-probability of the discriminator making a correct prediction, this generator loss function aims at maximising the log-probability of the discriminator making a mistake. In the former case, the generator receives a smaller loss if the discriminator outputs the correct label to a sample, hence the gradients of the generator parameters may vanish during training and the generator does not learn anything. This is especially the case in early training where the discriminator learns to distinguish between authentic and fake samples. Modifying the generator loss is therefore of practical importance to obtain a well-functioning generator [19, p.692][15, p.1-3]. Algorithm 5 in Appendix A summarises the GAN training.

Generative adversarial networks are notoriously difficult to train in practice due to instability and mode collapse/dropping. The phenomenon of **mode collapse** appears when the generator samples "collapse" to one mode, reducing the diversity of the generated samples. To gain a better understanding of GAN training and of the potential problems, see [26] where the problems are introduced and discussed with the aim of overcoming them. An example of a problem where mode collapse can arise is the creation of a GAN to sample 28×28 pixel-images of handwritten digits (0 to 9) in accordance with the MNIST dataset [27]. Mode collapse would result in a generator that only generates images that look identical and probably look like one of the ten digits.

3.3 Conditional GAN

For the previously introduced (vanilla) GAN, the user has no control over which sample that should be generated since the input $\boldsymbol{z}$ to the generator itself is sampled from a specified latent space. For generators that represent some distribution p_G corresponding to a multi-class distribution like the MNIST data for instance, the generated image could be any of the ten digits. If additional information is given to the networks by conditioning them on some class label (digit) during training, the generator may be able to produce the corresponding sample given a target label, resulting in what is called **Conditional Generative Adversarial Networks** (CGAN). Conditioning on some auxiliary data $\boldsymbol{c}$ that describes some data point $\boldsymbol{x}$ is done by feeding both the generator G and discriminator D the auxiliary data $\boldsymbol{c}$ together with the latent vector $\boldsymbol{z}$ and data point $\boldsymbol{x}$ giving outputs $G(\boldsymbol{z}, \boldsymbol{c})$ and $D(\boldsymbol{x}, \boldsymbol{c})$ respectively. The discriminator output $D(\boldsymbol{x}, \boldsymbol{c})$ can be seen as the probability that the input $\boldsymbol{x}$ is authentic given the condition $\boldsymbol{c}$. The training data for a CGAN consist of labelled data pairs $(\boldsymbol{x}, \boldsymbol{c})$, thus the goal of a CGAN is to fit the conditional distribution $p_{\text{data}}(\boldsymbol{x}|\boldsymbol{c})$. The minimax game for a CGAN can be seen as optimising the minimax game for a GAN according to equation

(3.3), but for each condition $\boldsymbol{c}$. More formally, this can be written as

$$\begin{aligned}\min_G \max_D \mathbb{E}_{\boldsymbol{c}\sim p_{\text{data}}(\boldsymbol{c})}\big[&\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}|\boldsymbol{c})}[\log(D(\boldsymbol{x},\boldsymbol{c}))] \\ &+\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z},\boldsymbol{c}),\boldsymbol{c}))]\big]\end{aligned}$$

[28, p.1-3]. A schematic illustration of the CGAN framework can be seen in Figure 3.3.

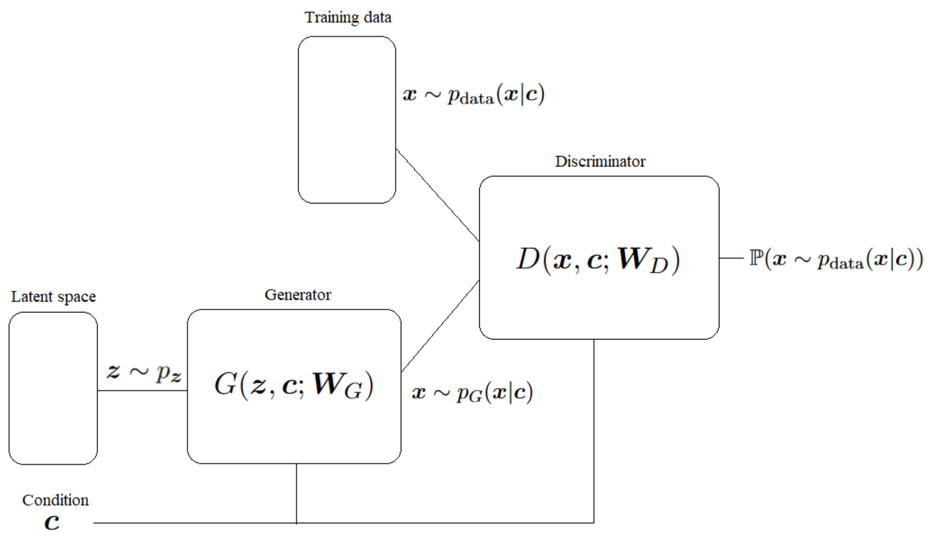

Training data
$\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}|\boldsymbol{c})$
Discriminator
$D(\boldsymbol{x},\boldsymbol{c};\boldsymbol{W}_D)$
$\mathbb{P}(\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}|\boldsymbol{c}))$
Latent space
Generator
$\boldsymbol{z} \sim p_{\boldsymbol{z}}$
$G(\boldsymbol{z},\boldsymbol{c};\boldsymbol{W}_G)$
$\boldsymbol{x} \sim p_G(\boldsymbol{x}|\boldsymbol{c})$
Condition
$\boldsymbol{c}$

Figure 3.3: Schematic illustration of the CGAN setup.

In practice, the conditioning of the networks on additional information $\boldsymbol{c}$ can be done in several ways. Kwak and Zhang explore four different architectures to incorporate the condition into the networks [29, p.2-3]. One of these (also used in the original CGAN paper) lets $\boldsymbol{c}$ be a **one-hot vector**, thus $\boldsymbol{c}$ is a zero vector apart from the element which corresponds to the particular class in question that instead takes value one. For example, if there are $C \in \mathbb{N}$ classes in a given dataset, the vector of size C with zeros everywhere except element k (instead having value one) would be a one-hot vector corresponding to the k:th class. The inputs to the generator and discriminator are then the concatenation of the data point $\boldsymbol{x}$ and condition $\boldsymbol{c}$.

The case when the conditional information belongs to some continuous space instead of a finite set of values is treated in the next subsection, presenting the continuous CGAN.

3.3.1 Continuous Conditional GAN

The CGAN framework works well for categorical conditions that correspond to class labels. However, this does not work equally well for conditions/scalar conditions lying in some continuous space, called **regression labels**. This is mainly due to two problems:

1. The CGAN training relies on the principle of so-called **Empirical Risk Minimisation** (ERM), which seeks to minimise the empirical version of the specified loss function given some training data [30, Section 4]. For this method to work well, the training data should have many data points for each distinct condition. In practice, this can be difficult to achieve since the regression label lies in a continuous space, resulting in a training dataset with few or no samples for each of the uncountable number of labels.

2. For conditions corresponding to class labels, the label is often encoded by a one-hot vector which is then fed into the generator and discriminator with their corresponding inputs using concatenation. The number of distinct labels is fixed in these scenarios. For regression labels however, the number of possible distinct labels are often uncountable, making the usage of one-hot vectors infeasible.

One might approach these two problems by naively "binning" the regression labels into disjoint intervals treated as independent classes and train a CGAN to fit this binned data. There are some problems with this approach however. For instance, the represented distribution by the generator can only be conditioned on the interval a label belongs to but not the label itself, large intervals lead to poor results and correlations between the different classes is not considered [31, p.1-2].

Ding et. al. [31] proposed the CCGAN framework to deal with the two problems stated above. In their work, they use the CCGAN for image generation where the regression labels are of dimension one. The regression label c is therefore assumed to be of dimension one for the rest of this section. For the purpose of this thesis, the regression label will be of higher dimension, but more on this in Section 5.5. The first problem is addressed by the introduction of two empirical discriminator losses, labelled the **Hard Vicinal Discriminator Loss** (HVDL) and **Soft Vicinal Discriminator Loss** (SVDL), and an empirical generator loss. In this thesis, only the HVDL is used, hence the SVDL will be disregarded. The second problem was dealt with by the introduction of a novel label input method [31, p.2]. The new input method uses a label projection, but this is not used in this thesis, see [32] for more details.

The HVDL is based on the principal of **vicinal risk minimisation** (VRM) instead of ERM. ERM makes use of empirical distributions calculated from an underlying training dataset, forming Dirac masses at these data points. In VRM, these empirical distributions are replaced with **vicinity functions** with domains $\Omega(\boldsymbol{x}_i) := \{\boldsymbol{x} : d(\boldsymbol{x}, \boldsymbol{x}_i) \leq \nu_{\boldsymbol{x}_i}\}$ for each data point $\boldsymbol{x}_i$, where $\boldsymbol{x}$ is some data point lying in the same space as the training data points, $d(\cdot,\cdot)$ is some specified metric, $\nu_{\boldsymbol{x}_i}$ is some small real number, and the set $\Omega(\boldsymbol{x}_i)$ is the **vicinity** of the data point $\boldsymbol{x}_i$ [33, Section 1.1].

As mentioned, the goal of the (continuous) CGAN is to fit the conditional distribution $p_{\text{data}}(\boldsymbol{x}|c)$. It is reasonable to assume that a small perturbation of a label c produces a small change to $p_{\text{data}}(\boldsymbol{x}|c)$, it is this property that VRM takes into account by letting data points in their vicinities share labels. Letting the data points resemble images of people's faces and the regression labels be the corresponding age of the person, one can imagine that the distribution of the facial features of eight-year-old children should be close to the ones of nine-year-old children.

In what follows, the HVDL and CCGAN generator losses are loosely derived from the CGAN discriminator and generator losses, respectively. The discriminator and generator losses for a CGAN are (compare with equations (3.1) and (3.2), respectively):

$$\begin{aligned}
L_D = &-\mathbb{E}_{c\sim p_{\text{data}}(c)}\left[\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}|c)}[\log(D(\boldsymbol{x},c))]\right] && \text{(3.6a)}\\
&- \mathbb{E}_{c\sim p_G(c)}\left[\mathbb{E}_{\boldsymbol{x}\sim p_G(\boldsymbol{x}|c)}[\log(1-D(\boldsymbol{x},c))]\right] && \text{(3.6b)}\\
= &-\int \log(D(\boldsymbol{x},c))p_{\text{data}}(\boldsymbol{x},c)\,d\boldsymbol{x}dc && \text{(3.6c)}\\
&-\int \log(1-D(\boldsymbol{x},c))p_G(\boldsymbol{x},c)\,d\boldsymbol{x}dc && \text{(3.6d)}
\end{aligned}$$

and

$$\begin{aligned}
L_G &= -\mathbb{E}_{c\sim p_G(c)}\left[\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(D(G(\boldsymbol{z},c),c))]\right]\\
&= -\int \log(D(G(\boldsymbol{z},c),c))p_{\boldsymbol{z}}(\boldsymbol{z})p_G(c)\,d\boldsymbol{z}dc,
\end{aligned}$$

where $p_{\text{data}}(c)$ and $p_G(c)$ are the authentic and fake label marginal distributions, respectively, $p_{\text{data}}(\boldsymbol{x}|c)$ and $p_G(\boldsymbol{x}|c)$ are the authentic and fake data point distributions conditioned on c respectively, $p_{\text{data}}(\boldsymbol{x},c)$ and $p_G(\boldsymbol{x},c)$ are the authentic and fake joint distributions of $\boldsymbol{x}$ and c, respectively, and $p_{\boldsymbol{z}}(\boldsymbol{z})$ is the prior probability density function over the latent space. Apart from the latent space prior the rest of the distributions are unknown, thus the corresponding empirical losses (indicated by the superscript $(\cdot)^\delta$) are minimised instead, in the CGAN framework:

$$\hat{L}_D^\delta = -\frac{1}{N^a}\sum_{i=1}^{C}\sum_{j=1}^{N_{c_i}^a}\log\left(D\big(\boldsymbol{x}_{c_i,j}^a,c_i\big)\right) - \frac{1}{N^G}\sum_{i=1}^{C}\sum_{j=1}^{N_{c_i}^G}\log\left(1-D\big(\boldsymbol{x}_{c_i,j}^G,c_i\big)\right)$$

$$\hat{L}_G^\delta = -\frac{1}{N^G}\sum_{i=1}^{C}\sum_{j=1}^{N_{c_i}^G}\log(D(G(\boldsymbol{z}_{c_i,j},c_i),c_i)),$$

where C is the number of distinct labels, N^a and N^G are the number of authentic and fake samples, respectively, $N^a_{c_i}$ and $N^G_{c_i}$ are the number of authentic and fake samples with label c, respectively, $\boldsymbol{x}^a_{c_i,j}$ and $\boldsymbol{x}^G_{c_i,j}$ are the j:th authentic and fake samples with label c_i, respectively, and $\boldsymbol{z}_{c_i,j}$ is independently and identically sampled from the prior $p_{\boldsymbol{z}}(\boldsymbol{z})$. In the empirical discriminator loss, the original joint probability densities $p_{\text{data}}(\boldsymbol{x}, c)$ and $p_G(\boldsymbol{x}, c)$ from the original discriminator loss have been estimated by their empirical probability density functions

$$\begin{aligned} p^{\delta}_{\text{data}}(\boldsymbol{x}, c) &= \frac{1}{N^a} \sum_{c_i}^{C} \sum_{j=1}^{N^a_{c_i}} \delta\big(\boldsymbol{x} - \boldsymbol{x}^a_{c_i,j}\big)\delta(c - c_i) \\ p^{\delta}_{G}(\boldsymbol{x}, c) &= \frac{1}{N^G} \sum_{c_i}^{C} \sum_{j=1}^{N^G_{c_i}} \delta\big(\boldsymbol{x} - \boldsymbol{x}^G_{c_i,j}\big)\delta(c - c_i), \end{aligned}$$

where $\delta(\cdot)$ indicates the Dirac delta mass centered at zero. It is these empirical probability density functions that VRM replaces with vicinity functions with the purpose of alleviating the first stated problem in the beginning of this subsection.

Using the chain rule of probabilities, the joint probability densities of $p_{\text{data}}(\boldsymbol{x}, c)$ and $p_G(\boldsymbol{x}, c)$ can be written

$$\begin{aligned} p_{\text{data}}(\boldsymbol{x}, c) &= p_{\text{data}}(c) p_{\text{data}}(\boldsymbol{x}|c) \\ p_G(\boldsymbol{x}, c) &= p_G(c) p_G(\boldsymbol{x}|c). \end{aligned}$$

Using so-called **kernel density estimation** (KDE), the marginal distributions $p_{\text{data}}(c)$ and $p_G(\boldsymbol{x})$ can be estimated using the given training and generated data with Gaussian kernels with variance, also called bandwidth, σ^2:

$$p_{\text{data}}(\boldsymbol{c}) \propto \frac{1}{N^a} \sum_{j=1}^{N^a} \exp\left(- \frac{\big(c - c^a_j\big)^2}{2\sigma^2} \right) \tag{3.7}$$

$$p_G(\boldsymbol{c}) \propto \frac{1}{N^G} \sum_{j=1}^{N^G} \exp\left(- \frac{\big(c - c^G_j\big)^2}{2\sigma^2} \right), \tag{3.8}$$

where $\boldsymbol{c}^a_j$ and $\boldsymbol{c}^G_j$ are the j:th authentic and fake training labels, respectively. Density estimation seeks to make inferences about an underlying probability density function from a finite number of samples seen as generated from this distribution, saying something about the probability of points not included in the data [34]. The parameter σ^2 can be seen as a hyperparameter for these marginal probability estimations. The conditional probability density functions $p_{\text{data}}(\boldsymbol{x}|\boldsymbol{c})$ and $p_G(\boldsymbol{x}|\boldsymbol{c})$ are estimated using the following vicinity functions:

$$p_{\text{data}}(\boldsymbol{x}|\boldsymbol{c}) \propto \frac{1}{N^a_{c,\nu}} \sum_{i=1}^{N^a} \mathbb{1}_{\{|c-c^a_i|\leq\nu\}} \delta\big(\boldsymbol{x} - \boldsymbol{x}^a_i\big) \tag{3.9}$$

$$p_G(\boldsymbol{x}|\boldsymbol{c}) \propto \frac{1}{N^G_{c,\nu}} \sum_{i=1}^{N^G} \mathbb{1}_{\{|c-c^G_i|\leq\nu\}} \delta\big(\boldsymbol{x} - \boldsymbol{x}^G_i\big), \tag{3.10}$$

where $\boldsymbol{x}^a_i$ and $\boldsymbol{x}^G_i$ are the i:th authentic and fake data points with corresponding labels c^a_i and c^G_i, respectively. The hyperparameter ν is a small real number determining how close regression labels should be in order to be considered belonging to each other's vicinities, which within their respective data points should be treated as similar. The integer $N^a_{c,\nu}$ is the number of authentic samples with regression label c^a_i that satisfy $|c - c^a_i| \leq \nu$ and the integer $N^G_{c,\nu}$ is the number of fake samples with regression label c^G_i that satisfy $|c - c^G_i| \leq \nu$. One can view these constraints as having "hard vicinities" defined by $c \pm \nu$, hence the name HVDL. Combining the marginal KDEs from equations (3.7) and (3.8) with the conditional vicinity functions from equations (3.9) and (3.10), respectively, results in the hard vicinal estimates (HVE) of the joint probability estimates as:

$$\begin{aligned}
&\hat{p}^{\text{HVL}}_{\text{data}}(\boldsymbol{x}, c) \\
&\quad = C_1 \left[\frac{1}{N^a} \sum_{j=1}^{N^a} \exp\left(- \frac{(c - c^a_j)^2}{2\sigma^2} \right) \right] \left[\frac{1}{N^a_{c,\nu}} \sum_{i=1}^{N^a} \mathbb{1}_{\{|c-c^a_i|\leq\nu\}} \delta(\boldsymbol{x} - \boldsymbol{x}^a_i) \right] \\
&\hat{p}^{\text{HVL}}_{G}(\boldsymbol{x}, c) \\
&\quad = C_2 \left[\frac{1}{N^G} \sum_{j=1}^{N^G} \exp\left(- \frac{\big(c - c^G_j\big)^2}{2\sigma^2} \right) \right] \left[\frac{1}{N^G_{c,\nu}} \sum_{i=1}^{N^G} \mathbb{1}_{\{|c-c^G_i|\leq\nu\}} \delta\big(\boldsymbol{x} - \boldsymbol{x}^G_i\big) \right],
\end{aligned}$$

where the constants C_1 and C_2 ensures that the estimates are valid probability density functions integrating to one. Inserting these estimates into the CGAN discriminator loss seen in equation (3.6) results in the CCGAN discriminator HVDL:

$$\begin{aligned}
&\hat{L}^{\text{HVDL}}_{D} \\
&\quad = -\frac{C_3}{N^a} \sum_{j=1}^{N^a} \sum_{i=1}^{N^a} \mathbb{E}_{\epsilon^a \sim \mathcal{N}(0,\sigma^2)} \left[\frac{\mathbb{1}_{\{|c^a_j+\epsilon^a-c^a_i|\leq\nu\}}}{N^a_{c^a_j+\epsilon^a,\nu}} \log\big(D\big(\boldsymbol{x}^a_i, c^a_j + \epsilon^a\big)\big) \right] \\
&\qquad - \frac{C_4}{N^G} \sum_{j=1}^{N^G} \sum_{i=1}^{N^G} \mathbb{E}_{\epsilon^G \sim \mathcal{N}(0,\sigma^2)} \left[\frac{\mathbb{1}_{\{|c^G_j+\epsilon^G-c^G_i|\leq\nu\}}}{N^a_{c^G_j+\epsilon^G,\nu}} \log\big(D\big(\boldsymbol{x}^G_i, c^G_j + \epsilon^G\big)\big) \right],
\end{aligned} \tag{3.11}$$

where the constants C_3 and C_4 ensures that the estimates are some constants, $\epsilon^a := c - c_j^a$ and $\epsilon^G := c - c_j^G$. The loss function of the generator of a CCGAN is

$$\hat{L}_G^\epsilon = \frac{1}{N^G} \sum_{i=1}^{N^G} \mathbb{E}_{\epsilon^G \sim \mathcal{N}(0,\sigma^2)} \Big[\log \big(D\big(G\big(\boldsymbol{z}_i, c_i^G + \epsilon^G\big), c_i^G + \epsilon^G\big)\big) \Big] \tag{3.12}$$

[35, Section 2].

The CCGAN discriminator and generator losses enable training on the vicinities of the regression labels c through the addition of Gaussian noise ϵ with variance σ^2 to the labels, lessening the problem of the training data not containing enough data points for each of the infinite number of labels. There are still some problems that may arise, as will be discussed in Section 5.5. As seen in the generator CCGAN loss in equation (3.12), Gaussian noise is added to the seen labels c_i^G to train the generator on unseen labels, enabling the generator to estimate $p_{\text{data}}(\boldsymbol{x}|c')$ where c' does not belong to the training dataset. In this way generalisation is encouraged, thus the hyperparameters σ and ν determining the noise and vicinity size respectively act as regularisation parameters. This is how the CCGAN solves the first problem stated in the beginning of this subsection.

Algorithm 6 in Appendix A summarises the CCGAN training. In practice, the normalising constants $N^a_{c_j^a+\epsilon^a,\nu}$ and $N^a_{c_j^G+\epsilon^G,\nu}$ in the HVDL CCGAN discriminator loss, seen in equation (3.11), are set to one. The regularisation parameters σ and ν are selected according to a rule of thumb, as explained in the CCGAN paper [31, Section S.4]. A clear motivation for the rule of thumb was not given, except that it seemed to work well for their experiments. All regression labels are normalised to the unit interval $[0, 1]$, the bandwidth of the KDEs is then selected as

$$\sigma = \hat{\sigma}_{c^a} \left(\frac{4}{3N^a} \right)^{1/5}, \tag{3.13}$$

where $\hat{\sigma}_{c^a}$ is the sample standard deviation of the normalised training labels. The hyperparameter σ can be seen as a parameter for how much the labels in the dataset should be perturbed, without representing a completely different distribution. The vicinity function radius ν is set to

$$\nu = m_\nu \nu_{\text{base}},$$

where m_ν is generally set to 1 or 2, standing for half of the minimum number of adjacent labels used for estimating $p_{\text{data}}(\boldsymbol{x}|c)$ given a label c, and

$$\nu_{\text{base}} = \max \Big(c^a_{[2]} - c^a_{[1]}, c^a_{[3]} - c^a_{[2]}, \dots, c^a_{[N^a_{uc}]} - c^a_{[N^a_{uc}-1]} \Big), \tag{3.14}$$

where $c^a_{[i]}$ is the i:th smallest normalised distinct label and N^a_{uc} is the number of distinct labels in the dataset.

3.4 Evaluating GAN performance

There are many techniques to evaluate GANs, both qualitative through visualisation of generated samples (images) and quantitative through metrics like the **Wasserstein-1** (W_1) distance, also called Earth-Mover distance, or **Energy distance** (ED) for instance. However, there is no consensus as to how exactly GANs should be evaluated in general [36]. In the following sections, qualitative evaluation and the two metrics, $W1$ distance and ED, will be presented.

Qualitative evaluation of GANs through visualisation of the generated samples work well as the visualisation makes sense to a human. The samples should thus be images or a collection of samples seen as drawn from a (multivariate) probability distribution of maximum dimension 2 so that the empirical distributions in the form of histograms or KDEs can be visualised. Comparisons are then made between the samples and the authentic data by an annotator. An example of a visualisation which upon qualitative evaluation is made where a GAN was used to model a standard univariate Gaussian $\mathcal{N}(0, 1)$ is shown in Figure 3.4. One can observe that the distribution the generator represents, p_G, is quite close to the true underlying data distribution p_{data}. This evaluation is highly subjective, which is why quantitative should be considered.

