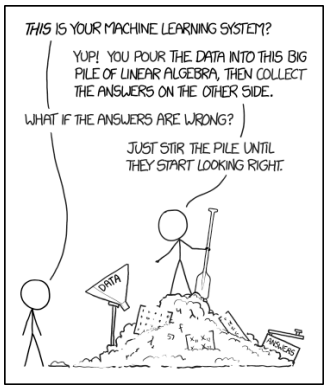
TÉCNICO LISBOA



Variational Mixture of Normalizing Flows

Guilherme P. Grijó Pires

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Mário Alexandre Teles de Figueiredo

Examination Committee

Chairperson: Prof. Full Name
Supervisor: Prof. Full Name 1 (or 2)
Member of the Committee: Prof. Full Name 3

Month Year

What I cannot create, I do not understand.

- Richard Feynman

Acknowledgments

A few words about the university, financial support, research advisor, dissertation readers, faculty or other professors, lab mates, other friends and family...

Resumo

Inserir o resumo em Português aqui com o máximo de 250 palavras e acompanhado de 4 a 6 palavras-chave...

Palavras-chave: palavra-chave1, palavra-chave2,...

Abstract

Keywords:

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α	Angle of attack.
β	Angle of side-slip.
κ	Thermal conductivity coefficient.
μ	Molecular viscosity coefficient.
ρ	Density.

Roman symbols

C_D	Coefficient of drag.
C_L	Coefficient of lift.
C_M	Coefficient of moment.
p	Pressure.
$\mathbf{u}$	Velocity vector.
u, v, w	Velocity Cartesian components.

Subscripts

∞	Free-stream condition.
i, j, k	Computational indexes.
n	Normal component.
x, y, z	Cartesian components.
ref	Reference condition.

Superscripts

*	Adjoint.
T	Transpose.

Glossary

CFD Computational Fluid Dynamics is a branch of fluid mechanics that uses numerical methods and algorithms to solve problems that involve fluid flows.

CSM Computational Structural Mechanics is a branch of structure mechanics that uses numerical methods and algorithms to perform the analysis of structures and its components.

MDO Multi-Disciplinar Optimization is an engineering technique that uses optimization methods to solve design problems incorporating two or more disciplines.

Chapter 1

Introduction

Insert your chapter material here...

1.1 Motivation

Relevance of the subject...

1.2 Topic Overview

Provide an overview of the topic to be studied...

1.3 Objectives

Explicitly state the objectives set to be achieved with this thesis...

1.4 Thesis Outline

Briefly explain the contents of the different chapters...

Chapter 2

Probabilistic Modelling

2.1 Introduction

Probabilistic modelling is a set of techniques that leverage probability distributions and random variables to posit, test and refine hypothesis about the behaviour of systems. Given observations of a system, the task of probabilistic modelling normally boils down to finding a probability distribution which:

- Is consistent with the observed data;
- Is consistent with **new**, previously unobserved data, originated in the same system.

This probability distribution is commonly called the *model*. A good model will be a good *emulator* of the true generative process that originated the observed data. In loose terms, this can be summarized as:

$$\text{data} \sim p(\text{data}|\text{hypothesis}^*), \tag{2.1}$$

where hypothesis* is the optimal hypothesis.

Via Bayes' Law, we can write:

$$p(\text{hypothesis}|\text{data}) = \frac{p(\text{data}|\text{hypothesis})p(\text{hypothesis})}{p(\text{data})} \tag{2.2}$$

In practice, a modeller will search for an hypothesis that maximizes (some form, or approximation of) this expression. For simpler problems, this search happens in closed-form, i.e., there is an expression to compute the optimal hypothesis, given data. However, for most real-world problems there is no closed-form solution, and the modeller has to resort to algorithms and approximations, and will only be able to find **local** optima for the above expression, in most cases.

It's also worth noting that there are effectively infinite candidate distributions - each one an hypothesis for how the system at hand generates data. It is common to make use of domain knowledge and assume the true system has a certain intrinsic structure and form, and to use these assumptions to constrain the space of candidate hypothesis. Assumptions about structure usually translate to conditional independence claims between some or all of the observed variables; assumptions about form translate

into the use of specific parametric families to govern some or all of the observed variables. These assumptions are commonly connected between themselves (for example, when conjugate likelihood-prior pairs are used).

