Optimal Importance Sampling in Quantum Monte Carlo for Lattice Models

Blaž Stojanovič

Supervisor: Prof. A. Lamacraft

Department of Physics

This dissertation is submitted for the degree of
Master of Philosophy in Scientific Computing

St. John's College April 2021

I would like to dedicate this thesis to …

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 15,000 words including appendices, figure legends, and tables.

Blaž Stojanovič
April 2021

Acknowledgements

And I would like to acknowledge ...

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

CNN	Convolutional Neural Network
DL	Deep Learning
DMC	Diffusion Quantum Monte Carlo
GAN	General Adversarial Network
ML	Machine Learning
NN	Neural Network
QMC	Quantum Monte Carlo
VAE	Variational Autoencoder
VMC	Variational Quantum Monte Carlo

Chapter 1

Introduction

Chapter 2

Background

2.1 Lattice models

2.1.1 Historical introduction

2.1.2 The Schrödinger equation

2.1.3 Canonical examples of lattice models

Quantum Ising model

Heisenberg model

Bose-Hubbard model

2.2 Quantum Monte Carlo

2.2.1 Overview

Quantum Monte Carlo is a class of methods that uses statistical sampling to directly deal with high-dimensional integration that arises from working with the many-body wave function. QMC methods are among the most accurate achieving chemical accuracy for smaller systems [15], and can achieve any degree of statistical precision sought. Quantum Monte Carlo is also very versatile and can be applied at both zero and finite temperatures [4]. The most basic zero temperature QMC method is variational QMC (VMC). The method is composed of two parts, firstly it directly evaluates the variational energy $E_V = \langle\Psi_T|\hat{H}|\Psi_T\rangle/\langle\Psi_T|\Psi_T\rangle$ of the system using Monte Carlo integration and a trial wave function Ψ_T. Secondly the parameters of the trial wave function are optimised such as to minimise the variational energy E_V, giving the method its name. The first application of VMC was to ground state ^{4}He [29] and was later extended for studying many-body fermionic systems [8]. A way of obtaining excitation energies using VMC is to use a trial wave function that models an excited state of the system, if the trial wave function obeys a certain symmetry, the variational principle guarantees that this VMC energy calculation gives an upper bound on the lowest exact eigenstate of this symmetry. Furthermore, the method can be extended to study non-equilibrium properties of bosonic [7, 6], and fermionic [22] systems. The main advantage of VMC is its simplicity while the main drawback is that the accuracy is limited by the flexibility and form of the trial wave function [4]. As such VMC is usually employed as a first step in more advanced QMC simulations.

Projector quantum Monte Carlo (PMC) is a class of QMC methods which are in essence nothing more than stochastic implementations of the power method to obtain the dominant eigenvector of a matrix or a kernel function [18]. Their distinct advantage over VMC is that they are not constrained by our parametrisation of the trial wave function, as they can describe arbitrary probability distributions. The projector $\hat{P}$ has to be chosen in such a way, that the ground state of the system becomes the dominant eigenvector, i.e. $|\Psi_0\rangle = \lim_{n\to\infty}\hat{P}^n|\Psi_T\rangle$. Different ways of achieving this give rise to different flavours of PMC methods, e.g choice of space (real or orbital space) in which the walk is done and choosing either first or second quantisation. Using an exponential projector $\hat{P} = e^{\tau(E_T\mathbb{1}-\hat{H})}$ can be interpreted as propagation in imaginary time $\tau \to it$ in turn transforming the Schrödinger equation into a diffusion equation, which is a continuous limit of the random walk and lends itself to stochastic integration [35]. Directly sampling from the exact Green function is known as Projector Green Function Monte Carlo (GFMC) method [23, 24]. A convenient approximation to GFMC is its short-time approximation which leads to one of the most popular QMC methods, diffusion Monte Carlo (DMC) [15, 35]. In this regime one can exploit analytical solutions to diffusion and rate problems to write an explicit form of the Green's function. Additionally, by using the Trotter-Suzuki formula the time-step bias can be expressed and accounted for [4]. DMC is statistically implemented by using a population of walkers which either branch or die, over which the average is calculated. Reptation quantum Monte Carlo [35] (RMC) is an alternative formulation which only uses a single walker, and instead of branching and dying the MC moves mutate the path of that single walker. The use of a guiding wave function for importance sampling greatly improves the statistical efficiency of PMC methods. The guiding wave function is usually obtained by means of VMC or some mean field calculation.