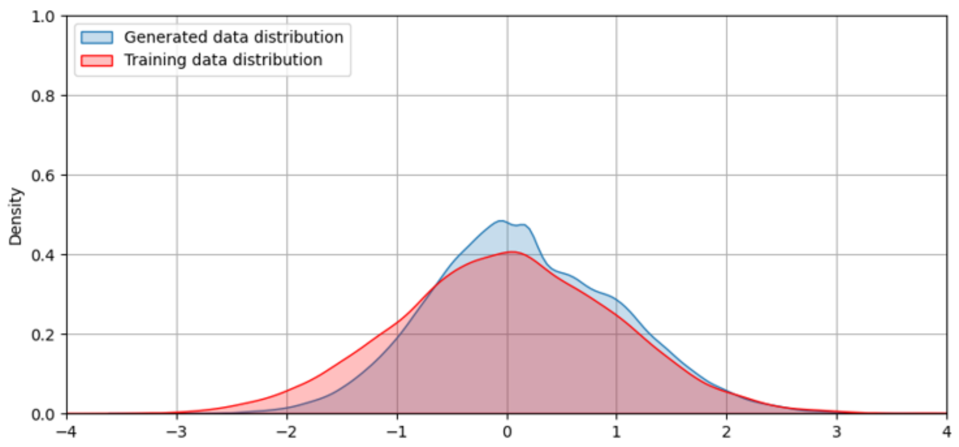


Figure 3.4: Visualisation which upon qualitative evaluation is made where a GAN was used to model a standard univariate Gaussian $\mathcal{N}(0, 1)$. The two densities were calculated using KDE from the training data sampled from a univariate standard normal distribution (red) and samples from the generator (blue).

As mentioned above, GAN-training seeks to minimise the distance between the distribution the generator represents, p_G, and the unknown data generating distribution p_{data}. How this distance is measured can be done in several ways, but the presented way was through JS divergence. Unfortunately, when visualising the losses of the discriminator and generator for each training epoch, they are not as informative as for the losses minimised for supervised learning problems. For supervised learning problems, there is a negative correlation between the value of the loss function to be minimised and the accuracy of the model. A lower loss function value corresponds to a higher accuracy of the model. For GANs, this is not the case. What one instead can look for in the discriminator and generator losses is if they converge and stabilise during training. Convergence of the losses tend to indicate that the networks have settled on some model parameters. One should however make a qualitative evaluation to make sure that the generator produces reasonable samples similar to the ones in the training data.

Another way of measuring how close two distributions p and q are, given a finite number of samples from them, is by using the Wasserstein-1 distance:

$$W_1(p,q) = \inf_{\pi \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\pi}\left[\|x-y\|_1\right],$$

where $\Pi(p,q)$ is the set of all joint probability distributions $\pi(x,y)$ with marginals p and q respectively [37, equation (1)]. This distance arose from the theory of optimal transport [38, p.22]. As opposed to the JS divergence, it has been experimentally shown that a lower W_1 distance are correlated with better qualitative results [24].

A metric that is similar to the Wasserstein-1 distance is the energy distance. Assuming X and X' are two independent random variables distributed according to p, and Y and Y' are two independent random variables distributed according to q, then the energy distance between the two variables X and Y is

$$\mathrm{ED}(p,q) = \left(2\mathbb{E}|X-Y| - \mathbb{E}|X-X'| - \mathbb{E}|Y-Y'|\right)^{1/2} \tag{3.15}$$

[39].

Chapter 4

Signal Modelling and Target Scene Estimation

This chapter details the methodology for one approach for solving the non-linear problem of estimating the target scene of a signal that is robust to grid mismatch. Section 4.1 provides a sparse signal representation using a Fourier dictionary taking into account potential grid point offsets. In Section 4.2, a hierarchical Bayesian model for the signal is proposed. To estimate the signal parameters, a hybrid-Gibbs sampler is used. The construction of this is detailed in Section 4.3, and the evaluation is presented in Section 4.4. Finally, suggestions for accelerating the Bayesian inference using deep generative models are given in Section 4.5.

4.1 Signal model

Upon receiving a signal $\boldsymbol{y} \in \mathbb{C}^M$, the goal is to estimate the target scene. Often, depending on how the signal is modelled, the signal consists of a sum of cisoids embedded in white noise:

$$\boldsymbol{y} = \sum_{n=1}^{N} \alpha_n \boldsymbol{a}_n + \boldsymbol{n}, \tag{4.1}$$

where M is the size of the observation space, N is the number of target signals (sources), $\alpha_n \in \mathbb{C}$, $n = 1, \dots, N$, are the complex amplitudes and $\boldsymbol{a}_n \in \mathbb{C}^M$, $n = 1, \dots, N$, the **steering vectors** defined as

$$[\boldsymbol{a}_n]_m = \exp(j2\pi f_n m)$$

with frequency f_n of the n:th target signal, and noise vector $\boldsymbol{n}$. The sought-after target scene from the signal is then $(N, \alpha_1, f_1, \dots, \alpha_N, f_N)$. The size of the observation space M may in practice be equal to the number of antenna elements seen in Figure 1.1. Assuming that the signal is sparse, namely that the number of targets is small, the signal may assume a sparse representation. By using a Fourier basis, the signal model in equation (4.1) can be rewritten as

$$\boldsymbol{y} = \boldsymbol{F}\boldsymbol{x} + \boldsymbol{n}, \tag{4.2}$$

where $\boldsymbol{F} \in \mathbb{C}^{M \times \overline{M}}$ is a Fourier dictionary where usually $\overline{M} \geq M$ and $\boldsymbol{x} \in \mathbb{C}^{\overline{M}}$ is the sparse vector having ideally N nonzero components. The elements in the Fourier dictionary are defined as

$$[\boldsymbol{f}_{\overline{m}}]_m = \frac{1}{\sqrt{M}} \exp\left(j2\pi m \frac{\overline{m}}{\overline{M}}\right),$$

where $m = 0, 1, \ldots, M-1$, $\overline{m} = 0, 1, \ldots, \overline{M}-1$, and $\boldsymbol{f}_{\overline{m}}$ is the $\overline{m}$:th column of $\boldsymbol{F}$. This basis is appropriate if the frequencies of the signals match the ones on the grid

$$0, \frac{2\pi}{\overline{M}}, 2\frac{2\pi}{\overline{M}}, \ldots, (\overline{M}-1)\frac{2\pi}{\overline{M}}.$$

However, in practice, the frequencies may have an offset, making the true frequencies lie outside or between the grid frequencies, giving rise to the **grid mismatch problem**. Targets with frequencies not aligning with the grid frequencies are so-called **off-grid targets**. As proposed in [40], the possible grid error or grid mismatch, namely the difference between a target's frequency and the closest frequency on the frequency grid, is modelled by a vector, $\boldsymbol{\epsilon} \in \mathbb{R}^{\overline{M}}$, corresponding to the perturbations on the frequency axis. This so-called **grid mismatch vector** is denoted as

$$\boldsymbol{\epsilon} = \left[\epsilon_0, \ldots, \epsilon_{\overline{m}}, \ldots, \epsilon_{\overline{M}-1}\right]^T.$$

The Fourier dictionary is thus parameterised by the grid mismatch vector as follows:

$$\boldsymbol{F}(\boldsymbol{\epsilon}) = \left[\boldsymbol{f}_0(\epsilon_0), \ldots, \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}), \ldots, \boldsymbol{f}_{\overline{M}-1}(\epsilon_{\overline{M}-1})\right],$$

where the m:th component of the vector in column $\overline{m}$ is

$$\left[\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})\right]_m = \frac{1}{\sqrt{M}} \exp\left(j2\pi m \frac{\overline{m} + \epsilon_{\overline{m}}}{\overline{M}}\right).$$

To prevent overlapping between the frequency bins of the Fourier dictionary, the grid mismatch elements are assumed to be bounded to the interval $\epsilon_{\overline{m}} \in [-0.5, 0.5)$. Nevertheless, if the frequencies of multiple different sources lie on the same grid point, these are not possible to distinguish. The sparse signal model in equation (4.2) is hence reformulated as

$$\boldsymbol{y} = \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{n}. \tag{4.3}$$

In what follows, a hierarchical Bayesian model for the observed signal $\boldsymbol{y}$ depending on parameters including the target amplitude vector $\boldsymbol{x}$ and grid mismatch vector $\boldsymbol{\epsilon}$ is presented, with the goal of estimating the target scene $(\boldsymbol{x}, \boldsymbol{\epsilon})$ using a hybrid-Gibbs sampler.

4.2 Hierarchical Bayesian Signal Model

A hierarchical Bayesian model for the observed signal is proposed. This model can be seen in Figure 4.1.

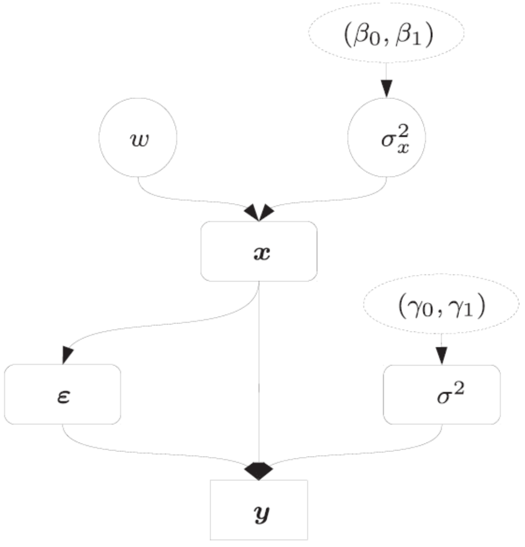


Figure 4.1: Hierarchical Bayesian model of an observed signal $\boldsymbol{y}$ given the parameters collected in $\boldsymbol{\theta} = \left[\sigma^2, w, \sigma_x^2, \boldsymbol{x}^T, \boldsymbol{\epsilon}^T\right]^T$ and hyperparameters β_0, β_1, γ_0 and γ_1.

Bayesian inference will be used to compute the posterior of the model parameters generating the observed signal. The joint probability of all the parameters should therefore in some sense reflect the dependencies between them [7, Chapter 5]. The goal after receiving a signal $\boldsymbol{y}$ is to compute the target scene ($\boldsymbol{x}$, $\boldsymbol{\epsilon}$). This can be done by either point estimates of the target scene or full distributions given by the Bayesian framework, but the posterior of the target scene must be known beforehand in the Bayesian case.

Using Bayes' theorem, the joint posterior of the model parameters, given the observed signal, can be written as

$$\begin{aligned} f\left(\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2 | \boldsymbol{y}\right) &= \frac{f\left(\boldsymbol{y} | \sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2\right) f\left(\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2\right)}{f(\boldsymbol{y})} \\ &\propto f\left(\boldsymbol{y} | \sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2\right) f\left(\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2\right). \end{aligned}$$

The symbol $\propto$ denotes proportionality. Proportionality holds because the left-hand side is a function of the model parameters, so discarding any factor solely depending on the observed signal $\boldsymbol{y}$ on the right-hand side, the marginal likelihood in this case, preserves proportionality. The marginal likelihood can be obtained by marginalising or integrating over the model parameter space, which can be a complicated task since the integral may not have a closed form. Denoting the collection of model parameters as

$$\boldsymbol{\theta} = \left[\sigma^2, w, \sigma_x^2, \boldsymbol{x}^T, \boldsymbol{\epsilon}^T\right]^T,$$

the marginal likelihood can be computed as

$$f(\boldsymbol{y}) = \int f(\boldsymbol{y}|\boldsymbol{\theta}) f(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}.$$

An estimate of the target scene $(\boldsymbol{x}, \boldsymbol{\epsilon})$, can be obtained by the minimum mean square errors (MMSE) of the estimators

$$\hat{\boldsymbol{x}}_{\text{MMSE}} = \int \boldsymbol{x} f(\boldsymbol{x}|\boldsymbol{y}) \mathrm{d}\boldsymbol{x}$$
$$\hat{\boldsymbol{\epsilon}}_{\text{MMSE}} = \int \boldsymbol{\epsilon} f(\boldsymbol{\epsilon}|\boldsymbol{y}) \mathrm{d}\boldsymbol{\epsilon}.$$

These estimates have no closed form solutions due to the intractability caused by the distributions $f(\boldsymbol{x}|\boldsymbol{y})$ and $f(\boldsymbol{\epsilon}|\boldsymbol{y})$. To overcome this problem, approximative methods are used, MCMC methods in our case. For reasons that will become apparent when the conditional posteriors of the target amplitude vector $\boldsymbol{x}$ and the grid mismatch vector $\boldsymbol{\epsilon}$ are derived, both of the vectors are sampled elementwise using a Gibbs sampler, where each of the grid mismatch elements are sampled using the MH algorithm, nested in the Gibbs sampler. The MCMC method used is therefore a hybrid-Gibbs sampler.

The hybrid-Gibbs sampler is implemented to iteratively sample the model parameters $\sigma^{2^{(n)}}, w^{(n)}, \sigma_x^{2^{(n)}} \boldsymbol{x}^{(n)}, \boldsymbol{\epsilon}^{(n)}$, from their corresponding conditional posterior distributions. The superscript $(\cdot)^{(n)}$ indicates an MCMC sample at iteration n. The conditional posterior distributions of each of the parameters can readily be written as

$$f(\theta_i|\boldsymbol{y}, \boldsymbol{\theta}_{-i}),$$

where $\boldsymbol{\theta}_{-i}$ is the same as $\boldsymbol{\theta}$ but were the element at position i has been removed. When enough samples have been collected, the MMSE estimators can be approximated empirically as

$$\hat{\theta}_{i,\,\text{MMSE}} = \frac{1}{N_r} \sum_{n=1}^{N_r} \theta_i^{(n+N_{bi})} \tag{4.4}$$

where N_{bi} and N_r are the number of burn-in samples and number of kept samples, respectively.

Because the parameters that compose the observed signal are sampled from their corresponding conditional posteriors, it is necessary to have access to these. In their turn, the conditional posteriors are derived from the likelihood of the observed signal and the corresponding priors of the parameters. Therefore, the likelihood is first derived, the priors of the parameters introduced afterwards, and the conditional posteriors derived lastly before constructing and using the hybrid-Gibbs sampler to estimate the minimum mean square error estimators of the target scene.

4.2.1 Signal likelihood derivation

The additive white background noise, $\boldsymbol{n}$, in the signal model

$$\boldsymbol{y} = \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{n}$$

is assumed to be a complex Gaussian with variance σ^2 of the form

$$\boldsymbol{n}|\sigma^2 \sim \mathcal{CN}\big(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_{M\times M}\big),$$

where $\boldsymbol{I}_{M\times M}$ is the identity matrix of size $M \times M$. This subscript will be disregarded for the ease of notation.

Assuming that the frequency mismatch $\boldsymbol{\epsilon}$ and target amplitude vector $\boldsymbol{x}$ are known, one can observe that the signal is a linear combination of a complex multivariate Gaussian, hence the signal itself must also be a complex multivariate Gaussian depending on the constant $\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}$ and the white noise $\boldsymbol{n}$. The likelihood of the observed signal can thus be written as

$$\boldsymbol{y}|\boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2 \sim \mathcal{CN}\Big(\mathbb{E}[\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{n}], \mathrm{Cov}(\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{n}, \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{n})\Big), \tag{4.5}$$

where the resulting expectation is computed as

$$\mathbb{E}[\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{n}] = \mathbb{E}[\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}] + \mathbb{E}[\boldsymbol{n}] = \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{0} = \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}$$

and the covariance computed as

$$\begin{aligned}
&\mathrm{Cov}(\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{n}, \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{n}) \\
&\quad = \mathrm{Cov}(\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}, \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}) + \mathrm{Cov}(\boldsymbol{n}, \boldsymbol{n}) + 2\mathrm{Cov}(\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}, \boldsymbol{n}) \\
&\quad = \boldsymbol{0} + \sigma^2 \boldsymbol{I} + \boldsymbol{0} \\
&\quad = \sigma^2 \boldsymbol{I},
\end{aligned}$$

where it was used that the covariance of the sum of independent random variables is the sum of the covariances. Using the resulting expectation and variance of the likelihood of the observed signal, the distribution of the observed signal given in equation (4.5) can be rewritten as

$$\boldsymbol{y}|\boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2 \sim \mathcal{CN}\big(\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}, \sigma^2 \boldsymbol{I}\big).$$

Using the expression of a generic complex multivariate Gaussian given by equation (2.8), the pdf of the likelihood can be written as follows:

$$
\begin{aligned}
&f\big(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x},\sigma^2\big) && (4.6a)\\
&= \frac{1}{\pi^M \det(\sigma^2 \boldsymbol{I})} \exp\Big(-\big(\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\big)^H \big(\sigma^2\boldsymbol{I}\big)^{-1}\big(\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\big)\Big) && (4.6b)\\
&= \frac{1}{\pi^M \prod_{m=0}^{M-1}\sigma^2} \exp\left(-\frac{\big(\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\big)^H \boldsymbol{I}\big(\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\big)}{\sigma^2}\right) && (4.6c)\\
&= \frac{1}{\pi^M \sigma^{2M}} \exp\left(-\frac{\|\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\|_2^2}{\sigma^2}\right). && (4.6d)
\end{aligned}
$$

Now that the distribution of the likelihood is known, the priors of the parameters making up the observed signal are introduced, some of which are chosen in such a way as to be conjugate priors with respect to the signal likelihood.

4.2.2 Signal parameter priors

The choice of the prior distributions of the parameters are made in such a way as to make the calculations of the posteriors mathematically tractable, making use of conjugate priors, at the same time as the physical interpretations are preserved.

As the target scene $(\boldsymbol{x},\boldsymbol{\epsilon})$ is assumed to be sparse, namely having a small number of nonzero elements, the priors of the target scene model parameters $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$ should enforce sparsity. With this in mind, the prior of the target amplitudes are assumed to be Bernoulli-complex Gaussian distributed, $x_{\overline{m}}|w,\sigma_x^2 \sim Ber\mathcal{CN}\big(w,0,\sigma_x^2\big)$, with the mixture pdf

$$
f(x_{\overline{m}}|w,\sigma_x^2) = (1-w)\delta(|x_{\overline{m}}|) + w\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right), \tag{4.7}
$$

which enforces sparsity. The level of sparsity is determined by the parameter $w \in [0,\, 1]$, whose value is the probability that the corresponding variable $x_{\overline{m}}$ is nonzero. In other words, this prior represents a target being present at the $\overline{m}$:th frequency bin with probability w and power σ_x^2.

If a target is present at the $\overline{m}$:th frequency bin, the grid mismatch for this target should be determined. It therefore seems reasonable to model the grid matches $\epsilon_{\overline{m}}$ depending on the reciprocal incoming target $x_{\overline{m}}$, and if no target is present in a specific frequency bin, there should be no grid mismatch. The prior for the grid mismatches is therefore set as

$$
f(\epsilon_{\overline{m}}|x_{\overline{m}}) = \begin{cases} \mathbb{1}_{[-0.5,0.5]}(\epsilon_{\overline{m}}), \text{ if } x_{\overline{m}} \neq 0 \\ \delta(\epsilon_{\overline{m}}), \text{ if } x_{\overline{m}} = 0. \end{cases} \tag{4.8}
$$

The prior of the noise power σ^2 is given by an inverse-gamma distribution $\sigma^2|\gamma_0,\gamma_1 \sim \mathcal{IG}(\gamma_0,\gamma_1)$ mostly because this prior is conjugate to the signal likelihood, seen in equation (4.6d). The hyperparameters γ_0 and γ_1 are the shape and scale parameters , respectively, for the inverse-gamma distribution, enabling the selection of an informative or flat prior depending on how much information is available about the noise power. The pdf of the inverse-gamma prior for the signal power is

$$f\big(\sigma^2|\gamma_0,\gamma_1\big) \propto \frac{e^{-\gamma_1/\sigma^2}}{(\sigma^2)^{\gamma_0+1}}\mathbb{1}_{[0,+\infty[}\big(\sigma^2\big).$$

The prior of the target signal power σ_x^2 is also given by an inverse-gamma prior, denoted as $\sigma_x^2|\beta_0,\beta_1 \sim \mathcal{IG}(\beta_0,\beta_1)$. As for the noise power, the hyperparameters for the target signal power are chosen carefully with respect to the prior knowledge about the target scene. The pdf of this distribution is

$$f\big(\sigma_x^2|\beta_0,\beta_1\big) \propto \frac{e^{-\beta_1/\sigma_x^2}}{(\sigma_x^2)^{\beta_0+1}}\mathbb{1}_{[0,+\infty[}\big(\sigma_x^2\big).$$

The hyperparameters γ_0, γ_1, β_0 and β_1 are chosen in a suitable way as to give the sought-after mean and variance of the prior distributions of the thermal noise power σ^2 and target power σ_x^2. The mean $\mathbb{E}[g]$ and variance $\mathrm{Var}[g]$ of a generic inverse gamma distribution $g \sim \mathcal{IG}(\nu_0,\nu_1)$ are given by

$$\mathbb{E}[g] = \frac{\nu_1}{\nu_0-1}, \qquad \nu_0 > 1 \tag{4.9a}$$

$$\mathrm{Var}[g] = \frac{\nu_1^2}{(\nu_0-1)^2(\nu_0-2)}, \quad \nu_0 > 2 \tag{4.9b}$$

[41]. Hyperparameter tuning is done using these equations.

The prior of the level of occupancy w is given by a uniform distribution on the unit interval, $w \sim \mathcal{U}_{[0,1]}$, since no knowledge about the sparsity of the signal is available. The pdf of this prior is

$$f(w) = 1 \text{ for } w \in [0,1].$$

Given the signal likelihood and signal parameter priors, the signal parameter posteriors can be derived.

4.2.3 Signal parameter posteriors

The final components needed for retrieving the estimated target scene using Bayesian inference are the posterior distributions of the model parameters. These are derived with the aid of the known likelihood and given priors above. The detailed derivations of the posterior distributions can be seen in Appendix B, this section serves as a summary.

As mentioned, the model parameters

$$\boldsymbol{\theta} = \left[\sigma^2, w, \sigma_x^2, \boldsymbol{x}^T, \boldsymbol{\epsilon}^T\right]^T$$

are to be sampled elementwise using a hybrid-Gibbs sampler. For this purpose, the posterior distributions of the elements $x_{\overline{m}}$ and $\epsilon_{\overline{m}}$ of the amplitude vector $\boldsymbol{x}$ and grid mismatch vector $\boldsymbol{\epsilon}$, respectively, also have to be known.

Starting from the joint posterior pdf of $\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2 | \boldsymbol{y}$, the posteriors of the model parameters are derived. Utilising the dependencies between the model parameters seen in the hierarchical Bayesian model in Figure 4.1, the joint posterior pdf can be written as

$$f\left(\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2 | \boldsymbol{y}\right) = f\left(\boldsymbol{y}|\boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2\right) f(\boldsymbol{\epsilon}|\boldsymbol{x}) f\left(\boldsymbol{x}|w, \sigma_x^2\right) f(w) f\left(\sigma_x^2\right) f\left(\sigma^2\right). \tag{4.10}$$

Using the fact that the following equality holds:

$$\boldsymbol{y} - \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} = \boldsymbol{e}_{\overline{m}} - \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) x_{\overline{m}},$$

where

$$\boldsymbol{e}_{\overline{m}} \coloneqq \boldsymbol{y} - \sum_{i \neq \overline{m}} \boldsymbol{f}_i(\epsilon_i) x_i,$$

one can obtain expressions of the posterior distributions of each separate element $x_{\overline{m}}$ and $\epsilon_{\overline{m}}$. Conditioning on all the parameters in the joint posterior pdf apart from $x_{\overline{m}}$ and $\epsilon_{\overline{m}}$ gives the conditional joint posterior

$$\begin{aligned} & f\left(\epsilon_{\overline{m}}, x_{\overline{m}} | \boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2\right) \\ & \quad \propto \exp\left(- \frac{|x_{\overline{m}}|^2 - x_{\overline{m}} \boldsymbol{e}_{\overline{m}}^H \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) - x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}}}{\sigma^2} \right) \\ & \qquad \times f(\epsilon_{\overline{m}} | x_{\overline{m}}) f\left(x_{\overline{m}} | w, \sigma_x^2\right), \end{aligned}$$

from which the conditional posteriors of $x_{\overline{m}}$ and $\epsilon_{\overline{m}}$ are derived.

