Understanding Normalizing Flows

W.H. van den Bos

Bachelor Project
2024
Applied Mathematics

TUDelft

Understanding Normalizing Flows

by

W.H. van den Bos

to obtain the degree of Bachelor of Science
at the Delft University of Technology,

Student number:	5323924	
Project duration:	April 2, 2024 – June 18, 2024	
Thesis committee:	Dr. ir. J. Bierkens,	TU Delft, supervisor
	Dr. D. de Laat,	TU Delft

TU Delft

Abstract

Normalizing flows are a probabilistic method to estimate the underlying density of data samples. The method is flow based and non-parametric, with the aim of being flexible, but still computationally manageable. This report aims to explain the process of normalizing flows by the underlying principles of this probabilistic method. Both the method itself and the underlying principles are explained and reduced. An example of a normalizing flow is then implemented to showcase the method and support the underlying principles.

W.H. van den Bos
Delft, June 2024

Contents

1 introduction

Normalizing flows are a probabilistic models to approximate the underlying density function from sample data. Approximating distributions from data samples is known as the density estimation problem. This problem is faced in many fields. Environmental scientists encounter it when estimating the distribution of pollutant concentrations in air or water samples for the identification of contamination hotspots and informing regulatory actions [5]. In medical imaging, density estimation in segmenting MRI scans by estimating the distribution of pixel intensities helps differentiate between healthy and diseased tissues, improving diagnostic accuracy and treatment planning [9]. Normalizing flows are a probabilistic method for density estimation. Normalizing flows are particularly an increasingly active area of machine learning research. This is because normalizing flows are commonly implemented for the training of deep neural networks because of their computationally simple architecture [10].
These are just some of many examples of the use of probabilistic models that showcase the importance of them in various fields. One of the issues probabilistic models face when estimating densities from data samples is data sparsity. Data sparsity refers to the situation where the available data points are insufficient or unevenly distributed across the space being analyzed. This lack of data density can lead to several significant issues. The model may struggle to identify true patterns and may instead capture noise, leading to unreliable estimations. This is called overfitting on the data. Conversely, data sparsity can also lead to underfitting, where the model is too simplistic and fails to capture the complexity of the data. Another challenge for probabilistic models is overcoming the curse of dimensionality: As dimensions increase, the data volume grows exponentially, requiring more data, more complex models and more computational resources to produce reliable underlying densities. A related issue is that of computational efficiency. To produce reasonable estimates of underlying densities, probabilistic models can require a large amounts of data. The model needs to efficiently compute with these large amounts of data.
Many probabilistic models have been developed for the density estimation problem, often specified for a particular application. Depending on the application, there is for example a choice between parametric methods, which assume a specific form for the distribution of the sample data, and non-parametric methods, which make minimal assumptions about the distribution shape. Normalizing flows are non-parametric, flow based models. The term flow based describes the algorithmic approach of this probabilistic method. The model starts with a simple estimation of the data samples. When the data samples follow a complex distribution, this simple density estimation might not be a reasonable estimate of the real density function. The goal of this flow based model is to push this estimation through a flow of many small and simple transformations to produce a density function that better describes the underlying distribution.
This paper aims to provide an overview of the probabilistic model called normalizing flows and the underlying principles of this method. This is done based on the structure of normalizing flows as explained in [13], [10] and [12]. Principles underlying this probabilistic method provide an understanding of why normalizing flows work and are useful for density estimation of data samples. Additionally, the underlying principles of normalizing flows are widely used in many other probabilistic models [3]. Explaining normalizing flows will be done by dividing this paper into two parts. In **chapter 2, Underlying principles**, the focus is on the underlying principles of normalizing flows. In **chapter 3, Applying a linear normalizing flow**, to showcase the underlying principles of normalizing flows indeed result in a working probabilistic model, the model is executed on real sample data.

Chapter 2 provide a general overview of the normalizing flow probabilistic model. In section 2.1 the normalizing flow algorithm is introduced by first demonstrating that the goal of the model is to estimate the map $y(x)$ from the samples of random variable x to samples of random variable y with a flow function. In section 2.2 we then focus on how we can use the log-likelihood to produce the flow functions. To showcase how this is done, in chapter 2.3 we look at the continuous case. For this case we can derive expressions for the evolution of the flow function that increases the log-likelihood. Finally in section 2.3.4, two examples of normalizing flows in the continuous case are provided in figures 2.8 and 2.9.
After showing the basic principles of normalizing flows in the continuous case, in section 2.4 the Kullback-Leibler divergence is used to prove that normalizing flows can be used to estimate probability density functions from samples that follow arbitrary distribution. The final part of chapter 2 is about individual flow requirements that are proposed for better results of the normalizing flow algorithm.
Chapter 3 serves to support the knowledge gained about the underlying principles of normalizing flows in chapter 2. This is done by applying a normalizing flow based on linear flow functions. Section 3.1 introduces the framework of the normalizing flow algorithm as described in 2.1 in the context of linear flow functions. To understand the effect of the linear flow we first introduce a parametric procedure, where we impose that the samples are made from a distribution with a linear relation to the standard Gaussian distribution. Then we introduce a flow based approach, where each of the flows is the result of a linear parametric procedure. Section 3.2 uses a 5 parameter flow function to illustrate a normalizing flow model based on linear flow functions. The examples showcase the statements from previous sections on real data samples.

2 Underlying principles

In this chapter we will explain the underlying principles of normalizing flows. normalizing flows are probabilistic models to approximate the underlying density of data samples. We will introduce normalizing flows by first explaining the process and the goal, which is to approximate a function that maps the data samples of an unknown distribution into a new random variable with a relative simple and known density. In practice a normally distributed random variable is chosen for the known density, hence the name normalizing flows. This map is approximated using flow functions that together form a flow. From this flow we can then retrieve an estimation of the density underlying the data and generate new samples of the data. We will explain how to construct a flow function that increases the log-likelihood. Finally we will explain why the normalizing flow process works as a probabilistic model based on the Kullback-Leibler divergence and individual flow requirements.

2.1. The normalizing flow process

We have observations x^j with $j = 1...m$ of continuous random variable x in $\mathbb{R}^n$. The goal is to find the unknown probability density function $\rho(x)$, or generate new samples of the data. This section will introduce the process of a normalizing flow to estimate this unknown probability density function. We will first show how this can be done when we know a function $y(x)$ mapping the samples to samples of random variable y with known density function, such as the standard normal density function $\mu(y)$. We will then demonstrate this with an example. We will then introduce how flow based models estimate this function $y(x)$, and explain how this estimation can be used to estimate the underlying density $\rho(x)$ of the samples.

2.1.1. The idea behind normalizing flows

Let $y(x)$ be a function that maps data x^j into a new set of variables y^j of random variable y with known density $\mu(y)$. In practice, $\mu(y)$ is taken to be a simple density. For example,[13] gives a motivation for choosing the isotropic Gaussian. One defining property of flow-based models is that $y(x)$ is invertible and differentiable and $y^{-1}(x)$ is invertible and differentiable. A consequence of the change of variables (appendix A.2) is that we can compute the density $\rho(x)$:

Proposition 2.1.1. *Let $x \in A \subset \mathbb{R}^n$. Now suppose $y(x)$ maps random variable x to random variable y. When $y(x)$ is differentiable and invective, then*

$$\rho(x) = J_y(x)\mu(y(x)). \tag{2.1}$$

In this equation $J_y(x)$ is the Jacobian determinant of the map $y(x)$ that maps random variable x to y. For the definition of the Jacobian matrix, see appendix A.1. Notice that the Jacobean determinant, $J_y(x)$, can be computed since $y(x)$ is differentiable. In figure 2.1 the goal of the normalizing flow is visualized.

Proof. Notice that the function $y(x)$ is injective since it in invertible. This means we can apply the change of variables formula from appendix A.2 to find that for all $A \subset \mathbb{R}^n$: [4]:

$$\mathbb{P}(x \in A) = \int_A \rho(x)\,dx = \mathbb{P}(y \in y(A)) = \int_{y(A)} \mu(y)\,dy = \int_A \mu(y(x)) J_y(x)\,dx. \tag{2.2}$$

Since it holds for all A, we must have that $\rho(x) = J_y(x)\mu(y(x))$. □

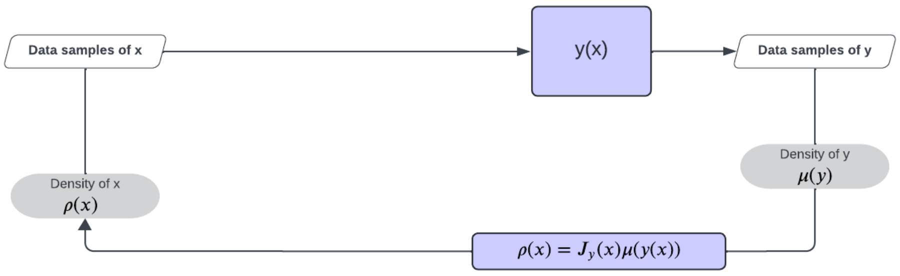


Figure 2.1: The normalizing flow algorithm constructs $y(x)$.

The function $y(x)$ allows both sampling from the model, and evaluating the model's density. Sampling from the model means generating a sample of variable y via the distribution $\mu(y)$ and calculating from it a new sample of variable x using the map $y(x)^{-1}$. For this application of the flow we make the restriction on $y(x)$ that it has to be invertible. The second operation is evaluating the model's density. This can be done from equation (2.1), and therefore requires the calculation of the Jacobian determinant of $y(x)$. Therefore we have that $y(x)$ must be differentiable.The application of the flow will determine which of these operations will be used. For sampling via y it is beneficial to have $y^{-1}(x)$ to be efficient. Evaluating the density would focus more on efficiently computing $J_y(x)$. However, the function $y(x)$ is unknown. We want to estimate the function $y(x)$ based on the data x. This will be done by introducing a flow $\phi_t(x)$.

2.1.2. Example of the idea of normalizing flows

We will now showcase the idea behind normalizing flows with an example in $\mathbb{R}^1$. In this example, 100 samples x^j, $j = 1,2,..,100$ are generated from random variable x which is distributed according to a normal distribution with mean 2 and standard deviation 1. This is visualized in figure 2.2.

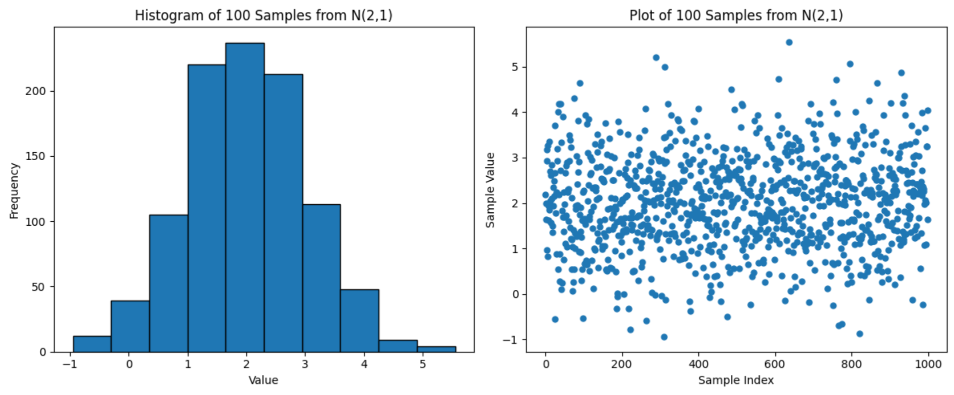


Figure 2.2: 100 samples from N(0,2).

Suppose that we do not know that the samples x^j follow this distribution. However, we do know a function $y(x)$ from random variable x towards random variable y, where y has known probability density function $\mu(y)$. In this example $\mu(y)$ is the density function of the standard normal:

$$\mu(y) = \frac{1}{\sqrt{2\pi}} e^{\frac{y^2}{2}}. \tag{2.3}$$

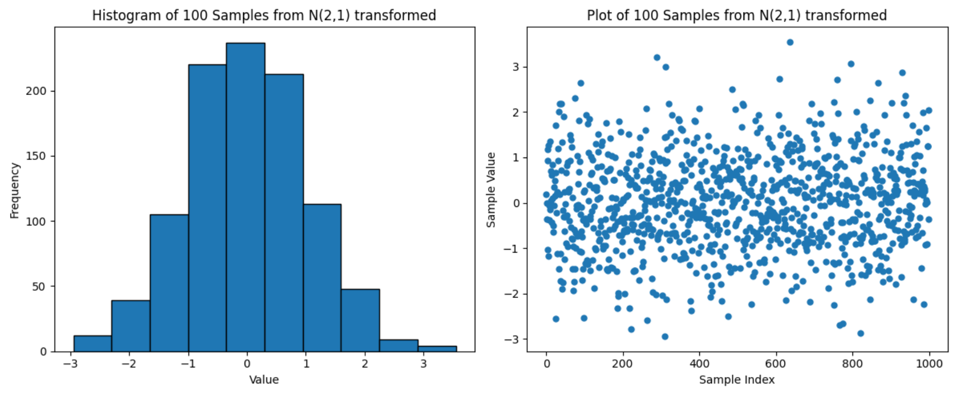


Figure 2.3: 100 samples from N(0,2) transformed by $y(x)$.

We know the function $y(x) = x - 2$. This function is applied to the data to get the resulting data of figure 2.3. The density $\rho(x)$ can be computed with (2.1). When we do not know the function y, an an estimate of $\rho(x)$ would be made by choosing $y(x)$ as the identity map. In other words, the best guess at $\rho(x)$ is saying it is the standard normal density function. Now we will proceed with computing the density of x when we do know the function y. The Jacobian determinant in $\mathbb{R}^1$ is just the derivative. Therefore, the Jacobian of the transformation $y(x)$, $j_y(x) = \frac{dy(x)}{dx} = 1$. This means equation (2.1) becomes

$$\rho(x) = J_y(x)\mu(y(x)) = \frac{1}{\sqrt{2\pi}} e^{\frac{(x-2)^2}{2}}. \tag{2.4}$$

This is exactly the density function belonging to the normal distribution with standard deviation - 1 and mean 2. This was the indeed the distribution the 100 samples where generated from. Figure 2.4 is the plot of this density function $\rho(x)$.

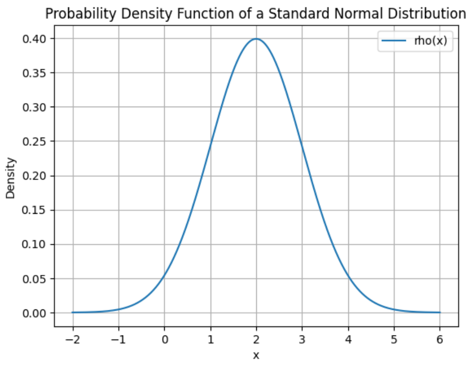


Figure 2.4: $\rho(x)$ computed with $y(x) = x - 2$.

We can use the result to generate new samples in two ways:

- $y(x) = x - 2$ has inverse function $y^{-1}(x)$. We can generate samples by generating a sample of random variable y and computing the inverse $x = y + 2$.
- We can sample directly from the computed $\rho(x)$ we computed with equation (2.1).

In this simple example we made the assumption that the function $y(x)$ was known to illustrate that we can use this function to evaluate the density of the samples $\rho(x)$, and generate new samples of random variable x. Normalizing flows are used to construct an approximation of this function $y(x)$. In the next section we will introduce the flow based structure of a normalizing flow.

2.1.3. Introducing density estimation via flow functions

In practice, a function $y(x)$ is unknown. Also, function $y(x)$ can be very complex. This is because we want apply the model to samples from random variable x with arbitrary density function $\rho(x)$. For example, there might not exist a linear function $y(x)$ to map the samples from random variable x to samples from random variable y. To construct an approximation of unknown function $y(x)$ for arbitrary density function $\rho(x)$, we can view this function as an infinite composition of infinitesimal transformations [13]. For this, we introduce the flow $z = \phi_t(x)$ such that for $t = 0$ this is the identity map. Additionally, the flow will evolve in the direction of a real $y(x)$ that maps x to y. This is done by evolving $\phi_t(x)$ in a way that increases the Log-likelihood, which will be discussed in section 2.2. Since ϕ_t evolves in the direction of a real $y(x)$, the infinite composition of infinitesimal transformations maps the observations x to a variable y with known probability density $\mu(y)$. For individual transformations we use $\varphi_t(x)$. The composition of these smaller transformations together form the current flow $\phi_t(x)$. Therefore we introduce the following notation:

$$\phi_0(x) = x, \qquad \phi_t(x) = (\varphi_t \circ ... \circ \varphi_3 \circ \varphi_2 \circ \varphi_1 \circ \varphi_0(x)), \qquad y(x) = \lim_{t\to\infty} \phi_t(x) = (... \circ \varphi_3 \circ \varphi_2 \circ \varphi_1 \circ \varphi_0(x)). \tag{2.5}$$

The flow based model must allow two operations: sampling from the model, and evaluating the model's density. Sampling from the model means generating a sample of variable y via the distribution $\mu(y)$ and calculating from it a new sample of variable x using a map $y(x)^{-1}$. For this application of the flow we made the restriction on $y(x)$ that it has to be invertible. Since the composition of invertible functions is again invertible, the flow functions $\phi_t(x)$ are all invertible. Suppose for example that the current flow is $\phi_1(X) = \varphi_1(x)$. We will evolve the flow function by using the composition with a new flow φ_2 to construct:

$$\phi_2(x) = \varphi_2(x) \circ \phi_1(x) \tag{2.6}$$

We can then compute the inverse as follows:

$$(\varphi_2 \circ \phi_1)^{-1} = \phi_1^{-1} \circ \varphi_2^{-1}. \tag{2.7}$$

The second operation that the normalizing flow algorithm will allow is evaluating the model's density. This can be done from equation (2.1). This means that we need to compute the Jacobian determinant of $y(x)$. Since the composition of differentiable functions is again differentiable, the flow functions $\varphi_t(x)$ that are used are differentiable. We can compute the Jacobian determinant of the composition with the chain rule for the Jacobian and using that the determinant is a multiplicative image [1].

$$J_{\phi_2}(x) = J_{\varphi_2 \circ \phi_1}(x) = J_{\varphi_2}(\phi_1(x)) J_{\phi_1}(x) \tag{2.8}$$

This means that invertible and differentiable transformations have the property that they are composable: given ϕ_1 and φ_2, their composition $\varphi_2 \circ \phi_1$ is also invertible and differentiable. This is useful since "in consequence we can built complex transformations by composing multiple instances of simpler transformations, without compromising the requirements of invertibility and differentiability" [10]. The application of the flow will determine which of these operations will be used. For sampling via y it is beneficial to have $y^{-1}(x)$ to be efficient. Evaluating the density would focus more on efficiently computing $J_y(x)$. Depending on the application, certain constrains can be made on the maps $\varphi_t(x)$ [10]. This composition of flow functions is incorporated to the diagram of figure 2.1 in figure 2.5.

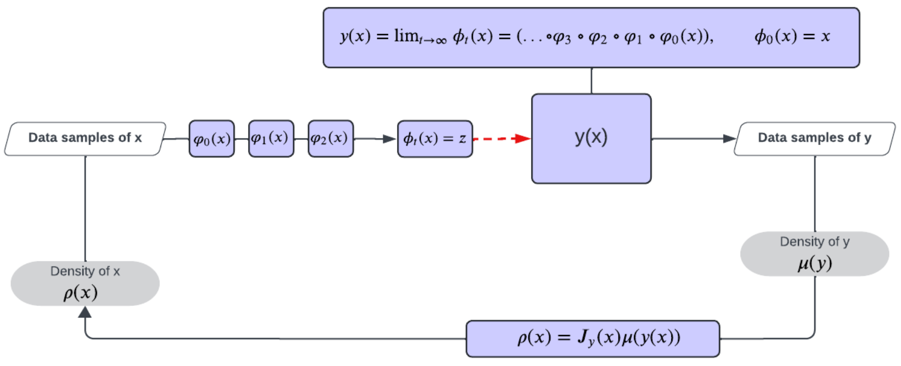

$y(x) = \lim_{t\to\infty} \phi_t(x) = (... \circ \varphi_3 \circ \varphi_2 \circ \varphi_1 \circ \varphi_0(x)),$ $\phi_0(x) = x$
Data samples of x
$\varphi_0(x)$
$\varphi_1(x)$
$\varphi_2(x)$
$\phi_t(x) = z$
y(x)
Data samples of y
Density of y $\mu(y)$
Density of x $\rho(x)$
$\rho(x) = J_y(x)\mu(y(x))$

Figure 2.5: $y(x)$ is approximated by the flow $\phi_t(x)$. The red dotted arrow shows that the flow will evolve in the direction of the real $y(x)$.