PMC methods suffer from the *sign problem*, which is present in Markov chain simulation of distributions that are not strictly positive, thus in fermionic and frustrated systems [18]. The problem refers to an exponential decrease in sampling efficiency with system size. The search for solutions of this problem is still an area of active research [15] but is in practice remedied by the *fixed-node* approximation [2]. It imposes a boundary condition into the projection, such that the projected state shares the same zero crossings (nodal surface) with a trial wave function, which is again usually obtained with VMC. The projected state is now only exact when the nodal surface is exact. Nevertheless this approximation is quite accurate [15]. Fixed node is widely used, one of its first applications being the electron gas [10], which is used in parameterisations of the exchange correlation functional in LSDA [42].

Quantum Monte Carlo methods have had a lot of success at finite temperatures. Auxiliary-field Monte Carlo, or Path Integral Monte Carlo [9], which leads to ring-polymer molecular dynamics, may be used for this purpose. Additionally QMC is not limited to continuum space applications and has been extensively used to study lattice models, notable examples being the cluster/loop algorithm and the worm algorithm [18, 33].

Quantum Monte Carlo methods are generally more computationally expensive than DFT approaches, but on the other hand QMC codes are, as a rule of thumb, simpler to implement. Furthermore, since the wave function does not need to be stored directly, QMC has reasonable storage requirements. The high computational cost of the QMC methods is remedied by the fact that they are intrinsically parallelisable, the core calculation involves generating (pseudo)-random numbers, performing a simple calculation and in the end averaging over the results. Therefore, implementations of QMC algorithms that have been applied to practical problems are optimised to run on massively parallel hardware with little overhead [31]. Finally, the repetitive nature of the Monte Carlo calculation lends itself to hardware acceleration using either graphical processing units (GPUs) or field-programmable gate arrays (FPGAs) [4].

2.2.2 Implementation

Monte Carlo Importance Sampling

The most common application of Monte Carlo methods is evaluation of integrals in high dimensional space. There MC has a distinct advantage over quadrature methods, as the statistical error decreases with the square root of samples irregardless of the dimensionality of the problem. Integrals of a function $g(\mathbf{R})$

$$I = \int g(\mathbf{R})\mathrm{d}\mathbf{R}, \tag{2.1}$$

where $\mathbf{R}$ is the *configuration* of the system or simply a *walker*, can be integrated by use of an *importance function* $\mathrm{P}(\mathbf{R})$, where $\int d\mathbf{R}\mathrm{P}(\mathbf{R}) = 1$ and $\mathrm{P}(\mathbf{R}) \geq 0$. The integral can be rewritten in the form

$$\int g(\mathbf{R})\mathrm{d}\mathbf{R} = \int \frac{g(\mathbf{R})}{\mathrm{P}(\mathbf{R})}\mathrm{P}(\mathbf{R})\mathrm{d}\mathbf{R} = \int f(\mathbf{R})\mathrm{P}(\mathbf{R})\mathrm{d}\mathbf{R}, \tag{2.2}$$

where $f(\mathbf{R}) = g(\mathbf{R})/\mathrm{P}(\mathbf{R})$. The importance function $\mathrm{P}(\mathbf{R})$ can be interpreted as a probability density. If we generate an infinite number of random uncorrelated configurations $\mathbf{R}_m$ from the distribution $\mathrm{P}(\mathbf{R})$, the sample average is a good estimator of the integral I

$$I = \lim_{M\to\infty} \left\{ \frac{1}{M} \sum_{m=1}^{M} f(\mathbf{R}_m) \right\}, \tag{2.3}$$

and for an approximation with a finite number of samples

$$I \approx \frac{1}{M} \sum_{m=1}^{M} f(\mathbf{R}_m). \tag{2.4}$$

Under conditions where the central limit theorem holds [15], the estimator is normally distributed with variance σ_f^2/M, which can also be estimated from the samples as

$$\frac{\sigma_f^2}{M} \approx \frac{1}{M(M-1)} \sum_{m=1}^{M} \left[f(\mathbf{R}_m) - \frac{1}{M} \sum_{n=1}^{M} f(\mathbf{R}_n) \right]^2. \tag{2.5}$$

In the case of Hookium, the configurations $\mathbf{R}$ are the positions of the electrons.