Following the derivation given in Appendix B.1, the $\overline{m}$:th element of $\boldsymbol{x}$ is distributed according to a Bernoulli Gaussian distribution

$$x_{\overline{m}} | \boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2 \sim \mathit{Ber}\mathcal{CN}\left(w_{\overline{m}}, \mu_{\overline{m}}, \eta_{\overline{m}}^2\right)$$

with parameters

$$\begin{aligned} \eta_{\overline{m}}^2 &\coloneqq \frac{\sigma^2 + \sigma_x^2}{\sigma^2 \sigma_x^2} \\ \mu_{\overline{m}} &\coloneqq \frac{\eta_{\overline{m}}^2}{\sigma^2} \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}} \\ w_{\overline{m}} &\coloneqq \frac{w \frac{\eta_{\overline{m}}^2}{\sigma_x^2} \exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)}{1 - w + w \frac{\eta_{\overline{m}}^2}{\sigma_x^2} \exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)}. \end{aligned} \tag{4.11}$$

The derivation of the posterior distribution of $\epsilon_{\overline{m}}$ in Appendix B.2 concludes that the $\overline{m}$:th element of $\boldsymbol{\epsilon}$ is distributed according to a dilated and truncated generalised von Mises distribution of order M:

$$\epsilon_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}, \sigma^2 \sim \mathrm{dGvM}_{M\,[-0.5,0.5]}(\boldsymbol{\kappa}, \boldsymbol{\phi})$$

with parameters

$$\begin{aligned} \boldsymbol{\kappa} &\in \mathbb{R}^M & \boldsymbol{\phi} &\in \mathbb{R}^M \\ \kappa_m &:= \frac{2}{\sigma^2\sqrt{M}}|b_m| & \phi_m &:= \arg(b_m) \\ \\ \boldsymbol{b} &:= x_{\overline{m}}^* \boldsymbol{u}_{\overline{m}}^* \odot \boldsymbol{e}_{\overline{m}} & \boldsymbol{u} &\in \mathbb{C}^{M\times\overline{M}} \\ b_m &:= [\boldsymbol{b}]_m & [\boldsymbol{u}_{\overline{m}}]_m &:= \exp\big(j2\pi m\overline{m}/\overline{M}\big). \end{aligned}$$

It has a density

$$f\big(\epsilon_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}, \sigma^2\big) \tag{4.12a}$$

$$\propto \exp\left(\sum_{m=0}^{M-1} \kappa_m \cos\left(2\pi\frac{\epsilon_{\overline{m}}}{\overline{M}}m - \phi_m\right)\right) f(\epsilon_{\overline{m}}|x_{\overline{m}}). \tag{4.12b}$$

Given an observed signal $\boldsymbol{y}$ and given target scene $(\boldsymbol{x}, \boldsymbol{\epsilon})$, the conditional posteriors of the parameters σ^2, w and σ_x^2 can be readily derived as in Appendices B.3, B.4 and B.5, respectively. Assuming that the amplitude vector $\boldsymbol{x}$ has n_1 nonzero elements, hence $n_0 = \overline{M} - n_1$ components that equals zero, the conditional posterior distributions can be summarised as

$$\begin{aligned} \sigma^2|\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\epsilon} &\sim \mathcal{IG}\big(\gamma_0 + M, \gamma_1 + \|\boldsymbol{y} - \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\|_2^2\big) \\ w|\boldsymbol{x} &\sim \mathrm{Beta}(n_1 + 1, n_0 + 1) \\ \sigma_x^2|\boldsymbol{x} &\sim \mathcal{IG}\big(\beta_0 + n_1, \beta_1 + \|\boldsymbol{x}\|_2^2\big). \end{aligned}$$

Note that the conditional posteriors of σ^2 and σ_x^2 also are inverse-gamma distributed as their priors due to the priors being conjugate priors to the likelihood of the observed signal.

With the given priors, signal likelihood and posteriors of the model parameters, the hybrid-Gibbs sampler for performing Bayesian inference can be constructed.

4.3 Signal parameter sampling with hybrid-Gibbs

In this section, the construction of the hybrid-Gibbs sampler is described. After the overall algorithm has been summarised, the details of how the target amplitude elements $x_{\overline{m}}$ and the grid mismatch elements $\epsilon_{\overline{m}}$ are sampled are given.

The goal upon receiving a signal $\boldsymbol{y}$ is to compute the target scene $(\boldsymbol{x}, \boldsymbol{\epsilon})$, that is modelled using the Bayesian hierarchy given in Figure 4.1. Because the conditional distributions of the model parameters are known and easy to sample from, apart from the grid mismatch vector $\boldsymbol{\epsilon}$ which will be sampled using a Metropolis-Hastings approach, the Gibbs sampler is suitable for sampling from the joint conditional posterior

$$f\big(\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2 | \boldsymbol{y}\big)$$

given by equation (4.10). This leads to the hybrid-Gibbs sampler specified in the following subsection.

4.3.1 Constructing the hybrid-Gibbs sampler

Apart from the observed signal $\boldsymbol{y}$, the hyperparameters (β_0, β_1), (γ_0, γ_1) for the noise and target signal powers σ^2 and σ_x^2, respectively, are needed for running the hybrid-Gibbs sampler since the priors of the model parameters must be defined beforehand. The hybrid-Gibbs sampler for sampling the model parameters and computing the MMSEs is summarised in Algorithm 1. Note that the initial grid mismatch samples only are drawn for the elements of $\boldsymbol{\epsilon}^{(0)}$ corresponding to the nonzero components of $\boldsymbol{x}^{(0)}$ according to the prior of $\epsilon_{\overline{m}}$ seen in equation (4.8). From this prior one observes that $x_{\overline{m}} = 0$ implies that $\epsilon_{\overline{m}} = 0$ due to the point mass $\delta(\epsilon_{\overline{m}})$, meaning that $\epsilon_{\overline{m}} = 0$ with probability 1. Sampling a uniform $\epsilon_{\overline{m}}$ is thus only necessary if $x_{\overline{m}} \neq 0$, otherwise one simply lets $\epsilon_{\overline{m}} = 0$. From the inner loop of the hybrid-Gibbs sampler algorithm, one notes that a slight difference in notation has been introduced. In this inner loop, the target amplitude and grid mismatch vectors $\boldsymbol{x}_{-\overline{m}}^{(n)}$ and $\boldsymbol{\epsilon}_{-\overline{m}}^{(n)}$, respectively, have the following definitions (pay attention to the superscripts)

$$\boldsymbol{x}_{-\overline{m}}^{(n)} = \Big[x_0^{(n)}, \dots, x_{\overline{m}-1}^{(n)}, x_{\overline{m}+1}^{(n-1)}, \dots, x_{\overline{M}-1}^{(n-1)}\Big] \tag{4.13}$$

$$\boldsymbol{\epsilon}_{-\overline{m}}^{(n)} = \Big[\epsilon_0^{(n)}, \dots, \epsilon_{\overline{m}-1}^{(n)}, \epsilon_{\overline{m}+1}^{(n-1)}, \dots, \epsilon_{\overline{M}-1}^{(n-1)}\Big], \tag{4.14}$$

where after adding the sample $x_{\overline{m}}^{(n)}$ to the target amplitude vector $\boldsymbol{x}_{-\overline{m}}^{(n)}$ gives

$$\boldsymbol{x}^{(n)} = \Big[x_0^{(n)}, \dots, x_{\overline{m}-1}^{(n)}, x_{\overline{m}}^{(n)}, x_{\overline{m}+1}^{(n-1)}, \dots, x_{\overline{M}-1}^{(n-1)}\Big], \tag{4.15}$$

which is conditioned upon for sampling the next grid mismatch $\epsilon_{\overline{m}}^{(n)}$.

Algorithm 1: Hybrid-Gibbs sampler for estimating target scene of a given signal $\boldsymbol{y}$.

```
Require: Signal $\boldsymbol{y}$, hyperparameters $(\beta_0, \beta_1)$, $(\gamma_0, \gamma_1)$, burn-in
  samples $N_{bi}$, samples to keep $N_r$, and subsample size $K$.
Result: Samples: $\left(\sigma^{2^{(n)}}, w^{(n)}, \sigma_x^{2^{(n)}}, \boldsymbol{x}^{(n)}, \boldsymbol{\epsilon}^{(n)}\right)_{n=1,2,\ldots,N_r}$
        Estimators: $\hat{\sigma}^2_{\mathrm{MMSE}}, \hat{w}_{\mathrm{MMSE}}, \hat{\sigma}^2_{x\,\mathrm{MMSE}}, \hat{\boldsymbol{x}}_{\mathrm{MMSE}}, \hat{\boldsymbol{\epsilon}}_{\mathrm{MMSE}}$
{Initialisation}
$\mathrm{w}^{(0)} \sim \mathcal{U}_{[0,\,1]}$
$\sigma_x^{2^{(0)}} \sim \mathcal{IG}(\beta_0, \beta_1)$
$\boldsymbol{x}^{(0)} \sim \prod_{i=0}^{\overline{M}-1} Ber\mathcal{CN}\left(w^{(0)}, 0, \sigma_x^{2^{(0)}}\right)$
$\boldsymbol{\epsilon}^{(0)} \sim \mathcal{U}_{[-0.5,\,0.5]}$
{Iterations}
for n=1 to $N_{bi} + KN_r$ do
    $\sigma^{2^{(n)}}|\boldsymbol{y}, \boldsymbol{x}^{(n-1)}, \boldsymbol{\epsilon}^{(n-1)} \sim \mathcal{IG}\left(\gamma_0 + M, \gamma_1 + \left\|\boldsymbol{y} - \boldsymbol{F}\left(\boldsymbol{\epsilon}^{(n-1)}\right)\boldsymbol{x}^{(n-1)}\right\|_2^2\right)$
    $w^{(n)}|\boldsymbol{x}^{(n-1)} \sim \mathrm{Beta}(n_1 + 1, n_0 + 1)$
    $\sigma_x^{2^{(n)}}|\boldsymbol{x}^{(n-1)} \sim \mathcal{IG}\left(\beta_0 + n_1, \beta_1 + \left\|\boldsymbol{x}^{(n-1)}\right\|_2^2\right)$
    for $\overline{m}$=0 to $\overline{M} - 1$ do
        $x_{\overline{m}}^{(n)}|\boldsymbol{y}, \boldsymbol{x}_{-\overline{m}}^{(n)}, w^{(n)}, \sigma_x^{2^{(n)}}, \sigma^{2^{(n)}}, \boldsymbol{\epsilon}^{(n-1)} \sim Ber\mathcal{CN}\left(w_{\overline{m}}, \mu_{\overline{m}}, \eta_{\overline{m}}^2\right)$
        if $x_{\overline{m}}^{(n)} = 0$ then
            $\epsilon_{\overline{m}}^{(n)} = 0$
        else
            $\epsilon_{\overline{m}}^{(n)}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}^{(n)}, \boldsymbol{x}^{(n)}, \sigma^{2^{(n)}} \sim \mathrm{dGvM}_{[-0.5,\,0.5]}(\boldsymbol{\kappa}, \boldsymbol{\phi})$ according
             to Algorithm 4 in Appendix A
        end
    end
end
{Estimators}
$\hat{\theta}_{\mathrm{MMSE}} = \frac{1}{N_r}\sum_{n=1}^{N_r} \theta^{(tK+N_{bi})}$ for each model parameter according to
  equation (4.4)
```

In practice, the level of occupancy $w_{\overline{m}}$ defined in equation (4.11) for determining the probability that the element $x_{\overline{m}}$ is nonzero can potentially become undefined. This is a consequence of the potential simultaneous overflow of the exponential factors in the numerator and denominator, which would lead to an undefined $w_{\overline{m}}$ (`NaN`). To overcome this problem, the value of $w_{\overline{m}}$ is set to one when this simultaneous overflow of the numerator and denominator occurs. This is reasonable since the limit of the fraction approaches one when both exponents tend to infinity.

The posteriors of the model parameters w, σ^2, σ_x^2, and $x_{\overline{m}}$ follow well known real distributions and are therefore easy to sample from. Algorithm 3 in Appendix A is used to sample the target amplitude elements, with the additional step of adding a complex mean to the sample. A more intricate method for sampling the grid mismatch elements $\epsilon_{\overline{m}}$ is needed. The details of how this is done are given in the following subsection.

4.3.2 Grid mismatch sampling with Metropolis-Hastings

The elementwise sampling of $\epsilon_{\overline{m}}$ is done using the MH algorithm since the conditional posterior is a dilated and truncated generalised von Mises distribution given in equation (4.12b), that has no easy way of being sampled. Instead of trying to directly sample from its conditional posterior,

$$f\left(\epsilon_{\overline{m}}|\boldsymbol{y},\boldsymbol{\epsilon}_{-\overline{m}},\boldsymbol{x},\sigma^2,w,\sigma_x^2\right) \propto \exp\left(\sum_{m=0}^{M-1}\kappa_m\cos\left(2\pi\frac{\epsilon_{\overline{m}}}{M}m-\phi_m\right)\right)f(\epsilon_{\overline{m}}|x_{\overline{m}}), \tag{4.16}$$

the idea of the MH algorithm is to sample a Markov chain with the distribution given by equation (4.16) as its stationary distribution. For the same reason as explained above, in the initialisation of $\boldsymbol{\epsilon}^{(0)}$ in the hybrid-Gibbs sampler, the next sample $\epsilon_{\overline{m}}^{(n)}$ is only sampled using the MH algorithm if the corresponding target amplitude is nonzero, $x_{\overline{m}}^{(n)} \neq 0$. Only grid points having a nonzero target amplitude can have a grid mismatch.

The proposal distribution in the MH algorithm is chosen to be a distribution that is efficient to sample from and as similar as possible to the target distribution. In Lasserre's article [3, Section 2.2.2], the proposal distribution was chosen as to depend on the estimated target power, $|x_{\overline{m}}|^2/\sigma^2$. If the estimated target power was below a fixed threshold, the proposal was chosen to be flat, and if it was above the threshold, the proposal was chosen to be a Gaussian. Unlike this choice of proposal, the method used in this thesis does not make use of such a proposal depending on the estimated target power. Instead, independently of the estimated target power, the proposal distribution is simply chosen to be a Gaussian with a mean taking the value of the previous sample $\epsilon_{\overline{m}}^{(n-1)}$ and some suitable variance σ_p^2. The idea behind choosing the Gaussian proposal mean to be the previous sample is to iteratively converge towards the true grid mismatch. A suitable variance is one that balances exploration and exploitation of the state space, hence the variance should neither be too small nor too large. More specifically, the proposal distribution $q\left(\epsilon_{\overline{m}}^{*}|\epsilon_{\overline{m}}^{(n-1)}\right)$ for the next sample of an element in the grid mismatch vector, $\epsilon_{\overline{m}}^{*}$, given the previous sample $\epsilon_{\overline{m}}^{(n-1)}$ is chosen as:

$$q\left(\epsilon_{\overline{m}}^{*}|\epsilon_{\overline{m}}^{(n-1)}\right) = \mathcal{N}\left(\epsilon_{\overline{m}}^{*}|\epsilon_{\overline{m}}^{(n-1)}; \epsilon_{\overline{m}}^{(n-1)}, \sigma_p^2\right).$$

Using the fact that this proposal is symmetric,

$$q\left(\epsilon_{\overline{m}}^{*}|\epsilon_{\overline{m}}^{(n-1)}\right) = q\left(\epsilon_{\overline{m}}^{(n-1)}|\epsilon_{\overline{m}}^{*}\right),$$

the acceptance probability can be written as

$$\begin{aligned}\alpha\left(\epsilon_{\overline{m}}^{*}, \epsilon_{\overline{m}}^{(n-1)}\right) &= \min\left(1, \frac{\mathrm{dGvM}_{[-0.5,\,0.5]}\left(\epsilon_{\overline{m}}^{*}; \boldsymbol{\kappa}, \boldsymbol{\phi}\right) q\left(\epsilon_{\overline{m}}^{(n-1)}|\epsilon_{\overline{m}}^{*}\right)}{\mathrm{dGvM}_{[-0.5,\,0.5]}\left(\epsilon_{\overline{m}}^{(n-1)}; \boldsymbol{\kappa}, \boldsymbol{\phi}\right) q\left(\epsilon_{\overline{m}}^{*}|\epsilon_{\overline{m}}^{(n-1)}\right)}\right)\\ &= \min\left(1, \frac{\mathrm{dGvM}_{[-0.5,\,0.5]}\left(\epsilon_{\overline{m}}^{*}; \boldsymbol{\kappa}, \boldsymbol{\phi}\right)}{\mathrm{dGvM}_{[-0.5,\,0.5]}\left(\epsilon_{\overline{m}}^{(n-1)}; \boldsymbol{\kappa}, \boldsymbol{\phi}\right)}\right).\end{aligned}$$

In practice, the fraction

$$\frac{\mathrm{dGvM}_{[-0.5,\,0.5]}\left(\epsilon_{\overline{m}}^{*}; \boldsymbol{\kappa}, \boldsymbol{\phi}\right)}{\mathrm{dGvM}_{[-0.5,\,0.5]}\left(\epsilon_{\overline{m}}^{(n-1)}; \boldsymbol{\kappa}, \boldsymbol{\phi}\right)}$$

may be undefined (`NaN`) when first computing the numerator and the denominator separately before taking the fraction due to the simultaneous overflow of both the numerator and denominator. To overcome this problem, the following rewrite is done before computing the acceptance probability in order to avoid overflow:

$$\begin{aligned}&\frac{\mathrm{dGvM}_{[-0.5,\,0.5]}\left(\epsilon_{\overline{m}}^{*}; \boldsymbol{\kappa}, \boldsymbol{\phi}\right)}{\mathrm{dGvM}_{[-0.5,\,0.5]}\left(\epsilon_{\overline{m}}^{(n-1)}; \boldsymbol{\kappa}, \boldsymbol{\phi}\right)}\\ &\quad= \frac{\exp\left(\sum_{m=0}^{M-1} \kappa_m \cos\left(2\pi \frac{\epsilon_{\overline{m}}^{*}}{M} m - \phi_m\right)\right)}{\exp\left(\sum_{m=0}^{M-1} \kappa_m \cos\left(2\pi \frac{\epsilon_{\overline{m}}^{(n-1)}}{M} m - \phi_m\right)\right)}\\ &\quad= \exp\left(\sum_{m=0}^{M-1} \kappa_m \left(\cos\left(2\pi \frac{\epsilon_{\overline{m}}^{*}}{M} m - \phi_m\right) - \cos\left(2\pi \frac{\epsilon_{\overline{m}}^{(n-1)}}{M} m - \phi_m\right)\right)\right).\end{aligned}$$

The truncation of the generalised von Mises distribution is taken care of a while-loop guaranteeing that the proposed sample $\epsilon_{\overline{m}}^{*}$ is in the interval $[-0.5,\, 0.5]$. A summary of the MH algorithm for sampling grid mismatch elements using the chosen proposal distribution is given in Algorithm 4 in Appendix A.

4.4 Estimated target scene evaluation

To evaluate the hybrid-Gibbs sampler, the target scene of a synthetically generated signal was estimated. The choice of noise power, hyperparameters, and hybrid-Gibbs parameters were all chosen in accordance with Lasserre's article [3, Chapter 4].

The synthetic target scene assumed an observation space of size $M = 32$ with $\overline{M} = M$ frequency grid points and three targets, $N = 3$, with the corresponding grid mismatches $\{0, 0.15, 0.45\}$ and post-processing signal-to-noise ratio (SNR) of 20 dB. The targets assumed to be on the indices 9, 17 and 24, respectively. The grid mismatch vector $\boldsymbol{\epsilon}$ was thus constructed as a zero vector of size $\overline{M}$ containing the values $[0, 0.15, 0.45]$ on the indices 9, 17 and 24, respectively, and the target amplitude vector $\boldsymbol{x}$ was constructed in the same way as $\boldsymbol{\epsilon}$ with the exception that each nonzero element was set to a complex number with amplitude 20 (dB units) and uniformly randomly chosen angle on the interval $[0, 2\pi]$. Figure 4.2 shows the true target scene as red stem plots. The synthetic signal was then constructed as

$$\boldsymbol{y} = \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} + \boldsymbol{n},$$

where the noise was created according to Algorithm 2 in Appendix A, with power $\sigma^2 = 1$.

The hyper-parameters (β_0, β_1), (γ_0, γ_1) were chosen as to give the desired mean and variance of the priors of the noise and target signal powers σ^2 and σ_x^2 respectively. The desired mean and variance for the noise and target signal powers were chosen as

$$\begin{aligned}
\left(\mathbb{E}[\sigma^2], \sqrt{\mathrm{Var}[\sigma^2]}\right) &= (0, 2.4) \text{ dB} \\
\left(\mathbb{E}[\sigma_x^2], \sqrt{\mathrm{Var}[\sigma_x^2]}\right) &= M_{\mathrm{dB}} + (0, 3.5) \text{ dB},
\end{aligned}$$

where $M_{\mathrm{dB}} = 10 \log_{10}(M)$. Expressing the means and variances in normal form and using the expressions given by equation (4.9) for the mean and variance of a generic inverse gamma distribution, the hyper-parameters can be computed. The resulting hyperparameters, given the specified inverse-gamma means and variances, were

$$\begin{aligned}
\gamma_0 &= \frac{1}{\left(10^{\frac{2.4}{10}}\right)^2} + 2 \\
\gamma_1 &= \gamma_0 - 1 \\
\beta_0 &= \frac{M^2}{\left(10^{\frac{M_{\mathrm{dB}}+3.5}{10}}\right)^2} + 2 \\
\beta_1 &= M(\beta_0 - 1).
\end{aligned}$$

To evaluate the hybrid-Gibbs sampler, it was executed a number of times with the given setup. The estimated target scenes for each execution were stored in order to compute the confidence intervals for each grid point. The resulting estimate of the target scene with confidence intervals is presented in Figure 4.2.

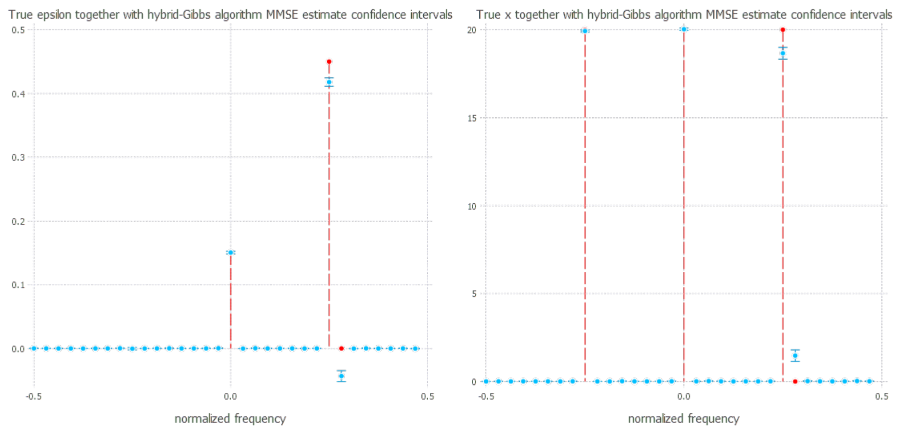


Figure 4.2: Resulting estimated target scene (blue) using the hybrid-Gibbs sampler with parameters as follows: $N_{\text{bi}} = 100$, $N_r = 1000$ and subsampling of size $K = 5$. The number of MH iterations for sampling grid mismatch elements was set to 20 and the proposal variance was set to $\sigma_p^2 = 0.05$. The synthetic signal $\boldsymbol{y}$ of size $M = 32$ was constructed with a signal power $\sigma^2 = 1$. The size of the frequency grid was set to $\overline{M} = 32$, the grid mismatch vector was constructed according to the left part of the plot (red) and target amplitude vector according to the right part of the plot (red). The estimated target scene with 95%-confidence intervals is shown in blue.