2.1.4. Density estimation for flow based models

The flow based structure is used to construct $y(x)$ for arbitrary densities $\rho(x)$. Next, we will look at how to use a flow based construction of $y(x)$ for estimating the density $\rho(x)$. Introduce $\tilde{\rho}_t(x)$ as the density approximation of the random variable x when using the current approximation of $y(x)$, with flow $\phi_t(x)$. This current density estimation of the density of random variable x, $\tilde{\rho}_t(x)$, is given by:

$$\tilde{\rho}_t(x) = J_{\phi_t}(x)\mu(\phi_t(x)). \tag{2.9}$$

This formula can be derived from change of variables in a similar way as equation (2.1). We will use (2.5) and that the Jacobian determinant of the identity map $J_{\phi_0}(x) = 1$ to find the expression for the estimated density after no time step and after infinite time steps:

$$\tilde{\rho}_0(x) = J_{\phi_0}(x)\mu(\phi_0(x)) = \mu(x), \quad \text{and} \tag{2.10}$$

$$\lim_{t\to\infty} \tilde{\rho}_t(x) = J_y(x)\mu(y(x)) \overset{(2.1)}{=} \rho(x). \tag{2.11}$$

The estimated density of x after no time steps is $\mu(y)$. We will show later that (by choosing the flow functions such that the log-likelihood increases) the estimated density, $\tilde{\rho}(x)$ evolves into the direction of the real density $\rho(x)$. This convergence is proved with the use of Kullback-Leibler divergence in section 2.83. This convergence is incorporated into the diagram of figure 2.5 in figure 2.6.

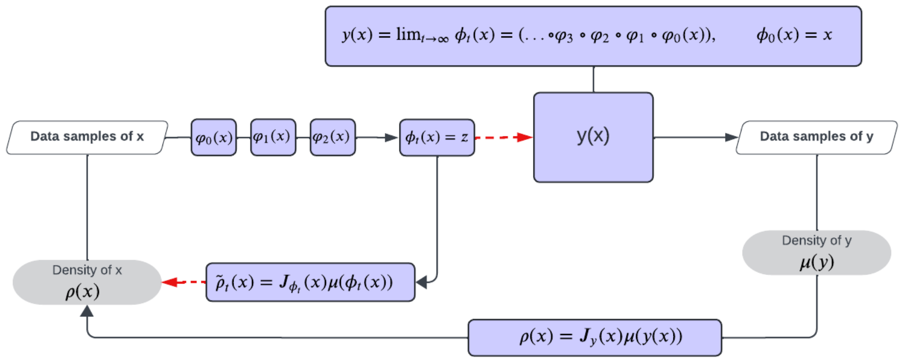

$y(x) = \lim_{t\to\infty} \phi_t(x) = (\ldots\circ\varphi_3 \circ \varphi_2 \circ \varphi_1 \circ \varphi_0(x)), \quad \phi_0(x) = x$
Data samples of x
$\varphi_0(x)$
$\varphi_1(x)$
$\varphi_2(x)$
$\phi_t(x) = z$
y(x)
Data samples of y
Density of y $\mu(y)$
Density of x $\rho(x)$
$\tilde{\rho}_t(x) = J_{\phi_t}(x)\mu(\phi_t(x))$
$\rho(x) = J_y(x)\mu(y(x))$

Figure 2.6: The estimated density $\tilde{\rho}(x)$ evolves into the direction of $\rho(x)$

We will now derive an expression for the density of the current flow of the samples. Recall that we use the variable z for the current flow: $z = \phi_t(x)$. Also equation (2.5) stated that

$$y(x) = \lim_{t\to\infty} \phi_t(x) = (\ldots \circ \varphi_3 \circ \varphi_2 \circ \varphi_1 \circ \varphi_0(x)).$$

Therefore the density of the current flow z converges to $\mu(y)$. This convergence is proved using Kullback-Leibler divergence in section 2.4.1. The density of z can be given by

$$\rho_t(z) = \frac{\rho(x)}{J_{\phi_t}(x)}. \tag{2.12}$$

This equation is the result of change of variables by similar arguments as (2.1). Now we have that $\rho(x)$ is the probability density function of the random variable x, and $\rho_t(z)$ is the probability density function of $z = \phi_t(x)$ [13]. Figure 2.7 gives a schematic overview of the basic principles of the normalizing flow algorithm. The red dotted arrows resemble convergence. $\tilde{\rho}_t(x)$ converges to the unknown density $\rho(x)$ and $\rho_t(z)$ converges to the density $\mu(y)$. This is called dual ascent of the Log-likelihood. Next, we will introduce the log-likelihood function and show how to use it to determine the flow functions $\phi_t(x)$.

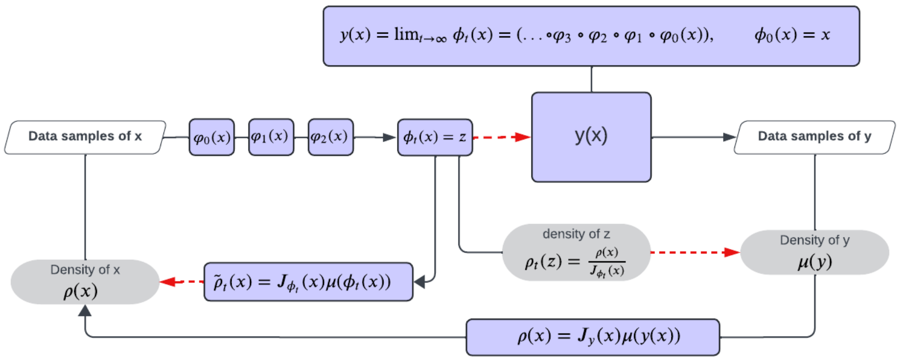

$y(x) = \lim_{t\to\infty} \phi_t(x) = (\ldots \circ \varphi_3 \circ \varphi_2 \circ \varphi_1 \circ \varphi_0(x)), \quad \phi_0(x) = x$
Data samples of x
$\varphi_0(x)$
$\varphi_1(x)$
$\varphi_2(x)$
$\phi_t(x) = z$
y(x)
Data samples of y
density of z
$\rho_t(z) = \frac{\rho(x)}{J_{\phi_t}(x)}$
Density of y
$\mu(y)$
Density of x
$\rho(x)$
$\tilde{\rho}_t(x) = J_{\phi_t}(x)\mu(\phi_t(x))$
$\rho(x) = J_y(x)\mu(y(x))$

Figure 2.7: Flow chart of the normalizing flow algorithm. The red dotted lines resemble convergence, the blue squares are functions, the white trapezoids are data samples and the gray ovals are density functions.

2.2. Log-likelihood function

The goal of the normalizing process is to find a good estimate of $\rho(x)$. To measure the quality of the estimated density at time t, $\tilde{\rho}_t(x)$, we use the log-likelihood of the sample x with respect to this density. The likelihood sums for all observed x^j the probability of observing this x^j when the data would have been extracted from the estimated probability density function $\tilde{\rho}_t(x)$. In this context it is clear that when $\tilde{\rho}_t(x)$ is a good estimate of the true underlying density $\rho(x)$, the log-likelihood is high. We will now introduce the log-likelihood more formally using definitions from [2].

Definition 2.2.1 (Likelihood function)**.** *[2] Let X be a random vector with probability density p_θ that depends on a parameter $\theta \in \Theta$. For x fixed, then the function*

$$\theta \to L(\theta; x) := p_\theta(x), \tag{2.13}$$

seen as a function $\phi \in \Theta$ (where Θ is the parameter space), is called the likelihood function. When $X = (X_1, X_2, \ldots, X_n)$ is a vector with independent identically distributed coordinates X_i. The density of X is the product $\prod_{i=1}^n p_\theta(x_i)$ of the marginal probability densities of $X_1, \ldots, X_n$, and the likelihood function is equal to

$$\theta \to L(\theta, x_1, \ldots, x_n) = \prod_{i=1}^n p_\theta(x_i), \tag{2.14}$$

where $p_\theta(x)$ is the marginal density of one X_i.

The logarithm of the likelihood is often taken since the logarithm is convenient for a maximization process. For example since we can write products inside the logarithm as sums. Since the Logarithm is a monotonically increasing function of its argument, maximizing the logarithm of a function is equivalent to maximizing the function itself.

Definition 2.2.2 (Log-likelihood)**.** *[2] If the observation $X = (X_1, \ldots, X_n)$ consists of independent, identically distributed subobservations X_i, then the log-likelihood is*

$$\theta \to \log L(\theta; x_1, \ldots, x_n) = \log \prod_{i=1}^n p_\theta(x_i) = \sum_{i=1}^n \log p_\theta(x_i). \tag{2.15}$$

We can use the definitions above for the log-likelihood function related to the flow $\phi_t(x)$ and corresponding estimated density $\tilde{\rho}_t(x)$

Definition 2.2.3 (Log-likelihood of the flow function)**.** *Let X be a random vector of m observations $x^1, x^2, \ldots, x^m$ with estimated probability density $\tilde{\rho}_t(x)$. $\phi_t(x)$ is a differentiable, invertible transformation of x. Let μ be a probability density function. When $\tilde{\rho}_t(x)$ is given by*

$$\tilde{\rho}_t(x) = J_{\phi_t}(x)\mu(\phi_t(x)),$$

then the log-likelihood of the flow function $\phi_t(x)$ is defined as

$$L[\phi_t] = \frac{1}{m} \sum_{j=1}^m \log(\tilde{\rho}_t(x^j)). \tag{2.16}$$

Lemma 2.2.4. *We can rewrite the log-likelihood using the definition of $\tilde{\rho}_t(x)$ to*

$$L[\phi_t] = \frac{1}{m}\sum_{j=1}^{m}\log(\tilde{\rho}_t(x^j)) = \frac{1}{m}\sum_{j=1}^{m}\log(J_{\phi_t}(x^j)\mu(\phi_t(x^j))) = \frac{1}{m}\sum_{j=1}^{m}(\log(J_{\phi_t}(x^j)) + \log(\mu(\phi_t(x^j)))). \tag{2.17}$$

Since the log-likelihood is a measure for the quality of $\tilde{\rho}_t(x)$ that increases for when $\tilde{\rho}_t(x)$ gets closer to the unknown density $\rho(x)$, we want to maximize the log likelihood of $z = \phi_t(x)$. Therefore, we will construct the flow $\phi_t(x)$ by following a direction of ascent of $L[\phi_t]$, such that the log-likelihood is always increasing,

$$\frac{dL[\phi_t]}{dt} \geq 0. \tag{2.18}$$

When this is the case, the map $y(x) = \lim_{t\to\infty}\phi_t(x)$ is a local maximizer of the log-likelihood function of the sample with respect to $\rho(x) = \tilde{\rho}_\infty(x)$ [13]. This maximum is then given by:

$$L[y] = L[\lim_{t\to\infty}\phi_t(x)] = \frac{1}{m}\sum_{j=1}^{m}\log(\lim_{t\to\infty}\tilde{\rho}_t(x^j)) \overset{(2.11)}{=} \frac{1}{m}\sum_{j=1}^{m}\log(J_y(x^j)\mu(y(x^j))) = \frac{1}{m}\sum_{j=1}^{m}(\log(J_y(x^j))+\log(\mu(y(x^j)))) \tag{2.19}$$

We will evolve $\phi_t(x)$ such that the Log-likelihood is increasing. One of the main ideas of normalizing flows is that the direction of ascent can be determined based on the current values of $z^j = \phi_t(x^j)$, without reference to the original sample, x^j [13]. Therefore, in section 2.3.1, $\phi_t(x)$ evolves in a way that increases the log-likelihood, but also ensures that we only need the current values of z.

2.3. Density estimation - the continuous case

In this section we assume there is an infinite amount of data samples available. This is called the continuous case, since in this case $\tilde{\rho}_t(x^j)$ is defined for all x^j in the domain of ρ_t. Therefore, for all x:

$$\tilde{\rho}_t(x) = \tilde{\rho}_t(x^j) \quad \text{for } x = x^j. \tag{2.20}$$

This case allows us to find an integral expression of the log-likelihood. This will help us to find an evolution of the flow ϕ_t into the direction of $y(x)$. To do this, we first seek the direction of ϕ_t that increases the log-likelihood the most (2.29). From this, we make a choice for the evolution of ϕ_t (2.43) that ensures that only the current values of $\phi_t(x)$ are needed to compute this evolution, instead of the original data points.

2.3.1. Log-likelihood in the continuous case

The case where infinite amount of data is available means that the number of observations, m, tends to infinity. We will use theorems from [7] to derive an expression for the log-likelihood in the continuous case:

Lemma 2.3.1 (The log-likelihood for the continuous case)**.** *When the number of observations of random variable x with density function $\rho(x)$ tends to infinity, and after the map $\phi_\infty(x)$ the density function is given by μ, then the log-likelihood of the flow function can be written as*

$$L_\rho[\phi_t] = \int (\log(J_{\phi_t}(x)) + \log(\mu(\phi_t(x))))\rho(x)\,dx. \tag{2.21}$$

Proof. (2.21) First we rewrite the log likelihood in the case where the number of observations tends to infinity:

$$L_\rho[\phi_t] = \lim_{m\to\infty}\frac{1}{m}\sum_{j=1}^{m}(\log(J_{\phi_t}(x^j)) + \log(\mu(\phi_t(x^j)))) \tag{2.22}$$

Now we will use the law of the subconscious statistician for discrete random variables, which can be found in the appendix A.6. Substitute $g(x) = \log(J_{\phi_t}(x)) + \log(\mu(\phi_t(x)))$ inside equation (A.5) , and use that we have m observations of random variable x. When we assume the observations are distinct we get $P(x = x^j) = 1/m$, and therefore

$$\mathbb{E}(g(x)) = \frac{1}{m}\sum_{j=1}^{m}\log(J_{\phi_t}(x^j)) + \log(\mu(\phi_t(x^j))) = L_\rho(\phi_t(x)). \tag{2.23}$$

Recall that the density of random variable x is $\rho(x)$. As the amount of observations tend to infinity, the data x can be viewed as a continuous random variable. This means we can apply the law of the subconscious statistician for continuous random variables that is stated in appendix A.7. This results in

$$\mathbb{E}(g(x)) = \int (\log(J_{\phi_t}(x)) + \log(\mu(\phi_t(x))))\rho(x)\,dx \tag{2.24}$$

Combining this gives the desired expression:

$$L_\rho(\phi_t(x)) = \int (\log(J_{\phi_t}(x)) + \log(\mu(\phi_t(x))))\rho(x)\,dx$$

□

Now that we have an expression for the log-likelihood in the continuous case, we will use it to decide on a direction to evolve ϕ_t in. We do this by first finding the direction with respect to ϕ_t in which the log-likelihood increases the most. For this we will first introduce the functional derivative:

Definition 2.3.2 (Functional derivative)**.** *[6] Let $F[\phi]$ be a mapping from a normed linear space of functions (a Banach space) $M = \{\phi(x) : x \in \mathbb{R}\}$ to the field of real or complex numbers, $F : M \to \mathbb{R}$ or $\mathbb{C}$. Just as could be expected for a derivative, $\frac{\delta F[\phi]}{\delta\phi(x)}$ tells how the value of the functional changes if the function $\phi(x)$ is changed at the point x. The functional derivative is defined as*

$$\delta F[\phi] = \int dx \frac{\delta F[\phi]}{\delta\phi(x)}\delta\phi(x). \tag{2.25}$$

Equivalently, we can write the functional derivative as the limit of divided differences. To see this we construct a variation of the the function $\phi(x)$ which is localized at the point y having strength ϵ:

$$\delta\phi(x) = \epsilon\delta(x-y). \tag{2.26}$$

Inserted into (2.25) *this leads to*

$$\delta F[\phi] = F[\phi + \epsilon\delta(x-y)] - F[\phi] = \int dx \frac{\delta F[\phi]}{\delta\phi(x)}\epsilon\delta(x-y) = \epsilon\frac{\delta F}{\delta\phi(y)}, \tag{2.27}$$

Now we use that the derivative is the limit of vanishing ϵ of the divided differences:

$$\frac{\delta F[\phi]}{\delta\phi(y)} = \lim_{\epsilon\to 0}\frac{F[\phi+\epsilon\delta(x-y)] - F[\phi]}{\epsilon}. \tag{2.28}$$

We can now use this definition to calculate the functional derivative of the log-likelihood function:

Proposition 2.3.3. *Let $L(\phi_t(x))$ be the log-likelihood of the flow function. The functional derivative [6] of the log-likelihood of the flow function is given by*

$$\frac{\delta L_\rho}{\delta\phi_t} = J_{\phi_t}(x)\left(\frac{\nabla_z\mu(z)}{\mu(z)}\rho_t(z) - \nabla_z\rho_t(z)\right), \tag{2.29}$$

where $z = \phi_t(x)$ and $\rho_t(z)$ is as described in equation (2.12)*. Recall from section 2.1 that $\rho(x)$ is the probability density function of the variable x, then $\rho_t(z)$ is the probability density function of $z = \phi_t(x)$.*

Proof. We will prove equation (2.29) in the one dimensional case. The statement is also used for higher dimensions [13]. To prove (2.29) we will use the following expression to determine the functional derivative [6]:

$$\lim_{\epsilon\to 0}\delta L_\rho[\phi_t] = \lim_{\epsilon\to 0}\big(L_\rho(\phi_t(x)+\epsilon\eta(x)) - L_\rho(\phi_t(x))\big) = \int\frac{\delta L_\rho}{\delta\phi_t}\epsilon\eta(x)\,dx. \tag{2.30}$$

In this equation $L(\phi_t(x)+\epsilon\eta(x))$ resembles the log-likelihood function after a small perturbation, $\epsilon\eta(x)$, in $\phi_t(x)$, where $\eta(x)$ is assumed to vanish at the boundary (in infinity). Recall the equation of the log-likelihood (equation (2.21)):

$$L_\rho(\phi_t) = \int (\log(J_{\phi_t}(x)) + \log(\mu(\phi_t(x))))\rho(x)\,dx.$$

In the one dimensional case the Jacobian determinant is the derivative. Therefore (2.21) becomes

$$\begin{aligned} L_\rho(\phi_t) &= \int \left(\log\left(\frac{d\phi_t(x)}{dx}\right) + \log(\mu(\phi_t(x)))\right)\rho(x)\,dx \\ &= \int \left(\log\left(\frac{d\phi_t(x)}{dx}\right)\right)\rho(x)\,dx + \int (\log(\mu(\phi_t(x))))\rho(x)\,dx. \end{aligned} \tag{2.31}$$

The perturbation results in a perturbed log-likelihood given by

$$\begin{aligned} L_\rho(\phi_t(x)+\epsilon\eta(x)) &= \int (\log(J_{\phi_t(x)+\epsilon\eta(x)}(x)) + \log(\mu(\phi_t(x)+\epsilon\eta(x))))\rho(x)\,dx \\ &= \int (\log(J_{\phi_t(x)+\epsilon\eta(x)}(x))\rho(x)\,dx + \int \log(\mu(\phi_t(x)+\epsilon\eta(x))))\rho(x)\,dx \end{aligned} \tag{2.32}$$

For the one dimensional case, (2.32) becomes

$$L_\rho(\phi_t(x)+\epsilon\eta(x)) = \int \log\left(\frac{d\phi_t(x)}{dx} + \epsilon\frac{d\eta(x)}{dx}\right)\rho(x)\,dx + \int (\log(\mu(\phi_t(x)+\epsilon\eta(x))))\rho(x)\,dx \tag{2.33}$$

Now consider inside equation (2.33) the perturbed target density,

$$\mu(\phi_t(x)) \to \mu(\phi_t(x)+\epsilon\eta(x)).$$

The perturbed target density can be estimated by it's first order Taylor polynomial around $z = \phi_t(x)$:

$$\mu(\phi_t(x)+\epsilon\eta(x)) \approx \mu(z) + \epsilon\eta(x)\frac{d\mu(z)}{dz} \tag{2.34}$$

To further simplify (2.33) we will rewrite the formula of the logarithm of the perturbed target density (equation (2.34)). For readability we introduce $\nabla_z\mu$ as a notation for $\frac{d\mu(z)}{dz}$,

$$\begin{aligned} \log\big(\mu(\phi_t(x)) + \epsilon\eta(x)\big) &\approx \log\big(\mu(\phi_t(x)) + \epsilon\eta(x)\nabla_z\mu(\phi_t(x))\big) \\ &= \log(\mu(\phi_t(x)))\left(1 + \frac{\epsilon\eta(x)\nabla_z\mu(\phi_t(x))}{\mu(\phi_t(x))}\right) \\ &= \log(\mu(\phi_t(x))) + \log\left(1 + \frac{\epsilon\eta(x)\nabla_z\mu(\phi_t(x))}{\mu(\phi_t(x))}\right) \\ &\approx \log(\mu(\phi_t(x))) + \frac{\epsilon\eta(x)\nabla_z\mu(\phi_t(x))}{\mu(\phi_t(x))}. \end{aligned} \tag{2.35}$$