Metropolis-Hastings algorithm

The integration technique from the previous section relies on our ability to obtain samples from a probability distribution $\mathrm{P}(\mathbf{R})$. In the case of QMC these distributions are high-dimensional and cannot be directly sampled from. Moreover their normalisations are usually not known. The Metropolis-Hastings algorithm [19], see Algorithm 1, avoids direct sampling from the distribution $\mathrm{P}(\mathbf{R})$ and is insensitive to its normalisation. It uses a Markov process whose stationary distribution $\pi(\mathbf{R})$ is the same as $\mathrm{P}(\mathbf{R})$ to generate a sequence of configurations $\{\mathbf{R}_n\}_{\mathrm{P}}$ that are drawn from $\mathrm{P}(\mathbf{R})$. A Markov process is completely defined with its transition probability $\mathrm{P}(\mathbf{R} \to \mathbf{R}')$, which is the probability of transitioning from state $\mathbf{R}$ to state $\mathbf{R}'$. For the process to have a unique stationary distribution two conditions must be met, the process must be *ergodic* and it must obey *detailed balance*

$$\mathrm{P}(\mathbf{R})\mathrm{P}(\mathbf{R} \to \mathbf{R}') = \mathrm{P}(\mathbf{R}')\mathrm{P}(\mathbf{R}' \to \mathbf{R}), \tag{2.6}$$

rewritten as

$$\frac{\mathrm{P}(\mathbf{R})}{\mathrm{P}(\mathbf{R}')} = \frac{\mathrm{P}(\mathbf{R}' \to \mathbf{R})}{\mathrm{P}(\mathbf{R} \to \mathbf{R}')}. \tag{2.7}$$

The right transition probability $\mathrm{P}(\mathbf{R} \to \mathbf{R}')$ is not known, but we can express it with a trial move transition probability $\mathrm{T}(\mathbf{R} \to \mathbf{R}')$ which we sample and acceptance probability $\mathrm{A}(\mathbf{R} \to \mathbf{R}')$ as

$$\mathrm{P}(\mathbf{R} \to \mathbf{R}') = \mathrm{T}(\mathbf{R} \to \mathbf{R}')\mathrm{A}(\mathbf{R} \to \mathbf{R}'). \tag{2.8}$$

For equation (2.7) to hold, the acceptance probability must be

$$A\left(\mathbf{R} \to \mathbf{R}'\right) = \min\left(1, \frac{\mathrm{T}\left(\mathbf{R}' \to \mathbf{R}\right)\mathrm{P}\left(\mathbf{R}'\right)}{\mathrm{T}\left(\mathbf{R} \to \mathbf{R}'\right)\mathrm{P}\left(\mathbf{R}\right)}\right). \tag{2.9}$$

Thus to sample from any probability distribution we need only have the ability to calculate probabilities $\mathrm{P}(\mathbf{R})$ and to sample from a trial transition probability $\mathrm{T}(\mathbf{R} \to \mathbf{R}')$. The efficiency of the algorithm depends on the amount of trial moves that we reject. All trial moves would be accepted if $\mathrm{T}(\mathbf{R} \to \mathbf{R}') = \mathrm{P}(\mathbf{R}')$, which would just mean sampling from P directly and is the very problem we are trying to solve with Metropolis-Hastings.