4.5 Acceleration of the hybrid-Gibbs sampler

Lasserre et.al.'s work in [3] describes a method for target scene estimation of a signal embedded in white noise. They use a sparse signal representation, where the signal is described by a Fourier dictionary parameterised by the potential grid mismatch, shown in equation (4.3). To estimate the target scene, they use a hierarchical Bayesian model, including the target scene as parameters, and Bayesian inference in the form of a hybrid-Gibbs sampler to estimate it. While this resulted in high estimation performance of the target scene, the computational load remained heavy. The goal of this thesis is to accelerate the hybrid-Gibbs sampler using deep generative models to estimate the target scene of a signal without losing too much of the estimation performance.

Examining the hybrid-Gibbs sampler in Algorithm 1 and the MH-part for sampling the grid mismatch elements in Algorithm 4 in Appendix A, one observes that the hybrid-Gibbs sampler has a triple loop. Even if the number of iterations in the second and third loops are significantly smaller than the outer one, the computational load of the algorithm can be substantially reduced if the third loop is replaced with a method of sampling the grid mismatch elements of constant computational complexity.

A way to replace the third loop would be to use a generative model for sampling the grid mismatch elements instead the MH-algorithm. The next chapter describes an attempt for how a CCGAN could be used for this task. Given the dependencies of the grid mismatch element $\epsilon_{\overline{m}}$, given as

$$\epsilon_{\overline{m}}^{(n)} | \boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}^{(n)}, \boldsymbol{x}^{(n)}, {\sigma^2}^{(n)},$$

the CCGAN seeks to model the corresponding dGvM distribution by generating grid mismatch elements as if generated from this distribution.

The second loop in the hybrid-Gibbs sampler loops over the elements of the target amplitude and grid mismatch vectors. Due to the assumption of sparseness of the signal, only a few elements of the target amplitude and grid mismatch vectors are nonzero. Using this sparsity structure, there may be a way to create a generative model that fully replaces the second loop by directly sampling the full target amplitude and grid mismatch vectors. However, this is not investigated in this thesis, but could be of interest in future work.

Chapter 5

Generative Modelling of Univariate Conditional Densities

In this chapter, the methodology for constructing a CCGAN to sample grid mismatch elements according to a dilated and truncated generalised von Mises distribution, as opposed to with the MH algorithm (described in Section 4.3.2), is described. In Section 5.1, a CCGAN is created to replicate the results presented in the CCGAN article [31, Section 4.1], where bivariate Gaussians with means placed on a unit circle parametrised by an angle were simulated, as a sanity check to ensure that the CCGANs implemented thereafter were correctly implemented. Section 5.2 describes how this CCGAN is applied to the problem of simulating univariate normal distributions with a fixed variance and varying mean on a specified interval. A similar problem is solved in Section 5.3, but with the variance also varying. Since the von Mises distribution can be seen as a normal distribution wrapped around a circle, the CCGAN used for simulating univariate Gaussians is then modified to simulate von Mises distributions with varying expectations. How this is done is presented in Section 5.4). Finally, in Section 5.5, how a CCGAN is trained to fit the grid mismatch samples generated from the MH sampler nested in the hybrid-Gibbs algorithm is detailed.

The purpose of solving these subtasks instead of directly trying to construct a CCGAN to model a truncated dGvM distribution was to examine where problems could arise and facilitate troubleshooting for how the models should be modified to finally work. Through this methodology, the understanding of the problem at hand was made more robust.

5.1 Bivariate Gaussians with means on unit circle

The results of the CCGAN paper [31, Section 4.1] were reproduced in order to ensure that the subsequent CCGANs were correctly implemented. Here, bivariate Gaussians with means placed on a unit circle parameterised by an angle were modelled using a CCGAN. Given an angle representing the mean of a Gaussian with given covariance matrix, the CCGAN should thus generate samples as if they were sampled from this specified Gaussian.

The training data for this problem consisted of a collection of 1200 samples: ten samples for each Gaussian with means uniformly placed on a unit circle and common covariance matrix $\tilde{\sigma}^2 \boldsymbol{I}_{2\times 2}$, where $\tilde{\sigma} = 0.02$. Figure 5.1 illustrates the training samples for this problem. One can view the mean as parametrised by an angle $\theta \in [0, 2\pi]$, where the angle $\theta = 0$ corresponds to the point A in Figure 5.1 and goes clockwise. The training data thus consisted of pairs $(\boldsymbol{x}, \theta)$ of data points $\boldsymbol{x} \in \mathbb{R}^2$ constituting the two-dimensional samples and corresponding angles θ giving the means of the Gaussians from which the data points are sampled.

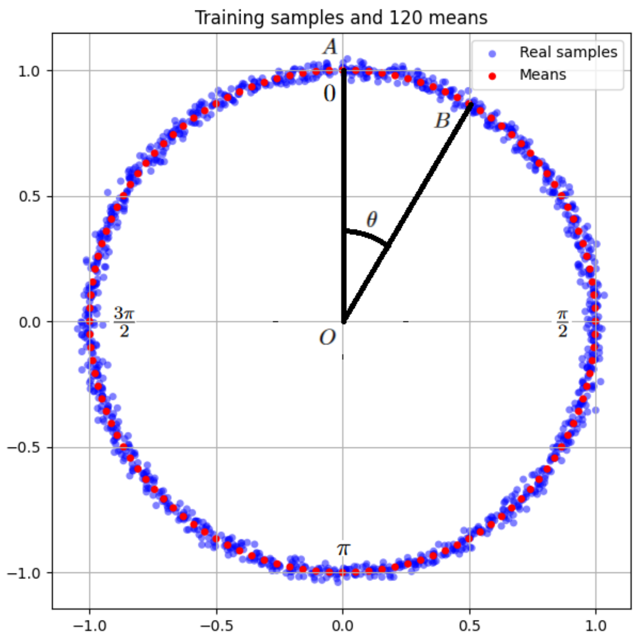


Figure 5.1: Visualisation of the 1200 training samples (blue) generated from 120 distinct Gaussians with means (red) uniformly placed on the unit circle and common covariance matrix $\tilde{\sigma}^2 \boldsymbol{I}_{2\times 2}$, where $\tilde{\sigma} = 0.02$. The label θ can be viewed as the angular parametrisation of each mean, with $\theta = 0$ corresponding to point A.

In this problem, the angle θ for a data sample is actually not the regression label, but is rather used for computing the HVDL regularisation parameters ν and σ according to the rule of thumb described in equations (3.13) and (3.14). The regression labels that are fed to the CCGAN are instead the coordinates of the mean representation. For a specific angle θ, the regression label is thus $(\sin(\theta), \cos(\theta))$. The network architecture for the CCGAN used in this experiment can be seen in Table C.1 in Appendix C. The choice of hyperparameters and setup of the training of this CCGAN is summarised in Table C.2. Note that the ADAM parameters were set differently than the default values mentioned in Section 3.1, this was because the parameters were chosen to be same as in the CCGAN article [31]. The qualitative results of this experiment can be seen in Figure 5.2. Here, 12 angles that were not part of the training data were chosen, from each of which the CCGAN generated 100 samples to be compared with an equal number of authentic Gaussian samples in a scatter plot.

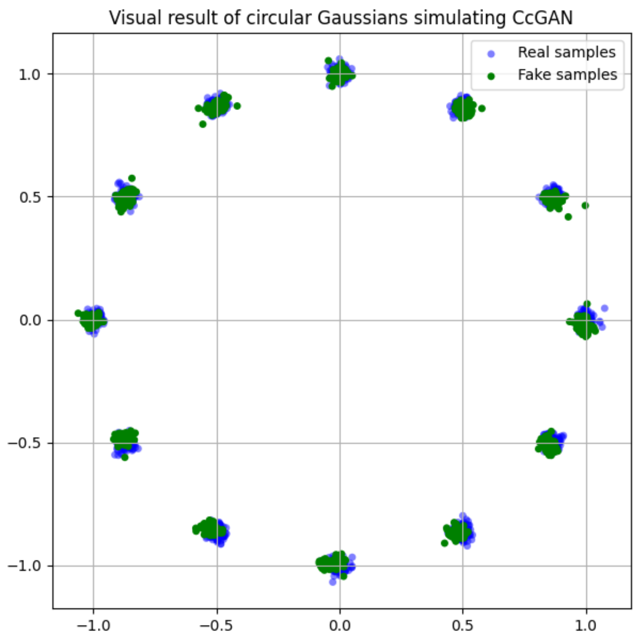


Figure 5.2: Qualitative result of the CCGAN simulating 2D Gaussians with fixed covariance matrix and expectations taking values on the unit circle. The blue points are true Gaussian samples, while the green ones were generated by the CCGAN.

The qualitative result shows that the CCGAN is able to simulate bivariate Gaussians with fixed covariance matrix $\tilde{\sigma}^2 \boldsymbol{I}_{2\times 2}$, conditioned on an angle $\theta \in [0, 2\pi]$ that parameterises the mean on the unit circle. This is seen since the real and the fake samples somewhat overlap. This gives confidence in the implementation of the CCGAN. Using this CCGAN as a steppingstone, the subsequent subtasks are hoped to be solved.

5.2 Univariate Gaussians with varying mean

In this section, the CCGAN in the previous section is modified to approximately sample a univariate Gaussian with fixed standard deviation and varying expectation on a specified real interval. The training data consisted of a collection of 10 samples drawn from 40 Gaussians with common standard deviation $\tilde{\sigma} = 0.02$ and distinct expectations μ uniformly placed on the interval $[0, 4]$, giving 400 samples in total. The regression label in this task is thus a mean $\mu \in [0, 4]$.

The CCGAN architecture for this task is presented in Table C.3. One notes that it is similar to the previous CCGAN in Table C.1, except that the regression label now instead is the one-dimensional mean $\mu \in [0, 4]$ and the samples x also are one-dimensional. The hyperparameters and training setup is presented in Table C.4.

A qualitative evaluation was done by comparing the KDEs of authentic samples and generated CCGAN samples for 12 means, not appearing in the training data, uniformly placed on the interval $[0, 4]$. See Figure 5.3 for this result.

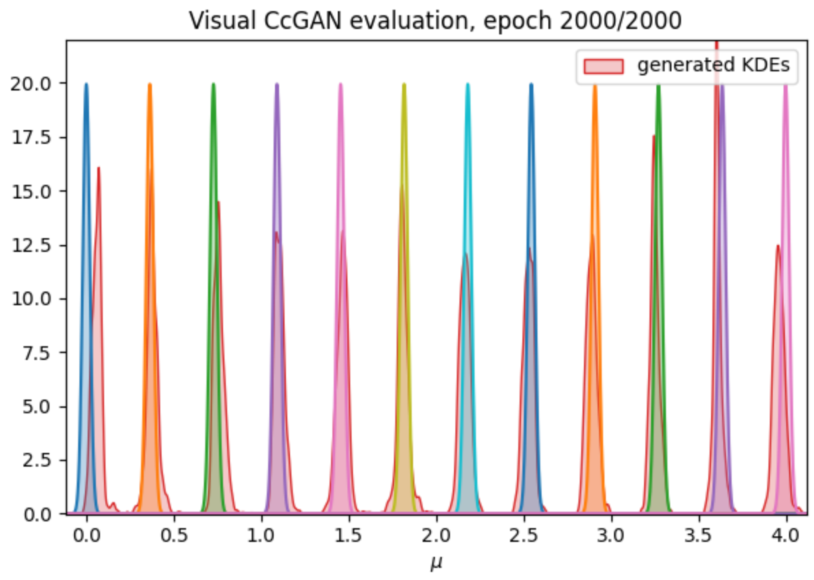


Figure 5.3: Qualitative result of the univariate Gaussians simulating CCGAN with fixed standard deviation and expectations taking values on the interval $[0, 4]$. Twelve means not part of the training set were uniformly picked from the given interval. For each of these, the true pdfs were plotted, seen in different colors. The red KDEs are the corresponding KDEs of the CCGAN generated samples.

Observing the qualitative results in Figure 5.3, one notes that the CC-GAN is fairly able to simulate univariate Gaussians with fixed standard deviation $\tilde{\sigma} = 0.02$, conditioned on a mean $\mu \in [0, 4]$. There appears to be a small bias in the generated KDEs, both in the mean and standard deviation. Considering the mean, this is more apparent further out the boundaries of the interval the mean takes values on. This is seen by how well the peaks of the true pdfs and corresponding generator KDEs align. The CCGAN estimates the standard deviation reasonably well but seems to have a larger estimation due to the peaks of the generated KDEs not reaching as high as the true pdfs.

5.3 Univariate Gaussians with varying mean and variance

While the standard deviation in the previous example was fixed, the CC-GAN in this subtask conditions on both the mean and standard deviation. The training data for this task consisted of a collection of 20 samples generated from 120 distinct Gaussians, giving a total of 2400 data points. The parameters of the Gaussians were uniformly chosen from the specified two-dimensional regression label space as $(\mu, \tilde{\sigma}) \in [0, 4] \times [0.05, 0.1]$.

The CCGAN architecture for this task is presented in Table C.5, and the hyperparameters and training setup is summarised in Table C.6. The qualitative evaluation was done by plotting the true densities and CCGAN generated KDEs of 12 Gaussians, with parameters uniformly placed in the specified regression label space, not contained in the training data. The result is illustrated in Figure 5.4.

The qualitative result of this CCGAN shows poor results on the estimation of the standard deviation, and it seems that a change in the conditioning on the standard deviation does not have an effect on the standard deviation estimation. Conditioning the CCGAN on standard deviations throughout the interval $[0.05, 0.1]$ does not seem to influence the estimated CCGAN distribution. This can be seen in how the red KDEs seem to have very similar standard deviations for the three different standard deviation labels. However, the means of the Gaussians are fairly well estimated, but with a slight bias. A reason why the CCGAN generated samples may be independent of the addition of the standard deviation to the regression labels may be that it is more difficult to discriminate between two Gaussians having the same mean and a similar standard deviation, than discriminating between two Gaussians having the same standard deviation and slightly different means. This may be a reason why the CCGAN has a tendency to generate samples that are very close to the conditioned mean, because these are easier to distinguish between as long as the standard deviation is sufficiently small.

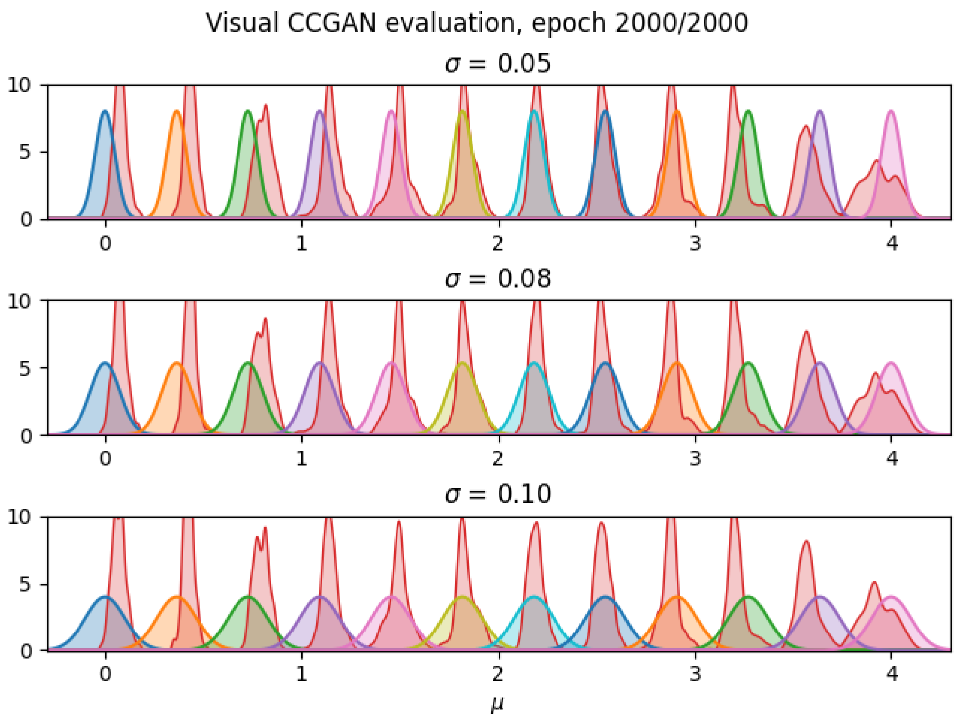


Figure 5.4: Qualitative result of the univariate Gaussians simulating CCGAN with varying expectation and variance. Twelve means and three standard deviations not part of the training set were uniformly picked from the given regression label space. For each of these, the corresponding true Gaussian was plotted, seen in different colors in the figure. The red KDEs are the corresponding KDEs of the CCGAN generated samples.

5.4 von Mises distributions with varying mean

In this section, CCGANs are trained to simulate von Mises distributions $\mathrm{vM}(\kappa, \phi)$ with a common fixed concentration κ and conditioned on the mean $\phi \in [-\pi, \pi]$, acting as the regression label, using a CCGAN. In a similar fashion as for the previous subtask, the training data consisted of a collection of 20 samples from 120 distinct von Mises distributions with common concentration $\kappa = 20$ and varying mean ϕ uniformly placed on the interval $[-\pi, \pi]$, giving 2400 samples in total. Because the von Mises distribution wraps around its domain, the regression labels $c = -\pi$ and $c = \pi$ correspond to the same distribution. In the training data, the regression labels therefore do not include the label $c = \pi$.

What makes simulating a von Mises distribution more complicated compared to a Gaussians is that the distribution wraps around the interval $[-\pi, \pi]$. This was circumvented by modifying the generator output x in the following way:

$$x_{\text{mod}} = \begin{cases} x + 2\pi \text{ if } x < -\pi \\ x - 2\pi \text{ if } x > \pi \end{cases}$$

to ensure that the output belonged to the interval $[-\pi, \pi]$.

The CCGAN architecture for this problem is identical to the preceding one seen in Figure C.3, except that the regression label instead is the mean $\phi \in [-\pi, \pi]$. The training setup and hyperparameters are summarised in Table C.8.

A qualitative evaluation was done by choosing five regression labels ϕ uniformly placed on the interval $[-\pi, \pi]$ and for each of these, the true von Mises pdf for the fixed concentration $\kappa = 20$ and the histograms of the CCGAN generated samples, were plotted. The result can be seen in Figure 5.5.

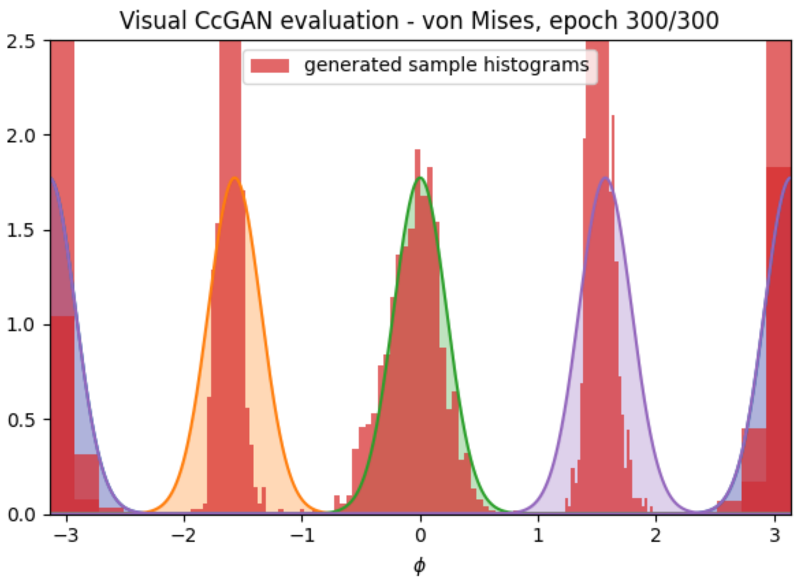


Figure 5.5: Qualitative result of the von Mises simulating CCGAN with fixed standard concentration $\kappa = 20$ and expectations taking values on the interval $[-\pi, \pi]$. Five means not part of the training set were uniformly picked from the given interval. For each of these, the true pdf was computed and plotted, seen as the pdfs of different colors in the plot. The red histograms are the corresponding CCGAN generated samples.

The qualitative evaluation shows that the CCGAN performs reasonably well on estimating the mean, but poorly on the estimation of the concentration. Like in the subtask of simulating univariate Gaussians with fixed variance and varying mean, the CCGAN performs worse on labels placed further away from the centre of the regression label space. Reminiscent of the discussion of the qualitative evaluation of the previous subtask, the CCGAN in this problem also has a tendency to estimate the concentration substantially higher than the true one. The suggested reason for this phenomenon is that it is easier to distinguish between two von Mises distributions with shared (small) concentration and slightly different means, than if the concentration was slightly different and the mean was the same. Thus, when the CCGAN conditions on a particular mean, it tends to sample close to this mean.

5.5 Dilated and truncated generalised von Mises distributions

The purpose of simulating $\mathrm{dGvM}_{[-0.5,0.5]}$ distributions using a CCGAN is to replace the MH algorithm for sampling the grid mismatch elements in order to accelerate the hybrid-Gibbs sampler for estimation of the target scene, given an observed signal. Using a CCGAN for grid mismatch generation would reduce the complexity of the algorithm since the MH algorithm requires several iterations to reach the stationary distribution from which the final grid mismatch sample is drawn from. The dimension of data used in Lasserre's article [3] was $M = 32$, implying that the order of the $\mathrm{dGvM}_{[-0.5,0.5]}$ distributions would be 32. However, the easier task of simulating $\mathrm{dGvM}_{[-0.5,0.5]}$ distributions of order 2 is first examined.

As mentioned in Section 4.3.2, the elements $\epsilon_{\overline{m}}$ of the grid mismatch vector $\boldsymbol{\epsilon}$ are to be sampled elementwise using the MH algorithm. By inspecting the posterior of each distinct grid mismatch element $\epsilon_{\overline{m}}$ in equation (4.12b), one observes that they are dependent on the grid point $\overline{m}$. If one were to replace the MH sampler to simulate $\epsilon_{\overline{m}} \sim \mathrm{dGvM}_{[-0.5,0.5]}(\boldsymbol{\kappa}, \boldsymbol{\phi})$ using a CCGAN, an option could therefore be to use regression labels $(\boldsymbol{\kappa}, \boldsymbol{\phi}, \overline{\boldsymbol{m}})$ of dimensions $2M + \overline{M}$, where $\overline{\boldsymbol{m}}$ is one-hot vectors corresponding to the grid elements in question, namely having value one at the grid element index $\overline{m}$. However, to reduce the complexity of the CCGAN, one CCGAN was constructed for each grid element in order to avoid having the CCGAN conditioning on regression labels containing one-hot vectors. In other words, $\overline{M}$ distinct CCGANs were constructed, one for each grid element, trained on distinct datasets. Examining the inner for-loop in the hybrid-Gibbs sampler in Algorithm 1, the sampling of $\epsilon_{\overline{m}}^{(n)}$ would then be done by subsequent CCGANs.

In what follows, a number of input architectures for the CCGAN are presented, the methodology for the dataset collection is described, and the evaluation of these CCGANs is given.

5.5.1 Input architectures

As the goal is to simulate a dGvM distribution parameterised by the parameters $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$, it seems reasonable to let the regression labels consist of these vectors. Nevertheless, other regression labels are suggested, with the reason for exploring other input architectures that may perform better. More specifically, four input architectures/regression labels have been examined. What the four cases have in common is that they all use the structure of the parameters $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$. The reader may want to be reminded of the following parameters, given in Section 4.2.3:

$$\begin{aligned} \boldsymbol{\kappa} &\in \mathbb{R}^M & \boldsymbol{\phi} &\in \mathbb{R}^M \\ \kappa_m &:= \frac{2}{\sigma^2\sqrt{M}}|b_m| & \phi_m &:= \arg(b_m) \end{aligned}$$

$$\begin{aligned} \boldsymbol{b} &:= x_{\overline{m}}^* \boldsymbol{u}_{\overline{m}}^* \odot \boldsymbol{e}_{\overline{m}} & \boldsymbol{u} &\in \mathbb{C}^{M\times\overline{M}} \\ b_m &:= [\boldsymbol{b}]_m & [\boldsymbol{u}_{\overline{m}}]_m &:= \exp\big(j2\pi m\overline{m}/\overline{M}\big). \end{aligned}$$

Moreover, only the input architecture distinguishes the CCGANs, and the training setup are the same.