In the last step we used that for small a,

$$\log(1+a) \approx a. \tag{2.36}$$

We can now use this to find a final form for the log-likelihood of the perturbed flow:

$$L_\rho(\phi_t(x)+\epsilon\eta(x)) = \int \log\left(\frac{d\phi_t(x)}{dx} + \epsilon\frac{d\eta(x)}{dx}\right)\rho(x)\,dx + \int \left(\log(\mu(\phi_t(x))) + \frac{\epsilon\eta(x)\nabla_z\mu(\phi_t(x))}{\mu(\phi_t(x))}\right)\rho(x)\,dx. \tag{2.37}$$

Now, we will calculate the form where we can recognize the the functional derivative of equation 2.30. For this, we calculate the difference of the log-likelihood and the perturbed log-likelihood. We can simplify the expression by using linearity of the integral and calculation rules for the logarithm. For readability and since this proof is for one dimension we use $J_{\phi_t}(x)$ for $\frac{d\phi_t(x)}{dx}$, and $J_\eta(x)$ for $\frac{d\eta_t(x)}{dx}$. This results in

$$\begin{aligned} L_\rho(\phi_t(x)+\epsilon\eta(x)) - L_\rho(\phi_t(x)) &= \int \left(\log\left(1 + \epsilon\frac{J_\eta(x)}{J_{\phi_t}(x)}\right) + \frac{\epsilon\eta(x)\nabla_z\mu(\phi_t(x))}{\mu(\phi_t(x))}\right)\rho(x)\,dx \\ &\overset{(2.36)}{=} \epsilon\int \frac{J_\eta(x)}{J_{\phi_t}(x)}\rho(x)\,dx + \int \frac{\epsilon\eta(x)\nabla_z\mu(\phi_t(x))}{\mu(\phi_t(x))}\rho(x)\,dx \end{aligned} \tag{2.38}$$

We will now first simplify the first integral with integration by parts, which can be found in appendix A.5. Since this proof is for $\mathbb{R}^1$, for readability we introduce the notation $\nabla_x = \frac{d}{dx}$. We will also use the formula for $\rho_t(z)$ from equation (2.12). Additionally, we use the assumption that $\eta(x)$ vanishes at the boundary. The simplification we find is

$$\int \frac{J_\eta(x)}{J_{\phi_t}(x)}\rho(x)\,dx = 0 - \int \nabla_x\left(\frac{\rho(x)}{J_{\phi_t}(x)}\right)\eta(x)\,dx = -\int \nabla_x(\rho_t(z))\eta(x)\,dx \tag{2.39}$$

Now we will use the following expression, where we again use the notation for $\nabla_z = \frac{d}{dz}$:

$$\nabla_x = \frac{d}{dx} = \frac{dz}{dx}\frac{d}{dz} = \frac{d\phi_t}{dx}\frac{d}{dz} = J_{\phi_t}(x)\frac{d}{dz} = J_{\phi_t}(x)\nabla_z. \tag{2.40}$$

Substituting this result into (2.39) results in the expression

$$\int \frac{J_\eta(x)}{J_{\phi_t}(x)}\rho(x)\,dx = -\int J_{\phi_t}(x)\nabla_z(\rho_t(z))\eta(x)\,dx \tag{2.41}$$

Substitute this back in the expression (2.38) to find

$$\begin{aligned} L_\rho(\phi_t(x)+\epsilon\eta(x)) - L_\rho(\phi_t(x)) &= \int\left(-J_{\phi_t}(x)\nabla_z(\rho_t(z))\eta(x) + \frac{\epsilon\eta(x)\nabla_z\mu(\phi_t(x))}{\mu(\phi_t(x))}\right)\rho(x)\,dx \\ &= \int\left(-J_{\phi_t}(x)\nabla_z(\rho_t(z))\eta(x) + \frac{\epsilon\eta(x)\nabla_z\mu(z)}{\mu(z)}\right)\rho(x)\,dx \\ &\overset{(2.12)}{=} \int\left(J_{\phi_t}(x)\left(\frac{\nabla_z\mu(z)}{\mu(z)}\rho_t(z) - \nabla_z\rho_t(z)\right)\right)\epsilon\eta(x)\,dx. \end{aligned} \tag{2.42}$$

In this form we finally recognize the form as in equation (2.30)

□

2.3.2. Evolution of the flow function ϕ_t

Equation (2.29) is the gradient of the log-likelihood function, and therefore represents the direction of $\phi_t(x)$ in which the log likelihood will increase the most. In [13] it is suggested to evolve $\phi_t(x)$ according to

$$\dot{\phi}_t(x) = u_t(\phi_t(x)) \tag{2.43}$$

where

$$u_t(z) = \frac{\nabla_z\mu(z)}{\mu(z)}\rho_t(z) - \nabla_z\rho_t(z). \tag{2.44}$$

Notice that $u_t(z)$ is the gradient of the log-likelihood function divided by $J_{\phi_t}(x)$. This guarantees that $\phi_t(x)$ evolves in an ascending direction of the log-likelihood function. Although choosing the gradient as a whole for the direction would be the steepest ascent, we will give a motivation for this choice of $u_t(z)$. One important characterization for normalizing flows is that only the current flow of the samples, $z = \phi_t(x)$ is needed for the computation of the next flow. When we use the current flow of the samples z as input and find the gradient of the log-likelihood from this, and use that we can always apply the identity map, we find the following expression:

$$\left.\frac{\delta L_\rho[\varphi\circ\phi_t]}{\delta\varphi}\right|_{\varphi=\text{id}} = \frac{\nabla_z\mu(z)}{\mu(z)}\rho_t(z) - \nabla_z\rho_t(z), \tag{2.45}$$

Thus,(2.44) corresponds to the evolution by steepest ascent on a modified log-likelihood function in which, at time t, where at time t we use $z = \phi_t(x)$ as the current flow rather than the original x [13].

2.3.3. Evolution of the density function ρ_t

The flow $\phi_t(x)$ evolves over time in the direction of $y(x)$. Now let us look at the evolution of $\rho_t(z)$. Recall that $\rho_t(z)$ is the probability density of the variable $z = \phi_t(x)$ given that x is distributed according to $\rho(x)$[13]. We want the distribution of the transformed data, $\rho_t(z)$, to converge to the target distribution μ. Therefore, we are interested in the evolution of the density $\rho_t(z)$ over time. To formulate an expression for the evolution of $\rho_t(z)$, $\frac{\partial\rho_t}{\partial t}$, we use that this satisfies the differential form of the continuity equation [14]. The continuity equation describes the transport of for example particles. To apply it to a normalizing flow, we interpret the current samples of the flow $z = \phi_t(x)$ as particles. In differential form the continuity equation is given by:

Lemma 2.3.4 (Differential form of the continuity equation)**.** *When we have that*

- *$\rho_t(z)$ is the density function of continuous random variable z, that is interpreted as a moving particle, $z = \phi_t(x)$, with density $\rho_t(z) = \frac{\rho(x)}{J_{\phi_t}(x)}$, where $\rho(x)$ the density function of continuous random variable x. The evolution of the particles $z = \phi_t(x)$ is according $u_t(z)$ and $\rho_t(\phi_t(x))$ is differentiable with respect to t.*
- *f is the flux density of z. $f = \rho_t u_t$.*
- *σ is the generation of particles z per unit volume per unit time.*

Then the differential form of the continuity equation [14] says that

$$\frac{\partial \rho_t(z)}{\partial t} + \nabla \cdot f = \sigma. \tag{2.46}$$

To understand why the function satisfies the continuity equation, we can interpret the situation in terms of a particle flow. As the particles x^j flow from x to $y(x)$ via $\phi_t(x)$, their probability density ρ_t evolves from the (unknown) initial ρ toward the target μ. We can view ρ_t as a conserved quantity that cannot be created or destroyed, and therefore $\sigma = 0$. Intuitively this makes sense because the total probability is always equal to 1: no particles are destroyed or created during the process. We now substitute $f = \rho_t u_t$ in the equation (2.46):

$$\frac{\partial \rho_t(z)}{\partial t} + \nabla \cdot (\rho_t u_t) = 0. \tag{2.47}$$

We will now use equation (2.47) to derive an explicit expression for $\frac{\partial \rho_t(z)}{\partial t}$ as in [13] by substituting $u_t(z)$ from equation (2.44).

Lemma 2.3.5 (Explicit expression for the evolution of the density of the flow)**.** *Let $\rho_t(z)$ be the density function of continuous random variable z, that is interpreted as a moving particle, $z = \phi_t(x)$, with density $\rho_t(z) = \frac{\rho(x)}{J_{\phi_t}(x)}$, where $\rho(x)$ the density function of continuous random variable x. let μ be a density function. Let $\phi_t(x)$ evolve according $u_t(z)$ from equation* (2.44)*. The evolution of the density of the flow z is then given by*

$$\frac{\partial \rho_t(z)}{\partial t} = \nabla \cdot \left(\left(\nabla_z \rho_t(z) - \frac{\nabla_z \mu}{\mu} \rho_t(z) \right) \rho_t(z) \right). \tag{2.48}$$

Proof. Substitute $u(z)$ from equation (2.44) into the continuity equation (2.47)

$$\frac{\partial \rho_t}{\partial t} + \nabla \cdot \left(\rho_t \frac{\nabla_z \mu(z)}{\mu(z)} \rho_t(z) - \rho_t \nabla_z \rho_t(z) \right) = 0. \tag{2.49}$$

Therefore

$$\begin{aligned} \frac{\partial \rho_t}{\partial t} &= \nabla \cdot \big(\rho_t(z) \nabla_z \rho_t(z)\big) - \nabla \cdot \left(\rho_t(z) \nabla_z \mu(z) \frac{\rho_t(z)}{\mu(z)} \right) \\ &= \nabla \cdot \big(\rho_t(z) \nabla_z \rho_t(z)\big) - \left(\nabla_z \rho_t(z) \left(\nabla_z \mu(z) \frac{\rho_t(z)}{\mu(z)} \right) + \rho_t(z) \nabla \cdot \left(\nabla_z \mu(z) \frac{\rho_t(z)}{\mu(z)} \right) \right) \end{aligned} \tag{2.50}$$

Recall that the goal is to proof equation (2.48). We will do this by rewriting equation (2.48) in the form of (2.50). We rewrite this using the product rule and use that $\nabla \cdot (\rho_t \nabla \rho_t) = \nabla \rho_t \nabla \rho_t + \rho_t \nabla \cdot (\nabla \rho_t)$ holds also by the product rule. This gives us the desired expression of equation

$$\begin{aligned} \nabla \cdot \left(\left(\nabla_z \rho_t(z) - \frac{\nabla_z \mu}{\mu} \rho_t(z) \right) \rho_t(z) \right) &= \nabla_z \rho_t(z) \left(\nabla_z \rho_t(z) - \frac{\nabla_z \mu}{\mu} \rho_t(z) \right) + \rho_t(z) \nabla \cdot \left(\nabla_z \rho_t(z) - \frac{\nabla_z \mu}{\mu} \rho_t(z) \right) \\ &= \nabla_z \rho_t(z) \nabla_z \rho_t(z) + \rho_t(z) \nabla \cdot (\nabla_z \rho_t(z)) - \nabla_z \rho_t(z) \left(\frac{\nabla_z \mu}{\mu} \rho_t(z) \right) - \rho_t(z) \left(\frac{\nabla_z \mu}{\mu} \rho_t(z) \right) \\ &= \nabla \cdot \big(\rho_t(z) \nabla_z \rho_t(z)\big) - \left(\nabla_z \rho_t(z) \left(\nabla_z \mu(z) \frac{\rho_t(z)}{\mu(z)} \right) + \rho_t(z) \nabla \cdot \left(\nabla_z \mu(z) \frac{\rho_t(z)}{\mu(z)} \right) \right) \\ &= \text{equation (2.50)} \end{aligned} \tag{2.51}$$

□

We want the distribution of the transformed data, $\rho_t(z)$ to converge to the target distribution μ. The first step in showing this, is proving that the desired target μ is a stationary solution of the partial differential equation (2.48). The consequence of this is that when ρ_t is equal to the target μ, it will stay at the target. Therefore we rewrite equation (2.48) in the form where we can recognize that $\rho_t(z) = \mu(z)$ is a stationary solution:

Proposition 2.3.6. *Let $\rho_t(z)$ be the density function of continuous random variable z, that is interpreted as a moving particle, $z = \phi_t(x)$, with density $\rho_t(z) = \frac{\rho(x)}{J_{\phi_t}(x)}$, where $\rho(x)$ the density function of continuous random variable x. let μ be a density function. When $\phi_t(x)$ evolves according $u_t(z)$ from equation* (2.44). *The evolution of the density of the flow z can be written as*

$$\frac{\partial \rho_t}{\partial t} = \nabla \cdot \left(\mu(z)^2 \nabla \left(\frac{1}{2} \left(\frac{\rho_t(z)}{\mu(z)} \right)^2 \right) \right) \tag{2.52}$$

Proof. We will substitute $v = \frac{\rho_t}{\mu}$ into equation (2.48), since then

$$\rho_t = \mu v.$$

Now taking the gradient. For the notation we use ∇ for ∇_z. This gives

$$\nabla \rho_t = \nabla(\mu v) = \mu \nabla v + v \nabla \mu.$$

This gives the following expressions:

$$\begin{aligned} \rho_t \nabla \rho_t &= \mu v (\mu \nabla v + v \nabla \mu) \\ &= \mu^2 v \nabla v + \mu v^2 \nabla \mu \end{aligned} \tag{2.53}$$

$$\frac{\nabla \mu}{\mu} \rho_t^2 = \frac{\nabla \mu}{\mu} (\mu v)^2 = \mu v^2 \nabla \mu. \tag{2.54}$$

We will now substitute these expressions into equation (2.48):

$$\begin{aligned} \frac{\partial \rho_t}{\partial t} &\overset{(2.48)}{=} \nabla \cdot \left(\left(\nabla \rho_t - \frac{\nabla \mu}{\mu} \rho_t \right) \rho_t \right) \\ &= \nabla \cdot \left((\nabla \rho_t) \rho_t - \frac{\nabla \mu}{\mu} \rho_t^2 \right) \\ &= \nabla \cdot \left(\mu^2 v \nabla v + \mu v^2 \nabla \mu - \mu v^2 \nabla \mu \right) \\ &= \nabla \cdot (\mu^2 v \nabla v) \end{aligned} \tag{2.55}$$

By the chain rule

$$v \nabla v = \nabla \left(\frac{1}{2} v^2 \right). \tag{2.56}$$

Substitute in equation (2.55) gives the form we desire in equation (2.52):

$$\frac{\partial \rho_t}{\partial t} = \nabla \cdot \left(\mu^2 \nabla \left(\frac{1}{2} v^2 \right) \right) = \nabla \cdot \left(\mu^2 \nabla \left(\frac{1}{2} \left(\frac{\rho_t}{\mu} \right)^2 \right) \right) \tag{2.57}$$

□

From the form in equation (2.52) we can conclude that $\rho_t = \mu$ is a stationary solution:

Lemma 2.3.7. *$\rho_t = \mu$ is a stationary solution for $\frac{\partial \rho_t}{\partial t}$.*

Proof. A stationary solution is a constant solution. Therefore, the time derivative must be 0. Substituting $\rho_t = \mu$ into (2.52) gives

$$\frac{\partial \rho_t}{\partial t} = \; = \nabla \cdot \left(\mu^2 \nabla \left(\frac{1}{2} \left(\left(\frac{\rho_t}{\mu} \right)^2 \right) \right) \right) = \nabla \cdot \left(\mu^2 \nabla \left(\frac{1}{2} \left(\frac{\mu}{\mu} \right)^2 \right) \right) = \nabla \cdot \left(\mu^2 \nabla \left(\frac{1}{2} \right) \right) = \nabla \cdot \left(\mu^2 0 \right) = 0 \tag{2.58}$$

□

In the section 2.4, Kullback-Leibler divergence will be used to show that for all initial probability densities ρ_0 converge to μ. In the next section, the evolution of $\rho_t(x)$ we found in equation (2.52) will be applied to the one-dimensional case, where x, y and z are one dimensional.

2.3.4. Example of the continuous case

Equation (2.52) provides a way to evolve ρ_t towards the target μ. To showcase that this evolution works, we will find the numerical solution of the partial differential equation by implementing a finite difference method. In this example we will look at the one dimensional case. Therefore, the divergence operator has the same result as the gradient, which in the one dimensional case is the derivative. We will use central differences to estimate the derivatives. The central difference method is a numerical method to approximate derivatives. In the one dimensional case, equation (2.52) becomes

$$\frac{\partial \rho_t}{\partial t} = \frac{\partial}{\partial z} \cdot \left(\mu^2 \frac{\partial}{\partial z} \left(\frac{1}{2} \left(\left(\frac{\rho_t}{\mu} \right)^2 \right) \right) \right) \tag{2.59}$$

Definition 2.3.8. *For a function $f(z)$ and a small step size Δz, the central difference approximation for the first derivative at a point z_0 is given by:*

$$\frac{d}{dz} f(z_0) \approx \frac{f(z_0 + \Delta z) - f(z_0 - \Delta z)}{2\Delta z} \tag{2.60}$$

In the case of formula (2.52),

$$f(z) = \mu^2 \frac{\partial}{\partial z} \left(\frac{1}{2} \left(\left(\frac{\rho_t}{\mu} \right)^2 \right) \right) \tag{2.61}$$

To approximate the derivative inside the function $f(z)$ we will apply the central differences again, but to

$$g(z) = \frac{1}{2} \left(\left(\frac{\rho_t}{\mu} \right)^2 \right) \tag{2.62}$$

From this approximation and formula (2.52), we can use finite differences to estimate $\rho_t(z)$:

$$\rho_t(z) \approx \rho_{t-1}(z) + \Delta t \frac{\partial \rho_t}{\partial t} \tag{2.63}$$

Figure 2.8 is an example that shows the evolution from a uniform distribution on [-0.2, 0.2] towards the target Gaussian using central differences. Figure 2.9 shows the evolution from an exponential distribution with $\lambda = 1$ towards the target Gaussian. These examples use time steps, $\Delta t = 0.001$ and $\Delta z = 0.1$. The code can be found in appendix B.1 and B.2.

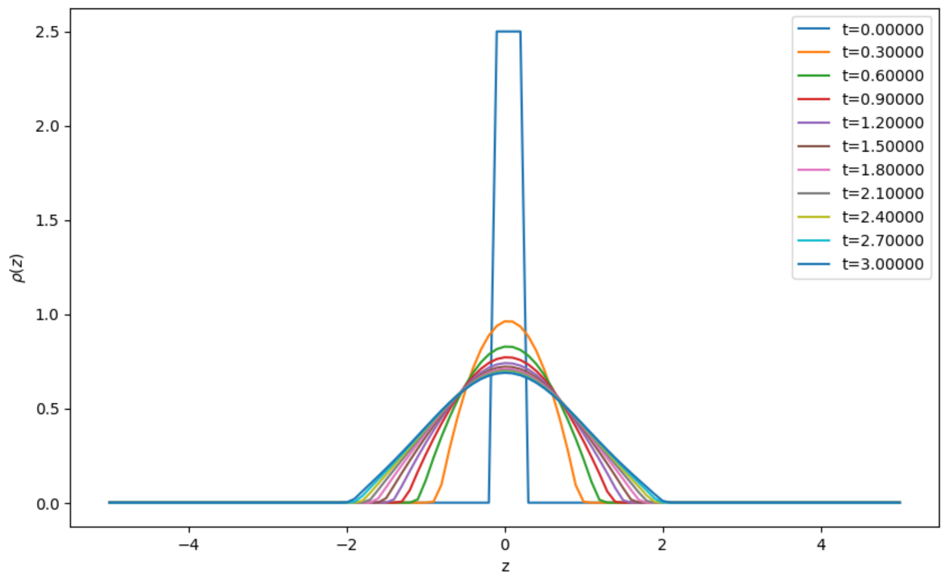


Figure 2.8: Numerical solution of (2.52), displaying the flow from a uniform distribution on [-0.2, 0.2] evolving towards the target Gaussian. This example uses central differences with delta x = 0.1 and delta t = 0.001.

In these examples we see that the density of the current flow $\rho_t(x)$, which in this plot is shown as $\rho(z)$, converges towards the target μ. To prove convergence for general initial initial probability density function $\rho(x)$ we will introduce Kullback-Leibler divergence in the next section.