Algorithm 1: Metropolis-Hastings

```
Result: A set of configurations $\{\mathbf{R}_n\}_{\mathrm{P}}$ sampled from P
Initialize walker at random configuration $\mathbf{R}$;
while no. samples less than N do
    Generate new configuration $\mathbf{R}'$ with transition probability $\mathrm{T}(\mathbf{R} \to \mathbf{R}')$;
    Accept the move $(\mathbf{R} \to \mathbf{R}')$ with probability
     $A(\mathbf{R} \to \mathbf{R}') = \min\left(1, \frac{\mathrm{T}(\mathbf{R}' \to \mathbf{R})\mathrm{P}(\mathbf{R}')}{\mathrm{T}(\mathbf{R} \to \mathbf{R}')\mathrm{P}(\mathbf{R})}\right)$;
    Append $\mathbf{R}$ to the set of configuration;
end
```

Variational Quantum Monte Carlo

Variational quantum Monte Carlo uses a trial wave function Ψ_T which is an approximation to the true ground state wave function to directly evaluate the expectation value of $\hat{H}$. This provides an upper bound on the ground state energy

$$E_V = \frac{\langle \Psi_T | H | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} = \frac{\int \Psi_T^*(\mathbf{R}) \hat{H} \Psi_T(\mathbf{R}) d\mathbf{R}}{\int \Psi_T^*(\mathbf{R}) \Psi_T(\mathbf{R}) d\mathbf{R}} \geq E_0. \tag{2.10}$$

The expression for the variational energy E_V can be rewritten as

$$E_V = \frac{\int |\Psi_T(\mathbf{R})|^2 \left(\Psi_T(\mathbf{R})^{-1} \hat{H} \Psi_T(\mathbf{R}) \right) d\mathbf{R}}{\int |\Psi_T(\mathbf{R})|^2 d\mathbf{R}}. \tag{2.11}$$

The above integral is estimated by using Metropolis-Hastings to sample a set of configurations $\{\mathbf{R}_n\}_\mathrm{P}$ from the probability distribution given by the trial wave function as $\mathrm{P}(\mathbf{R}) = |\Psi_T(\mathbf{R})|^2 \mathrm{d}\mathbf{R}$ and averaging these *local* E_L contributions

$$E_V \approx \frac{1}{M} \sum_{m=1}^{M} E_L(\mathbf{R}_m), \tag{2.12}$$

where

$$E_L(\mathbf{R}) = \Psi_T(\mathbf{R})^{-1} \hat{H} \Psi_T(\mathbf{R}). \tag{2.13}$$

The procedure is analogous for any other calculation of expectation value. Trial moves may be chosen in a variety of ways depending on the system studied. In the case of Hookium we will use a Gaussian distribution centered at the current position of the walker.

Estimation of the variational energy E_V is only one part of a variational Monte Carlo simulation. The second part is the variational optimisation of the trial wave function. The trial wave function Ψ_T is parameterized with a set of variational parameters $\{\alpha_k\}$. Historically the number of parameters has been low due to high computational cost [15]. The optimal parameters for the system are found by minimizing the *cost function*. A straightforward choice of the cost function is the variational energy E_V, eq. (2.11). Given that its value is bounded below due to the variational principle, eq (2.10), its minimisation gives parameters $\{\alpha_k\}$ that give the best energy estimate for given parameterisation. An alternative is to minimize the variance of energy

$$\sigma_E^2(\{\alpha_k\}) = \frac{\int \Psi_T^2(\{\alpha_k\}) \left[E_L(\{\alpha_k\}) - E_V(\{\alpha_k\}) \right]^2 d\mathbf{R}}{\int \Psi_T^2(\{\alpha_k\}) d\mathbf{R}}, \tag{2.14}$$

which minimizes the statistical error of VMC estimation of energy. Most practical calculations are done by minimizing energy variance [15]. Minimisation of energy variance works because of the *zero-variance* property, which is exclusive for quantum expectation values. If the trial wave function Ψ_T is an exact eigenfunction of the Hamiltonian

$$\hat{H}|\Psi_T\rangle = E_V|\Psi_T\rangle, \tag{2.15}$$

then the local energy E_L, a random variable, does not depend on the sampled configuration $\mathbf{R}$

$$E_L(\mathbf{R}) = \Psi_T(\mathbf{R})^{-1}\hat{H}\Psi_T(\mathbf{R}) = \Psi_T(\mathbf{R})^{-1}E_V\Psi_T(\mathbf{R}) = E_V, \tag{2.16}$$

is constant with zero variance. This equality holds only when Ψ_T is an exact eigenfunction of the Hamiltonian. However, zero-variance property has important consequences for numerical stability of optimisation, it means that energy variance minima are robust to finite sampling. Minimizing the variance of energy drives the trial wave function towards eigenstates of the Hamiltonian. Moreover, the statistical error associated with estimation of any expectation value $\langle\hat{O}\rangle$ is proportional to the variance of $\hat{O}$. One can use the zero-variance condition to define a renormalized observable $\tilde{O}$ with the same average and smaller variance [3] for more efficient sampling.