Regression label of the form $(\boldsymbol{\kappa}, \boldsymbol{\phi})$

This is the most natural regression label, since $(\boldsymbol{\kappa}, \boldsymbol{\phi})$ directly parameterise the dGvM distribution. As $\boldsymbol{\kappa}, \boldsymbol{\phi} \in \mathbb{R}^M$, the regression labels are in this case of dimension $2M$. Reminiscent of the structures of $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$, presented in Section 4.2.3, one may imagine other parameters that indirectly parameterise the dGvM distribution. Some suggestions are presented in what follows.

Regression label of the form $(x_{\overline{m}}, \boldsymbol{e}_{\overline{m}})$

The complex vector $\boldsymbol{b}$ is one representation of the parameters $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$, but anything that determines $\boldsymbol{b} := x_{\overline{m}}^* \boldsymbol{u}_{\overline{m}}^* \odot \boldsymbol{e}_{\overline{m}}$ works too. The vector $\boldsymbol{u}_{\overline{m}}$ is determined by $\overline{m}$ only and $x_{\overline{m}}$ is a single complex number. It is $\mathbf{e}_{\overline{m}} := \boldsymbol{y} - \sum_{i\neq\overline{m}} \boldsymbol{f}_i(\epsilon_i)x_i$ that is most involved. It approximately represents the part of $\boldsymbol{y}$ corresponding to signal component $\overline{m}$ plus noise, given that $\boldsymbol{x}_{-\overline{m}}$ and $\boldsymbol{\epsilon}_{-\overline{m}}$ are roughly correct.

It seems that $(x_{\overline{m}}, \boldsymbol{e}_{\overline{m}})$ is a less complicated representation to use. It does in particular not contain the varying periodicity introduced by $\boldsymbol{u}_{\overline{m}}$. As mentioned, one distinct CCGAN is trained for each grid point, with the hope of incorporating the dependency of $\overline{m}$. A choice of regression label could therefore be $(x_{\overline{m}}, \mathbf{e}_{\overline{m}})$. Because $x_{\overline{m}}$ is complex $\mathbf{e}_{\overline{m}}$ is a complex vector, they cannot be directly inputted to the CCGAN, which only takes real values inputs. Complex features should be represented by their real and imaginary parts or by their amplitude and phase. In this work, a complex value is represented by its real and imaginary parts, thus one complex value corresponds to two input nodes of the CCGAN. It thus follows that the regression label $(x_{\overline{m}}, \boldsymbol{e}_{\overline{m}}) \in \mathbb{R}^{2+2M}$.

Regression label of the form $(|x_{\overline{m}}|, \boldsymbol{\kappa}/|x_{\overline{m}}|, \boldsymbol{\phi})$

The vector $\boldsymbol{u}_{\overline{m}}$ does not change the magnitude of $\boldsymbol{b}$, so the proposed representation is close to the $(\boldsymbol{\kappa}, \boldsymbol{\phi})$ representation, at least for $\boldsymbol{\kappa}$, which is the same up to a factor $2x_{\overline{m}}/\sigma^2\sqrt{M}$. The factor $2/\sigma\sqrt{M}$ seems like a quite natural normalisation factor. An option is to factor out $|x_{\overline{m}}|$ from $\boldsymbol{\kappa}$ as a separate feature and use the same $\boldsymbol{\phi}$. This yields a representation $(|x_{\overline{m}}|, \boldsymbol{\kappa}/|x_{\overline{m}}|, \boldsymbol{\phi})$. The dimension of this regression label is $1 + 2M$.

Regression label of the form $(|x_{\overline{m}}|, \boldsymbol{\kappa}/|x_{\overline{m}}|, \angle\boldsymbol{e}_{\overline{m}})$

Combining the above ideas, another choice of regression label is $(|x_{\overline{m}}|, \boldsymbol{\kappa}/|x_{\overline{m}}|, \boldsymbol{\phi})$, but with $\boldsymbol{\phi}$ replaced by $\angle\boldsymbol{e}_{\overline{m}}$, in order to not have the periodicity stemming from $\boldsymbol{u}_{\overline{m}}$. This yields a representation $(|x_{\overline{m}}|, \boldsymbol{\kappa}/|x_{\overline{m}}|, \angle\boldsymbol{e}_{\overline{m}})$. This probably has the advantage that the training data for $\angle\boldsymbol{e}_{\overline{m}}$ is more uncorrelated. On the possible downside, the representation is further away to the $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$ for the dGvM. The dimension of this regression label is $1+2M$.

5.5.2 Data collection

To generate training data of the form $(\epsilon_{\overline{m}}, \boldsymbol{c})$, where $\boldsymbol{c}$ corresponds to the regression label taking any of the four forms described above, the hybrid-Gibbs sampler in Algorithm 1 was used. For a given signal $\boldsymbol{y}$ of size M, the hybrid-Gibbs sampler was executed, and for each generated grid mismatch $\epsilon_{\overline{m}}$, a data point of the form

$$(\epsilon_{\overline{m}}, x_{\overline{m}}, \overline{m}, \boldsymbol{\kappa}, \boldsymbol{\phi}, \boldsymbol{e}_{\overline{m}})$$

was stored, forming a collection of data

$$\mathcal{D} = \left\{ \left(\epsilon_{\overline{m}}^{(n)}, x_{\overline{m}}^{(n)}, \overline{m}^{(n)}, \boldsymbol{\kappa}^{(n)}, \boldsymbol{\phi}^{(n)}, \boldsymbol{e}_{\overline{m}}^{(n)} \right) \right\}_{n=1}.$$

The superscript $(\cdot)^{(n)}$ is not to be confused with the iteration index in the hybrid-Gibbs sampler. The one-hot vector $\overline{m}$ of size $\overline{M}$ determines the grid point a data point belongs to. Thus, with the use of this vector, the dataset $\mathcal{D}$ is split into $\overline{M}$ distinct datasets

$$\mathcal{D}_{\overline{m}} = \left\{ \left(\epsilon_{\overline{m}}^{(n)}, x_{\overline{m}}^{(n)}, \boldsymbol{\kappa}^{(n)}, \boldsymbol{\phi}^{(n)}, \boldsymbol{e}_{\overline{m}}^{(n)} \right) \right\}_{n=1},$$

which is the collection of all grid mismatches at grid point $\overline{m}$. Utilising these collections of grid point dependent datasets, the datasets for each of the four input architectures are obtained by simply removing or modifying the regression label elements of interest. The CCGAN corresponding to grid point $\overline{m}$ thus trains on the dataset given by $\mathcal{D}_{\overline{m}}$. If a dataset is to be generated from multiple signals, for each and every grid point, the datasets generated from each distinct signal are simply concatenated.

5.5.3 Evaluation

Three synthetically generated signals of size $M = 2$, grid size $\overline{M} = 2$, and power $\sigma^2 = 1$ were created: the first without any targets, the second with one target in the first grid point with amplitude 10 dB and grid mismatch 0.25, and the third one identical to the second one, but positioned in the second grid point. For each of these signals, data was collected in accordance with the method described in the preceding subsection. The hybrid-Gibbs sampler was executed with the same setup as described in Section 4.4, with the parameters given in Figure 4.2.

Apart from the input architecture, the CCGAN was the same for all four cases. Furthermore, for each case, the CCGAN has the same structure for all grid points. The CCGAN architecture for the first case, where the regression label is $(\boldsymbol{\kappa}, \boldsymbol{\phi})$, is summarised in Table C.9. The training setup and the hyperparameters were chosen to be identical between all the CCGANs, Table C.10. Moreover, for all of the different input architectures, the random seeds were set to the same value, in order to enable comparison between them.

As opposed to the preceding subtasks, the training dataset in this task is generated from a practical setting, namely from hybrid-Gibbs samples given a synthetically generated signal. The preceding problems generated data points where the labels were uniformly chosen from the regression label space. In this problem, the labels may be correlated, and thus not spread out evenly throughout the regression label space. Figure 5.6 shows scatter plots of the concentrations and means. As a consequence of this, it may happen that vicinities of labels that are perturbed do not have a nonempty intersection with any of the other label vicinities. This can happen if the rule of thumb for computing the regularisation parameters, σ and ν, is used. If this happens, the CCGAN training algorithm does not work. Overcoming this problem showed to be difficult, a discussion on this can be read in Chapter 6. The CCGAN regularisation parameters were therefore set to $\sigma = 0$ and $\nu = 0$, which is implied conditional GAN losses.

A qualitative evaluation was done by traversing through the training data points, where for each data point, the true $\mathrm{dGvM}_{[-0.5,0.5]}$ pdf and the corresponding histogram consisting of CCGAN samples was plotted. Figure 5.7 demonstrates the results for two different data points at indices zero and

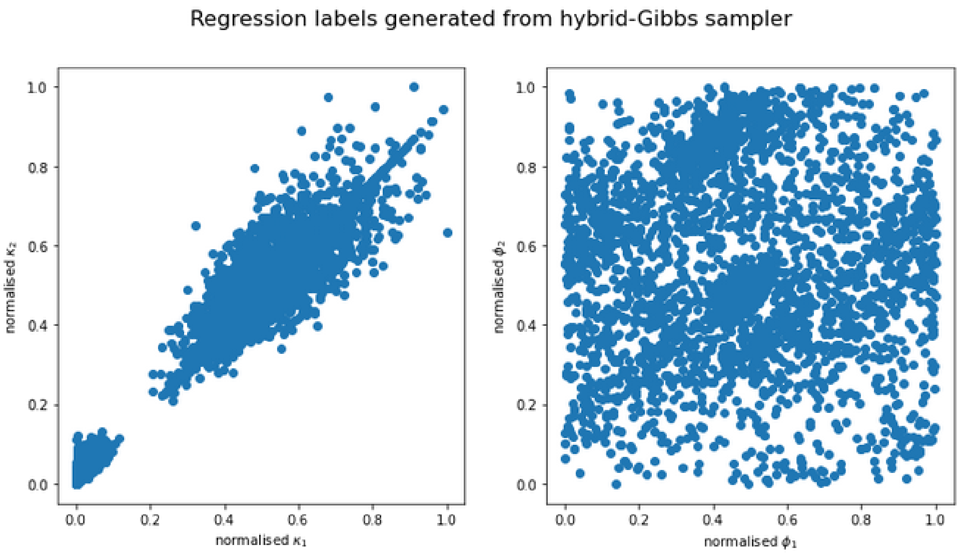
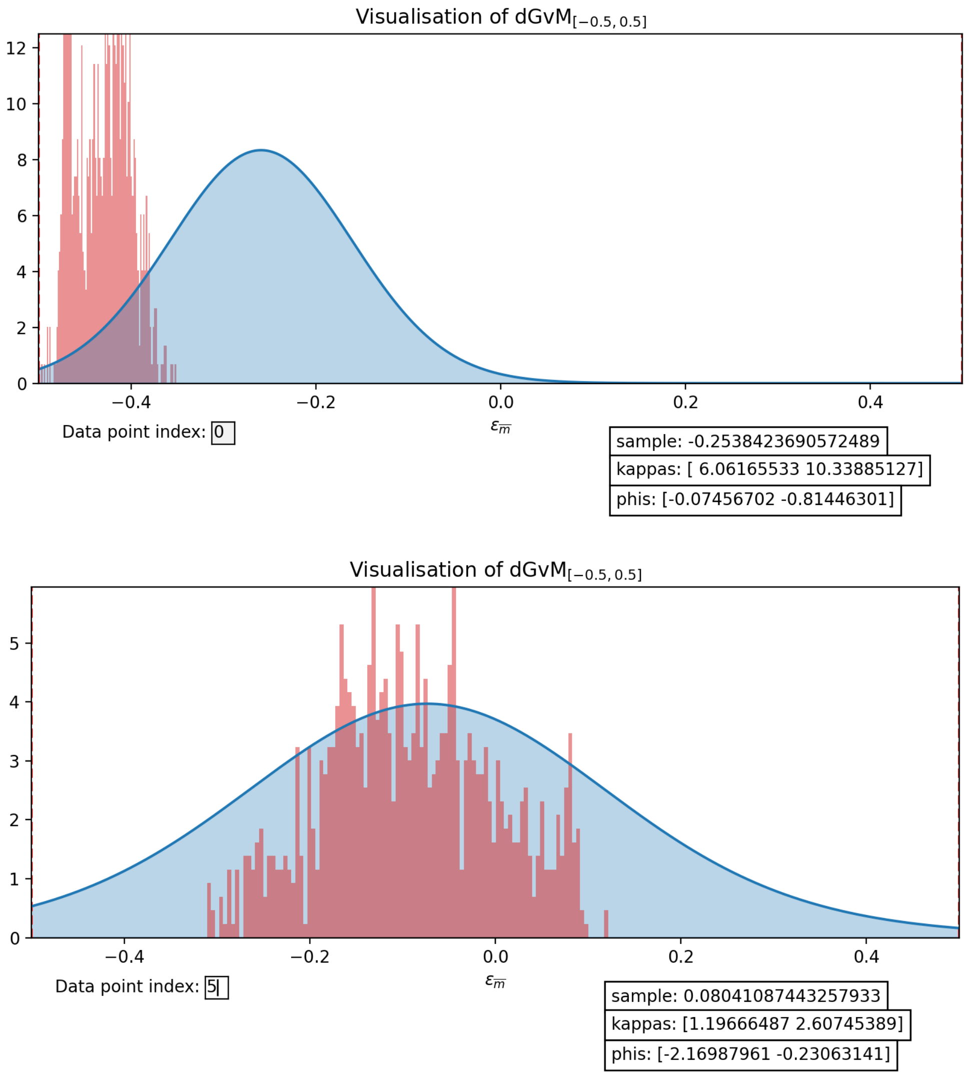


Figure 5.6: Scatter plot of the normalised four-dimensional regression labels of the form $(\boldsymbol{\kappa}, \boldsymbol{\phi})$ divided into two plots.

five of the training dataset for the first input case. Apart from the plots, the actual data sample, $\epsilon_{\overline{m}}$, and the dGvM parameters $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$ are also displayed. Because the qualitative result did not differ significantly between the four regression label cases, only the result of the first case is shown. It is clear that the CCGANs perform poorly on estimating dGvM$_{[-0.5,0.5]}$ distributions of order two conditioned on the dGvM parameters $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$. Ideally, the red CCGAN sample histograms should overlap with the true densities shown in blue. If the CCGANs would have been able to fit the training data well, they could have had practical usefulness for replacing the MH algorithm in the hybrid-Gibbs sampler for the sampling of grid mismatch elements. Ideally, the qualitative evaluation should have been in a similar fashion as the preceding subtasks, where the CCGAN histogram is compared to the true pdf of an arbitrary dGvM distribution, and not only the ones given by the labels in the training dataset.

A quantitative evaluation was done by calculating the generator and discriminator losses, Wasserstein-1 distance, and energy distance, for each epoch during training. Figures 5.8 and 5.9 summarise the quantitative evaluations of the four input cases. The quantitative results show that all CCGANs, independent of grid point or input architecture, have some sort of stabilisation in the generator and discriminator losses. Moreover, the Wasserstein-1 and energy distances decrease, indicating that the qualitative results improve during training.

Suggestions for further development of the CCGAN are given in the next chapter.



Data point index: 0

sample: -0.2538423690572489

kappas: [6.06165533 10.33885127]

phis: [-0.07456702 -0.81446301]

Data point index: 5

sample: 0.08041087443257933

kappas: [1.19666487 2.60745389]

phis: [-2.16987961 -0.23063141]

Figure 5.7: Qualitative evaluation-GUI for the dGvM$_{[-0.5,0.5]}$ simulating CCGANs. The GUI enabled traversal of the training data points. Here, two data points, at indices zero and five, from the dataset for the first regression label case, has been chosen. The blue plot is the true dGvM$_{[-0.5,0.5]}$ pdf given the parameters seen in each respective figure. The red histograms show the CCGAN samples for the given regression labels.

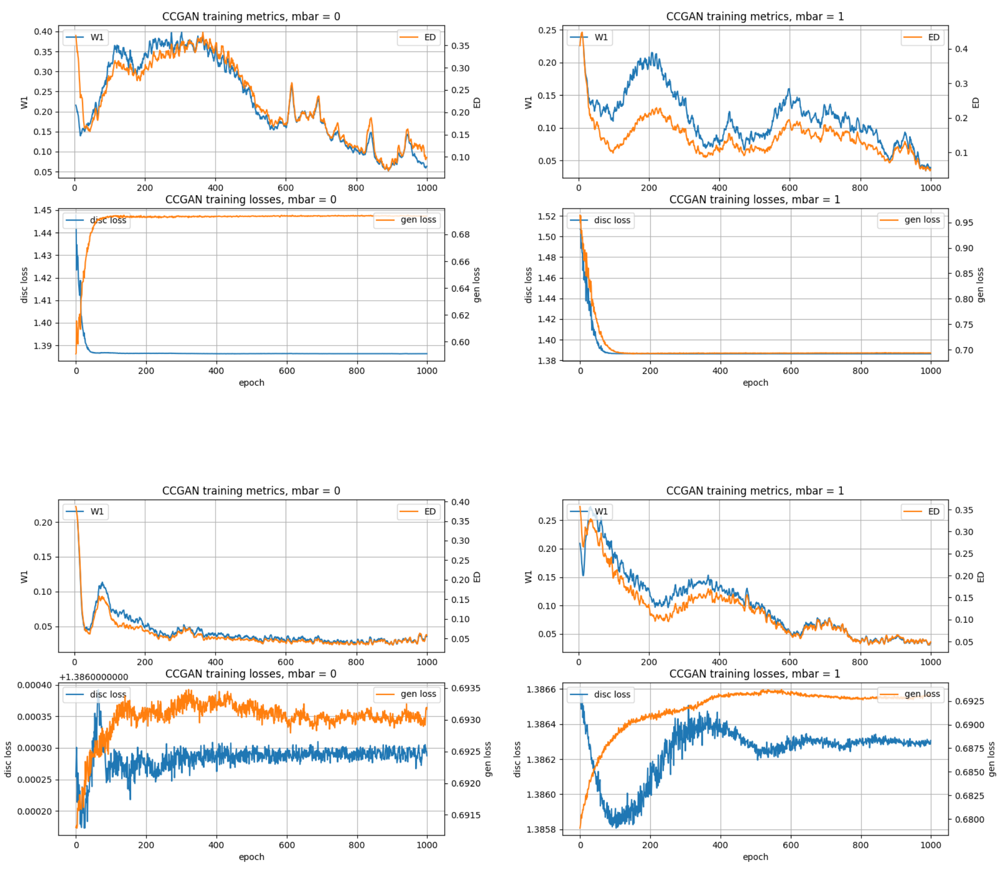


Figure 5.8: Quantitative evaluation of the CCGANs trained on data of the form $(\epsilon_{\overline{m}}, \boldsymbol{\kappa}, \boldsymbol{\phi})$ and $(\epsilon_{\overline{m}}, x_{\overline{m}}, \boldsymbol{e}_{\overline{m}})$, shown in the top and bottom row, respectively. The left column shows the results for the first grid point CCGAN, while the right column shows for the second grid point CCGAN.

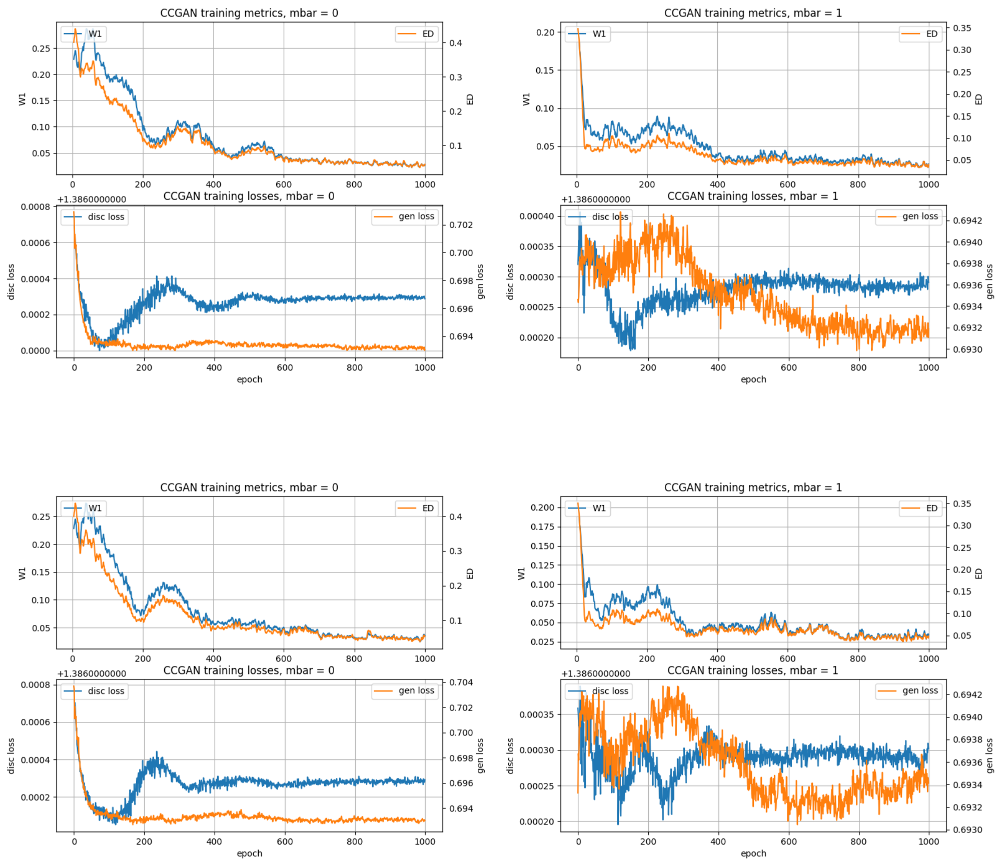


Figure 5.9: Quantitative evaluation of the CCGANs trained on data of the form $(\epsilon_{\overline{m}}, , |x_{\overline{m}}|, \boldsymbol{\kappa}/|x_{\overline{m}}|, \boldsymbol{\phi})$ and $(\epsilon_{\overline{m}}, |x_{\overline{m}}|, \boldsymbol{\kappa}/|x_{\overline{m}}|, \angle \boldsymbol{e}_{\overline{m}})$, shown in the top and bottom row, respectively. The left column shows the results for the first grid point CCGAN, while the right column shows for the second grid point CCGAN.

Chapter 6

Conclusion

In this thesis, a hierarchical Bayesian model for a sparse complex signal embedded in Gaussian noise was introduced. The choice of parameter priors was motivated, and the posteriors were derived. Given these, a hybrid-Gibbs sampler, robust to the grid mismatch problem, was successfully implemented to estimate the target scene of a synthetically generated signal. However, the complexity of this algorithm is high, due to a triple loop. The innermost loop consists of a Metropolis-Hastings sampler that samples a type of generalised (univariate and conditional) von Mises distribution. The main task of this thesis was to investigate the possibility of replacing this with a deep generative model, in the hope of a significant acceleration of the hybrid-Gibbs sampler. Specifically, a CCGAN was investigated for this purpose. A successful reproduction of a CCGAN from literature, used for sampling a two-dimensional Gaussian distribution with constant covariance, conditioned on the mean being on the unit circle, was done, which gave confidence in the implementation.