When parametric forms are used, an hypothesis is uniquely defined by the set of parameters it requires - commonly called θ[1].

2.2 Model Complexity

Intuitively, the dimension of θ is deeply connected with the expressiveness of the distribution. In practice, this translates to the observation that if we make the model[2] expressive enough, it can fit the observed data arbitrarily well. Naïvely, this would be a desirable characteristic to exploit - it's always possible to increase the likelihood by adding parameters to the model. However, increasing model complexity normally comes at the expense of generalization. This phenomenon is commonly referred to as *overfitting*, and there are several angles from which to explain it and interpret it. Namely:

- The classical perspective is that of the bias-variance tradeoff. To understand this, consider the concept of an Hypothesis Class - a set of hypotheses in which, via some procedure, the modeller will search for an hypothesis that is consistent with the observed data, and is expected to generalize to unseen data. Said procedure is what is normally referred to as *fitting* the model to the data. In the case of parametric models, the set of models of a given parametric form, with a parameter-vector of a certain fixed dimension, is an example of an Hypothesis Class. Intuitively, a more complex Hypothesis Class is more likely to contain the true hypothesis (or a good approximation to it). However, the more complex the Hypothesis Class, the larger the search-space - the higher the number of candidate hypothesis. In this sense, an increase in the size of the search-space often translates into an increase of the sensitivity to the problem variables (in the case of learning and inference, this means sensitivity to initialization and to the data used to fit). Conversely, a simpler model will constitute a smaller search-space, hence the search procedure will be less sensitive to initalization and problem variables. However, the true hypothesis (or a good approximation to it) is less likely to be contained in it - precisely because it is a smaller Hypothesis Class. The bias-variance tradeoff is a summary of these observations: a highly complex model is potentially able to achieve a low expected error on observed data (low bias), but will tend to be extremely sensible to small variations on its input (high variance). Conversely, a simpler model will be more robust to variations on its input (low variance), but won't have the same modelling capacity and will produce a larger expected error (high bias).[3]

[1] For the type of models and problems dealt with in this work, I will assume θ is finite, but it's worth noting that there are models for which the dimension of θ *grows* with the dataset size. These are called non-parametric models. They come with their own advantages and disadvantages, which are out of the scope of this work.

[2] Throughout this work I will be using the words *model* and *distribution* almost interchangeably, making it clear when context isn't enough.

[3] The number of parameters is far from being the best measure of complexity of a model. Nevertheless, it is a good proxy to compare model complexity between models of the same parametric family. However, recent work by Belkin et al. [1] shows that modern machine learning contexts, in which the number of parameters is far larger than in classical settings, have to be understood under a measure of model complexity different than the traditional ones. This is because it is now common practice to fit highly overparameterized models to a point of interpolation (close to zero training error), still being able to achieve good

- Andrey Kolmogorov's and Gregory Chaitin's ideas on Algorithmic Information Theory [2], and Kolmogorov complexity are another useful lens through which to regard this question. Consider that data are measurements of phenomena. Modelling is concerned with finding the laws that explain/govern these phenomena. Intuitively, if the laws are as complex as the data they intend to explain, they aren't explaining anything. AIT formalizes this notion by borrowing the concept of *program* to define the generative process by which observed data comes to existence. The complexity of a dataset is then easy to define: it is the size of the **smallest**[4] program that generates the observed data. And the appropriate unit with which to measure the size of a program - and, as we've now seen, the complexity of a dataset - is bits[5]. The parallel between these ideas and the question of overfitting is thus easy to make: a program (or a model and its parameter vector) is useful if it *compresses* the data, intuitively because to do so it leverages the patterns therein, which are the object of interest in the modelling task.

Both of these lines of reasoning make clear that there is a certain balance in complexity that a good model has to achieve: it should be parsimonious enough that it won't overfit, but flexible enough that it is able to properly explain the observed data. There are strategies to make this mathematically objective. Some of those methods are the Bayesian Information Criterion, the Akaike Information Criterion and the Minimum Description Length.