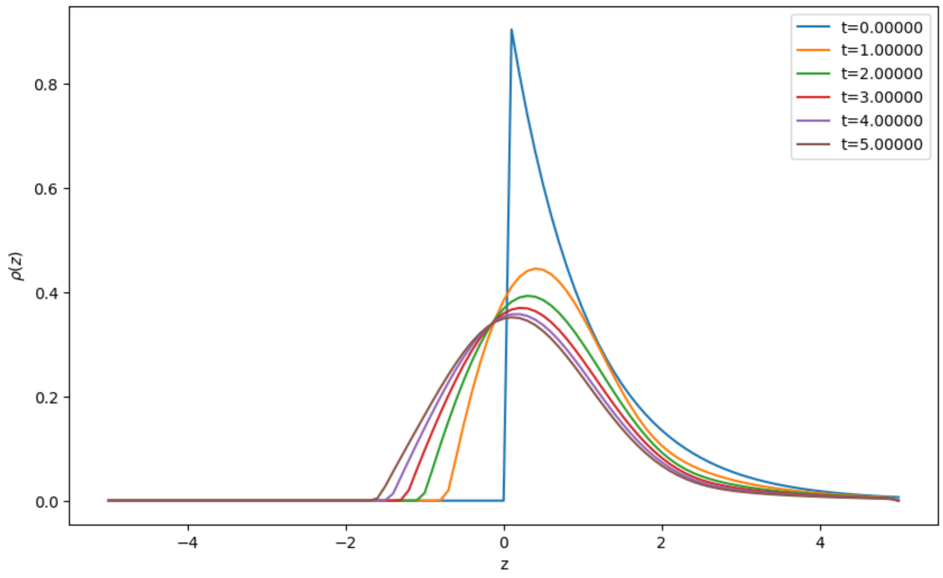


Figure 2.9: Numerical solution of (2.52), displaying the flow from an exponential distribution with $\lambda = 1$ evolving towards the target Gaussian. This example uses central differences with delta x = 0.1 and delta t = 0.001.

2.4. Kullback-Leibler divergence

This chapter aims to show convergence of the solution of equation (2.52) towards the target μ. We consider ρ_t to be the estimation of μ at time t. The Kullback-Leibler divergence, $D_{kl}(\mu, \rho_t)$, is a measure for dissimilarity of the two distributions μ and ρ_t [3]. To use the Kullback-Leibler divergence for the convergence of the solution of equation (2.52), we will also need the Kullback-Leibler divergence of $\tilde{\rho}_t$ and ρ.

Definition 2.4.1 (Kullback-Leibler divergence)**.** *[3] Let X be a continuous random variable. The formula of the Kullback-Leibler divergence for density functions $p(x)$ and $q(x)$ is given by*

$$\begin{aligned} D_{kl}(p,q) &= -\int p(x)\ln q(x)\,dx - \left(-\int p(x)\ln p(x)\,dx\right) \\ &= -\int p(x)\ln\left\{\frac{q(x)}{p(x)}\right\}dx. \end{aligned} \tag{2.64}$$

In the first part of this chapter (2.4.1), we will work towards this result by showing that minimizing $D_{kl}(\mu, \rho_t)$ is the same as maximizing $L_\rho(\phi_t(x))$ (lemma (2.4.3)). In lemma 2.4.7, we will show that $D_{kl}(\mu, \rho_t) \geq 0$, with equality if and only if $\rho_t = \mu$. Then, in 2.4.8 we show that $\frac{d}{dt}D_{KL}(\mu_t, \rho_t) \leq 0$. We will use this to show that $D_{kl}(\mu, \rho_t)$ does not stop decreasing until $\rho_t(z)$ reaches the target $\mu(z)$. In equation (2.58) it was shown that μ is a stationary solution. Therefore once ρ_t reaches the target μ, it will stay there.
However, the Kullback-Leibler divergence of μ and ρ_t is not enough for the proof of convergence. In section 2.4.2 we will look at the Kullback-Leibler divergence of ρ and $\tilde{\rho}_t$, $D_{kl}(\rho, \tilde{\rho}_t)$. Recall that $\tilde{\rho}_t(x)$ is the current estimation of the density $\rho(x)$. We will use $D_{kl}(\rho, \tilde{\rho}_t)$ to give a more general argument to prove convergence of the solution of equation (2.52) to μ.

2.4.1. Kullback-Leibler divergence of ρ_t and μ

We will use definition (2.64) to find the Kullback-Leibler divergence of μ and ρ_t, $D_{kl}(\mu, \rho_t)$:

Lemma 2.4.2.

$$D_{kl}(\mu, \rho_t) = \int \mu(z)\log\left(\frac{\mu(z)}{\rho_t(z)}\right)dz. \tag{2.65}$$

Proof.

$$\begin{aligned} D_{kl}(\mu,\rho_t) &:= -\int \mu(z)\log(\rho_t(z))\,dz - \left(-\int \mu(z)\log(\mu(z))\,dz\right) \\ &= -\int \mu(z)\log\left(\frac{\rho_t(z)}{\mu(z)}\right)dz \\ &= \int \mu(z)\log\left(\frac{\mu(z)}{\rho_t(z)}\right)dz. \end{aligned} \tag{2.66}$$

□

Suppose the data x is being generated from a distribution $\mu(z)$. The density $\rho_t(z)$ aims to model this distribution. In chapter 2.2 we suggest that finding this density function can be done by maximizing the log-likelihood. We will now show that minimizing the KL divergence with respect to t is the same as maximizing the log likelihood of $\phi_t(x)$. Therefore, one way to determine $\phi_t(x)$ such that eventually $\rho_t(z) \approx \mu(z)$ that is suggested in [13] is to minimize the Kullback-Leibler divergence between $\mu(z)$ and $\rho_t(z)$ with respect to t.

Lemma 2.4.3. *Minimizing the $D_{KL}(\mu_t,\rho_t)$ has the same result as maximizing $L_\rho[\phi_t]$.*

Proof. Finding a maximum $D_{KL}(\mu_t,\rho_t)$ can be done by looking at where the time derivative is equal to 0. We use linearity of the derivative and that the second integral is independent of t to find

$$\begin{aligned} \frac{d}{dt}D_{kl}(\mu,\rho_t) &:= \frac{d}{dt}\left(-\int \mu(z)\log(\rho_t(z))\,dz\right) - \frac{d}{dt}\left(-\int \mu(z)\log(\mu(z))\,dz\right) \\ &= -\frac{d}{dt}\left(-\int \mu(z)\log(\rho_t(z))\,dz\right) = \frac{d}{dt}L_\mu[\phi_t^{-1}] \end{aligned} \tag{2.67}$$

This last step follows from the invertibility of the flow function and lemma 2.3.1: the inverse flow function maps random variable z with density $\mu(z)$, and after this map the density is given by $\rho_t(z)$. This can then be applied to the lemma 2.3.1. Since $L_\mu[\phi_t^{-1}]$ is maximal for $\phi_t(x) = y(x)$ and so is $L_\rho[\phi_t]$, we get that minimizing $D_{KL}(\mu_t,\rho_t)$ has the same result as maximizing $L_\rho[\phi_t]$. □

We want to use the Kullback-Leibler divergence as a measure of the dissimilarity of the two distributions $\mu(z)$ and $\rho_t(z)$[13]. We will now work towards showing that the Kullback-Leibler divergence of μ and ρ_t does not stop decreasing until $\rho_t(z)$ reaches it's target $\mu(z)$ [13]. To proof this, we first introduce some useful lemmas about the Kullback-Leibler divergence.

Lemma 2.4.4.

$$D_{kl}(\mu,\rho_t) = 0 \iff \mu(z) = \rho_t(z) \quad \textit{for all } z, \tag{2.68}$$

Proof. We will prove both ways:

1. $D_{kl}(\mu,\rho_t) = 0 \quad \Rightarrow \quad \mu(z) = \rho_t(z)$, since

$$\begin{aligned} \int \mu(z)\log\left(\frac{\mu(z)}{\rho_t(z)}\right)dz = 0 &\implies \log\left(\frac{\mu(z)}{\rho_t(z)}\right) = 0 \\ &\implies \frac{\mu(z)}{\rho_t(z)} = 1 \\ &\implies \mu(z) = \rho_t(z) \end{aligned} \tag{2.69}$$

2. $\mu(z) = \rho_t(z) \quad \Rightarrow \quad D_{kl}(\mu,\rho_t) = 0$, since then for all z, $\log\left(\frac{\mu(z)}{\rho_t(z)}\right) = \log(1) = 0$

□

We will now work toward proving $D_{kl}(\mu,\rho_t) \geq 0$. For this, we first need to show $f(y) = -\log(y)$ is a convex function.

Definition 2.4.5 (Convex function)**.** *[3] A function $f(y)$ is said to be convex if it has the property that every chord lies on or above the function. This can be expressed as follows: $\forall y \in [a,b], \forall 0 \leq \lambda \leq 1$:*

$$f(\lambda a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda)f(b). \tag{2.70}$$

This is equivalent to the requirement that the second derivative of the function be everywhere positive. We call the function $f(y)$ strictly convex if equality only holds for $\lambda = 0$ and $\lambda = 1$.

Lemma 2.4.6. *$f(y) = -\log(y)$ is a strictly convex function*

Proof. This follows from taking the second derivative:

$$\frac{d^2}{dy^2}(-\log(y)) = \frac{1}{y^2} >= 0 \text{ for all } y, \text{ with equality only for } \lim_{y\to\pm\infty}. \tag{2.71}$$

□

Now we can use this result to apply Jensen's inequality for continuous random variables (which can be found in A.4) in the proof of the following:

Lemma 2.4.7.

$$D_{kl}(\mu, \rho_t) \geq 0, \quad \textit{with equality if and only if } \mu = \rho_t \tag{2.72}$$

Proof. We will apply Jensen's inequality for continuous random variables, which can be found in appendix A.4. Let $X = \frac{\rho_t(z)}{\mu(z)} = g(z)$. Then

$$\mathbb{E}(X) = \int \mu(z)g(z)\,dz = \int \mu(z)\frac{\rho_t(z)}{\mu(z)}\,dz = \int \rho_t(z)\,dz = 1. \tag{2.73}$$

Introduce $h(X) = -\log(X)$. We can show that $\mathbb{E}(h(X)) = D_{kl}(\mu, \rho_t)$:

$$\mathbb{E}(h(X)) = E(h \circ g(z)) = \int \mu(z)\left(-\log\left(\frac{\rho_t(z)}{\mu(z)}\right)\right)dz = D_{kl}(\mu, \rho_t). \tag{2.74}$$

Now we use that $h(X) = -\log(X)$ is a strictly convex function, so we can apply Jensen's inequality:

$$D_{kl}(\mu, \rho_t) = \mathbb{E}(h(X)) \geq h(\mathbb{E}(X)) = -\log(1) = 0, \quad \text{with equality if and only if } \mu = \rho_t. \tag{2.75}$$

□

We have just shown that the Kullback-Leibler divergence is 0 if and only if the corresponding densities are equal. We will now prove this as well for the derivative of the Kullback-Leibler divergence.

Lemma 2.4.8.

$$\frac{d}{dt}D_{KL}(\mu, \rho_t) = -\int \frac{\mu}{\rho_t}\frac{\partial \rho_t}{\partial t}dz = -\int \left(\frac{\mu^3}{\rho_t}\right)\left|\nabla\left(\frac{\rho_t}{\mu}\right)\right|^2 dz \leq 0, \tag{2.76}$$

Proof.

$$\begin{aligned}
\frac{d}{dt}D_{KL}(\mu, \rho_t) &\overset{(2.66)}{=} \frac{d}{dt}\int \mu(z)\left(\log\frac{\mu(z)}{\rho_t(z)}\right)dz \\
&= \int \mu(z)\frac{d}{dt}\left(\log\mu(z) - \log\rho_t(z)\right)dz \\
&= -\int \mu(z)\frac{d}{dt}\left(\log\rho_t(z)\right)dz \quad \text{(since } \mu(z) \text{ is independent of } t, \text{ the time derivative is 0)} \\
&= -\int \frac{\mu}{\rho_t}\frac{\partial \rho_t}{\partial t}dz.
\end{aligned} \tag{2.77}$$

This equation can be further simplified to the form

$$\begin{aligned}
\frac{d}{dt}D_{KL}(\mu, \rho_t) &= -\int \frac{\mu}{\rho_t}\nabla\cdot\left(\mu^2\nabla\left(\frac{1}{2}\left(\frac{\rho_t}{\mu}\right)^2\right)\right)dz \quad \text{(by substituting equation (2.52))} \\
&= \int \nabla\left(\frac{\mu}{\rho_t}\right)\mu^2\nabla\frac{1}{2}\left(\frac{\rho_t}{\mu}\right)^2 dx \quad \text{(by integration by parts, using that densities vanish at the border in infinity)} \\
&= \int \nabla\left(\frac{\mu}{\rho_t}\right)\mu^2\left(\frac{\rho_t}{\mu}\right)\nabla\left(\frac{\rho_t}{\mu}\right)dx \quad \text{(chain rule)} \\
&= \int \left(\frac{(\nabla\mu)\rho_t}{\rho_t^2} - \frac{\mu\rho_t}{\rho_t^2}\right)\mu\rho_t\cdot\nabla\left(\frac{\rho_t}{\mu}\right)dz \quad \text{(quotient rule)} \\
&= \int -\nabla\left(\frac{\rho_t}{\mu}\right)\frac{\mu^2}{\rho_t^2}\mu\rho_t\nabla\left(\frac{\rho_t}{\mu}\right)dz \quad \text{(quotient rule explained at the bottom of this proof)} \\
&= -\int \left(\frac{\mu^3}{\rho_t}\right)\left|\nabla\left(\frac{\rho_t}{\mu}\right)\right|^2 dz \leq 0.
\end{aligned} \tag{2.78}$$

In the second to last step we used the quotient rule:

$$\begin{aligned} \frac{-(\nabla\rho_t)\mu+\rho_t\nabla\mu}{\mu^2} &= -\nabla\left(\frac{\rho_t}{\mu}\right), \quad \text{and therefore} \\ \frac{(\nabla\mu)\rho_t}{\rho_t^2} - \frac{\mu\rho_t}{\rho_t^2} &= -\nabla\left(\frac{\rho_t}{\mu}\right)\frac{\mu^2}{\rho_t^2} \end{aligned} \tag{2.79}$$

□

Lemma 2.4.9.

$$\frac{d}{dt}D_{KL}(\mu,\rho_t) = 0 \iff \mu(z) = \rho_t(z) \quad \textit{for all } z, \tag{2.80}$$

Proof. We will prove both directions:

1. For the direction where we assume that $\mu(z) = \rho_t(z)$, we can use that $\mu(z)$ is a stationary solution from lemma 2.3.7. Therefore $\frac{\partial\rho_t}{\partial t} = 0$, and thus $\frac{d}{dt}D_{KL}(\mu,\rho_t) = 0$.

2. When $\frac{d}{dt}D_{KL}(\mu,\rho_t) = 0$, first notice that $\frac{\mu(z)}{\rho_t(z)} \neq 0$. Therefore we have that $\frac{\partial\rho_t}{\partial t} = 0$. We have shown that $\rho_t(z) = \mu(z)$ is a stationary solution for which this holds. However, we have not shown that it is the only stationary solution. To proof this direction and complete the proof we will use the Kullback-Leibler divergence of ρ and $\tilde{\rho}_t$. This will be done in the next section in 2.4.17.

□

We can combine the results of lemmas 2.4.8 and 2.4.9 to conclude that finding $\rho_t \approx \mu$ can be done by minimizing the Kullback-Leibler divergence, instead of maximizing the log-likelihood. During this minimization process, the Kullback-Leibler divergence of μ and ρ_t does not stop decreasing until $\rho_t(z)$ reaches it's target $\mu(z)$ [13]. However, the proof of lemma 2.4.9 is not complete yet. To finish this prove and then use this to show convergence of $\tilde{\rho}_t(x)$ to the target μ, in the next section we will look at the Kullback-Leibler divergence of ρ and $\tilde{\rho}_t$.

2.4.2. Kullback-Leibler divergence of ρ and $\tilde{\rho}_t$

The aim is still to to prove convergence of $\tilde{\rho}_t(x)$ to the target μ for every initial density of the samples, $\rho(x)$. To accomplish a more general argument for the convergence of the solution of equation (2.52) to μ and to complete the proof of lemma (2.4.9), in this section we consider the Kullback-Leibler divergence of ρ and $\tilde{\rho}_t$, $D_{\text{KL}}(\rho,\tilde{\rho}_t)$. Intuitively, this makes sense since equation (2.11) shows that we want interpret $\tilde{\rho}_t(x)$ as a current estimation of the density function $\rho(x)$. For $D_{\text{KL}}(\rho,\tilde{\rho}_t)$ we will also prove that the evolution is smaller or equal to 0, with equality only when $\tilde{\rho}_t = \rho$. From this we will retrieve that the Kullback-Leibler divergence of ρ and $\tilde{\rho}_t$ will keep decreasing until $\tilde{\rho}_t$ reaches the target density ρ. This can be used to show convergence of the solution of equation (2.52) to μ [13]. We start with using equation (2.64) from the definition to find the Kullback-Leibler divergence of ρ and $\tilde{\rho}_t$, $D_{kl}(\rho,\tilde{\rho}_t)$:

Lemma 2.4.10.

$$D_{kl}(\rho,\tilde{\rho}_t) = \int \rho(x)\log\left(\frac{\rho(x)}{\tilde{\rho}_t(x)}\right)dx. \tag{2.81}$$

We can derive the following form of $D_{\text{KL}}(\rho,\tilde{\rho}_t)$:

Lemma 2.4.11.

$$D_{\text{KL}}(\rho,\tilde{\rho}_t) = \int \log(\rho(x))\rho(x)\,dx - L_\rho(\phi_t). \tag{2.82}$$

Proof.

$$\begin{aligned} D_{\mathrm{KL}}(\rho,\tilde\rho_t) &= \int \log\left(\frac{\rho}{\tilde\rho_t}\right)\rho\, dx \\ &= \int \log(\rho)\rho\, dx - \int \log(\tilde\rho_t)\rho\, dx \\ &= \int \log(\rho)\rho\, dx - L_\rho(\phi_t). \\ &\text{Here we used that} \\ \int \log(\tilde\rho_t)\rho &\overset{(2.9)}{=} \int \log(J_{\phi_t}(x))\mu(\phi_t(x))\rho\, dx \\ &= \int \log(J_{\phi_t}(x))\mu(\phi_t(x))\rho(x)\, dx \\ &\overset{(2.21)}{=} L_\rho(\phi_t). \end{aligned} \tag{2.83}$$

□

Lemma 2.4.12. *Minimizing the Kullback-Leibler divergence of* ρ *and* $\tilde\rho_t$ *is the same as maximizing the log-likelihood* $L_\rho(\phi_t(x))$.