In order to facilitate troubleshooting and gain a better understanding of where problems may arise, a number of subtasks were examined before creating the CCGAN to simulate generalised von Mises distributions. The first subtask was the simulation of univariate Gaussians with fixed variance, conditioned on the mean, using a CCGAN. The qualitative result showed that the problem had been solved successfully. In the second subtask, the sampling of univariate Gaussians, conditioned on both mean and variance, was investigated. The performance was not satisfactory, since the CCGAN did not fit to the standard deviation. The third subtask was the simulation of von Mises distributions with a fixed concentration, conditioned on the mean. The CCGAN achieved good mean estimations, while it had a tendency to estimate the concentration significantly higher than its true value. The last two subtasks suggest that the CCGAN has a tendency to generate samples close to the conditioned mean. A reason for this may be that the CCGAN discriminator distinguishes between data points that are close to its conditioned mean, and data points that are far away from its conditioned mean. Thus, the CCGAN may have a tendency to classify samples that are close to their conditioned means as authentic, while the probability of samples further away from their conditioned means being authentic is smaller.

Finally, a CCGAN was created to simulate dGvM$_{[-0.5,0.5]}$ distributions conditioned on both a concentration and mean vector $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$, respectively. Four different representations of the dGvM parameters were explored as regression labels for the CCGAN. All four cases gave equally poor qualitative results. The quantitative results showed that the training was stable, indicated by the converging discriminator and generator losses and decreasing Wasserstein-1 and energy distances.

There is more work to be done in order to construct a well functioning CCGAN for dGvM$_{[-0.5,0.5]}$ distribution simulation. More generally, the study of how CCGANs can be used for simulation of distributions conditioned on multidimensional parameters is of interest. Specifically, how the CCGAN regularisation parameters are chosen should be further investigated. In this work, these parameters were set to zero for each regression label element, but in order to make use of the introduced HVDL loss, these should be set to some positive carefully chosen values. To achieve better results with the CCGAN, suggestions for future work are presented in the next section.

6.1 Future Work

During the course of the thesis, several interesting problems have presented themselves, in relation to generative modelling of univariate distributions. The main goal of the thesis was to construct a CCGAN to simulate dGvM$_{[-0.5,0.5]}$ distributions. This turned out to be a difficult problem, as has also been noted in the literature, investigating the potential of GANs simulating one-dimensional distributions [42]. Up to date, there seems to be no explanation for why this is the case.

6.1.1 Choice of CCGAN regularisation parameters

One obvious potential for improvement of this work, in relation to CCGAN, would be how the choice of the regularisation parameters σ and ν is made for multidimensional regression labels. The parameter σ signifies the bandwidth of the Gaussian kernels used for marginal distribution estimations of the regression labels, seen in equations (3.7) and (3.8). It can be seen as determining how much a label in the dataset should be perturbed, in the CCGAN algorithm, without representing a too different distribution. The value of parameter ν determines how large the vicinities of the labels are.

Labels belonging in each other vicinities are treated as describing similar distributions.

For the one-dimensional label case, a rule of thumb, given in equations (3.13) to (3.14), was used to calculate the two parameters. There is yet no corresponding rule of thumb for the multidimensional case. One approach, as in Section 5.3, is to use the rule of thumb for each dimension of the regression labels. When using this approach for the final CCGAN for dGvM$_{[-0.5,0.5]}$ simulation, this did not work. One reason for this may be that the training dataset was generated from a practical example not suitable for the rule of thumb, as opposed to the subtasks, where the labels were uniformly spaced in the regression label space. If the rule of thumb was used, the CCGAN training did not function, which was due to the perturbed labels not having any close neighbouring labels. This is a consequence of the labels being highly correlated between the $\boldsymbol{\kappa}$:s (although not between the $\boldsymbol{\phi}$:s). See Figure 5.6. Thus, perturbing a label with Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ (distinct for each label dimension), resulted in labels that were too far away from the training labels. Reducing the value of σ and increasing ν might alleviate this problem, but with the cost of the distributions represented by the different labels being smeared together, which would result in worse accuracy of the CCGAN. The difficulty of filling out the regression label space increases significantly for each added dimension.

A second approach to the choice of CCGAN regularisation parameters would be to perturb the labels according to some noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{H})$, where $\boldsymbol{H}$ would be a covariance matrix taking into account the correlations between the regression label dimensions. In the first approach, the covariance matrix is a diagonal matrix, where each entry is a variance computed according to the rule of thumb for each label element. Introducing correlation between the label elements may alleviate the problem of perturbing training labels too far away from the ones existing in the dataset.

6.1.2 Restrict generator output

The dGvM$_{[-0.5,0.5]}$ distribution is truncated on the interval $[-0.5, 0.5]$, thus the CCGAN generator should only output samples in this interval. A further improvement of the CCGAN could be the restriction of the generator outputs to the interval of interest. This is something that should be handled implicitly during training, since an optimal discriminator would classify samples outside this interval as fake, which is something the generator would realise. Using a sigmoid function to restrict the generator outputs could be a possibility. Another suggestion is to explicitly make the discriminator classify samples falling outside the interval as fake.

6.1.3 Intermediate tasks

In Chapter 5, subtasks were investigated before constructing a CCGAN for $\mathrm{dGvM}_{[-0.5,0.5]}$ simulation, where the preceding subtask was to simulate von Mises distributions with fixed concentration, conditioned on the mean. The difference in complexity between the two problems is big, due to the $\mathrm{dGvM}_{[-0.5,0.5]}$ distribution having potentially very high-dimensional parameters $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$. Intermediate tasks may be introduced in order to further facilitate troubleshooting. One suggestion for such a task could be to simulate two distinct $\mathrm{dGvM}_{[-0.5,0.5]}$ distributions of low order, say $M = 2$, having similar parameters, and investigating how well the CCGAN generalises to labels lying in between. One could also try fixating the concentration vector $\boldsymbol{\kappa}$, while only conditioning on the mean vector $\boldsymbol{\phi}$. Moreover, a training dataset where the labels are uniformly distributed in the regression label space, instead of having been generated from a practical experiment, could be examined. Using the first approach for choosing regularisation parameters would then work.

6.1.4 Replacing the MH algorithm with CCGAN

The resulting CCGAN for generating grid mismatch elements as if they were sampled from their corresponding $\mathrm{dGvM}_{[-0.5,0.5]}(\boldsymbol{\kappa}, \boldsymbol{\phi})$ distributions gave poor qualitative results. Naturally, they would therefore not be able to be used in the hybrid-Gibbs sampler for target scene estimation. However, the CCGAN is assumed to have given sufficiently good qualitative results for the rest of this subsection. Before the CCGAN framework can replace the MH algorithm in the hybrid-Gibbs sampler for grid mismatch sampling, there is an extra step that has to be introduced in the algorithm. Because the CCGAN generates samples, here denoted $\epsilon^*_{\overline{m}}$, that are not distributed according to the actual $\mathrm{dGvM}_{[-0.5,0.5]}(\boldsymbol{\kappa}, \boldsymbol{\phi})$ distribution, an acceptance/rejection step has to be taken. This has to be done if the grid mismatch samples are to be sampled from the correct (stationary) distribution. After the CCGAN sample has been sampled, an acceptance probability is computed as follows:

$$\alpha\Big(\epsilon^*_{\overline{m}}, \epsilon^{(n-1)}_{\overline{m}}\Big) = \min\left(1, \frac{\mathrm{dGvM}_{[-0.5,\,0.5]}\big(\epsilon^*_{\overline{m}}; \boldsymbol{\kappa}, \boldsymbol{\phi}\big)\, q\big(\epsilon^{(n-1)}_{\overline{m}} | \epsilon^*_{\overline{m}}\big)}{\mathrm{dGvM}_{[-0.5,\,0.5]}\big(\epsilon^{(n-1)}_{\overline{m}}; \boldsymbol{\kappa}, \boldsymbol{\phi}\big)\, q\big(\epsilon^*_{\overline{m}} | \epsilon^{(n-1)}_{\overline{m}}\big)}\right),$$

where q denotes the density of the distribution the CCGAN represent. The proposed sample is then accepted ($\epsilon^{(n)}_{\overline{m}} = \epsilon^*_{\overline{m}}$) with probability $\alpha\Big(\epsilon^*_{\overline{m}}, \epsilon^{(n-1)}_{\overline{m}}\Big)$ and otherwise rejected ($\epsilon^{(n)}_{\overline{m}} = \epsilon^{(n-1)}_{\overline{m}}$). A problem that arises here is that the distribution the CCGAN represents is not known explicitly. One way to circumvent this may be by using probability density estimation given a finite number of samples from the CCGAN. However, other methods for this may already exist.

6.1.5 Other generative models

There are other generative models than CCGAN that may be interesting to investigate for the simulation of $\mathrm{dGvM}_{[-0.5,0.5]}$. Specifically, the Conditional Variational Auto-Encoder (CVAE) could be of interest. Like the CCGAN, the CVAE is a conditional generative model that seeks to generate novel samples as if they were drawn from its training dataset. However, the underlying structure and training of these models are fundamentally different. The details for CVAEs are given in [43].

Bibliography

[1] W.T. Cochran, J.W. Cooley, D.L. Favin, H.D. Helms, R.A. Kaenel, W.W. Lang, G.C. Maling, D.E. Nelson, C.M. Rader, and P.D. Welch. What is the fast Fourier transform? *Proceedings of the IEEE*, pages 1664–1674, 1967.

[2] Maroua Taghouti. Chapter 10 - Compressed sensing. In Frank H.P. Fitzek, Fabrizio Granelli, and Patrick Seeling, editors, *Computing in Communication Networks*, pages 197–215. Academic Press, 2020.

[3] Marie Lasserre, Stéphanie Bidon, Olivier Besson, and François Le Chevalier. Bayesian sparse Fourier representation of off-grid targets with application to experimental radar data. *Signal Processing*, pages 261–273, 2015.

[4] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, USA, 3rd edition, 2009.

[5] F.J. Samaniego. *A Comparison of the Bayesian and Frequentist Approaches to Estimation.* Springer Series in Statistics. Springer New York, 2010.

[6] F. Liese and K.J. Miescke. *Statistical Decision Theory: Estimation, Testing, and Selection.* Springer Series in Statistics. Springer New York, 2008.

[7] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin. *Bayesian Data Analysis, Third Edition.* Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013.

[8] C.M. Bishop. *Pattern Recognition and Machine Learning.* Information Science and Statistics. Springer New York, 2016.

[9] C. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer Texts in Statistics. Springer New York, 2013.

[10] A. Lapidoth. *A Foundation in Digital Communication.* Cambridge University Press, 2017.

[11] H.H. Andersen, M. Hojbjerre, D. Sorensen, and P.S. Eriksen. *Linear and Graphical Models: for the Multivariate Complex Normal Distribution*. Lecture Notes in Statistics. Springer New York, 2012.

[12] N. Dobigeon, A.O. Hero, and J.-Y. Tourneret. Hierarchical Bayesian Sparse Image Reconstruction With Application to MRFM. *IEEE Transactions on Image Processing*, pages 2059–2070, sep 2009.

[13] D. J. Best and N. I. Fisher. Efficient Simulation of the von Mises Distribution. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, pages 152–157, 1979.

[14] Riccardo Gatto and Sreenivasa Rao Jammalamadaka. The generalized von Mises distribution. *Statistical Methodology*, pages 341–353, 2007.

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. 2014.

[16] Amit H. Bermano, Rinon Gal, Yuval Alaluf, Ron Mokady, Yotam Nitzan, Omer Tov, Or Patashnik, and Daniel Cohen-Or. State-of-the-Art in the Architecture, Methods and Applications of StyleGAN. 2022.

[17] This person does not exist. `https://this-person-does-not-exist.com/en`. Accessed: 2022-07-18.

[18] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, 2017.

[19] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016.

[20] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint*, 2016.

[21] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint*, 2015.

[22] Dmitry I. Belov and Ronald D. Armstrong. Distributions of the Kullback–Leibler divergence with applications. *British Journal of Mathematical and Statistical Psychology*, pages 291–309, 2011.

[23] Frank Nielsen. On the Jensen–Shannon Symmetrization of Distances Relying on Abstract Means. *Entropy*, 2019.

[24] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint*, 2017.

[25] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved Training of Wasserstein GANs. *arXiv preprint*, 2017.

[26] Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. 2017.

[27] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, pages 141–142, 2012.

[28] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv preprint*, 2014.

[29] Hanock Kwak and Byoung-Tak Zhang. Ways of Conditioning Generative Adversarial Networks. *arXiv preprint*, 2016.

[30] Vladimir Vapnik. Principles of Risk Minimization for Learning Theory. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, page 831–838, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[31] Xin Ding, Yongwei Wang, Zuheng Xu, William J. Welch, and Z. Jane Wang. Continuous Conditional Generative Adversarial Networks: Novel Empirical Losses and Label Input Mechanisms. *arXiv preprint*, 2021.

[32] Takeru Miyato and Masanori Koyama. cGANs with Projection Discriminator. *arXiv preprint*, 2018.

[33] Chao Zhang, Min-Hsiu Hsieh, and Dacheng Tao. Generalization Bounds for Vicinal Risk Minimization Principle. *arXiv preprint*, 2018.

[34] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1986.

[35] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal Risk Minimization. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2000.

[36] Ali Borji. Pros and Cons of GAN Evaluation Measures: New Developments. *arXiv preprint*, 2021.

[37] Aaditya Ramdas, Nicolas Garcia, and Marco Cuturi. On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests. *arXiv preprint*, September 2015.

[38] C. Villani. *Optimal Transport: Old and New.* Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008.

[39] Gábor J. Székely and Maria L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, pages 1249–1272, 2013.

[40] Marie Lasserre, Stéphanie Bidon, Olivier Besson, and François Le Chevalier. Bayesian sparse Fourier representation of off-grid targets with application to experimental radar data. *Signal Processing*, pages 261–273, 2015.

[41] Salah H. Abid and Saja A. Al-Hassany. On the Inverted Gamma Distribution. *International Journal of Systems Science and Applied Mathematics*, pages 16–22, 2016.

[42] Manzil Zaheer and Barnabás Póczos. Connoisseur : Can GANs Learn Simple 1D Parametric Distributions? 2018.

[43] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning Structured Output Representation using Deep Conditional Generative Models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems.* Curran Associates, Inc., 2015.

Appendix A

Algorithms

Algorithm 2: Simulating univariate complex Gaussian with expectation zero.

```
**Result:** Sample $z \sim \mathcal{CN}\big(0, \sigma^2\big)$
{Generate real and imaginary parts of $z$}
$\mathrm{Re}(z) \sim \frac{1}{\sqrt{2}}\mathcal{N}\big(0, \sigma^2\big)$
$\mathrm{Im}(z) \sim \frac{1}{\sqrt{2}}\mathcal{N}\big(0, \sigma^2\big)$
**Return:** $z = \mathrm{Re}(z) + j\,\mathrm{Im}(z)$
```

Algorithm 3: Simulating Bernoulli complex Gaussian.

```
**Result:** Sample $z \sim Ber\mathcal{CN}\big(w, \mu, \sigma^2\big)$
{Generate Bernoulli sample}
$x \sim Ber(w)$
{Sample $z$ according to outcome of $x$}
$$z = \begin{cases} 0, \text{ if } x = 0 \\ \sim \mathcal{CN}\big(\mu, \sigma^2\big), \text{ if } x = 1. \end{cases}$$
**Return:** $z$
```

Algorithm 4: MH algorithm for sampling $\epsilon_{\overline{m}}$.

```
Require: Specified number MH-iterations $N_r$, dGvM-parameters,
  proposal variance $\sigma_p^2$, and vectors $\boldsymbol{x}^{(n)}$, $\boldsymbol{\epsilon}_{-\overline{m}}^{(n)}$ as in (4.15) and
  (4.14), respectively. Moreover, the previous grid mismatch sample,
  $\epsilon_{\overline{m}}^{(n-1)}$, is needed. To simplify notation, the superscript is here
  assumed to denote the MH-iterations and not the Gibbs-iterations.
  Also, the subscript is neglected as it is clear that the grid point in
  question is $\overline{m}$. The previous grid mismatch sample is thus written
  as $\epsilon^{(0)}$ instead of $\epsilon_{\overline{m}}^{(n-1)}$.
Result: A sample $\epsilon_{\overline{m}}^{(n)}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}^{(n)}, \boldsymbol{x}^{(n)}, \sigma^{2^{(n)}} \sim \mathrm{dGvM}_{[-0.5,\,0.5]}(\boldsymbol{\kappa}, \boldsymbol{\phi})$
for t=1 to $N_r$ do
    {Proposal}
    while $\epsilon^* < -0.5$ or $\epsilon^* > 0.5$ do
        $\epsilon^* \sim \mathcal{N}\Big(\epsilon^* | \epsilon^{(t-1)}; \epsilon^{(t-1)}, \sigma_p^2\Big)$
    end
    {Acceptance}
    $\alpha(\epsilon^*, \epsilon^{(t-1)}) = \min\Bigg[1, \exp\Bigg(\sum_{m=0}^{M-1} \kappa_m \Bigg(\cos\Big(2\pi \frac{\epsilon^*}{M} m - \phi_m\Big) - \cos\Big(2\pi \frac{\epsilon^{(t-1)}}{M} m - \phi_m\Big)\Bigg)\Bigg)\Bigg]$
    {Accept or reject}
    $u \sim \mathcal{U}_{[0,\,1]}$
    if $u < \alpha\Big(\epsilon^*, \epsilon^{(t-1)}\Big)$ then
        {Accept}
        $\epsilon^{(t)} = \epsilon^*$
    else
        {Reject}
        $\epsilon^{(t)} = \epsilon^{(t-1)}$
    end
end
Return: $\epsilon_{\overline{m}}^{(n)} = \epsilon^{(N_r)}$
```

Algorithm 5: Training of the generative adversarial networks using mini-batch gradient descent.

```
Require: Generator network $G$, discriminator network $D$, specified
  gradient descent algorithm (for example ADAM), number of
  training iterations $N$, number of discriminator iterations per
  generator iteration $K$, mini-batch size $m$ and a training data set
  containing the authentic samples.
Result: Generator that can sample data points indistinguishable
        (measured by the discriminator) from the authentic ones.

{Iterations}
for $n$=1 to $N$ do
    {Discriminator iterations}
    for $k$=1 to $K$ do
        · Sample mini-batch of latent vectors $\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_m\}$ from the
          latent space prior $p_{\boldsymbol{z}}$.
        · Sample mini-batch of authentic data points $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}$
          from the training data.
        · Update the discriminator weights by descending the
          gradient:
              $-\nabla_{\boldsymbol{W}_D} \frac{1}{m} \sum_{i=1}^{m} \left[ \log(D(\boldsymbol{x}_i)) + \log(1 - D(G(\boldsymbol{z}_i))) \right]$
    end
    {Generator iteration}
    · Sample mini-batch of latent vectors $\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_m\}$ from the
      latent space prior $p_{\boldsymbol{z}}$.
    · Update the generator weights by descending the gradient:
              $-\nabla_{\boldsymbol{W}_G} \frac{1}{m} \sum_{i=1}^{m} \left[ \log(D(G(\boldsymbol{z}_i))) \right]$
end
Return: generator network $G$
```

Algorithm 6: Training of the continuous conditional generative adversarial networks using mini-batch gradient descent and HVDL.

```
Require: Generator network $G$, discriminator network $D$, training
  data set containing data points-regression label pairs
  $\Omega^a = \{(\boldsymbol{x}_1^a, c_1^a), \ldots, (\boldsymbol{x}_{N^a}^a, c_{N^a}^a)\}$, $N_{uc}^a$ ordered distinct labels
  $\Upsilon = \{y_{[1]}^a, \ldots, y_{[N_{uc}^a]}^a\}$ in the data set, regularisation parameters $\sigma$
  and $\nu$, specified gradient descent algorithm (for example ADAM),
  number of training iterations $K$ and mini-batch size $m$.
Result: Trained generator $G$.
{Iterations}
for $k$=1 to $K$ do
    {Train discriminator}
    · Sample mini-batch $C^D = \{c_1, \ldots, c_m\}$ with replacement from
      $\Upsilon$.
    · Create a set of perturbed target labels
      $C^{D,\epsilon} = \{c_i + \epsilon \,|\, c_i \in C^D, \epsilon \in \mathcal{N}(0, \sigma^2), i = 1, \ldots, m\}$ on which
      the discriminator will condition on.
    · Initialise the sets $\Omega_D^a = \emptyset$ and $\Omega_D^f = \emptyset$ that will contain
      authentic samples and generated fake samples with their
      corresponding perturbed labels, respectively.
    for $i$=1 to $m$ do
        · Randomly pick a pair $(\boldsymbol{x}, c) \in \Omega^a$ satisfying $|c - c_i - \epsilon| \leq \nu$,
          where $c_i + \epsilon \in C^{D,\epsilon}$ and let $\Omega_D^a = \Omega_D^a \cup (\boldsymbol{x}, c_i + \epsilon)$.
        · Randomly pick a label $c'$ from $U(c_i + \epsilon - \nu, c_i + \epsilon + \nu)$ and
          generate a fake sample $\boldsymbol{x}'$ by evaluating $G(\boldsymbol{z}, c')$, where
          $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$ and let $\Omega_D^f = \Omega_D^f \cup (\boldsymbol{x}', c_i + \epsilon)$.
    end
    · Update $D$ with the created sets $\Omega_D^a$ and $\Omega_D^f$ using the specified
      gradient descent optimiser based on equation (3.11).
    {Train generator}
    · Sample mini-batch $C^G = \{c_1, \ldots, c_m\}$ with replacement from
      $\Upsilon$.
    · Create another set of perturbed target labels
      $C^{G,\epsilon} = \{c_i + \epsilon \,|\, c_i \in C^G, \epsilon \in \mathcal{N}(0, \sigma^2), i = 1, \ldots, m\}$ on which
      the generator will condition on.
    · Generate $m$ fake samples conditional on $C^{G,\epsilon}$ and put the
      resulting data point-label pairs in $\Omega_G^f$.
    · Update $G$ with the samples in $\Omega_G^f$ using the specified gradient
      descent optimiser based on equation (3.12).
end
Return: Generator network $G$.
```

Appendix B

Derivation of Model Parameter Posteriors

Starting from the joint posterior pdf $\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2 | \boldsymbol{y}$, the posteriors of the signal model parameters can be derived given an observed signal $\boldsymbol{y}$. The joint posterior pdf of $\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2 | \boldsymbol{y}$ can be computed by rewriting it using Bayes' theorem, conditioning and using conditional independence between the parameters. The dependencies between the parameters are given in the hierarchical Bayesian model seen in Figure 4.1. The joint posterior pdf can thus be readily computed as

$$\begin{aligned}
&f(\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2 | \boldsymbol{y}) \\
&\quad \propto f(\boldsymbol{y} | \sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2) f(\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2) \\
&\quad = f(\boldsymbol{y} | \boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2 | \sigma^2) f(\sigma^2) \\
&\quad = f(\boldsymbol{y} | \boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2) f(\sigma^2) \\
&\quad = f(\boldsymbol{y} | \boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\boldsymbol{\epsilon}, \boldsymbol{x}, w | \sigma_x^2) f(\sigma_x^2) f(\sigma^2) \\
&\quad = f(\boldsymbol{y} | \boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\boldsymbol{\epsilon}, \boldsymbol{x} | w, \sigma_x^2) f(w | \sigma_x^2) f(\sigma_x^2) f(\sigma^2) \\
&\quad = f(\boldsymbol{y} | \boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\boldsymbol{\epsilon} | \boldsymbol{x}, w, \sigma_x^2) f(\boldsymbol{x} | w, \sigma_x^2) f(w) f(\sigma_x^2) f(\sigma^2) \\
&\quad = f(\boldsymbol{y} | \boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\boldsymbol{\epsilon} | \boldsymbol{x}) f(\boldsymbol{x} | w, \sigma_x^2) f(w) f(\sigma_x^2) f(\sigma^2).
\end{aligned}$$

The target amplitude and frequency mismatch vectors, $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$, respectively, are sampled elementwise from their corresponding conditional posterior distributions of $x_{\overline{m}}$ and $\epsilon_{\overline{m}}$, using the definitions $x_{\overline{m}} \coloneqq [\boldsymbol{x}]_{\overline{m}}$ and $\epsilon_{\overline{m}} \coloneqq [\boldsymbol{\epsilon}]_{\overline{m}}$. These posteriors are derived from their conditional joint posterior pdf

$$f(\epsilon_{\overline{m}}, x_{\overline{m}} | \boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2), \tag{B.2}$$

which in its turn is derived from the joint posterior given by equation (B.1). Here we have defined

$$\boldsymbol{x}_{-\overline{m}} \coloneqq (x_0, \ldots, x_{\overline{m}-1}, x_{\overline{m}+1}, \ldots, x_{\overline{M}-1})$$

and

$$\boldsymbol{\epsilon}_{-\overline{m}} \coloneqq (\epsilon_0, \ldots, \epsilon_{\overline{m}-1}, \epsilon_{\overline{m}+1}, \ldots, \epsilon_{\overline{M}-1}).$$

The conditional joint posterior pdf given by equation (B.2) can be derived by conditioning on all the other parameters except $\epsilon_{\overline{m}}$ and $x_{\overline{m}}$ and using proportionality to disregard all the factors that are independent of these two. Conditioning leads to

$$\begin{aligned} f(\sigma^2, \boldsymbol{\epsilon}, \boldsymbol{x}, w, \sigma_x^2|\boldsymbol{y}) = f(\epsilon_{\overline{m}}, x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) \qquad &\text{(B.3a)} \\ \times f(\boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2|\boldsymbol{y}). \qquad &\text{(B.3b)} \end{aligned}$$

Assuming that the factor given by (B.3b) is nonzero and rewriting the left-hand side of equation (B.1) using (B.3), one obtains the conditional joint posterior pdf

$$\begin{aligned} &f(\epsilon_{\overline{m}}, x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) &\text{(B.4a)} \\ &\propto \frac{f(\boldsymbol{y}|\boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\boldsymbol{\epsilon}|\boldsymbol{x}) f(\boldsymbol{x}|w, \sigma_x^2) f(w) f(\sigma_x^2) f(\sigma^2)}{f(\boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2|\boldsymbol{y})} &\text{(B.4b)} \\ &\propto f(\boldsymbol{y}|\boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\epsilon_{\overline{m}}|x_{\overline{m}}) f(x_{\overline{m}}|w, \sigma_x^2). &\text{(B.4c)} \end{aligned}$$

Remember that the amplitudes $x_{\overline{m}}$ are assumed to be i.i.d. and that the same applies to the frequency mismatches $\epsilon_{\overline{m}}$.