2.3 MAP and ML Estimation

Once the parametric form of the model is defined, the task at hand becomes the discovery of the parameter vector θ that best explains the observed data, within the defined parametric family.

The naïve (but often the only possible) approach is to maximize what is called the likelihood function, given by:

$$\mathcal{L}(\theta) = p(x|\theta) \tag{2.3}$$

This approach is called *Maximum Likelihood Estimation*. Note that $\mathcal{L}$ is a function of θ. Depending on the model, finding θ^* - the optimum - can be as simple as computing an analytical expression, or as difficult as using gradient-based methods to optimize a non-convex objective. In the latter case, a local optima is usually the best one can expect to obtain.

Another approach, called *Maximum a posteriori*, works by retrieving the mode of the posterior distribution of the parameter-vector, given by:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{2.4}$$

It's easy to see that these two approaches are intimately related. MAP differs from ML by the fact

generalization.

[4]Note the emphasis on "smallest" - this is because any program can be made arbitrarily redundant, and thus arbitrarily large.

[5]Or the basic unit of memory of the computer where the data generating program would run

that it employs the prior $p(\theta)$ to give different weights to different hypothesis (i.e., different instances of θ). This is useful if there is domain knowledge available that can be encoded in the prior.

It is worth noting that ML is a special case of MAP, when the prior is uniform.

2.4 Structure and Latent Variables

In some cases, one might want to leverage some available domain knowledge. This often translates into assuming that there is some latent structure in the data. This structure is commonly encoded into latent variables and their influence over the observable variables.

In this scenario, we become interested in the distribution given by $p(x, z, \theta_x, \theta_z)$, where z is the latent variable, θ_x is the parameter vector for the distribution over $\mathcal{X}$, and θ_z is the parameter vector for the distribution over $\mathcal{Z}$.

For structure and latent variables to be useful we normally make the additional assumption that we have the ability of factorizing their joint distribution in ways that make it tractable. If we have a dataset $\mathbf{X}$, with N i.i.d. samples $x_1, x_2, x_3, ..., x_N$ and N latent variables $z_1, z_2, z_3, ..., z_N$, one common factorization is:

$$p(\mathbf{X}, \mathbf{Z}, \theta) = \Big(\prod_{i=1}^{N} p(x_i|z_i, \theta_x) p(z_i|\theta_z) \Big) p(\theta_x) p(\theta_z). \tag{2.5}$$

It's also possible that the samples in $\mathbf{X}$ have some sort of causal relation, for instance if they occur ordered in time. In this case, they are not i.i.d. One way to encode this assumption is to posit an *autoregressive* model, i.e., a model in which a random variable depends on the variables that come before it. If each random variable depends solely on the random variable that precedes it, this is called a Markov Model. A common variation of the Markov Model is the Hidden Markov Model, where the autoregressive part of the model is present only in the latent variables:

$$p(\mathbf{X}, \mathbf{Z}, \theta) = p(z_1) \Big(\prod_{i=2}^{N} p(x_i|z_i, \theta_x) p(z_i|z_{i-1}, \theta_z) \Big) p(\theta_x) p(\theta_z). \tag{2.6}$$

These are merely examples of models with different structure assumptions encoded into them. Normally, if the structure has a certain regularity, it's possible to exploit it to obtain tractable (approximate) inference and estimation methods.

2.5 Mixture Models and the EM Algorithm

Mixture Models are a subset of the structure "family" described in 2.5, and they have a central role in this work.

In a Mixture Model there is a discrete latent variable z_i which selects one of K components from

which an observation x_i will be sampled. This can be summarized as:

$$z_i \sim p(z_i|\pi) \tag{2.7}$$

$$x_i \sim p(x_i|z_i) \tag{2.8}$$

The probability of x_i being sampled from component k (that is, the proability of $z_i = k$) is commonly referred to as the weight of component k.

It's common to assume that all of the K components are part of the same parametric family. In that case, we can rewrite the above as:

$$z_i \sim p(z_i|\pi) \tag{2.9}$$

$$x_i \sim p(x_i|\theta_{z_i}), \tag{2.10}$$

where it is made evident that the discrete variable z_i is selecting the **parameter vector** to be used for sample x_i.