Various approaches to minimize the cost function can be taken, the simplest being trial and error using simple fitting procedures, which only works for small numbers of parameters. Alternatively a reweighting technique can be used to evaluate the energy or energy variance of a wave function with slightly different parameters $\Psi_T(\{\alpha+\delta\alpha\})$ to the one already evaluated $\Psi_T(\{\alpha\})$ [41], this increases the number of variational parameters that can be treated in small systems. Another option is to evaluate energy derivatives and use some sort of stochastic optimisation technique, *stochastic gradient descent* being the simplest and *stochastic recofiguration* [40] being a more elaborate alternative.

Trial wave functions Ψ_T

The choice of trial wave function Ψ_T is the limiting factor for the performance of VMC as it determines its statistical efficiency and its final accuracy. The choice of trial wave function is flexible but must satisfy the following conditions [15]. Both the wave function and its gradient must be finite where the potential is finite, the wave function must have the appropriate symmetry and integrals

$$\int \Psi_T^*\Psi_T, \quad \int \Psi_T^*\hat{H}\Psi_T \quad \text{and} \quad \int \Psi_T^*\hat{H}^2\Psi_T \tag{2.17}$$

must exist. A popular choice, and the one we will use in this project, is the *Slater-Jastrow* state. The trial wave function is a product of a Slater determinant $D(\mathbf{R})$ of single particle states $\{\psi_k(\mathbf{r})\}$

$$D(\mathbf{X}) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_1(\mathbf{x}_1) & \psi_2(\mathbf{x}_1) & \cdots & \psi_N(\mathbf{x}_1) \\ \psi_1(\mathbf{x}_2) & \psi_2(\mathbf{x}_2) & \cdots & \psi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(\mathbf{x}_N) & \psi_2(\mathbf{x}_N) & \cdots & \psi_N(\mathbf{x}_N) \end{vmatrix} \tag{2.18}$$

and the *Jastrow correlation factor* $J(\mathbf{X})$

$$\Psi_{SJ}(\mathbf{X}) = e^{J(\mathbf{X})} D(\mathbf{X}), \tag{2.19}$$

where $\mathbf{X}$ is a configuration that contains both the spin and position degrees of freedom $\mathbf{x}_i = (\mathbf{r}_i, s_i)$. The discriminant is usually decomposed into spin up and down components as

$$D(\mathbf{X}) = D^{\uparrow}\left(\mathbf{r}_1, \ldots, \mathbf{r}_{N_\uparrow}\right) D^{\downarrow}\left(\mathbf{r}_{N_\uparrow+1}, \ldots, \mathbf{r}_N\right). \tag{2.20}$$

This is computationally beneficial as it results in smaller determinants and no explicit sum over spin. The Jastrow factor is heuristically constructed to incorporate electron correlation into the wave function. In most practical calculations it is limited to one- and two-body terms [15]

$$J(\mathbf{R}) = \underbrace{\sum_{i=1}^{N} \chi(\mathbf{r}_i)}_{\text{one-body}} \underbrace{- \frac{1}{2} \sum_{i=1}^{N} \sum_{j<i}^{N} u(\mathbf{r}_i, \mathbf{r}_j)}_{\text{two-body}}. \tag{2.21}$$

The first term describes the nuclear-electronic correlation and the second electron-electron correlation, which is the only term we consider in this project. Various forms of $u(\mathbf{r}_i, \mathbf{r}_j)$ exist depending on application area, a common form for atomic and molecular calculations is

$$u(r_{ij}) = \frac{a_{ij} r_{ij}}{1 + \sum_{\kappa=1} b_{ij}^{(\kappa)} r_{ij}^{\kappa}} \tag{2.22}$$

The parameters a_{ij} are chosen to satisfy the *cusp conditions*,

$$\left.\frac{du}{dr}\right|_{r=0} = \begin{cases} -\frac{1}{2}, & \text{for opposite spins,} \\ -\frac{1}{4}, & \text{for parallel spins.} \end{cases} \tag{2.23}$$

and the $b_{ij}^{(\kappa)}$ parameters remain to be optimised variationally.