To make derivations of the conditional posteriors of the amplitudes $x_{\overline{m}}$ and frequency mismatches $\epsilon_{\overline{m}}$ simpler later on, the following expression is introduced:

$$\boldsymbol{e}_{\overline{m}} \coloneqq \boldsymbol{y} - \sum_{i \neq \overline{m}} \boldsymbol{f}_i(\epsilon_i) x_i.$$

This expression lends a way of expressing the posteriors as a function of each separate component $x_{\overline{m}}$ and $\epsilon_{\overline{m}}$. One can check that

$$\boldsymbol{y} - \boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x} = \left(\boldsymbol{y} - \sum_{i \neq \overline{m}} \boldsymbol{f}_i(\epsilon_i) x_i\right) - \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) x_{\overline{m}} = \boldsymbol{e}_{\overline{m}} - \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) x_{\overline{m}}. \quad \text{(B.5)}$$

The likelihood of the observed signal in equation (4.6d) can be rewritten using (B.5) as

$$\begin{aligned}
&f(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x},\sigma^2)\\
&\quad\propto \exp\left(-\frac{\|\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\|_2^2}{\sigma^2}\right)\\
&\quad= \exp\left(-\frac{\big(\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\big)^H\big(\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\big)}{\sigma^2}\right)\\
&\quad= \exp\left(-\frac{\big(\boldsymbol{e}_{\overline{m}}-\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})x_{\overline{m}}\big)^H\big(\boldsymbol{e}_{\overline{m}}-\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})x_{\overline{m}}\big)}{\sigma^2}\right)\\
&\quad= \exp\left(-\frac{\boldsymbol{e}_{\overline{m}}^H\boldsymbol{e}_{\overline{m}}-\boldsymbol{e}_{\overline{m}}^H\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})x_{\overline{m}}-x_{\overline{m}}^*\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H\boldsymbol{e}_{\overline{m}}+x_{\overline{m}}^*\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})x_{\overline{m}}}{\sigma^2}\right)\\
&\quad= \exp\left(-\frac{|x_{\overline{m}}|^2+\boldsymbol{e}_{\overline{m}}^H\boldsymbol{e}_{\overline{m}}-x_{\overline{m}}\boldsymbol{e}_{\overline{m}}^H\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})-x_{\overline{m}}^*\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H\boldsymbol{e}_{\overline{m}}}{\sigma^2}\right),
\end{aligned}$$

where the computation in the last line made use of

$$\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) \overset{(4.1)}{=} \sum_{m=0}^{M-1}\frac{1}{\sqrt{M}}\exp(-\psi_m)\cdot\frac{1}{\sqrt{M}}\exp(\psi_m)=\sum_{m=0}^{M-1}\frac{1}{M}=1,$$

where

$$\psi_m := j2\pi m\frac{\overline{m}+\epsilon_{\overline{m}}}{M}.$$

The conditional joint posterior given by (B.4c) can thus be expressed as

$$\begin{aligned}
&f(\epsilon_{\overline{m}},x_{\overline{m}}|\boldsymbol{y},\boldsymbol{\epsilon}_{-\overline{m}},\boldsymbol{x}_{-\overline{m}},\sigma^2,w,\sigma_x^2) && \text{(B.6a)}\\
&\quad\propto f(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x},\sigma^2)f(\epsilon_{\overline{m}}|x_{\overline{m}})f(x_{\overline{m}}|w,\sigma_x^2) && \text{(B.6b)}\\
&\quad= \exp\left(-\frac{|x_{\overline{m}}|^2+\boldsymbol{e}_{\overline{m}}^H\boldsymbol{e}_{\overline{m}}-x_{\overline{m}}\boldsymbol{e}_{\overline{m}}^H\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})-x_{\overline{m}}^*\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H\boldsymbol{e}_{\overline{m}}}{\sigma^2}\right) && \text{(B.6c)}\\
&\qquad\times f(\epsilon_{\overline{m}}|x_{\overline{m}})f(x_{\overline{m}}|w,\sigma_x^2) && \text{(B.6d)}\\
&\quad\propto \exp\left(-\frac{|x_{\overline{m}}|^2-x_{\overline{m}}\boldsymbol{e}_{\overline{m}}^H\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})-x_{\overline{m}}^*\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H\boldsymbol{e}_{\overline{m}}}{\sigma^2}\right) && \text{(B.6e)}\\
&\qquad\times f(\epsilon_{\overline{m}}|x_{\overline{m}})f(x_{\overline{m}}|w,\sigma_x^2) && \text{(B.6f)}\\
&\quad= L(\epsilon_{\overline{m}},x_{\overline{m}},\sigma^2)f(\epsilon_{\overline{m}}|x_{\overline{m}})f(x_{\overline{m}}|w,\sigma_x^2), && \text{(B.6g)}
\end{aligned}$$

where

$$L(\epsilon_{\overline{m}},x_{\overline{m}},\sigma^2)=\exp\left(-\frac{|x_{\overline{m}}|^2-x_{\overline{m}}\boldsymbol{e}_{\overline{m}}^H\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})-x_{\overline{m}}^*\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H\boldsymbol{e}_{\overline{m}}}{\sigma^2}\right). \quad \text{(B.7)}$$

To simplify notation further, the shorthand L will be used for denoting $L(\epsilon_{\overline{m}}, x_{\overline{m}}, \sigma^2)$. This conditional joint posterior will be used for deriving the conditional posteriors for $\epsilon_{\overline{m}}$ and $x_{\overline{m}}$ in order to facilitate the usage of a Gibbs sampler.

B.1 Target amplitude vector elements

The elements of the target amplitude vector $\boldsymbol{x}$ are to be sampled elementwise in order to enable the sampling of $\boldsymbol{x}$ using a Gibbs sampler. Each element $x_{\overline{m}}$ is sampled from the conditional posterior of $x_{\overline{m}}$, thus its posterior must be derived. Firstly, a general expression of the conditional posterior is derived from the joint conditional posterior of $\epsilon_{\overline{m}}$ and $x_{\overline{m}}$ seen in equation (B.6g) by conditioning on $\epsilon_{\overline{m}}$ and then using proportionality in order to only keep factors with $x_{\overline{m}}$-dependence. Secondly, the pdf of the conditional posterior is computed using the pdfs of the corresponding distributions that make up the conditional posterior, namely the likelihood and priors. Lastly, from this resulting pdf of the conditional posterior, a known distribution will be recognised, and the derivation will be complete.

Conditioning the conditional joint posterior in equation (B.6g) on $\epsilon_{\overline{m}}$, it can be expressed as

$$\begin{aligned} f(\epsilon_{\overline{m}}, x_{\overline{m}} | \boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) &= f(x_{\overline{m}} | \boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) && \text{(B.8a)} \\ &\quad \times f(\epsilon_{\overline{m}} | \boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2). && \text{(B.8b)} \end{aligned}$$

Assuming that the factor given by (B.8b) is nonzero and using proportionality, one obtains a general expression for the conditional posterior of $x_{\overline{m}}$:

$$\begin{aligned} f(x_{\overline{m}} | \boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) &= \frac{f(\epsilon_{\overline{m}}, x_{\overline{m}} | \boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2)}{f(\epsilon_{\overline{m}} | \boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2)} && \text{(B.9a)} \\ &\propto f(\boldsymbol{y} | \boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\epsilon_{\overline{m}} | x_{\overline{m}}) f(x_{\overline{m}} | w, \sigma_x^2) && \text{(B.9b)} \\ &\propto L f(\epsilon_{\overline{m}} | x_{\overline{m}}) f(x_{\overline{m}} | w, \sigma_x^2). && \text{(B.9c)} \end{aligned}$$

Examining the conditional posterior, one can observe that the mixture prior $f(x_{\overline{m}} | w, \sigma_x^2)$ given in equation (4.7) is present, which complicates the derivation of the posterior of the target element $x_{\overline{m}}$ since it contains a Dirac delta. In order to overcome this difficulty, the derivation is done using an approximation of the Dirac delta function. The Dirac delta function is approximated in the suitable sense (weakly) by

$$\delta = \lim_{\nu \to 0} \frac{\mathbb{1}_{B_\nu}}{|B_\nu|} \qquad \text{(B.10)}$$

where $\nu > 0$ and B_ν is the closed ball in some arbitrary normed vector space S, for example $\mathbb{R}$ or $\mathbb{C}$, centred around origo with radius ν. Here, $|B_\nu|$ is the volume of the ball, ensuring that integration over the Dirac delta function results in the unit volume. Formally, the closed ball can be written as the set

$$B_\nu = \big\{x \in S \;\big|\; |x| \le \nu\big\}.$$

Assuming one has the closed balls $B_{\nu_x} \subset \mathbb{R}$ and $B_{\nu_\epsilon} \subset [-0.5, 0.5]$, using the approximation of the Dirac delta function, the priors of $\epsilon_{\overline{m}}$ and $x_{\overline{m}}$ can be approximated by $\tilde{f}(\epsilon_{\overline{m}}|x_{\overline{m}})$ and $\tilde{f}(x_{\overline{m}}|w, \sigma_x^2)$, respectively, as follows:

$$\tilde{f}(\epsilon_{\overline{m}}|x_{\overline{m}}) = \begin{cases} \frac{\mathbb{1}_{B_{\nu_\epsilon}}(\epsilon_{\overline{m}})}{|B_{\nu_\epsilon}|}, \text{ if } x_{\overline{m}} \in B_{\nu_x} \\ \mathbb{1}_{[-0.5,0.5]}(\epsilon_{\overline{m}}), \text{ otherwise} \end{cases}$$

and

$$\tilde{f}(x_{\overline{m}}|w, \sigma_x^2) = (1-w)\frac{\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)}{|B_{\nu_x}|} + w\frac{1}{\pi\sigma_x^2}\exp\Big(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\Big). \tag{B.11}$$

The conditional posterior of $x_{\overline{m}}$ can be written as

$$\begin{aligned} &f(x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) \\ &\quad = f(x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2)\big(\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|) + \mathbb{1}_{B^C_{\nu_x}}(|x_{\overline{m}}|)\big) \\ &\quad \propto Lf(\epsilon_{\overline{m}}|x_{\overline{m}})f(x_{\overline{m}}|w, \sigma_x^2)\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|) \\ &\qquad + Lf(\epsilon_{\overline{m}}|x_{\overline{m}})f(x_{\overline{m}}|w, \sigma_x^2)\mathbb{1}_{B^C_{\nu_x}}(|x_{\overline{m}}|), \end{aligned}$$

where Bayes' rule was applied in the form of equation (B.9c). Making use of the approximations of the priors then lends

$$\begin{aligned} &\tilde{f}(x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) \\ &\quad \propto Lf(x_{\overline{m}}|w, \sigma_x^2)\bigg(\frac{\mathbb{1}_{B_{\nu_\epsilon}}(\epsilon_{\overline{m}})}{|B_{\nu_\epsilon}|}\mathbb{1}_{\{0\}}(|x_{\overline{m}}|) + \mathbb{1}_{\mathbb{R}\setminus\{0\}}(|x_{\overline{m}}|)\bigg)\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|) \\ &\quad + Lf(x_{\overline{m}}|w, \sigma_x^2)\mathbb{1}_{B^C_{\nu_x}}(|x_{\overline{m}}|), \end{aligned}$$

where $\tilde{f}(x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2)$ is the approximation of the conditional posterior of the target amplitude elements. The product of the indicator functions $\mathbb{1}_{\mathbb{R}\setminus\{0\}}(|x_{\overline{m}}|)$ and $\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)$ in the first term can be seen to satisfy

$$\mathbb{1}_{\mathbb{R}\setminus\{0\}}(|x_{\overline{m}}|)\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|) = \mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)$$

since integrating over $\{0\}$ has no contribution. Similarly,

$$\mathbb{1}_{\{0\}}(|x_{\overline{m}}|)\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|) = 0.$$

Thus, it holds that

$$\begin{aligned} f(x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) &= Lf(x_{\overline{m}}|w, \sigma_x^2)\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|) \\ &\quad + Lf(x_{\overline{m}}|w, \sigma_x^2)\mathbb{1}_{B^C_{\nu_x}}(|x_{\overline{m}}|). \end{aligned}$$

Using the expression of the approximated target amplitude prior, $\tilde{f}(x_{\overline{m}}|w,\sigma_x^2)$, given in equation (B.11), lends

$$\begin{aligned}
&\tilde{f}(x_{\overline{m}}|\boldsymbol{y},\boldsymbol{\epsilon},\boldsymbol{x}_{-\overline{m}},\sigma^2,w,\sigma_x^2)\\
&\quad\propto L\left((1-w)\frac{\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)}{|B_{\nu_x}|}+w\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right)\right)\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)\\
&\qquad+L\left((1-w)\frac{\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)}{|B_{\nu_x}|}+w\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right)\right)\mathbb{1}_{B_{\nu_x}^C}(|x_{\overline{m}}|).
\end{aligned}$$

In the first term, one has $\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)=\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)$. In the second term, the product of the indicator functions $\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)$ and $\mathbb{1}_{B_{\nu_x}^C}(|x_{\overline{m}}|)$ is zero since the sets are complements, giving

$$\begin{aligned}
&\tilde{f}(x_{\overline{m}}|\boldsymbol{y},\boldsymbol{\epsilon},\boldsymbol{x}_{-\overline{m}},\sigma^2,w,\sigma_x^2)\\
&\quad\propto L(1-w)\frac{\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)}{|B_{\nu_x}|}\\
&\qquad+Lw\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right)\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)\\
&\qquad+Lw\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right)\mathbb{1}_{B_{\nu_x}^C}(|x_{\overline{m}}|)\\
&\quad=L(1-w)\frac{\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)}{|B_{\nu_x}|}\\
&\qquad+Lw\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right)\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)\\
&\qquad+Lw\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right)\mathbb{1}_{B_{\nu_x}^C}(|x_{\overline{m}}|).
\end{aligned}$$

Since $B_{\nu_x}\to\{0\}$ as $\nu_x\to 0$, it holds that $\mathbb{1}_{B_{\nu_x}}(|x_{\overline{m}}|)\to 0$ (in the integrated sense of a pdf) as $\nu_x\to 0$. Similarly, $B_{\nu_x}^C\to\mathbb{R}$ as $\nu_x\to 0$, and $\mathbb{1}_{B_{\nu_x}^C}(|x_{\overline{m}}|)\to 1$ as $\nu_x\to 0$. Letting $\nu_x\to 0$ therefore results in

$$\begin{aligned}
&\lim_{\nu_x\to 0}\tilde{f}(x_{\overline{m}}|\boldsymbol{y},\boldsymbol{\epsilon},\boldsymbol{x}_{-\overline{m}},\sigma^2,w,\sigma_x^2)\\
&\quad\propto L(\epsilon_{\overline{m}},x_{\overline{m}}=0,\sigma^2)(1-w)\delta(|x_{\overline{m}}|)\\
&\qquad+L(\epsilon_{\overline{m}},x_{\overline{m}},\sigma^2)w\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right)\\
&\quad\propto(1-w)\delta(|x_{\overline{m}}|)\\
&\qquad+L(\epsilon_{\overline{m}},x_{\overline{m}},\sigma^2)w\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right),
\end{aligned}$$

where it was used that $L(\epsilon_{\overline{m}},x_{\overline{m}}=0,\sigma^2)=1$. Inserting the expression of $L(\epsilon_{\overline{m}},x_{\overline{m}},\sigma^2)$ from equation (B.7) into this expression, the conditional posterior can be computed as

$$\begin{aligned}
&f(x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) \\
&\quad \propto (1-w)\delta(|x_{\overline{m}}|) \\
&\qquad + \exp\left(- \frac{|x_{\overline{m}}|^2 - x_{\overline{m}} \boldsymbol{e}_{\overline{m}}^H \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) - x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}}}{\sigma^2} \right) \\
&\qquad\quad \times w \frac{1}{\pi \sigma_x^2} \exp\left(- \frac{|x_{\overline{m}}|^2}{\sigma_x^2} \right) \\
&\quad = (1-w)\delta(|x_{\overline{m}}|) \\
&\qquad + w \frac{1}{\pi \sigma_x^2} \exp\left(- \frac{\sigma^2 + \sigma_x^2}{\sigma^2 \sigma_x^2} |x_{\overline{m}}|^2 - x_{\overline{m}}^* \frac{\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}}}{\sigma^2} - x_{\overline{m}} \frac{\boldsymbol{e}_{\overline{m}}^H \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})}{\sigma^2} \right).
\end{aligned}$$

Defining

$$\eta_{\overline{m}}^2 := \frac{\sigma^2 + \sigma_x^2}{\sigma^2 \sigma_x^2}$$

and

$$\mu_{\overline{m}} := \frac{\eta_{\overline{m}}^2}{\sigma^2} \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}}$$

gives the reformulation

$$\begin{aligned}
&f(x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) \\
&\quad \propto (1-w)\delta(|x_{\overline{m}}|) + w \frac{1}{\pi \sigma_x^2} \exp\left(- \frac{|x_{\overline{m}}|^2}{\eta_{\overline{m}}^2} - \frac{x_{\overline{m}}^* \mu_{\overline{m}}}{\eta_{\overline{m}}^2} - \frac{\mu_{\overline{m}}^* x_{\overline{m}}}{\eta_{\overline{m}}^2} \right) \\
&\quad = (1-w)\delta(|x_{\overline{m}}|) + w \frac{1}{\pi \sigma_x^2} \exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2} \right) \exp\left(- \frac{|x_{\overline{m}} - \mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2} \right).
\end{aligned}$$

Rewriting the factor in the first term as

$$(1-w) = \left(1 - w + w \frac{\eta_{\overline{m}}^2}{\sigma_x^2} \exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2} \right) - w \frac{\eta_{\overline{m}}^2}{\sigma_x^2} \exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2} \right) \right)$$

gives

$$\begin{aligned}
&f(x_{\overline{m}}|\boldsymbol{y},\boldsymbol{\epsilon},\boldsymbol{x}_{-\overline{m}},\sigma^2,w,\sigma_x^2)\\
&\quad\propto\left(1-w+w\frac{\eta_{\overline{m}}^2}{\sigma_x^2}\exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)-w\frac{\eta_{\overline{m}}^2}{\sigma_x^2}\exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)\right)\delta(|x_{\overline{m}}|)\\
&\qquad+w\frac{1}{\pi\sigma_x^2}\exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)\exp\left(-\frac{|x_{\overline{m}}-\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)\\
&\quad\propto\left(1-\frac{w\frac{\eta_{\overline{m}}^2}{\sigma_x^2}\exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)}{1-w+w\frac{\eta_{\overline{m}}^2}{\sigma_x^2}\exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)}\right)\delta(|x_{\overline{m}}|)\\
&\qquad+\frac{w\frac{\eta_{\overline{m}}^2}{\sigma_x^2}\exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)}{1-w+w\frac{\eta_{\overline{m}}^2}{\sigma_x^2}\exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)}\frac{1}{\pi\eta_{\overline{m}}^2}\exp\left(-\frac{|x_{\overline{m}}-\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right).
\end{aligned}$$

Defining the constant

$$w_{\overline{m}}:=\frac{w\frac{\eta_{\overline{m}}^2}{\sigma_x^2}\exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)}{1-w+w\frac{\eta_{\overline{m}}^2}{\sigma_x^2}\exp\left(\frac{|\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)},$$

one obtains the final expression of the conditional posterior of the target amplitude elements:

$$f(x_{\overline{m}}|\boldsymbol{y},\boldsymbol{\epsilon},\boldsymbol{x}_{-\overline{m}},\sigma^2,w,\sigma_x^2)\propto(1-w_{\overline{m}})\delta(|x_{\overline{m}}|)+w_{\overline{m}}\frac{1}{\pi\eta_{\overline{m}}^2}\exp\left(-\frac{|x_{\overline{m}}-\mu_{\overline{m}}|^2}{\eta_{\overline{m}}^2}\right)$$

Examining this expression of the conditional posterior pdf, one can observe that the conditional posterior of $x_{\overline{m}}$ is distributed according to

$$x_{\overline{m}}|\boldsymbol{y},\boldsymbol{\epsilon},\boldsymbol{x}_{-\overline{m}},\sigma^2,w,\sigma_x^2\sim Ber\mathcal{CN}\big(w_{\overline{m}},\mu_{\overline{m}},\eta_{\overline{m}}^2\big),$$

which concludes the derivation of the conditional posterior of the target amplitude elements.

B.2 Grid mismatch vector elements

As stated above, the elements of $\boldsymbol{\epsilon}$ are to be sampled elementwise in order to facilitate the usage of a Gibbs sampler. Therefore, the conditional posterior distribution of each component $\epsilon_{\overline{m}}$ must be derived. This is done using the conditional joint posterior seen in equation (B.4c).