Proof. Is similar to that the proof of 2.4.3. Just as before, to minimize $D_{\mathrm{KL}}(\rho,\tilde\rho_t)$ we are looking for when the time derivative equals 0. Notice that in the expression of equation (2.83), only $L_\rho(\phi_t)$ is time dependent. When taking the time derivative, all other parts of this expression become 0. Therefore

$$\frac{d}{dt}D_{KL}(\rho,\tilde\rho_t) = -\frac{d}{dt}L_\rho(\phi_t). \tag{2.84}$$

This shows that minimizing the Kullback-Leibler divergence is the same as maximizing the log-likelihood of ρ and $\tilde\rho_t$. □

The goal is to show that that the $D_{\mathrm{KL}}(\rho,\tilde\rho_t)$ will keep decreasing until $\tilde\rho_t$ reaches the target density ρ. For this we need want to show that

$$\frac{d}{dt}D_{KL}(\rho,\tilde\rho_t) \le 0, \quad \text{with equality if and only if } \rho(x) = \tilde\rho_t(x). \tag{2.85}$$

Recall that in equation (2.5), $y(x)$ maps the observed x_i from random variable x to random variable y with known density $\mu(y)$. In section 2.1.3 we introduced y as the composition of infinitesimal transformations [13]. Let us now consider $\phi_t(x)$ to be the composition of two maps, $\phi_t(x) = \phi_{t_1+t_2}(x) = (\phi_{t_2}\circ\phi_{t_1})(x)$. We will now analyze the log-likelihood of this composition:

Lemma 2.4.13. *Let* $\phi_t(x) = \phi_{t_1+t_2}(x) = (\phi_{t_2}\circ\phi_{t_1})(x)$. *Then define*

$$\tilde L_\rho[\phi_{t_1}] = \int \log(J_{\phi_{t_1}}(x))\rho(x)\, dx. \tag{2.86}$$

Using this notation, the log likelihood of the composition, $L_\rho[\phi_{t_1+t_2}(x)]$, *is given by*

$$L_\rho[\phi_{t_1+t_2}(x)] = L_{\rho_{t_1}}[\phi_{t_2}] + \tilde L_\rho[\phi_{t_1}], \tag{2.87}$$

Proof. From applying the definition of the log-likelihood we get

$$L_\rho[\phi_{t_1+t_2}(x)] \overset{(2.21)}{=} \int \left(\log\left(J_{\phi_{t_1+t_2}}(x)\right) + \log\left(\mu(\phi_{t_1+t_2}(x))\right)\right)\rho(x)\, dx. \tag{2.88}$$

By the chain rule, the Jacobian of $\phi_{t_1+t_2}$,

$$j_{\phi_{t_1+t_2}} = j_{\phi_{t_2}}(\phi_{t_1}(x))\, j_{\phi_{t_1}}(x). \tag{2.89}$$

Since the determinant is a multiplicative image [1],

$$J_{\phi_{t_1+t_2}}(x) = J_{\phi_{t_2}}(\phi_{t_1}(x))\, J_{\phi_{t_1}}(x). \tag{2.90}$$

Also notice that we can compute the log-likelihood associated with $\phi_{t_2}(y)$ under a distribution $\rho_{t_1}(y)$,

$$L_{\rho_{t_1}}[\phi_{t_2}] = \int \Big(\log(J_{\phi_{t_2}}(y)) + \log(\mu(\phi_{t_2}(y)))\Big)\rho_{t_1}(y)\,dy \tag{2.91}$$

We will now combine the above statements:

$$\begin{aligned}
L[\phi_{t_1+t_2}(x)] &= \int \Big(\log\Big(J_{\phi_{t_2}}(\phi_{t_1}(x))\,J_{\phi_{t_1}}(x)\Big) + \log\big(\mu(\phi_{t_1+t_2}(x))\big)\Big)\rho(x)\,dx \\
&= \int \Big(\log(J_{\phi_{t_2}}(\phi_{t_1}(x))) + \log(J_{\phi_{t_1}}(x)) + \log\big(\mu(\phi_{t_1+t_2}(x))\big)\Big)\rho(x)\,dx \\
&\quad \text{Now substitute } y=\phi_{t_1}(x) \text{ and } \log\big(\mu(\phi_{t_1+t_2}(x))\big) = \log(\mu(\phi_{t_2}(\phi_{t_1}(x)))), \\
&= \int \Big(\log(J_{\phi_{t_2}}(y)) + \log(J_{\phi_{t_1}}(x)) + \log\big(\mu(\phi_{t_2}(y))\big)\Big)\rho(x)\,dx \\
&= \int \Big(\log(J_{\phi_{t_2}}(y)) + \log\big(\mu(\phi_{t_2}(y))\big)\Big)\rho(x)\,dx + \int \Big(\log(J_{\phi_{t_1}}(x))\Big)\rho(x)\,dx. \\
&\quad \text{Now apply change of variables} \quad dx = \frac{dy}{J_{\phi_{t_1}}(x)}, \quad \rho_{t_1}(y) = \frac{\rho(x)}{J_{\phi_{t_1}}(x)} \text{ and therefore} \\
&= \int \Big(\log\Big(J_{\phi_{t_2}}(y)\Big) + \log\big(\mu(\phi_{t_2}(y))\big)\Big)\rho_{t_1}(y)dy + \int \log\Big(J_{\phi_{t_1}}(x)\Big)\rho(x)\,dx \\
&= L_{\rho_{t_1}}[\phi_{t_2}] + \tilde{L}_\rho[\phi_{t_1}]
\end{aligned} \tag{2.92}$$

□

After the map $\phi_{t_1+t_2}(x)$ we want to find the direction in which the log-likelihood of $L_\rho(\phi_{t_1+t_2}) = L_\rho(\phi_t)$ evolves. We compute the time derivative of $L_\rho(\phi_{t_1+t_2})$ at a fixed t_1. We use that $\tilde{L}_\rho[\phi_{t_1}]$ does not depend on t_2. Therefore, at fixed t_1 we get

$$\frac{d}{dt}D_{KL}(\rho,\tilde{\rho}_t) \overset{(2.84)}{=} -\frac{d}{dt}L_\rho(\phi_t) = -\frac{d}{dt_2}L_{\rho_{t_1}}(\phi_{t_2}). \tag{2.93}$$

In [13], it is stated that "on the other hand, modifying the intermediate time t_1 does not affect the value of D_{KL}, just the relative weight of its components under the partition (2.88)". To understand this, let us write explicitly what modifying the intermediate time t_1 to $\tilde{t}_1$ does. First notice that $t = t_1 + t_2 = \tilde{t}_1 + \tilde{t}_2$.

$$\begin{aligned}
\phi_t &= \phi_{t_1+t_2} = \phi_{t_2}(\phi_{t_1}) = \phi_{t_2}\circ\phi_{t_1}, \\
\phi_{t_1} &= \phi_{t_N}\circ\phi_{t_{N-1}}\circ\cdots\circ\phi_{t_1}, \\
\phi_{t_2} &= \phi_{t_K}\circ\phi_{t_{K-1}}\circ\cdots\circ\phi_{t_{K-K}}, \\
&\quad \text{Substitute } \phi_{t_1} \text{ and } \phi_{t_2} \text{ into the expression of } \phi_t, \\
\phi_t &= \phi_{t_K}\circ\phi_{t_{K-1}}\circ\cdots\circ\phi_{t_{u-u}}\circ\phi_N\circ\cdots\circ\phi_{N-N}, \\
&= \phi_{\tilde{t}_2}\circ\phi_{\tilde{t}_1} = \phi_{\tilde{t}_1+\tilde{t}_2}
\end{aligned} \tag{2.94}$$

Therefore, when modifying the intermediate time t_1 to $\tilde{t}_1$ in equation (2.88)

$$L[\phi_{t_1+t_2}(x)] = L[\phi_t(x)] = L[\phi_{\tilde{t}_1+\tilde{t}_2}] = L_{\rho_{\tilde{t}_1}}[\phi_{\tilde{t}_2}] + \tilde{L}_\rho[\phi_{\tilde{t}_1}]. \tag{2.95}$$

From this it is clear that $L_\rho(\phi_t)$ is not affected by modifying t_1, and from equation (2.83) it follows that the value of D_{KL} does not change. Since modifying the intermediate time t_1 does not affect the value of D_{KL}, in [13] it is suggested to calculate the limit in which $t_1 \uparrow t$ and $t_2 \downarrow 0$ with $t_1 + t_2 = t$. This is useful, since then we precisely obtain "the likelihood L_{ρ_t} that determines the flow $\phi_t$in (2.43)" [13], since

$$\frac{d}{dt_2}L_{\rho_{t_1}}(\phi_{t_2}) \to u_t(\phi_t) \text{ as } t_1 \uparrow t \text{ and } t_2 \downarrow 0 \text{ with } t_1 + t_2 = t. \tag{2.96}$$

This claim comes directly from article [13] and is in this paper not supported by a proof. This statement is used to prove the following lemma:

Lemma 2.4.14.

$$\frac{d}{dt}D_{KL}(\rho,\tilde{\rho}_t) = -\int \left|u_t(\phi_t(y))\right|^2 dy \le 0 \tag{2.97}$$

Proof.

$$\begin{aligned}\frac{d}{dt}D_{KL}(\rho,\tilde{\rho}_t) &= -\frac{d}{dt}L_{\rho_t} \\ &= -\lim_{\epsilon\to 0}\frac{1}{\epsilon}\left(L(\phi_{t+\epsilon}) - L(\phi_t)\right) \\ &= -\frac{1}{\epsilon}\left(L(\phi+\epsilon\dot{\phi}_t) - L(\phi_t)\right). \\ &\text{Now we will apply the formula of the functional derivative 2.30 on this} \\ &= -\frac{1}{\epsilon}\epsilon\int \frac{\delta L_{\rho_t}}{\delta\phi_t}\dot{\phi}_t(y)\,dy \\ &\overset{(2.96)}{=} -\int \left|u_t(\phi_t(y))\right|^2 dy \\ &\leq 0\end{aligned} \tag{2.98}$$

□

This can be used to show equation (2.97) holds, with equality only when $\tilde{\rho}_t = \rho$:

Lemma 2.4.15.

$$\frac{d}{dt}D_{KL}(\rho,\tilde{\rho}_t) = 0 \iff \rho = \tilde{\rho}_t \tag{2.99}$$

Proof. We use equation (2.97) find that

$$\frac{d}{dt}D_{KL}(\rho,\tilde{\rho}_t) = 0 \iff u_t(\phi_t(y)) = \dot{\phi}_t(x) = 0 \tag{2.100}$$

This means that the evolution of $\phi_t(x)$ is equal to 0, which means that the Jacobian is the identity matrix, and therefore the Jacobian determinant $J_{\phi_t}(x) = 1$. We find finish the proof by using the formula from (2.12): $\rho_t(z) = \frac{\rho(x)}{J_{\phi_t}(x)} = \rho(x)$. □

We can now derive the desired statement

Proposition 2.4.16. *The Kullback-Leibler divergence of ρ and $\tilde{\rho}_t$ will keep decreasing until $\tilde{\rho}_t$ reaches the target density ρ.*

Proof. Follows directly from the equivalences of lemmas 2.4.15 and 2.4.14. □

Now we can finally use this to finish the second statement of the proof of lemma 2.4.9. All we still needed was the following:

Proposition 2.4.17.

$$\frac{d}{dt}D_{KL}(\mu,\rho_t) = 0 \Rightarrow \mu(z) = \rho_t(z) \quad \textit{for all } z. \tag{2.101}$$

Proof. We will prove the statement by proving the contrapositive:

$$\mu(z) \neq \rho_t(z) \Rightarrow \frac{d}{dt}D_{KL}(\mu,\rho_t) \neq 0 \quad \text{for all } z. \tag{2.102}$$

Suppose $\mu \neq \rho_t$, then we get

$$\begin{aligned}\mu \neq \rho_t &\Rightarrow \rho \neq \tilde{\rho}_t \\ &\Rightarrow D_{KL}(\rho,\tilde{\rho}_t) \neq 0 \\ &\Rightarrow \text{ We have not reached a maximum of } L_\rho(\phi_t). \\ &\Rightarrow \text{ We have not reached the maximum of } D_{KL}(\mu,\rho_t). \\ &\Rightarrow \frac{d}{dt}D_{KL}(\mu,\rho_t) \neq 0.\end{aligned} \tag{2.103}$$

We conclude the prove by contrapositive □

We can now combine all the results into a theorem for convergence of $\tilde{\rho}_t(x)$ to the target $\mu(\phi_t(x))$:

Theorem 2.4.18 (Convergence of $\tilde{\rho}_t(x)$ to the target $\mu(\phi_t(x))$**.** *Let x be a random variable with density function $\rho(x)$. When $\phi_t(x)$ is given by* (2.43)*, then the estimated density $\tilde{\rho}_t(x)$, calculated by 2.9 converges to the target $\mu(\phi_t(x))$ for all initial densities of the random variable x.*

Proof. Follows from lemmas 2.4.8, 2.4.7 and 2.4.9. □

We have now shown convergence for all $\tilde{\rho}_t$ to ρ. However, when the flow functions keep the log-likelihood constant, we will not reach the desired target density. Therefore in [13] the following restriction is proposed:

$$\rho_t \neq \mu \quad \Rightarrow \quad \frac{\delta L_{\rho_t}[\phi_t]}{\delta \phi_t} \neq 0. \tag{2.104}$$

In this equation, the variation is taken at fixed ρ_t. In the next section we will discuss other requirements for the maps $\phi_t(x)$.

2.5. Individual flow requirements

It is clear from (2.7) and (2.8) that the functions $\phi_t(x)$ and $\phi_t(x)^{-1}$ need to be invertible and differentiable for the purpose of a normalizing flow. In the section 2.4 we saw that for convergence of $\tilde{\rho}_t$ to ρ we only need the constraint of equation (2.104). In this section we will discuss other requirements to enhance the performance of the normalizing flow procedure that are suggested in [13].

- **Over-resolution** occurs when the flow $\phi_t(x)$ is excessively fine-tuned to the data points. The estimated density function $\tilde{\rho}_t(x)$ then captures noise and minor variations rather than the underlying distribution. In other words, $\tilde{\rho}_t(x)$ then fails to represent a smooth continuous probability function. For this reason we need smoothness of the maps $\phi_t(x)$. This can be ensured by restricting to maps with length scale at all points larger than the typical distance between nearby flow-makers [13]. This smoothing is done with a mollification factor. This factor makes sharp features smooth, while still remaining close to the original. Later we will see how the parameter ϵ functions as mollifier to guarantee that when data points are closer together, more mollification of the map happens. Figure 2.10 provides an example of what overfitting would look like in the context of density estimation.

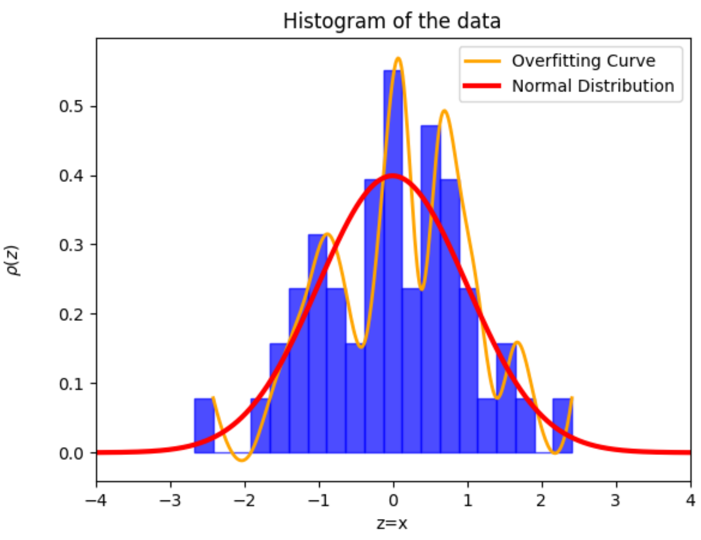


Figure 2.10: Histogram of 50 samples from the standard normal density. The density of the normal distribution is plotted as a red line. The data is actually distributed according to this is the density function. The orange curve is an example of what overfitting on the data would look like: the curve does not represent the correct underlying density, but rather captures minor variations in the data.

- **Flexible and Simple** maps that are computationally manageable are used as building blocks. These blocks have to be flexible to have a fast convergence to arbitrary distributions $\rho(x)$. One way of achieving flexibility would me to implement flow functions with many parameters. However, since for every iteration the parameters of the flow function need to be decided, this does have computational disadvantages. We also can achieve flexibility by keeping the flow simple and computationally manageable. This can be done by taking many near identity maps. This second method is extremely flexible since arbitrary densities can be built through the composition of simple maps [12], and computationally manageable since we only have to decide a few parameters per iteration of the algorithm. Therefore taking many near identity maps is preferred.

- **Explicit** forms of the flow and their parameters ensure our ability to easily compute the Jacobian and the flow functions. By keeping computations in explicit form, the algorithm stays computationally manageable.

3

Applying a linear normalizing flow

3.1. normalizing flow with $\mu(y)$ the standard Gaussian and a linear flow

The situation is the same as in section 2.1.1: we aim to estimate the underlying probability density function of random variable x, $\rho(x)$, from observations of this random x^j, $j = 1...m$. We do this by looking for a map $y(x)$ to map observations towards the random variable x with unknown density towards the random variable y with known density function $\mu(y)$. In this section, for $\mu(y)$ we will use the isotropic Gaussian.

$$\mu(y) = N(0, I_n) = \frac{1}{(2\pi)^{\frac{n}{2}}} e^{-\frac{\|y\|^2}{2}} \tag{3.1}$$

When function $y(x)$ and $\mu(y)$ are known we can apply equation (2.1) to retrieve the density of x:

$$\rho(x) = J_y(x)\mu(y(x)).$$

This section will first look at the case where we impose that the distribution of the sample data is a linear map from the isotropic Gaussian. We then use a flow based architecture so that this restriction is no longer necessary.

3.1.1. The map $y(x)$ as a linear function

In this section we will investigate a linear function $y(x)$. In some cases, it might be known or reasonable to assume the density, $\rho(x)$ of the random variable belonging to the samples is a linear transformation from the known density, $\mu(y)$. This means we assume that there is a linear map from random variable x to random variable y. Since we impose a linear transformation to the standard Gaussian, this is a parametric method. We will follow a similar procedure as in [12]. The parameter β determines the linear function that is used:

$$y_\beta(x) = A(x-b) \quad \left(\beta = \left\{A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n\right\}\right). \tag{3.2}$$

Just as before we choose β to maximize the log-likelihood of the data.

Lemma 3.1.1.

$$\beta = \arg\max_{\beta\in\{A,b\}} L = \arg\max_{\beta\in\{A,b\}} \sum_{j=1}^{m} \left[\log\left(J_{y_\beta}(x^j)\right) - \frac{\|y_\beta(x)\|^2}{2}\right] \tag{3.3}$$

Proof.

$$\begin{aligned}
\beta &= \arg\max_{\beta\in\{A,b\}} L[y_\beta(x)] \\
&= \arg\max_{\beta\in\{A,b\}} \sum_{j=1}^{m} \left[\log(\rho(x^j;\beta))\right] \\
&= \arg\max_{\beta\in\{A,b\}} \sum_{j=1}^{m} \left[\log(\mu(y_\beta(x^j))J_{y_\beta}(x^j))\right] \\
&= \arg\max_{\beta\in\{A,b\}} \sum_{j=1}^{m} \left[\log(J_{y_{\beta(x)}}(x^j)) + \log\left(\frac{1}{(2\pi)^n} e^{-\frac{1}{2}\|y_\beta(x^j)\|^2}\right)\right] \\
&= \arg\max_{\beta\in\{A,b\}} \sum_{j=1}^{m} \left[\log(J_{y_{\beta(x)}}(x^j)) - \frac{n}{2}\log(2\pi) - \frac{1}{2}\|y_\beta(x^j)\|^2\right]
\end{aligned}$$

In this form we recognize that $-\frac{n}{2}\log(2\pi)$ is independent of β, so

$$\beta = \arg\max_{\beta\in\{A,b\}} \sum_{j=1}^{m} \left[\log\left(J_{y_\beta}(x^j)\right) - \frac{\|y_\beta(x)\|^2}{2}\right]$$

□

Lemma 3.1.2. *The output of the maximization of* (3.3) *is given by*

$$b = \bar{x}, \quad A = \Sigma^{-\frac{1}{2}}, \tag{3.4}$$

Where Σ is the empirical covariance matrix, $\Sigma = \frac{1}{m}\sum_{j=1}^m x^j x_j^t$, of the data and $\bar{x}$ denotes the mean of the data (x_j^t resembles the transposed of x^j). $\bar{x} = \frac{1}{m}\sum_{j=1}^m x^j$.