The most discussed mixture model is the Gaussian Mixture Model, in which the K components of the model are Gaussian distributions.

The EM Algorithm

The Expectation-Maximization algorithm is the most commonly used algorithm to fit Mixture Models[6].

The starting point of EM is the realisation that if all variables were observed, it would be easy to apply ML or MAP estimation for the parameters. Given that, the algorithm can be generally described as an alternation between two steps:

- **E-step**: infer the most probable values of the unobserved variables. (In the case of Mixture Models, this corresponds to inferring the discrete variables that select the component from which each observed data point was sampled. This can be roughly thought of as a cluster assignment).
- **M-step**: given the observed variables and the values inferred on the previous step, optimize the model parameters. (In the case of Mixture Models, this correponds to inferring the parameters of each component, and its weight).

A more rigorous description of the procedure follows. Consider the following expression:

$$\ell_c(\theta) = \sum_{i=1}^{N} \log p(x_i, z_i|\theta) \tag{2.11}$$

This is the complete data log-likelihood. Like the likelihood function, it is a function of θ, which is easy to compute, given all the x_i and z_i. However, the z_i aren't observed.

To overcome this, let us define the expected complete data log likelihood:

$$Q(\theta, \theta^{(t-1)}) = \mathbb{E}_{Z|X,\theta^{(t-1)}}[\ell_c(\theta)|X, Z, \theta^{(t-1)}], \tag{2.12}$$

[6] Although it can be used to fit other types of models.

where $\theta^{(t)}$ represents the value of θ at time step t of the fitting procedure.

Note that the expectation is w.r.t. Z, given X and $\theta^{(t-1)}$. It is the expected value of $\ell_c(\theta)$, given the parameter values obtained at the previous step of the the algorithm. Depending on the nature of Z, this expectation can be obtained either in closed form or approximated, for instance via samples of z_i (if a sampling procedure is available).

In the case of Mixture Models, where the z_i are instances of a categorical random variable, the expression for $Q(\theta, \theta^{(t-1)})$ can be made simpler as such:

$$
\begin{aligned}
Q(\theta, \theta^{(t-1)}) &= \mathbb{E}_{Z|X,\theta^{(t-1)}}[\ell_c(\theta)|X, Z, \theta^{(t-1)}] && (2.13)\\
&= \mathbb{E}_{Z|X,\theta^{(t-1)}}[\sum_{i=1}^{N} \log p(x_i, z_i|\theta)] && (2.14)\\
&= \mathbb{E}_{Z|X,\theta^{(t-1)}}[\sum_{i=1}^{N} \log \prod_{k=1}^{K} \pi_k p(x_i|\theta_k)^{\mathbb{I}(z_i=k)}] && (2.15)\\
&= \mathbb{E}_{Z|X,\theta^{(t-1)}}[\sum_{i=1}^{N} \sum_{k=1}^{K} \log \Big(\pi_k p(x_i|\theta_k)^{\mathbb{I}(z_i=k)}\Big)] && (2.16)\\
&= \mathbb{E}_{Z|X,\theta^{(t-1)}}[\sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{I}(z_i=k) \log \Big(\pi_k p(x_i|\theta_k)\Big)] && (2.17)\\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} \underbrace{\mathbb{E}_{Z|X,\theta^{(t-1)}}[\mathbb{I}(z_i=k)]}_{\text{Let this quantity be represented by } r_{ik}} \log \Big(\pi_k p(x_i|\theta_k)\Big) && (2.18)\\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \log \Big(\pi_k p(x_i|\theta_k)\Big) && (2.19)\\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \log \pi_k + \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \log p(x_i|\theta_k) && (2.20)\\
& && (2.21)
\end{aligned}
$$