In a similar fashion as in the derivation of the conditional joint posterior given by equation (B.4) from the joint posterior seen in (B.1), the conditional joint posterior can be written as

$$f(\epsilon_{\overline{m}}, x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2) = f(\epsilon_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}, \sigma^2, w, \sigma_x^2) \\ \times f(x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2).$$

The conditional posterior of $\epsilon_{\overline{m}}$ can thus be computed from equation (B.6) as

$$\begin{aligned} &f(\epsilon_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}, \sigma^2, w, \sigma_x^2) \\ &= \frac{f(\boldsymbol{y}|\boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\epsilon_{\overline{m}}|x_{\overline{m}}) f(x_{\overline{m}}|w, \sigma_x^2)}{f(x_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}_{-\overline{m}}, \sigma^2, w, \sigma_x^2)} \\ &\propto f(\boldsymbol{y}|\boldsymbol{\epsilon}, \boldsymbol{x}, \sigma^2) f(\epsilon_{\overline{m}}|x_{\overline{m}}) f(x_{\overline{m}}|w, \sigma_x^2) \\ &= \exp\left(\frac{|x_{\overline{m}}|^2 - x_{\overline{m}} \boldsymbol{e}_{\overline{m}}^H \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) - x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}}}{-\sigma^{-2}} \right) \\ &\quad \times f(\epsilon_{\overline{m}}|x_{\overline{m}}) f(x_{\overline{m}}|w, \sigma_x^2) \\ &\propto \exp\Big(\frac{x_{\overline{m}} \boldsymbol{e}_{\overline{m}}^H \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) + x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}}}{\sigma^{-2}}\Big) f(\epsilon_{\overline{m}}|x_{\overline{m}}). \end{aligned}$$

Using the fact that $z + z^* = 2\,\mathfrak{Re}[z]$ for an arbitrary $z \in \mathbb{C}$ and that

$$\left(x_{\overline{m}} \boldsymbol{e}_{\overline{m}}^H \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) \right)^* = x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}},$$

where the left-hand side and the right-hand side are complex conjugates, one obtains

$$f(\epsilon_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}, \sigma^2, w, \sigma_x^2) \propto \exp\left(2\sigma^{-2} \mathfrak{Re}\left[x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}} \epsilon_{\overline{m}}^H \boldsymbol{e}_{\overline{m}} \right] \right) f(\epsilon_{\overline{m}}|x_{\overline{m}}). \tag{B.12}$$

Examining $x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}}$, one can rewrite it as

$$\begin{aligned} &x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}} \\ &= x_{\overline{m}}^* \sum_{m=0}^{M-1} \left[\boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}}) \right]_m^* [\boldsymbol{e}_{\overline{m}}]_m \\ &= x_{\overline{m}}^* \sum_{m=0}^{M-1} \frac{1}{\sqrt{M}} \exp\left(-j2\pi m \frac{\overline{m} + \epsilon_{\overline{m}}}{\overline{M}} \right) [\boldsymbol{e}_{\overline{m}}]_m \\ &= \sum_{m=0}^{M-1} \frac{1}{\sqrt{M}} x_{\overline{m}}^* \exp\left(-j2\pi m \frac{\overline{m}}{\overline{M}} \right) [\boldsymbol{e}_{\overline{m}}]_m \exp\left(-j2\pi m \frac{\epsilon_{\overline{m}}}{\overline{M}} \right). \end{aligned}$$

Defining the matrix

$$\boldsymbol{u} \in \mathbb{C}^{M \times \overline{M}}$$

$$[\boldsymbol{u}_{\overline{m}}]_m := \exp\left(j2\pi m \overline{m} / \overline{M} \right),$$

where $\boldsymbol{u}_{\overline{m}}$ is the $\overline{m}$:th column of $\boldsymbol{u}$, one further obtains:

$$x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}} = \sum_{m=0}^{M-1} \frac{1}{\sqrt{M}} x_{\overline{m}}^* [\boldsymbol{u}_{\overline{m}}]_m^* [\boldsymbol{e}_{\overline{m}}]_m \exp\Big(-j2\pi m \frac{\epsilon_{\overline{m}}}{\overline{M}} \Big),$$

using

$$\begin{aligned} \boldsymbol{b} &:= x_{\overline{m}}^* \boldsymbol{u}_{\overline{m}}^* \odot \boldsymbol{e}_{\overline{m}} \\ b_m &:= [\boldsymbol{b}]_m, \end{aligned}$$

where $\odot$ denotes the Hadamard product, results in

$$x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}} = \sum_{m=0}^{M-1} \frac{1}{\sqrt{M}} b_m \exp\Big(-j2\pi m \frac{\epsilon_{\overline{m}}}{\overline{M}} \Big). \tag{B.13}$$

Letting

$$z_m = b_m \exp\Big(-j2\pi m \frac{\epsilon_{\overline{m}}}{\overline{M}} \Big), \tag{B.14}$$

the polar form of z_m can in general be expressed as

$$z_m = |z_m|\, e^{j \arg(z_m)}. \tag{B.15}$$

Inserting the expression given by (B.14) into (B.15) lends

$$\begin{aligned} z_m &= |z_m|\, e^{j \arg(z_m)} && \text{(B.16a)} \\ &= \left| b_m \exp\Big(-j2\pi m \frac{\epsilon_{\overline{m}}}{\overline{M}} \Big) \right| e^{j \arg(b_m \exp(-j2\pi m \epsilon_{\overline{m}}/\overline{M}))} && \text{(B.16b)} \\ &= |b_m| \left| \exp\big(-j2\pi m \frac{\epsilon_{\overline{m}}}{\overline{M}} \big) \right| e^{j(\arg(b_m) + \arg(\exp(-j2\pi m \epsilon_{\overline{m}}/\overline{M})))} && \text{(B.16c)} \\ &= |b_m| e^{j(\phi_m - 2\pi m \epsilon_{\overline{m}}/\overline{M})}, && \text{(B.16d)} \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{\phi} &\in \mathbb{R}^M \\ \phi_m &:= \arg(b_m). \end{aligned}$$

Inserting the expression given by(B.16d) into (B.13) and taking the real part results in

$$\begin{aligned} \mathfrak{Re}\big[x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}} \big] &= \mathfrak{Re}\left[\sum_{m=0}^{M-1} \frac{1}{\sqrt{M}} |b_m| e^{j(\phi_m - 2\pi m \epsilon_{\overline{m}}/\overline{M})} \right] && \text{(B.17a)} \\ &= \sum_{m=0}^{M-1} \frac{1}{\sqrt{M}} |b_m| \cos(2\pi \frac{\epsilon_{\overline{m}}}{\overline{M}} m - \phi_m), && \text{(B.17b)} \end{aligned}$$

where it was used that

$$e^{j\theta} = \cos(\theta) + j\sin(\theta),$$

and the symmetry of the cosine function. In conclusion, the conditional posterior of $\epsilon_{\overline{m}}$ formulated as in equation (B.12) can be rewritten using (B.17) as

$$\begin{aligned}
&f(\epsilon_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}, \sigma^2, w, \sigma_x^2) && \text{(B.18a)}\\
&\quad \propto \exp\left(2\sigma^{-2}\mathfrak{Re}\big(x_{\overline{m}}^* \boldsymbol{f}_{\overline{m}}(\epsilon_{\overline{m}})^H \boldsymbol{e}_{\overline{m}}\big)\right) f(\epsilon_{\overline{m}}|x_{\overline{m}}) && \text{(B.18b)}\\
&\quad = \exp\left(2\sigma^{-2}\sum_{m=0}^{M-1}\frac{1}{\sqrt{M}}|b_m|\cos(2\pi\frac{\epsilon_{\overline{m}}}{M}m - \phi_m)\right) f(\epsilon_{\overline{m}}|x_{\overline{m}}) && \text{(B.18c)}\\
&\quad = \exp\left(\sum_{m=0}^{M-1}\kappa_m \cos\big(2\pi\frac{\epsilon_{\overline{m}}}{M}m - \phi_m\big)\right) f(\epsilon_{\overline{m}}|x_{\overline{m}}), && \text{(B.18d)}
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{\kappa} &\in \mathbb{R}^M\\
\kappa_m &:= \frac{2}{\sigma^2\sqrt{M}}|b_m|.
\end{aligned}$$

This concludes the derivation of the conditional posterior of the grid mismatch elements.

Observing the resulting expression of the conditional posterior of each grid mismatch $\epsilon_{\overline{m}}$ in equation (B.18d), one can see that it is the pdf of a dilated and truncated generalised von Mises distribution of order M [14, equation (7)], which is denoted

$$\epsilon_{\overline{m}}|\boldsymbol{y}, \boldsymbol{\epsilon}_{-\overline{m}}, \boldsymbol{x}, \sigma^2, w, \sigma_x^2 \sim \mathrm{dGvM}_{M\,[-0.5,0.5]}(\boldsymbol{\kappa}, \boldsymbol{\phi}).$$

The truncation is due to the multiplication of the posterior with the prior pdf of $\epsilon_{\overline{m}}$, $f(\epsilon_{\overline{m}}|x_{\overline{m}})$, which is zero when $x_{\overline{m}} = 0$ and otherwise $f(\epsilon_{\overline{m}}|x_{\overline{m}} \neq 0) = \mathbb{1}_{[-0.5,0.5]}(\epsilon_{\overline{m}})$ according to the definition in equation (4.8). The dilation is due to the scaling of the multiplication of $2\pi m/M$ with $\epsilon_{\overline{m}}$ in the cosine function of the posterior.

B.3 Noise power

In the same manner as for the elementwise amplitudes $x_{\overline{m}}$ and grid mismatches $\epsilon_{\overline{m}}$, the expression for the posterior of σ^2 is obtained by conditioning on all the other joint parameters seen in equation (B.1) and using proportionality. The resulting posterior is calculated as

$$\begin{aligned} f(\sigma^2|\boldsymbol{y},\boldsymbol{x},\boldsymbol{\epsilon}) &\propto f(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\epsilon},\sigma^2)f(\sigma^2) \\ &\propto \exp\left(-\frac{\|\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\|_2^2}{\sigma^2}\right)\frac{\exp\left(-\frac{\gamma_1}{\sigma^2}\right)}{(\sigma^2)^{\gamma_0+1}}\mathbb{1}_{[0,+\infty[}(\sigma^2) \\ &= \frac{1}{(\sigma^2)^{\gamma_0+M+1}}\exp\left(-\frac{\gamma_1+\|\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\|_2^2}{\sigma^2}\right)\mathbb{1}_{[0,+\infty[}(\sigma^2), \end{aligned}$$

one observes that this corresponds to the inverse-gamma distribution

$$\sigma^2|\boldsymbol{y},\boldsymbol{x},\boldsymbol{\epsilon} \sim \mathcal{IG}(\gamma_0+M,\gamma_1+\|\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{\epsilon})\boldsymbol{x}\|_2^2).$$

As expected, the prior of the noise power σ^2 is a conjugate prior to the likelihood of the observation signal since both the prior and posterior pdfs $f(\sigma^2)$ and $f(\sigma^2|\boldsymbol{y},\boldsymbol{x},\boldsymbol{\epsilon})$, respectively, belong to the same family of distributions.

B.4 Occupancy level

The same method as for the previous model parameters is used to obtain the conditional posterior for the occupancy level w. The general expression for it is

$$f(w|\boldsymbol{x}) \propto f(\boldsymbol{x}|w,\sigma_x^2)f(w).$$

The level of occupancy w is a measure of the probability that a specific component of the target amplitude vector is nonzero. Because the prior of $\boldsymbol{x}$ is a mixture distribution, more specifically $x_{\overline{m}}|w,\sigma_x^2 \sim Ber\mathcal{CN}(w,0,\sigma_x^2)$, we first assume that the number of nonzero components of $\boldsymbol{x}$ is n_1. We define the index set A of cardinality n_1 as the set of indices that correspond to the nonzero components of a target amplitude vector $\boldsymbol{x}$. The posterior of w can thus be readily computed as

$$\begin{aligned} f(w|\boldsymbol{x}) &\propto f(\boldsymbol{x}|w,\sigma_x^2)f(w) \\ &= \prod_{\overline{m}=0}^{\overline{M}-1} f(x_{\overline{m}}|w,\sigma_x^2)f(w). \end{aligned}$$

Making use of the known target amplitude vector $\boldsymbol{x}$ and where the indices of the nonzero components are located using the index set A, together with the approximation of the Dirac delta approximation given by equation (B.10), the posterior of w can be written as

$$f(w|\boldsymbol{x}) \propto \prod_{\overline{m}_c\in A^C}(1-w)\frac{1}{|B_\nu|}\mathbb{1}_{B_\nu}(|x_{\overline{m}_c}|)\prod_{\overline{m}\in A} w\frac{1}{\pi\sigma_x^2}\exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right),$$

where A^C is the complement of the index set A, thus containing the indices of the zero elements in $\boldsymbol{x}$. Using the fact that $|x_{\overline{m}_c}| = 0$, it follows that $\mathbb{1}_{B_\nu}(|x_{\overline{m}_c}|) = 1$ for all $\overline{m}_c \in A^C$ since $|x_{\overline{m}_c}| \leq \nu$ for all $\nu \geq 0$. Consequently,

$$\begin{aligned} f(w|\boldsymbol{x}) &\propto (1-w)^{\overline{M}-n_1} \frac{1}{|B_\nu|^{\overline{M}-n_1}} \prod_{\overline{m}\in A} w \frac{1}{\pi\sigma_x^2} \exp\left(- \frac{|x_{\overline{m}}|^2}{\sigma_x^2} \right) \\ &= (1-w)^{\overline{M}-n_1} w^{n_1} \frac{1}{|B_\nu|^{\overline{M}-n_1}} \prod_{\overline{m}\in A} \frac{1}{\pi\sigma_x^2} \exp\left(- \frac{|x_{\overline{m}}|^2}{\sigma_x^2} \right) \\ &\propto (1-w)^{\overline{M}-n_1} w^{n_1}. \end{aligned}$$

Letting $n_0 \coloneqq \overline{M} - n_1$, one can see that the posterior of w is a beta distribution

$$w|\boldsymbol{x} \sim \text{Beta}(n_1+1, n_0+1),$$

which concludes the derivation of the level of occupancy posterior.

B.5 Target signal power

The conditional posterior of σ_x^2 is derived in this section. The general expression of the posterior is

$$f(\sigma_x^2|\boldsymbol{x}) \propto f(\boldsymbol{x}|w,\sigma_x^2) f(\sigma_x^2).$$

As in the case of the derivation of the conditional posterior of w, it is assumed that the target amplitude vector $\boldsymbol{x}$ has exactly n_1 nonzero components, where the index set A is the same, the posterior can thus be computed in a similar fashion as for w. The posterior of σ_x^2 can thus be computed as

$$\begin{aligned} f(\sigma_x^2|\boldsymbol{x}) &\propto f(\boldsymbol{x}|w,\sigma_x^2) f(\sigma_x^2) \\ &= \prod_{\overline{m}=0}^{\overline{M}-1} f(x_{\overline{m}}|w,\sigma_x^2) f(\sigma_x^2) \\ &= \prod_{\overline{m}_c\in A^C} (1-w) \frac{1}{|B_\nu|} \mathbb{1}_{B_\nu}(|x_{\overline{m}_c}|) \\ &\quad \times \prod_{\overline{m}\in A} w \frac{1}{\pi\sigma_x^2} \exp\left(- \frac{|x_{\overline{m}}|^2}{\sigma_x^2} \right) \frac{e^{-\beta_1/\sigma_x^2}}{(\sigma_x^2)^{\beta_0+1}} \mathbb{1}_{[0,+\infty[}(\sigma_x^2). \end{aligned}$$

Using the fact that $|x_{\overline{m}_c}| = 0$, it follows that $\mathbb{1}_{B_\nu}(|x_{\overline{m}_c}|) = 1$ for all $\overline{m}_c \in A^C$ since $|x_{\overline{m}_c}| \leq \nu$ for all $\nu \geq 0$. It thus follows that

$$\begin{aligned}
&f(\sigma_x^2|\boldsymbol{x}) \\
&\quad \propto (1-w)^{\overline{M}-n_1} \frac{1}{|B_\nu|^{\overline{M}-n_1}} \prod_{\overline{m}\in A} w \frac{1}{\pi\sigma_x^2} \exp\left(-\frac{|x_{\overline{m}}|^2}{\sigma_x^2}\right) \frac{e^{-\beta_1/\sigma_x^2}}{(\sigma_x^2)^{\beta_0+1}} \mathbb{1}_{[0,+\infty[}(\sigma_x^2) \\
&\quad \propto \frac{1}{(\sigma_x^2)^{n_1}} \exp\left(-\frac{\|\boldsymbol{x}\|_2^2}{\sigma_x^2}\right) \frac{e^{-\beta_1/\sigma_x^2}}{(\sigma_x^2)^{\beta_0+1}} \mathbb{1}_{[0,+\infty[}(\sigma_x^2) \\
&\quad = \frac{\exp\big(-(\beta_1 + \|\boldsymbol{x}\|_2^2)/\sigma_x^2\big)}{(\sigma_x^2)^{\beta_0+n_1+1}} \mathbb{1}_{[0,+\infty[}(\sigma_x^2).
\end{aligned}$$

Examining the posterior of the target signal power, one can in conclusion observe that it is in fact an inverse-gamma distribution

$$\sigma_x^2|\boldsymbol{x} \sim \mathcal{IG}(\beta_0 + n_1, \beta_1 + \|\boldsymbol{x}\|_2^2).$$

Appendix C

Tables Summarising Network Architectures and Training Setups

C.1 Bivariate Gaussians with means on unit circle

Latent vector: $\boldsymbol{z} \sim \mathcal{N}(0, I_{2\times 2})$ Angle: $\theta \in [0, 2\pi]$
$\text{concat}(\boldsymbol{z}, \sin(\theta), \cos(\theta)) \in \mathbb{R}^4$
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 2

(a) Generator

Sample: $\boldsymbol{x} \in \mathbb{R}^2$ Angle: $\theta \in [0, 2\pi]$
$\text{concat}(\boldsymbol{x}, \sin(\theta), \cos(\theta)) \in \mathbb{R}^4$
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 1; Sigmoid

(b) Discriminator

Table C.1: CCGAN architecture for the simulation of Gaussians with common fixed covariance matrix and means taking values on the unit circle. The abbreviation "fc" stands for "fully connected" layers, where the right arrow indicates how many hidden nodes are in the next layer. "BN" stands for "batch normalisation" and one can see that the ReLU activation function was used.

Parameter	Value
Iterations	6000
Learning rate	$5 \cdot 10^{-5}$
Optimiser	ADAM (with $\beta_1 = 0.5$ and $\beta_2 = 0.999$)
Mini-batch size	128
σ	0.074
ν	0.0083

Table C.2: Training setup of the CCGAN for simulating bivariate Gaussians with fixed common covariance matrix and means taking values on the unit circle. The learning rate and mini-batch size is the same for both the generator and discriminator. The computed HVDL regularisation parameters σ and ν are also included, as defined in equations (3.13) to (3.14) respectively.

C.2 Univariate Gaussians with varying expectation

Latent vector: $\boldsymbol{z} \sim \mathcal{N}(0, I_{2\times 2})$ Mean: $\mu \in [0, 4]$
concat$(\boldsymbol{z}, \mu) \in \mathbb{R}^3$
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 1

(a) Generator

Sample: $x \in \mathbb{R}$ Mean: $\mu \in [0, 4]$
concat$(x, \mu) \in \mathbb{R}^2$
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 1; Sigmoid

(b) Discriminator

Table C.3: CCGAN architecture for the simulation of Gaussians with common fixed standard deviation and means taking values on the interval $[0, 4]$.

Parameter	Value
Iterations	2000
Learning rate	10^{-4}
Optimiser	ADAM (with $\beta_1 = 0.5$ and $\beta_2 = 0.999$)
Mini-batch size	128
σ	0.095
ν	0.025

Table C.4: Training setup of the CCGAN for simulating univariate Gaussians with fixed common standard deviation and means taking values on the interval $[0, 4]$. The learning rate and mini-batch size is the same for both the generator and discriminator.

C.3 Univariate Gaussians with varying expectation and variance

Latent vector: $\boldsymbol{z} \sim \mathcal{N}(0, I_{2\times 2})$ Mean: $\mu \in [0, 4]$ Standard deviation: $\sigma \in [0.05, 0.1]$
concat$(\boldsymbol{z}, \mu, \sigma) \in \mathbb{R}^4$
fc $\to$ 100; BN; ReLU
fc $\to$ 100; BN; ReLU
fc $\to$ 100; BN; ReLU
fc $\to$ 100; BN; ReLU
fc $\to$ 100; BN; ReLU
fc $\to$ 100; BN; ReLU
fc $\to$ 1

(a) Generator

Sample: $x \in \mathbb{R}$ Mean: $\mu \in [0, 4]$ Standard deviation: $\sigma \in [0.05, 0.1]$
concat$(x, \mu, \sigma) \in \mathbb{R}^3$
fc $\to$ 100; ReLU
fc $\to$ 100; ReLU
fc $\to$ 100; ReLU
fc $\to$ 100; ReLU
fc $\to$ 100; ReLU
fc $\to$ 1; Sigmoid

(b) Discriminator

Table C.5: CCGAN architecture for the simulation of Gaussians with means taking values on the interval $[0, 4]$ and standard deviations taking values on $[0.05, 0.1]$.

Parameter	Value
Iterations	2000
Learning rate	10^{-4}
Optimiser	ADAM (with $\beta_1 = 0.5$ and $\beta_2 = 0.999$)
Mini-batch size	64
σ	[0.021, 0.021]
ν	[0.0084, 0.0084]

Table C.6: Training setup of the CCGAN for simulating univariate Gaussians with means taking values on the interval $[0, 4]$ and standard deviations taking values on $[0.05, 0.1]$. The learning rate and mini-batch size is the same for both the generator and discriminator. Note that the regularisation parameters are two-dimensional since the regression labels are two-dimensional. The first element corresponds to the mean, while the second one corresponds to the standard deviation.

C.4 von Mises distributions with varying expectation

Latent vector: $\boldsymbol{z} \sim \mathcal{N}(0, I_{2\times 2})$ Mean: $\phi \in [-\pi, \pi]$
concat$(\boldsymbol{z}, \phi) \in \mathbb{R}^3$
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 1

(a) Generator

Sample: $x \in \mathbb{R}$ Mean: $\phi \in [-\pi, \pi]$
concat$(x, \phi) \in \mathbb{R}^2$
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 1; Sigmoid

(b) Discriminator

Table C.7: CCGAN architecture for the simulation of von Mises distributions with fixed concentration $\kappa = 20$, conditioned on means taking values on the interval $[-\pi, \pi]$.

Parameter	Value
Iterations	300
Learning rate	$5 \cdot 10^{-5}$
Optimiser	ADAM (with $\beta_1 = 0.5$ and $\beta_2 = 0.999$)
Mini-batch size	128
σ	0.065
ν	0.0084

Table C.8: Training setup of the CCGAN for simulating von Mises distributions with fixed common concentration κ and means ϕ taking values on the interval $[-\pi, \pi]$. The learning rate and mini-batch size is the same for both the generator and discriminator.

C.5 Dilated and truncated generalised von Mises distributions

Latent vector: $\boldsymbol{z} \sim \mathcal{N}(0, I_{2\times 2})$ Concentrations: $\boldsymbol{\kappa} \in \mathbb{R}^M$ Means: $\boldsymbol{\phi} \in \mathbb{R}^M$
concat$(\boldsymbol{z}, \boldsymbol{\kappa}, \boldsymbol{\phi}) \in \mathbb{R}^{2+2M}$
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 100; BN; ReLU
fc $\rightarrow$ 1

(a) Generator

Sample: $\epsilon_{\overline{m}} \in \mathbb{R}$ Concentrations: $\boldsymbol{\kappa} \in \mathbb{R}^M$ Means: $\boldsymbol{\phi} \in \mathbb{R}^M$
concat$(x, \boldsymbol{\kappa}, \boldsymbol{\phi}) \in \mathbb{R}^{1+2M}$
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 100; ReLU
fc $\rightarrow$ 1; Sigmoid

(b) Discriminator

Table C.9: CCGAN architecture for the simulation of dGvM$_{[-0.5,0.5]}$ distributions conditioned on the dGvM parameters $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$.

Parameter	Value
Iterations	1000
Learning rate	10^{-5}
Optimiser	ADAM (with $\beta_1 = 0.5$ and $\beta_2 = 0.999$)
Mini-batch size	512
σ	0
ν	0

Table C.10: Training setup of the CCGAN for simulating univariate $\mathrm{dGvM}_{[-0.5,0.5]}$ distributions conditioned on the parameters $\boldsymbol{\kappa}$ and $\boldsymbol{\phi}$. The learning rate and mini-batch size is the same for both the generator and discriminator. The CCGAN regularisation parameters are set to zero for all regression label elements.

Master's Theses in Mathematical Sciences 2022:E68
ISSN 1404-6342

LUTFMS-3458-2022

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/