Proof. The log-likelihood $L(y_\beta(x))$ is dependent on the choice of $\beta = \left\{A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n\right\}$. We will now derive the output of the maximization of equation (3.3) by setting the derivative of L with respect respect to b to 0. Notice that the Jacobian matrix of $y_\beta(x) = A(x-b)$ is given by

$$j_{y_\beta(x)} = \begin{bmatrix}
\frac{\partial y_{\beta_1}}{\partial x_1} & \frac{\partial y_{\beta_1}}{\partial x_2} & \cdots & \frac{\partial y_{\beta_1}}{\partial x_n} \\
\frac{\partial y_{\beta_2}}{\partial x_1} & \frac{\partial y_{\beta_2}}{\partial x_2} & \cdots & \frac{\partial y_{\beta_2}}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial y_{\beta_n}}{\partial x_1} & \frac{\partial y_{\beta_n}}{\partial x_2} & \cdots & \frac{\partial y_{\beta_n}}{\partial x_n}
\end{bmatrix}$$

$$\begin{aligned}
\frac{\partial y_{\beta_i}}{\partial x_j} &= \frac{\partial}{\partial x_j}\left(A_{i,1}(x_1-b_1) + A_{i,2}(x_2-b_2) + \cdots + A_{i,n}(x_n-b_n)\right) \\
&= A_{ij}.
\end{aligned} \tag{3.5}$$

Therefore we have that $j_{y_\beta(x)} = A$. Therefore we combine equation (3.5), (3.2) and (3.3) to get the following:

$$L(y_\beta(x)) = \sum_{j=1}^{m}\left(\log(A) - \frac{\|A(x^j-b)\|^2}{2}\right) \tag{3.6}$$

Now take the derivative with respect to b:

$$\begin{aligned}
\frac{L(y_\beta(x))}{\partial b} &= \frac{\partial}{\partial b}\sum_{j=1}^{m} -\frac{\|A(x^j-b)\|^2}{2} \\
&= -\sum_{j=1}^{m}\frac{\partial}{\partial b}\left(\frac{\|A(x^j-b)\|^2}{2}\right) \\
&= \sum_{j=1}^{m} A^T A(x^j-b)
\end{aligned} \tag{3.7}$$

Where we used the following:

$$\begin{aligned}\|A(x^j-b)\|^2 &= (A(x^j-b))^T(A(x^j-b))\\ &= (x^j-b)^T A^T A(x^j-b)\end{aligned} \tag{3.8}$$

$$\frac{\partial}{\partial b}(x^j-b)^T A^T A(x^j-b) = -2A^T A(x^j-b) \tag{3.9}$$

Now we want to find the optimum, and therefore we set the derivative equal to 0.

$$\frac{L(y_\beta(x))}{\partial b} = A^T A\sum_{j=1}^{m}(x^j-b) = 0 \tag{3.10}$$

Now use that $A^\top A \neq 0$ since A is invertible, and therefore

$$\begin{aligned}\sum_{j=1}^{m}(x^j-b) &= 0\\ \sum_{j=1}^{m}x^j &= mb\\ b &= \frac{1}{m}\sum_{j=1}^{m}x^j\\ &= \bar{x}\end{aligned} \tag{3.11}$$

Now we still need to derive the expression $A = \Sigma^{-\frac{1}{2}}$. Only the scalar of the transformation changes the covariance, the additive term does not. When the data is linearly transformed by matrix A, the covariance is transformed by $A\Sigma A^T$. In our case random variable y is distributed as an isotropic Gaussian distribution, so $y_\beta(x) \sim \mu(y) = N(0, I_n)$. The covariance is therefore transformed to I_n. Therefore we can calculate A as follows:

$$\begin{aligned}A\Sigma A^T &= I\\ A^{-1}A\Sigma A^T(A^T)^{-1} &= A^{-1}(A^T)^{-1}\\ \Sigma A^T(A^T)^{-1} &= A^{-1}(A^T)^{-1}\\ \Sigma &= A^{-1}(A^T)^{-1}\\ \Sigma^{-1} &= A^T A\\ A &= \Sigma^{-\frac{1}{2}}\end{aligned} \tag{3.12}$$

□

From equation (3.4) we can explain that a linear choice for $y_\beta(x)$ results in subtracting the mean and dividing by the square root of the covariance matrix [12]. Therefore, from equation (3.1) we find that the density function of x, $\rho(x)$ is given by:

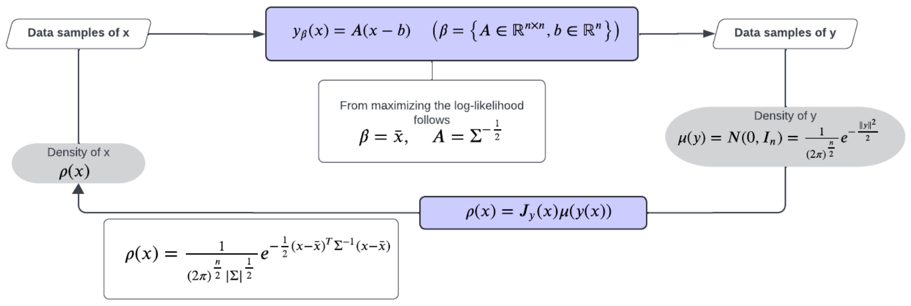
$$\rho(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\bar{x})^T\Sigma^{-1}(x-\bar{x})} \tag{3.13}$$

The diagram in figure 3.1 summarizes the knowledge we acquired for a linear map $y_\beta(x)$ onto the isotropic Gaussian. The problem with this parametric procedure is that we assume an underlying probability, since we assume there is a linear transformation from x onto y. However, this might not at all be the case. We solve this by factoring the map $y(x)$ into N parametric maps. This will be done in section 3.1.2

3.1.2. Linear flow functions to construct arbitrary $y(x)$

In section 3.1.1 we imposed that the data samples are from a particular distribution. We want the probabilistic method is able to estimate complex and ideally arbitrary densities. For example, for the application of probabilistic methods on deep neural networks we need to be able to estimate very complex densities. By the restriction of the map, this might not be possible. In this section, we solve this by factoring the map $y(x)$ into N parametric maps, the flow functions, $\varphi_{\beta_i}(z)$:

$$y_N(x) = \varphi_{\beta_N} \circ \varphi_{\beta_{N-1}} \circ \cdots \circ \varphi_{\beta_1}(x) \tag{3.14}$$



Data samples of x
$y_\beta(x) = A(x-b) \quad \left(\beta = \left\{A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n\right\}\right)$
Data samples of y
From maximizing the log-likelihood follows
$\beta = \bar{x}, \quad A = \Sigma^{-\frac{1}{2}}$
Density of y
$\mu(y) = N(0, I_n) = \frac{1}{(2\pi)^{\frac{n}{2}}} e^{-\frac{\|y\|^2}{2}}$
Density of x
$\rho(x)$
$\rho(x) = J_y(x)\mu(y(x))$
$\rho(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\bar{x})^T\Sigma^{-1}(x-\bar{x})}$

Figure 3.1: Flow chart of the normalizing flow algorithm. Where we assume the density $\rho(x)$ is a linear transformation from the isotropic Gaussian distribution.

The benefit of the flow based model is that the composition of many simple maps can be made arbitrarily complex [12]. Therefore we can estimate the density of arbitrary density functions $\rho(x)$. The map $y_\beta(x)$ depends on the family of parameters $\beta = (\beta_1 \cdots \beta_N)$. By considering $y_\beta(x)$ as a composition of functions we can compute the φ_{β_i}'s sequentially by following ascent of the log-likelihood as explained in section 2.3.1. First we calculate β_1 from $y_1(x) = \varphi_{\beta_1}(x)$ in (3.2), then we use β_1 fixed at the value found in the prior step to calculate $y_2(x)$ [12]:

$$y_2(x) = \varphi_{\beta_2}(\varphi_{\beta_1}(x)) = \varphi_{\beta_2}(y_1(x)) = \varphi_{\beta_2}(z). \tag{3.15}$$

Where just as before in section 2.1.3 we introduce the notation z as the current flow of the observations: after iteration i,

$$z^j = y_{i-1}(x^j) \tag{3.16}$$

We always include the identity map $\varphi_{\beta_i=0}$. Therefore each step will only increase the value of the log-likelihood. Note that we do not maximize the log-likelihood, but only ascend the log-likelihood when computing the next φ_{β_i}, since maximizing the log-likelihood depending on β would result in the linear mapping (3.2), but the idea is to also be able to construct non-linear maps via this composition.

Lemma 3.1.3. *We can calculate the Jacobian of the current map $y_i(x^j)$ sequentially based on the current flow of the observation z_i:*

$$J_{y_i}(x^j) = J_{\varphi_i}(z^j) J_{y_{i-1}}(x^j) \tag{3.17}$$

Proof.

From the chain rule we have that:

$$J_{f\circ g}(x) = J_f(g(x))J_g(x)$$

Now we write out the expression of $y_i(x^j)$:

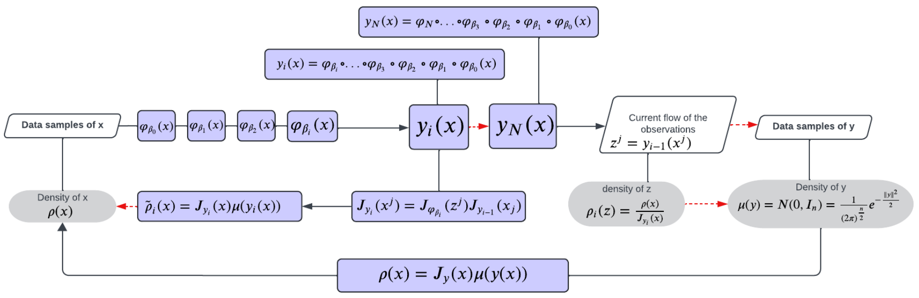
$$y_i(x^j) = \varphi_{\beta_i}(\varphi_{\beta_{i-1}}(\varphi_{\beta_{i-2}}(\ldots(\varphi_{\beta_1}(x^j))\ldots)))$$

Now we apply the chain rule to $y_i(x^j)$:

$$\begin{aligned} J_{y_i}(x^j) &= J_{\varphi_{\beta_i}}\left[\varphi_{\beta_{i-1}}\left(\varphi_{\beta_{i-2}}(\ldots(\varphi_{\beta_1}(x^j))\ldots)\right)\right] J_{\varphi_{\beta_{i-1}}\varphi_{\beta_{i-2}}(\ldots(\varphi_{\beta_1}(x))\ldots)}(x^j) \\ &= J_{\varphi_{\beta_i}}(y_{i-1}(x^j))J_{y_{i-1}}(x^j) \end{aligned} \tag{3.18}$$

□

In the diagram of figure 3.2, the scheme of figure 2.7 is rewritten in the case of a linear flow. Here we see again the duality of the algorithm: as $y_i(x)$ converges to the real $y(x)$, the estimated density, $\tilde{\rho}_i(x)$ converges to $\rho(x)$.



$y_N(x) = \varphi_N \circ \ldots \circ \varphi_{\beta_3} \circ \varphi_{\beta_2} \circ \varphi_{\beta_1} \circ \varphi_{\beta_0}(x)$
$y_i(x) = \varphi_{\beta_i} \circ \ldots \circ \varphi_{\beta_3} \circ \varphi_{\beta_2} \circ \varphi_{\beta_1} \circ \varphi_{\beta_0}(x)$
Data samples of x
$\varphi_{\beta_0}(x)$
$\varphi_{\beta_1}(x)$
$\varphi_{\beta_2}(x)$
$\varphi_{\beta_i}(x)$
$y_i(x)$
$y_N(x)$
Current flow of the observations $z^j = y_{i-1}(x^j)$
Data samples of y
Density of x $\rho(x)$
$\tilde{\rho}_i(x) = J_{y_i}(x)\mu(y_i(x))$
$J_{y_i}(x^j) = J_{\varphi_{\beta_i}}(z^j)J_{y_{i-1}}(x_j)$
density of z $\rho_i(z) = \frac{\rho(x)}{J_{y_i}(x)}$
Density of y $\mu(y) = N(0, I_n) = \frac{1}{(2\pi)^{\frac{n}{2}}} e^{-\frac{\|y\|^2}{2}}$
$\rho(x) = J_y(x)\mu(y(x))$

Figure 3.2: Flow chart of the normalizing flow algorithm with $\varphi_{\beta_i}(x)$ linear flow functions.

3.2. Example of a normalizing flow based on a linear flow function

To showcase that the normalizing flow probabilistic model based on a linear flow function can be used for density estimation, we will provide examples of the procedure on real data. To create the examples we will first specify a linear flow function, which is a parametric map that uses five parameters. This five parameter map is used to create two examples. The first example uses samples from a normal density with mean 2 and standard deviation 1. This showcases that this probabilistic model works, however, it could also be done with the parametric procedure of section 3.1.1, since there is a linear map from this density towards the standard normal density. The second example uses uses samples from an exponential density function with $\lambda = 1$. This showcases the power of a flow based model for non parametric density estimation, since there is not a linear map from the exponential density function towards this exponential density.

3.2.1. Five parameter linear flow function

In [13] a family of functions with five parameters $(\gamma, \sigma, x_0, \varphi_0, \epsilon)$ is used that satisfies the requirements from section 2.5, given by

$$\varphi_\alpha(x) = (1-\sigma)x + \varphi_0 + \gamma\sqrt{\epsilon^2 + [(1-\sigma)x - x_0]^2}, \tag{3.19}$$

with $\alpha = (\gamma, \sigma, \varphi_0)$. An explanation of this five parameter family of functions is given below. To support the explanation, figure 3.3 shows a plot of three cases of the five parameter function. The function $f_1(x) = x - 1 + \sqrt{(x-1)^2}$ is plotted in green, $f_2(x) = x - 1 + \sqrt{0.1 + (x-1)^2}$ is plotted in blue and $f_3(x) = x - 1 + \sqrt{1 + (x-1)^2}$ is plotted in red.

- Notice that when γ, σ, and φ_0 are zero, the map reduces to the identity. From the requirement of flexible and simple maps it follows that these are chosen close to 0 in the ascent direction of the log-likelihood. The parameters x_0 and ϵ are not selected by ascent of the log likelihood, but we will see that these are chosen to avoid over-resolution.
- The parameter σ quantifies the amount of stretching and φ_0 the displacement of the data x.
- To increase the flexibility, but maintain smoothness, we introduce the additional term $\gamma\sqrt{\epsilon^2 + [(1-\sigma)x - x_0]^2}$.
  - x_0 is the point where $\varphi_t(x)$ switches between $d\varphi/dx \approx 1 - \sigma - \gamma$ and $d\varphi/dx \approx 1 - \sigma + \gamma$. That x_0 is the location of the slope switch can clearly be seen in the figure 3.3. The value x_0 is picked at random from a standard normal distribution. This is motivated in [13] by the fact that that then, near convergence, the number of opportunities for local distortions is proportional to the density of the current flow of the samples $\rho_t(z)$. This is because $\rho_t(z)$ converges to $\mu(z)$, and in this example, $\mu(z)$ is the standard normal.
  - γ is the slope change at x_0.
  - The parameter ϵ mollifies the transition between the two slopes of the map to the left and right of x_0. The value of ϵ is x_0-dependent:

$$\epsilon = \sqrt{2\pi n_p} \exp\left(\frac{x_0^2}{2}\right), \tag{3.20}$$

where n_p is the desired average number of data points within the transition area. In figure 3.3, the effect of the mollifier is shown.

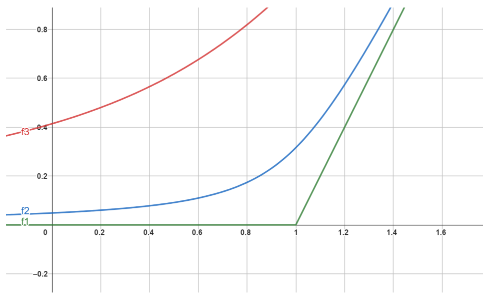


Figure 3.3: Plots made with Geogabra, in particular showcasing the effect of the mollifier. The function $f_1(x) = x-1+\sqrt{(x-1)^2}$ is plotted in green, $f_2(x) = x-1+\sqrt{0.1+(x-1)^2}$ is plotted in blue and $f_3(x) = x-1+\sqrt{1+(x-1)^2}$ is plotted in red.

As stated before, the parameter $\alpha = (\gamma, \sigma, \varphi_0)$ are chosen close to 0 and in direction of ascent of the log-likelihood. In [13] it is suggested to take at each step

$$\alpha = \Delta t \frac{\nabla_\alpha L}{\sqrt{1+\delta^2 |\nabla_\alpha L|^2}}. \tag{3.21}$$

In this formula Δt and δ are adjustable parameters which control the size of the steps.We will now sum op all the expressions needed to calculate $\nabla_\alpha L$ from equation (3.21). Since in the chapter 2 we used the notation $\varphi_t(x)$, we will adapt this notation now instead of $\varphi_\alpha(x)$.

$$L(x^j) = \frac{1}{m}\sum_{j=1}^{m} \log(\tilde{\rho}_t(x^j)) \tag{3.22}$$

$$\nabla_\alpha(L) = \begin{bmatrix} \frac{\partial L}{\partial \gamma} & \frac{\partial L}{\partial \sigma} & \frac{\partial L}{\partial \varphi_0} \end{bmatrix} \tag{3.23}$$

$$\frac{\partial L}{\partial \gamma} = \frac{1}{m}\sum_{j=1}^{m} \frac{\partial}{\partial \gamma} \log(\tilde{\rho}_t(x^j)) = \frac{1}{m}\sum_{j=1}^{m} \frac{1}{\tilde{\rho}_t(x^j)} \frac{\partial \tilde{\rho}_t(x^j)}{\partial \gamma} \tag{3.24}$$

$$\frac{\partial L}{\partial \sigma} = \frac{1}{m}\sum_{j=1}^{m} \frac{1}{\tilde{\rho}_t(x^j)} \frac{\partial \tilde{\rho}_t(x^j)}{\partial \sigma} \tag{3.25}$$

$$\frac{\partial L}{\partial \varphi_0} = \frac{1}{m}\sum_{j=1}^{m} \frac{1}{\tilde{\rho}_t(x^j)} \frac{\partial \tilde{\rho}_t(x^j)}{\partial \varphi_0} \tag{3.26}$$

$$\tilde{\rho}_t(x^j) = J_y(x^j)\mu(\tilde{\varphi}_t(x^j)) = J_{\varphi_t}(x^j) J_{y-1}(x^j)\mu(\varphi_t(x^j)) \tag{3.27}$$

We work in $\mathbb{R}^1$, so

$$J_{\varphi_t}(x^j) = \frac{d\varphi_t(x^j)}{dx} = (1-\sigma) + \gamma \frac{((1-\sigma)x^j - x_0)(1-\sigma)}{\sqrt{\epsilon^2 + ((1-\sigma)x^j - x_0)^2}}. \tag{3.28}$$

Also, μ is the standard Gaussian, so we have that

$$\mu(\varphi_t(x^j)) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(\varphi_t(x^j))^2}{2}} \tag{3.29}$$

- Formulas needed for $\frac{\partial L}{\partial \gamma}$:
  - An expression for $\frac{\partial \tilde{\rho}_t}{\partial \gamma}$:

$$\frac{\partial \tilde{\rho}_t}{\partial \gamma} = \left[\left(\frac{\partial}{\partial \gamma} J_{\varphi_t}(x^j) \right) \mu(\varphi_t(x^j)) + J_{\varphi_t}(x^j) \frac{\partial}{\partial \gamma} \mu(\varphi_t(x^j)) \right] J_{y-1} \tag{3.30}$$

- From equation (3.28)

$$\frac{\partial}{\partial\gamma}J_{\varphi_t}(x^j) = \frac{((1-\sigma)x^j - x_0)(1-\sigma)}{\epsilon^2 + ((1-\sigma)x^j - x_0)^2} \tag{3.31}$$

- From equation (3.29) and the chain rule

$$\frac{\partial}{\partial\gamma}\mu(\varphi_t(x^j)) = \frac{1}{\sqrt{2\pi}}e^{-\frac{(\varphi_t(x^j))^2}{2}}(-\varphi_t(x^j))\frac{\partial}{\partial\gamma}\varphi_t(x^j) = \frac{1}{\sqrt{2\pi}}e^{-\frac{(\varphi_t(x^j))^2}{2}}(-\varphi_t(x^i))\sqrt{\epsilon^2 + ((1-\sigma)x^j - x_0)^2} \tag{3.32}$$

- Formulas needed for $\frac{\partial L}{\partial\sigma}$:
  - An expression for $\frac{\partial\tilde{\rho}_t}{\partial\sigma}$:

$$\frac{\partial\tilde{\rho}_t(x^j)}{\partial\sigma} = \left[\left(\frac{\partial}{\partial\sigma}J_{\varphi_t}(x^j)\right)\mu(\varphi_t(x^j)) + J_{\varphi_t}(x^j)\frac{\partial}{\partial\sigma}\mu(\varphi_t(x^j))\right]J_{y-1} \tag{3.33}$$

  - Let $u = (1-\sigma)x^j - x_0$.
  - From equation (3.28) and the quotient rule

$$-1 + \gamma\frac{(-2(1-\sigma)x^j + x_0)\sqrt{\epsilon^2+u^2} + \frac{x^j u(1-\sigma)u}{\sqrt{\epsilon^2+u^2}}}{\epsilon^2 + u^2} \tag{3.34}$$

  - From equation (3.29) and the chain rule

$$\begin{aligned}\frac{\partial}{\partial\sigma}\mu(\varphi_t(x^j)) &= \frac{1}{\sqrt{2\pi}}e^{-\frac{(\varphi_t(x^j))^2}{2}}(-\varphi_t(x^j))\frac{\partial}{\partial\sigma}\varphi_t(x^j)\\ &= \frac{1}{\sqrt{2\pi}}e^{-\frac{(\varphi_t(x^j))^2}{2}}(-\varphi_t(x^j))\left(-x^j + \gamma\frac{u(-x^j)}{\sqrt{\epsilon^2+u^2}}\right)\end{aligned} \tag{3.35}$$

- Formulas needed for $\frac{\partial L}{\partial\varphi_0}$:
  - An expression for $\frac{\partial\tilde{\rho}_t}{\partial\varphi_0}$:

$$\frac{\partial\tilde{\rho}_t}{\partial\varphi_0} = \left[\left(\frac{\partial}{\partial\varphi_0}J_{\varphi_t}(x^j)\right)\mu(\varphi_t(x^j)) + J_{\varphi_t}(x^j)\frac{\partial}{\partial\varphi_0}\mu(\varphi_t(x^j))\right]J_{y-1} \tag{3.36}$$

  - Note that $\frac{\partial}{\partial\varphi_0}J_{\varphi_t}(x^j) = 0$
  - Also note that $\frac{\partial}{\partial\varphi_0}\varphi_t(x^j) = 1$ simplifies the expression for $\frac{\partial}{\partial\varphi_0}\mu(\varphi_t(x^j))$

$$\frac{\partial}{\partial\varphi_0}\mu(\varphi_t(x^j)) = \frac{1}{\sqrt{2\pi}}e^{-\frac{(\varphi_t(x^j))^2}{2}}(-\varphi_t(x^j)) \tag{3.37}$$

  - Therefore

$$\frac{\partial\tilde{\rho}_t}{\partial\varphi_0} = J_{\varphi_t}(x^j)\frac{1}{\sqrt{2\pi}}e^{-\frac{(\varphi_t(x^j))^2}{2}}(-\varphi_t(x^j)) \tag{3.38}$$

3.2.2. Implementing a five parameter linear flow function

In this section we will be implementing the linear flow function of section 3.2.1 and evaluate the results. The result of the prescribed normalizing flow in section 3.2.1 are plotted by Python code that can be found in the appendix B.3. The result is the histogram of the data, the evolution of the log-likelihood and the estimated densities of the data points. Recall that the aim of the flow is to transform the initial data samples into samples of a standard normal random variable. The flow does so by increasing the log-likelihood in every iteration of the flow. We can use the current flow to give an estimation of the $\tilde{\rho}_t(x)$.