The quantity defined above as r_{ik} is referred to as the responsability. It is trivial to arrive at a closed-form expression for it, given the value of θ arrived at the previous iteration, i.e. $\theta^{(t-1)}$:

$$
\begin{aligned}
r_{ik} &= p(z_i = k|x_i\,; \theta^{(t-1)}) && (2.22)\\
&= \frac{p(z_i = k, x_i\,; \theta^{(t-1)})}{p(x_i\,; \theta^{(t-1)})} && (2.23)\\
&= \frac{p(z_i = k, x_i\,; \theta^{(t-1)})}{p(x_i\,; \theta^{(t-1)})} && (2.24)\\
&= \frac{p(z_i = k, x_i\,; \theta^{(t-1)})}{\sum_{k'} p(z_i = k', x_i\,; \theta^{(t-1)})} && (2.25)\\
&= \frac{\pi_k p(x_i|\theta_k^{(t-1)})}{\sum_{k'} \pi_{k'} p(x_i|\theta_{k'}^{(t-1)})} && (2.26)
\end{aligned}
$$

The EM algorithm for Mixture Models then becomes an alternation between the following two steps:

- **E-step**: Compute the responsabilies r_{ik} for each x_i.

- **M-step**: Given r_{ik}, solve the following optimization problem:

$$\theta^{(t)} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta, \theta^{(t-1)}) \underbrace{+ \log p(\theta)}_{\text{Optional, if we want to do MAP estimation}}, \tag{2.27}$$

  which can have a closed-form solution in some of the simplest Mixture Models, like Gaussian Mixture Models, but can require a gradient-based optimization procedure for more flexible models.

2.6 Approximate Inference

Take the expression $p(x, z, \theta)$. For simplicity, let us consider θ as part of the latent variables z. This means that the model is simply written as the joint distribution: $p(x, z)$. *Inference*[7] is the task of finding the most probable z after having observed x . Specifically, the goal is to find the posterior distribution of z, given x, i.e.: $p(z|x)$.

Recall Bayes' Law:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{2.28}$$

$$= \frac{p(x|z)p(z)}{\int p(x|z')p(z')dz'} \tag{2.29}$$

For the vast majority of cases, the integral on the denominator will be intractable. To overcome this difficulty we normal resort to two families of methods: Monte-Carlo methods - which are out of the scope of this work - and Variational methods.

2.6.1 Variational Methods

Variational methods work by turning the problem of integration into one of optimization. They propose a family of parametric distributions, and then optimize the parameters so as to minimize the "distance" between the approximate (normally called "variational") distribution and the distribution of interest.

There are two ways to derive the most commonly used objective function for this problem, which will be detailed in the two following subsections.

Kullback-Leibler Divergence

The Kullback-Leibler divergence is a measure[8] of the distance between two probability distributions p, and q. It is given by:

[7] If we hadn't collapsed θ into z and were instead handling separately, we would call **inference** to the task of finding z and **learning** to the task of finding θ

[8] Note that the KL divergence isn't symmetric and as such I haven't called it a *metric*

$$KL(q||p) = \int q \log \frac{q}{p} \tag{2.30}$$

In the setting of inference, p is the posterior $p(z|x)$ and q is a distribution in some parametric family, with parameters ϕ, i.e., $q(z;\phi)$. However, it's clear that we can't compute the Kullback-Leibler directly, because it requires the knowledge of both the distributions, and finding $p(z|x)$ is precisely the task at hand. Let us expand the KL divergence expression:

$$\begin{aligned} KL(q||p) &= \int q(z)(\log q(x) - \log p(z|x))dz && (2.31)\\ &= \int q(z)(\log q(z) - (\log p(x,z) - \log p(x)))dz && (2.32)\\ &= \mathbb{E}_q[\log q(z)] - \mathbb{E}_q[\log p(x,z)] + \mathbb{E}_q[\log p(x)] && (2.33)\\ &= \mathbb{E}_q[\log q(z)] - \mathbb{E}_q[\log p(x,z)] + \log p(x) && (2.34)\\ & && (2.35) \end{aligned}$$

The last term is constant w.r.t $q(z)$. In that sense, for a fixed $p(x)$, minimizing the KL divergence is equivalent to minimizing

$$\mathbb{E}_q[\log q(z)] - \mathbb{E}_q[\log p(x,z)], \tag{2.36}$$

which is equivalent to maximizing

$$\mathbb{E}_q[\log p(x,z)] - \mathbb{E}_q[\log q(z)]. \tag{2.37}$$

This quantity is commonly refered to as ELBO - Evidence Lower BOund. It can be rewritten as:

$$\begin{aligned} ELBO(q) &= \mathbb{E}_q[\log p(x,z)] - \mathbb{E}_q[\log q(z)] && (2.38)\\ &= \mathbb{E}_q[\log p(x|z)] + \mathbb{E}_q[\log p(z)] - \mathbb{E}_q[\log q(z)] && (2.39) \end{aligned}$$

In this form, each term of the ELBO has an easily interpretable role:

- $\mathbb{E}_q[\log p(x|z)]$ tries to maximize the conditional likelihood of x. That can be seen as assigning high probability mass to values of z that *explain* x well.
- $\mathbb{E}_q[\log p(z)]$ is the symmetric of the crossentropy between $q(z)$ and $p(z)$. Maximizing this quantity is equivalent of minimizing that crossentropy. This can be regarded as a regularizer that discourages $q(z)$ of being too different from the prior $p(z)$.
- $-\mathbb{E}_q[\log q(z)]$ is the entropy of $q(z)$. Maximizing this term incentivizes the probability mass of $q(z)$ to be spread out: another form of regularization.

A lower bound on $\log p(x)$

Another way of approaching the intractable posterior is to start by stating that our inherent goal is to maximize $p(x)$, or equivalently $\log p(x)$. Given that, consider the following:

$$\log p(x) = \log \int p(x, z) dz \tag{2.40}$$

$$= \log \int q(z) \frac{p(x, z)}{q(z)} dz \tag{2.41}$$

$$= \log \mathbb{E}_q [\frac{p(x, z)}{q(z)}] \tag{2.42}$$

$$\geq \mathbb{E}_q [\log \frac{p(x, z)}{q(z)}] \tag{2.43}$$

$$\geq \mathbb{E}_q [\log p(x, z)] - \mathbb{E}_q [q(z)] \tag{2.44}$$

To understand this derivation, consider Jensen's inequality, given (in one of its forms) by:

$$\phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(X)], \tag{2.45}$$

where $\phi(.)$ is a convex function.

If $\xi(.)$ is a concave function, then $-\xi(.)$ is a convex function, and we obtain the reverse inequality (substituting $\phi(.)$ with $-\xi(.)$ in the inequality in 2.45):

$$-\xi(\mathbb{E}[X]) \leq \mathbb{E}[-\xi(X)] \tag{2.46}$$

$$\xi(\mathbb{E}[X]) \geq \mathbb{E}[\xi(X)] \tag{2.47}$$

This form is the most useful for us, since $\log$ is a concave function. Using this, the step between 2.42 and 2.43 is made obvious.

Note that the right-hand side of 2.44 is the same quantity we arrived at in 2.37, and that it is a lower-bound on the quantity we want to maximize, and so we want to maximize it. It's worth noting that when $q(z) = p(z|x)$, the bound is tight.

Chapter 3

Normalizing Flows

3.1 Introduction

The best known and studied probability distributions are rarely expressive enough for real-world datasets. However, they have properties that make them amenable to work with, for instance: tractable parameter estimation and closed-form likelihood functions.

As has been described, one way to obtain more expressive models is to assume the existence of latent variables, leverage certain factorization structures, and to use well-known distributions for the individual factors of the product that constitutes the model's joint distribution. By using these structures and choosing specific combinations of distributions (namely, conjugate prior-likelihood pairs), these models are able to stay tractable - normally via bespoke estimation/inference/learning algorithms.

Another approach to obtaining expressive probabilistic models is to apply transformations to a simple distribution, and use the Change of Variables formula to compute probabilities in the transformed space. This is the basis of Normalizing Flows, an approach proposed by in [**?**], and which has since evolved and developed into the basis of multiple SoA techniques for density estimation [TODO: cite examples here].