The first experiment is executing this five parameter normalizing flow on 100 samples of a random variable x that is distributed as a normal random variable with mean 2 and standard deviation 1.

$$\rho(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{(x-2)^2}{2}}. \tag{3.39}$$

The normalizing flow of section 3.2.1 is implemented on 100 samples generated from normal distribution with standard deviation 1 and mean 2. Other parameters are $\Delta t = 1$, $\delta = 1000$, $n_p = 1$. 500 iterations of the described five parameter normalizing flow is given in figures 3.4, 3.5, 3.6 and 3.7. We see the expected results: the histogram of the data after the flow looks like that of standard normal distributed samples. We see that the log-likelihood converges to a maximum. We also see that the estimated densities, $\tilde{\rho}(x)$ converge to the real densities, $\rho(x)$ of the standard normal with mean 2 and deviation 1.

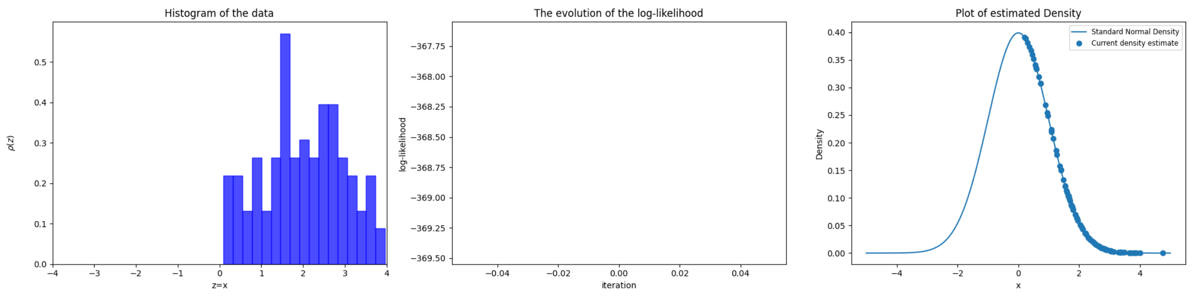


Figure 3.4: Normalizing flow of 5 parameter flow functions applied to samples from a normal density with mean 2 and standard deviation 1 after 0 iterations. Left shows the histogram of the data, the middle shows the evolution of the log-likelihood and the right shows the estimated density of the data points. This flow uses parameters $\Delta t = 1$, $\delta = 1000$, $n_p = 1$.

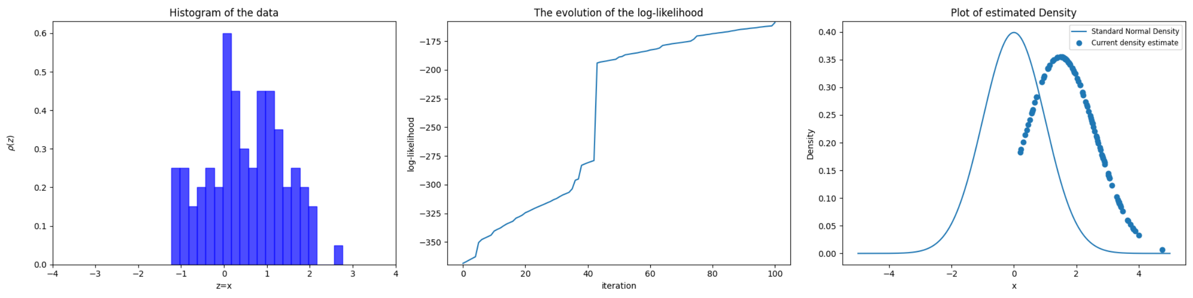


Figure 3.5: Normalizing flow of 5 parameter flow functions applied to samples from a normal density with mean 2 and standard deviation 1 after 100 iterations.

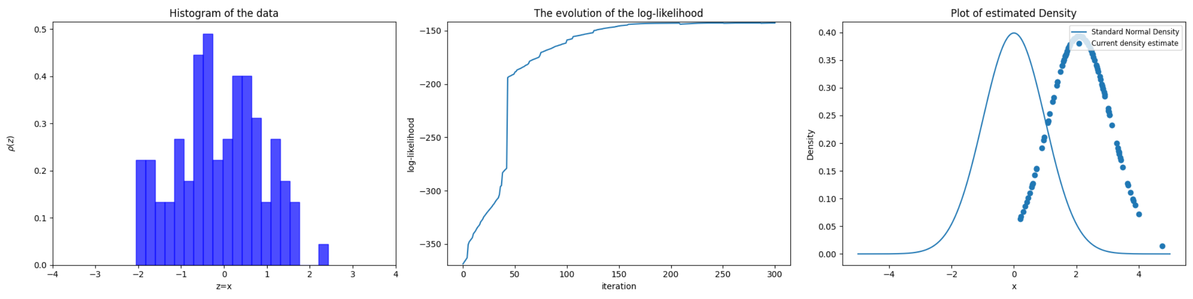


Figure 3.6: Normalizing flow of 5 parameter flow functions applied to samples from a normal density with mean 2 and standard deviation 1 after 300 iterations.

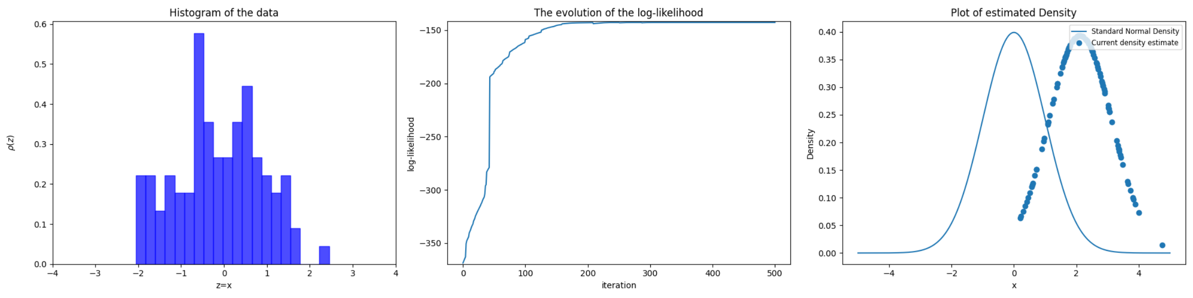


Figure 3.7: Normalizing flow of 5 parameter flow functions applied to samples from a normal density with mean 2 and standard deviation 1 after 500 iterations.

The second experiment is executing this five parameter normalizing flow on 100 samples of a random variable x that is distributed according the exponential distribution with $\lambda = 1$. The exponential density function is:

$$\rho(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.40}$$

The normalizing flow of section 3.2.1 implemented on 100 samples generated from the exponential with $\lambda = 1$, $\Delta t = 1$, $\delta = 1000$, $n_p = 0.01$ and 18000 iterations is given in figures 3.8, 3.9, 3.10 and 3.11.

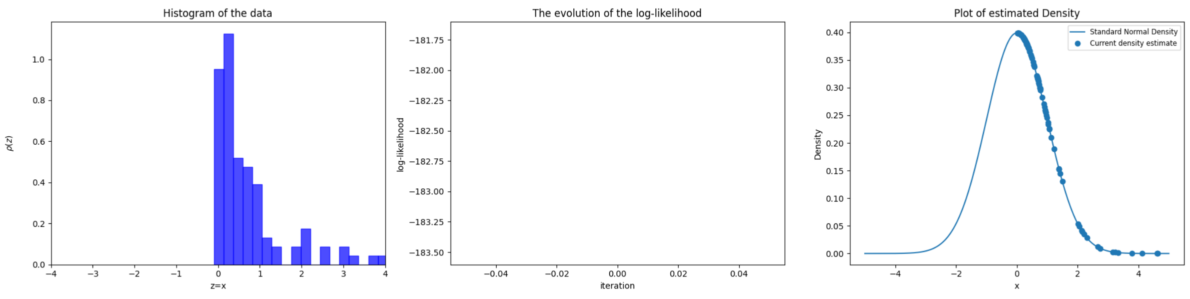


Figure 3.8: Normalizing flow of 5 parameter flow functions applied to samples from an exponential density with $\lambda = 1$ after 0 iterations. Left shows the histogram of the data, the middle shows the evolution of the log-likelihood and the right shows the estimated density of the data points. This flow uses parameters $\Delta t = 1$, $\delta = 1000$, $n_p = 0.01$.

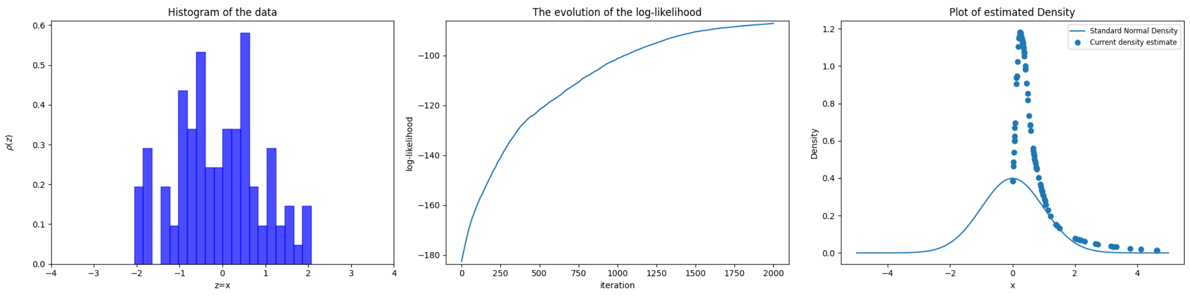


Figure 3.9: Normalizing flow of 5 parameter flow functions applied to samples from an exponential density with $\lambda = 1$ after 2000 iterations.

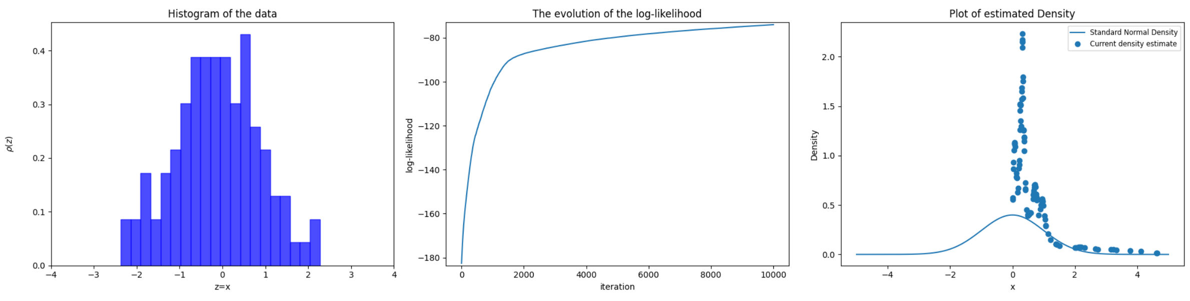


Figure 3.10: Normalizing flow of 5 parameter flow functions applied to samples from an exponential density with $\lambda = 1$ after 10000 iterations.

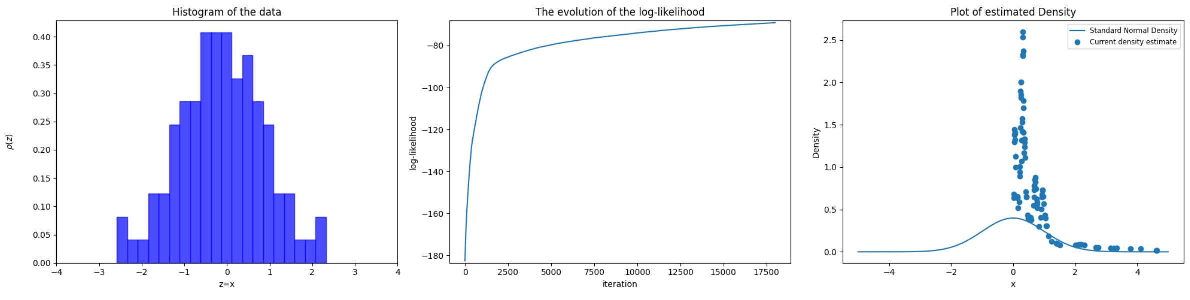


Figure 3.11: Normalizing flow of 5 parameter flow functions applied to samples from an exponential density with $\lambda = 1$ after 18000 iterations.

The choice for δ ensures that α is not drastically changing when the size of $\nabla_\alpha L$ is large, while keeping the process computationally possible. This is necessary for some form of numerical stability: although the steps of α are in a direction to increase the log-likelihood, if the size of $\nabla_\alpha L$ is very large, we might change α so much that the log-likelihood after $\varphi_\alpha(x)$ actually decreases. While avoiding steps in the wrong direction, this does mean we take smaller steps for increasing the log-likelihood. In the next experiment, this five parameter normalizing flow is executed on 100 new samples of the exponential with $\lambda = 1$. However, we change δ to 100 instead. The result is plotted in figures 3.12 and 3.13. After 4000 iterations we see a better result with respect to the log-likelihood than before, however, the evolution of the log-likelihood is less smooth. Figure 3.12 clearly shows a big step that decreases the log-likelihood. This misstep is a numerical issue. Figure 3.13 also clearly shows that the the following steps quickly increase the log-likelihood. The values log-likelihood clearly show when a step is taken into the wrong direction. This suggests that although a smaller δ makes the ascent of the log-likelihood less smooth, it can be beneficial for the maximization process.
The same result can be seen when we look at the scenario of the first experiment with samples from the normal density with mean 2 and standard deviation 1. For this experiment the value $\delta = 1000$ was used. When we change this to $\delta = 100$ and run the described 5 parameter normalizing flow algorithm we see faster, but less smooth convergence. This is illustrated in figure 3.14.

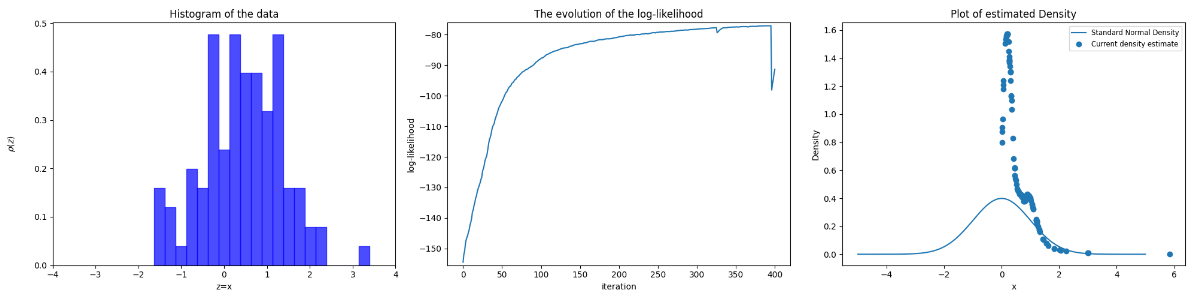


Figure 3.12: Normalizing flow of 5 parameter flow functions applied to samples from an exponential density with $\lambda = 1$ after 400 iterations. Left shows the histogram of the data, the middle shows the evolution of the log-likelihood and the right shows the estimated density of the data points. This flow uses parameters $\Delta t = 1$, $\delta = 100$, $n_p = 0.01$. The middle image clearly shows a step decreasing the log-likelihood.

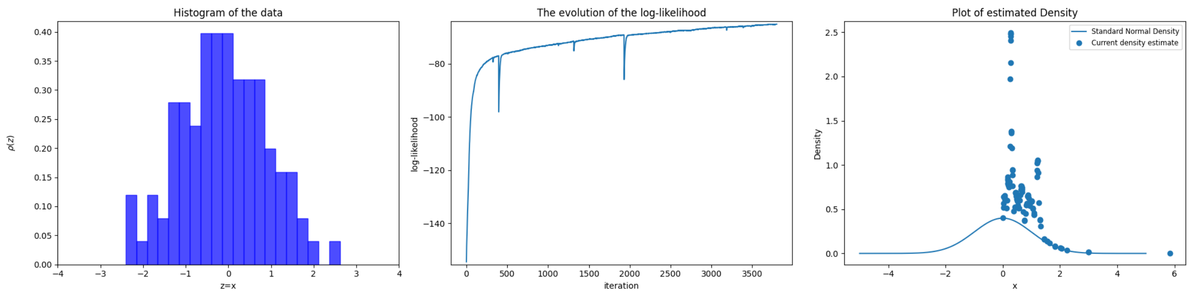


Figure 3.13: Normalizing flow of 5 parameter flow functions applied to samples from an exponential density with $\lambda = 1$ and parameter $\delta = 100$ after 4000 iterations, showing the limited impact of the missteps on the log-likelihood.

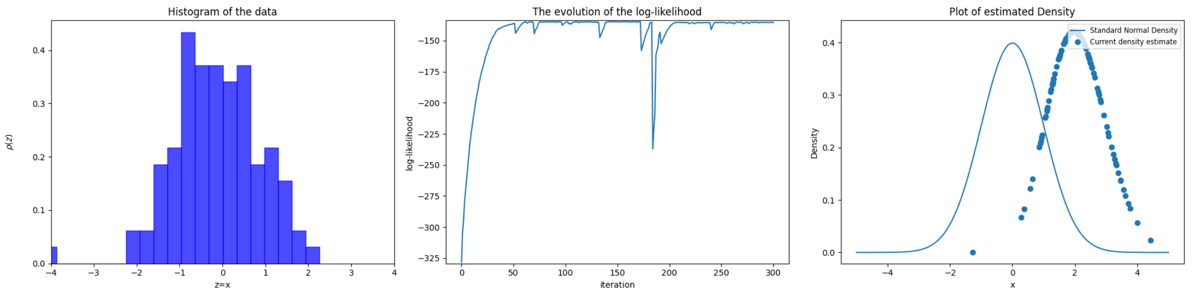


Figure 3.14: Normalizing flow of 5 parameter flow functions applied to samples from a normal density with mean 2 and standard deviation 1 after 300 iterations. Left shows the histogram of the data, the middle shows the evolution of the log-likelihood and the right shows the estimated density of the data points. This flow uses parameters $\Delta t = 1$, $\delta = 100$, $n_p = 1$. The evolution of the log-likelihood shows some steps decreasing the log-likelihood. However, we also see fast convergence to a maximum of the log-likelihood.

4
Discussion

This paper explains the normalizing flow model for density estimation by their underlying principles. Many of these underlying principles are motivated by results of sources [13] and [12]. Many of the statements of the follow directly from these two sources. Not all statements from these papers are included in this paper. Although this paper aims to give a comprehensive view on the underlying flow model, this is not fully accomplished. For example, we only proof proposition 2.3.3 for the one dimensional case. We also only use examples in the one dimensional case in sections 2.3.4 and 3.2. There are also other parts of these papers not included. For example, much about computational efficiency of the flows is not included, although this is an important reason for the popularity of normalizing flows. Additionally, sometimes the motivations given in these papers are not entirely understood. For example the proof of equation (2.96) is missing. Finally, there are possibly many more sources about normalizing flows with different perspectives and explanations.
One example of a different source it [10]. This paper attempts to provide a unified perspective. Therefore, looking into this paper could be interesting future research. Additionally, [10] focuses on the computational trade-offs that need to be made when choosing a particular flow function. And relates it to subjects as generative modelling and supervised learning. It would be interesting to show the application of flow models in areas related to machine learning, since there is a lot of research going on in this field currently.
Lastly, in this paper we only implement the five parameter flow from [13]. Future research could look at the implementation of different flow functions and compare these. Future research could also include comparing normalizing flows to other probabilistic models.

5 Conclusion

In this thesis, we explored the concept and application of normalizing flows as a method for density estimation of data samples. We started by outlining the underlying principles of normalizing flows, which are non-parametric, flow based probabilistic models. The model is non parametric and therefore does not assume an underlying density of the data. This makes normalizing flows flexible and applicable to complex datasets. The model is flow based since it approximates the underlying density function of sample data through a sequence of invertible and differentiable transformations. These transformations, or "flows," are designed to progressively map a complex data distribution to a simpler one, often a standard Gaussian distribution. This flow is then use to find an estimation of the density of the data. The success of the model is based on the convergence of the estimated density towards the real density of the data when using a flow that increases the log-likelihood. Additionally, this choice in flow function ensures that the current flow of the data converges towards a standard Gaussian distribution. The convergence is shown with the help of the Kullback-Leibler divergence, which is an estimate for the similarity of two density functions. Finally, this convergence is shown in examples where a five parameter flow is applied to real data samples. Due to the flow based architecture, where the samples are transformed through many relative simple maps, the normalizing flow model has the flexibility to estimate density functions from complex data samples, while staying computationally manageable.