3.2 Change of Variables

Given a probability distribution $p(z)$, with probability density function $f_Z(.)$, and a bijective and continuous function g, it's possible to write an expression for the probability density function $f_Y(.)$ of the random variable y that is obtained by applying g to samples of $p(z)$:

$$\text{if } z \sim p(z) \tag{3.1}$$

$$\text{and } y = g(z) \tag{3.2}$$

$$\text{then } f_Y(y) = f_Z(g^{-1}(y)) \left| \det \left(\frac{d}{dy} g^{-1}(y) \right) \right| \tag{3.3}$$

For this to be useful, some objects have to be easily computable:

- $f_Z(z)$ - the starting distribution's probability density function. It is assumed that there is a closed-form expression to compute this. In practice, this is normally one of the basic distributions (Gaussian, Uniform, etc.)
- $\det\left(\frac{d}{dy}g^{-1}(y)\right)$ - the determinant of the Jacobian matrix of $g^{-1}(.)$. For most transformations this is not "cheap" to compute. As will be shown, the main challenge of Normalizing Flows is to find transformations that are expressive and for which the determinants of their Jacobian matrices are "cheap" to compute.

3.3 Normalizing Flows

Let us have L transformations h_ℓ that fulfill the above two points, and let z_ℓ be the result of applying transformation $h_{\ell-1}$, with the exception of z_0, which is obtained by sampling from $p(z_0)$, the base distribution. Furthermore, let g be the composition of the L transformations.

Applying the Change of Variables formula:

$$\text{if } z_0 \sim p(z_0) \tag{3.4}$$

$$\text{and } y = h_L \circ h_{L-1} \circ ... \circ h_0(z_0) \tag{3.5}$$

$$\text{then } f_Y(y) = f_Z(g^{-1}(y))\left|\det\left(\frac{d}{dy}g^{-1}(y)\right)\right| \tag{3.6}$$

$$= f_Z(g^{-1}(y))\prod_{\ell=1}^{L}\left|\det\left(\frac{d}{dz_{\ell+1}}h_\ell^{-1}(z_{\ell+1})\right)\right| \tag{3.7}$$

$$= f_Z(g^{-1}(y))\prod_{\ell=1}^{L}\left|\det\left(\frac{d}{dz_\ell}h_\ell\left(h_\ell^{-1}(z_{\ell+1})\right)\right)\right|^{-1} \tag{3.8}$$

Replacing $h_\ell^{-1}(z_{\ell+1}) = z_\ell$ in 3.8:

$$f_Y(y) = f_Z(g^{-1}(y))\prod_{\ell=1}^{L}\left|\det\left(\frac{d}{dz_\ell}h_\ell\left(z_\ell\right)\right)\right|^{-1} \tag{3.9}$$

$$\log f_Y(y) = \log f_Z(g^{-1}(y)) - \sum_{\ell=1}^{L}\log\left|\det\left(\frac{d}{dz_\ell}h_\ell(z_\ell)\right)\right| \tag{3.10}$$

Depending on the task, one might prefer to replace the second term in 3.10 with a sum of log-abs-determinants of the Jacobians of the inverse transformations. This switch would imply replacing the minus sign before the sum with a plus sign:

$$\log f_Y(y) = \log f_Z(g^{-1}(y)) + \sum_{\ell=0}^{L-1}\log\left|\det\left(\frac{d}{dz_{\ell+1}}h_\ell^{-1}(z_{\ell+1})\right)\right| \tag{3.11}$$

3.4 Flow examples

In the Normalizing Flows literature, a Flow is the name given to each of the individual transformations that are composed to form a Normalizing Flow.

3.5 Real NVP

Chapter 4

Variational Mixture of Normalizing Flows

4.1 Introduction

Lorem ipsem Lorem ipsemLorem ipsemLorem ipsemLorem ipsem Lorem ipsemLorem ipsemLorem ipsemLorem ipsemLorem ipsemLorem ipsem Lorem ipsemLorem ipsemLorem ipsemLorem ipsem Lorem ipsemLorem ipsemLorem ipsemLorem ipsemLorem ipsem Lorem ipsemLorem ipsemLorem ipsemLorem ipsemLorem ipsem

Chapter 5

Conclusions

Insert your chapter material here...

5.1 Achievements

The major achievements of the present work...

5.2 Future Work

A few ideas for future work...

Bibliography

[1] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine learning and the bias-variance trade-off. *arXiv*, Dec 2018. URL `https://arxiv.org/abs/1812.11118`.

[2] G. Chaitin. Doing mathematics differently. *Inference - International Review of Science*, Volume Two(Issue One), February 2016. URL `https://inference-review.com/article/doing-mathematics-differently`.