A
Additional definitions and theorems

A.1. Definition of the Jacobian matrix

Definition A.1.1 (Jacobian matrix). *Let $y(x)$ be a function that maps a vector $x \in \mathbb{R}^n$ to a vector $y \in \mathbb{R}^n$: $y : \mathbb{R}^n \to \mathbb{R}^n$, given by*

$$y(x) = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_n(x) \end{bmatrix}.$$

The Jacobian matrix, j_y, of function $y(x)$ is the $n \times n$ matrix that contains all the first-order partial derivatives of the function. Each element $(j_y)_{kl}$ is the partial derivative of y_k with respect to x_l:

$$j_y(x) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_n} \end{bmatrix}$$

We use the notation J_y for the determinant of the Jacobean matrix of transformation y.

A.2. Change of variables in $\mathbb{R}^n$

Theorem A.2.1. *[4] If the mapping F is injective on U, then, for any measurable set $A \subset U$ and any Borel function $g \in L^1(\mathbb{R}^n)$, one has the equality*

$$\int_A g(F(x)) J_{F(x)}\, dx = \int_{F(A)} g(y)\, dy.$$

That g is a Borel function $g \in L^1(\mathbb{R}^n)$ means that g is Lebesque measurable. This is necessary so the integral is finite. [4].

A.3. Divergence theorem

The divergence theorem, commonly referred to as Gauss's theorem, states that the volume integral of the divergence of a vector field within a region enclosed by a surface is equivalent to the surface integral of the vector field over that closed surface. A more formal statement is given in [8]: suppose V is a subset of $\mathbb{R}^n$, which is closed and bounded with piecewise smooth boundary $S = \delta V$. let $U \subset \mathbb{R}^n$. If $\mathbb{F} : U \to \mathbb{R}^n$ is a continuously differentiable vector field, then

$$\int (\delta \cdot \mathbb{F})\, dV = \int_{\delta V} \mathbb{F} \hat{n}\, dS. \tag{A.1}$$

A.4. Jensen's inequality

Jensen's inequality for the probability distribution of random variable X and a convex function $f(X)$, where X has probability density function $p(X)$ can be written as

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]) \tag{A.2}$$

$$\int f(X)p(X)\,dX \geq f\left(\int Xp(X)\,dX\right). \tag{A.3}$$

A.5. Integration by parts

Theorem A.5.1. *[11] Suppose F and G are differentiable functions on $[a,b]$, $F' = f \in \mathbb{R}, and G' = g \in \mathbb{R}$. Then*

$$\int_a^b F(x)g(x)dx = F(b)G(b) - F(a)G(b) - \int_a^b f(x)G(x)dx. \tag{A.4}$$

A.6. Law of the subconscious statistician for discrete random variables

Lemma A.6.1 (Law of the subconscious statistician for discrete random variables)**.** *[7] If x is a discrete random variable and $g : \mathbb{R} \to \mathbb{R}$, then*

$$\mathbb{E}(g(x)) = \sum_{x^j \in Imx} g(x^j)P(x = x^j) \tag{A.5}$$

whenever this sum converges absolutely.

A.7. Law of the subconscious statistician for continuous random variables

Lemma A.7.1 (Law of the subconscious statistician for continuous random variables)**.** *[7] If x is a continuous random variable with density function $\rho(x)$ and $g : \mathbb{R} \to \mathbb{R}$, then*

$$\mathbb{E}(g(x)) = \int g(x)\mu(x)dx \tag{A.6}$$

whenever this integral converges absolutely.

B
Code

B.1. Numerical solution of equation (2.52) as plotted in figure 2.8

The code below has $\rho(x)$ as the uniform density on $[-0.2, 0.2]$ as in figure 2.8.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Define the standard normal PDF
def mu(z):
    return norm.pdf(z)

# Initial condition: exponential pdf for z > 0, and 0 for z <= 0
def initial_condition(z, lambda_param):
    return np.where(np.abs(z) <= 0.2, 1/0.4, 0)
    #return np.where(z > 0, lambda_param * np.exp(-lambda_param * z), 0)

# Boundary condition (assuming rho is zero at boundaries)
def boundary_condition(t):
    return (0, 0)

# Discretization parameters
dx = 0.1
dt = 0.001
t_max = 3
L = 5.0
z = np.arange(-L, L+dx, dx)
nz = len(z)
t = np.arange(0, t_max+dt, dt)
nt = len(t)

# Initialize the solution matrix
rho = np.zeros((nt, nz))

# Set initial condition with a lambda parameter for the exponential distribution
lambda_param = 1.0
rho[0, :] = initial_condition(z, lambda_param)

# Time-stepping loop
for n in range(0, nt-1):
    for i in range(1, nz-1):
        mu_z = mu(z[i])
        mu_z_plus = mu(z[i+1])
        mu_z_minus = mu(z[i-1])

        mu_phi_plus = mu(z[i+1])**2 * ((rho[n, i+1] / mu_z_plus)**2 - (rho[n, i] /
mu_z)**2) / (2 * dx)
        mu_phi_minus = mu(z[i-1])**2 * ((rho[n, i] / mu_z)**2 - (rho[n, i-1] /
mu_z_minus)**2) / (2 * dx)

        d_mu_phi_dz = (mu_phi_plus - mu_phi_minus) / (2 * dx)
```

```

            # Update rho
            rho[n+1, i] = rho[n, i] + dt * d_mu_phi_dz

        # Apply boundary conditions
        rho[n+1, 0], rho[n+1, -1] = boundary_condition(t[n+1])

        # Print the solution at multiple time steps
        if n % 300 == 0:  # Adjust the step size as needed
            print(f"Time step {n+1}, Time {t[n+1]:.5f}")
            print(rho[n+1, :])

    # Plotting the results
    plt.figure(figsize=(10, 6))
    for i in range(0, len(t), 300):  # Adjust the step size as needed
        plt.plot(z, rho[i, :], label=f't={t[i]:.5f}')
    plt.xlabel('z')
    plt.ylabel(r'$\rho(z)$')
    plt.legend()
    plt.show()
```

B.2. Numerical solution of equation (2.52) as plotted in figure 2.9

The code below has $\rho(x)$ as the exponential density with $\lambda = 1$ as in figure 2.8.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Define the standard normal PDF
def mu(z):
    return norm.pdf(z)

# Initial condition: exponential pdf for z > 0, and 0 for z <= 0
def initial_condition(z, lambda_param):
    return np.where(z > 0, lambda_param * np.exp(-lambda_param * z), 0)

# Boundary condition (assuming rho is zero at boundaries)
def boundary_condition(t):
    return (0, 0)

# Discretization parameters
dx = 0.1
dt = 0.001
t_max = 5
L = 5.0
z = np.arange(-L, L+dx, dx)
nz = len(z)
t = np.arange(0, t_max+dt, dt)
nt = len(t)

# Initialize the solution matrix
rho = np.zeros((nt, nz))

# Set initial condition with a lambda parameter for the exponential distribution
lambda_param = 1.0
rho[0, :] = initial_condition(z, lambda_param)

# Time-stepping loop
for n in range(0, nt-1):
    for i in range(1, nz-1):
        mu_z = mu(z[i])
        mu_z_plus = mu(z[i+1])
        mu_z_minus = mu(z[i-1])

        mu_phi_plus = mu(z[i+1])**2 * ((rho[n, i+1] / mu_z_plus)**2 - (rho[n, i] /
mu_z)**2) / (2 * dx)
        mu_phi_minus = mu(z[i-1])**2 * ((rho[n, i] / mu_z)**2 - (rho[n, i-1] /
mu_z_minus)**2) / (2 * dx)

```

```
        d_mu_phi_dz = (mu_phi_plus - mu_phi_minus) / (2 * dx)

        # Update rho
        rho[n+1, i] = rho[n, i] + dt * d_mu_phi_dz

    # Apply boundary conditions
    rho[n+1, 0], rho[n+1, -1] = boundary_condition(t[n+1])

    # Print the solution at multiple time steps
    if n % 1000 == 0:  # Adjust the step size as needed
        print(f"Time step {n+1}, Time {t[n+1]:.5f}")
        print(rho[n+1, :])

# Plotting the results
plt.figure(figsize=(10, 6))
for i in range(0, len(t), 1000):  # Adjust the step size as needed
    plt.plot(z, rho[i, :], label=f't={t[i]:.5f}')
plt.xlabel('z')
plt.ylabel(r'$\rho(z)$')
plt.legend()
plt.show()
```

B.3. Example five parameter normalizing flow

This is the code for the example using samples from the exponential density function. This code is very similar to the code used for the example with samples from the normal density with mean 2 and standard deviation 1. To find this a similar figure, change line 40 to

```
x = np.random.normal(loc=2, scale=1, size=100)
```

and change line 173 to "delta = 1000" and line 174 to "n_p=1". For the experiment. To find similar figures of the experiment in this paper, is also advisable to change line 177 to "for t in range(0,1000):" and line 201 to if "t%100 == 0:". To find the examples from the figures, the code is run in Jupyter Notebook.

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   from scipy.stats import expon
4   from scipy.stats import norm
5   from scipy.stats import expon
6
7
8   def generate_centered_exponential_samples(lam=1, num_samples=100):
9
10      # Generate samples from the exponential distribution
11      exponential_samples = expon.rvs(scale=1/lam, size=num_samples)
12
13      # Center the samples
14      centered_samples = exponential_samples #- np.mean(exponential_samples)
15
16      return centered_samples
17
18  def plot_samples_histogram(samples):
19
20      #plt.figure(figsize=(8, 6))
21
22      # Calculate the histogram
23      counts, bins = np.histogram(samples, bins=20)
24
25      # Normalize the histogram
26      bin_width = bins[1] - bins[0]
27      normalized_counts = counts / (counts.sum()* bin_width)
28
29      # Plot the histogram
30      plt.bar(bins[:-1], normalized_counts, width=bin_width, edgecolor='blue', color=
    'blue', alpha=0.7)
31
32      plt.title('Histogram of the data')
33      plt.xlabel('z=x')
34      plt.ylabel(r'$\rho(z)$', labelpad=20)
35      #plt.grid(True)
```

```
        plt.xlim(-4, 4)
        #plt.show()

    #Generate and plot the samples
    x = generate_centered_exponential_samples()
    plot_samples_histogram(x)



    def plot_likelihood(data):
        # Find the minimum and maximum values in the array
        min_val = np.min(data)
        max_val = np.max(data)

        # Create a plot
        plt.plot(data, linestyle='-')

        # Set the y-axis limits based on the min and max values
        plt.ylim(min_val - 1, max_val + 1)

        # Adding labels and title
        plt.xlabel('iteration')
        plt.ylabel('log-likelihood')
        plt.title('The evolution of the log-likelihood')

    def Log_likelihood(rho_tilde):
        return np.sum(np.log(rho_tilde))

    def plot_estimated_density(x,y):
        #initialization: we start with gues rho(x) is the standard normal density
        x_for_density = np.linspace(-5,5,1000)
        rhox_for_density = 1/np.sqrt(2*np.pi)*np.exp(-((x_for_density)**2)/2)
        plt.plot(x_for_density, rhox_for_density, label='Standard Normal Density')
        plt.scatter(x, y, label='Current density estimate')
        plt.xlabel('x')
        plt.ylabel('Density')
        plt.title('Plot of estimated Density')
        plt.legend(loc='upper right', fontsize='small')
        #plt.grid(True)
        #plt.show()
    #plot_estimated_density(x_for_density, rhox_for_density)

    x_axis = x
    #initial values
    #rho is at the start  the standard normal density function
    #the phi is at the start just the identity, so the total jacobian is 1
    rho0 = np.array([(1/(np.sqrt(2*np.pi)))*np.exp(-((xj)**2)/2) for xj in x])

    #de begin jacobian is de identiteit want phi0 is de identiteit
    Jacobian_t = np.array([1 for s in x])
    #we beginnen met phi0 = de identiteit, dan is alfa 0 en dus sigma, gamma en phi_0
    sigma = 0
    gamma = 0
    phi_0 = 0
    #after the identity tranformation, the data is still the same, thus phix = x
    phix = x

    #the current estimated density of the samples
    rho_tilde = rho0


    #mu(y) with mu as N(0,1)
    def mu(phix):
        return np.array([(1/(np.sqrt((2*np.pi))))*np.exp(-((zj)**2)/2) for zj in phix])


    def rho_tilde_t(phix, Jacobian_t):
        #jacobian_t is the jacobian det of the total transformation this far, not a
    smaller the jacobian of a smaller phi
        return np.array([Jacobian_t[j] * mu(phix)[j] for j in range(0, len(phix))])

```

```python

    def Jacobian_phi(x, sigma,gamma, epsilon, x0):
        return np.array([1-sigma + gamma*((1-sigma)*xj- x0)*(1-sigma)/np.sqrt(epsilon**2 +((1-sigma)*xj - x0)**2) for xj in x])


    def delta_Jacobian_t_gamma(x, sigma, epsilon, x0):
        return np.array([((1-sigma)*xj - x0)*(1-sigma)/(np.sqrt(epsilon**2+((1-sigma)*xj - x0)**2)) for xj in x])

    def delta_mu__t_gamma(phix, sigma, epsilon, x0, x):
        return np.array([(1/np.sqrt((2*np.pi)))*np.exp(-(phix[j]**2)/2)*(-phix[j])*np.sqrt(epsilon**2+((1-sigma)*x[j]-x0)**2) for j in range(0,len(phix))])

    def delta_rho_tilde_gamma(phix, Jacobian_t,sigma, epsilon, x0, x):
        Jacobian_phi1 = Jacobian_phi(x, sigma,gamma, epsilon, x0)
        delta_Jacobian_t_gamma1 = delta_Jacobian_t_gamma(x, sigma, epsilon, x0)
        delta_mu__t_gamma1 = delta_mu__t_gamma(phix, sigma, epsilon, x0,x)
        return (delta_Jacobian_t_gamma1 * mu(phix) + Jacobian_phi1*delta_mu__t_gamma1)*Jacobian_t

    def dLdgamma(phix, rho_tilde, delta_rho_tilde_gamma):
        m = len(phix)
        summation = 0
        for j in range(0,m):
            summation += (1/rho_tilde[j])*delta_rho_tilde_gamma[j]
        #print(summation)
        return 1/m*summation

    def delta_Jacobian_t_sigma(phix, sigma,gamma, epsilon, x0,x):
        return np.array([-1+gamma*(((-2*(1-sigma)*x[j] +x0)*np.sqrt(epsilon**2+((1-sigma)*x[j]-x0)**2)+x[j]*((1-sigma)*x[j]-x0)*(1-sigma)*((1-sigma)*x[j]-x0))/(np.sqrt(epsilon**2+((1-sigma)*x[j]-x0)**2)))/(epsilon**2 + ((1-sigma)*x[j]-x0)**2) for j in range(0, len(phix))])

    def delta_mu__t_sigma(phix, sigma,gamma, epsilon, x0,x):
        return np.array([(1/np.sqrt(2*np.pi))*np.exp(-(phix[j]**2)/2)*(-phix[j])*((-x[j])+(gamma*((1-sigma)*x[j]-x0)*(-x[j]))/np.sqrt(epsilon**2+((1-sigma)*x[j]-x0)**2)) for j in range(0, len(phix))])

    def delta_rho_tilde_sigma(phix, Jacobian_t,sigma, gamma, epsilon, x0,x):
        Jacobian_phi1 = Jacobian_phi(x, sigma,gamma, epsilon, x0)
        delta_Jacobian_t_sigma1 = delta_Jacobian_t_sigma(phix, sigma,gamma, epsilon, x0,x)
        delta_mu__t_sigma1 = delta_mu__t_sigma(phix, sigma,gamma,  epsilon,x0,x)
        return (delta_Jacobian_t_sigma1 * mu(phix) + Jacobian_phi1*delta_mu__t_sigma1)*Jacobian_t


    def dLdsigma(phix, rho_tilde, delta_rho_tilde_sigma):
        m = len(phix)
        summation = 0
        for j in range(0,m):
            summation += (1/rho_tilde[j])*delta_rho_tilde_sigma[j]
        return 1/m*summation


    def delta_mu__t_phi_0(phix, sigma,gamma,  epsilon,x0,x):
        return np.array([(1/np.sqrt(2*np.pi))*np.exp(-(zj**2)/2)*(-zj) for zj in phix])


    def delta_rho_tilde_phi_0(phix, Jacobian_t,sigma, gamma, epsilon, x0,x):
        Jacobian_phi1 = Jacobian_phi(x, sigma,gamma, epsilon, x0)
        delta_mu_t_phi_0 = delta_mu__t_phi_0(phix, sigma,gamma,  epsilon,x0,x)
        return (Jacobian_phi1*delta_mu_t_phi_0)*Jacobian_t#*(-Jacobian_t)

    def dLdphi_0(phix, rho_tilde, delta_rho_tilde_phi_0):
        m = len(phix)
        summation = 0
        for j in range(0,m):
            summation += (1/rho_tilde[j])*delta_rho_tilde_phi_0[j]
    return 1/m*summation




Delta_t = 1
delta = 100
n_p = 0.01
Log_evolution = []

for t in range(0,4000):
    #x0 is random sample of N(0,1)
    x0 =  np.random.normal(loc=0, scale=1)
    epsilon = np.sqrt(2*np.pi)*n_p*np.exp((x0**2)/2)
    delta_rho_tilde_gamma1 = delta_rho_tilde_gamma(phix, Jacobian_t,sigma, epsilon, x0,x)

    delta_rho_tilde_sigma1 = delta_rho_tilde_sigma(phix, Jacobian_t,sigma,gamma, epsilon, x0,x)

    delta_rho_tilde_phi_01 = delta_rho_tilde_phi_0(phix, Jacobian_t,sigma,gamma, epsilon, x0,x)

    gradL = [dLdgamma(phix,rho_tilde, delta_rho_tilde_gamma1), dLdsigma(phix, rho_tilde, delta_rho_tilde_sigma1), dLdphi_0(phix,rho_tilde, delta_rho_tilde_phi_01)]
    alpha = np.array([ Delta_t * grad / np.sqrt(1 + delta**2 * np.linalg.norm(grad)**2) for grad in gradL])
    gamma, sigma, phi_0 = alpha[0], alpha[1], alpha[2]
    x = phix
    phix = (1 - sigma) * phix + phi_0 + gamma * np.sqrt(epsilon**2 + ((1 - sigma) * phix - x0)**2)
    phix = np.nan_to_num(phix, nan=0.0)


    Jacobian_t = Jacobian_t * np.array([(1-sigma) + gamma * ((1-sigma)*xj-x0)*(1-sigma)/(np.sqrt(epsilon**2 + ((1-sigma)*xj-x0)**2)) for xj in x])

    #update the estimate density rho tilde t

    rho_tilde = rho_tilde_t(phix, Jacobian_t)
    Log_evolution.append(Log_likelihood(rho_tilde))
    if t%200 == 0:
            plt.figure(figsize=(20, 5))

            plt.subplot(1, 3, 1)
            plot_samples_histogram(x)

            plt.subplot(1, 3, 2)
            plot_likelihood(Log_evolution)

            plt.subplot(1,3,3)
            plot_estimated_density(x_axis,rho_tilde)
            plt.tight_layout()
            plt.show()
            print("log-likelihood", Log_likelihood(rho_tilde))
```

Bibliography

[1] Fraleigh Beauregard. *Linear algebra.* Pearson South Africa, 2000.

[2] Fetsje Bijma, Marianne Jonker, and Aad van der Vaart. Inleiding in de statistiek. epsilon uitgaven, 2017.

[3] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[4] Vladimir I Bogachev. Operations on measures and functions. *Measure Theory*, pages 175–248, 2007.

[5] Prachi Goyal, Sunil Gulia, and S.K. Goyal. Identification of air pollution hotspots in urban areas - an innovative approach using monitored concentrations data. *Science of The Total Environment*, 798: 149143, 2021. ISSN 0048-9697. doi: https://doi.org/10.1016/j.scitotenv.2021.149143. URL `https://www.sciencedirect.com/science/article/pii/S0048969721042169`.

[6] Walter Greiner and Joachim Reinhardt. *Field quantization.* Springer Science & Business Media, 2013. Page 37.

[7] Geoffrey Grimmett and Dominic JA Welsh. *Probability: an introduction.* Oxford University Press, 2014.

[8] Erwin Kreyszig, Herbert Kreyszig, and Edward J. Norminton. *Advanced Engineering Mathematics.* John Wiley and Sons, 10 edition, 2011. ISBN 978-0-470-45836-5.

[9] Reabal Najjar. Redefining radiology: a review of artificial intelligence integration in medical imaging. *Diagnostics*, 13(17):2760, 2023.

[10] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[11] Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964.

[12] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[13] Esteban G Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

[14] EM Wahba. Derivation of the differential continuity equation in an introductory engineering fluid mechanics course. *International Journal of Mechanical Engineering Education*, 50(2):538–547, 2022.