2.3 Feynman-Kac: connecting Quantum mechanics and Stochastic Processes

2.3.1 Stochastic processes

2.3.2 Feynman-Kac formula

2.3.3 Lattice-model representations

Quantum Ising model

Heisenberg model

Bose-Hubbard model

2.4 Machine Learning

2.4.1 Generative modelling

Depending on what we learn in machine learning we differentiate between *discriminative* and *generative* modelling. While there is some inconsistency in the terminology, at their core discriminative models only have the ability to discriminate between targets Y and inputs X while generative models "simulate" the underlying data generation process and have the ability to generate new samples from the underlying distribution of data. Generative modelling is very much similar to the scientific process of providing competing models of how physical phenomena work and testing them through observations. The models are considered highly intuitive and interpretable as we can encode our prior knowledge into them while treating unknown details as noise [25].

While there has been a recent surge of popularity in generative modeling, with models like the *variational autoencoder* (VAE) [27, 37], *Generative Adversarial Network* (GAN) [16] and *Normalizing Flows* [36] being popular examples, generative modelling in ML has a rich history. Models such as Gaussian mixture models [14], Hidden Markov models [34], Autoregressive models, Latent Dirichlet allocation [5], Boltzmann and Helmholtz machines [1, 12] have long been known and used. With increase in computational power, modern generative models have achieved astounding results in speech synthesis [32], image super-resolution [28] and have been implemented to aid supervised learning [26].

Ian Goodfellow provided a helpful taxonomy of generative models in his tutorial on GANs [16]. He differentiates generative models that maximise likelihood between how they compute or approximate the likelihood and its gradients. The model can construct an explicit probability density $p_\theta(\mathbf{x})$ and consequently an explicit likelihood which is then optimised. This explicit density can either be tractable as is the case in *fully observed Bayesian networks*, or it could be intractable requiring either variational approaches, as in the VAE, or MCMC for example in Boltzmann machines. On the other hand we have models that do not construct the probability distribution explicitly, but instead provide a less direct way to interact with it i.e. draw samples from it. GANs fall into this category.

2.4.2 Deep latent variable models

We assume that an observable value $\mathbf{x}$ is generated by an unknown underlying random process with unknown probability distribution $p^*(\mathbf{x})$. This is the process we try to approximate using a generative model $p_\theta(\mathbf{x})$ parameterised with parameters θ. The process of learning is finding parameters θ such that our model distribution $p_\theta(\mathbf{x})$ is as close to the underlying distribution

$p^*(\mathbf{x})$ as possible. Normally though, this is the case for regression and classification problems, we are modelling the conditional distribution $p^*(\mathbf{y}|\mathbf{x})$ with our model $p_\theta(\mathbf{y}|\mathbf{x})$, where $\mathbf{y}$ is conditioned on the input $\mathbf{x}$. The approaches described and implemented in this written assignment are equally applicable to both conditional and unconditional modelling. In either case we want our model to be versatile enough to represent the underlying structure of the data. The last two decades have seen a renaissance of neural networks as models of complex processes [17], they are *universal function approximators*, generally made of compositions of smaller differentiable sub-units with nonlinear activation functions and trained using gradient-based approaches via backpropagation. They can be used to parameterize conditional distributions and form basis of probabilistic models thus combining the best of both worlds, a probabilistic description and *hopefully* better scalability by making use of gradient-based learning. The categorical distribution $p_\theta(y|\mathbf{x})$ in the case of classification, where y is the class, given the input $\mathbf{x}$, can be parameterised as

$$\begin{aligned} \mathbf{p} &= f(\mathbf{x};\theta), \\ p_\theta(y|\mathbf{x}) &= \text{Categorical}(y;\mathbf{p}), \end{aligned} \tag{2.24}$$

where $f(\cdot,\theta):\mathbb{R}^{n_\text{in}} \to \mathbb{R}^{n_\text{out}}$ is the neural network, with a final layer such that the probabilities $\sum_i^{n_\text{out}} p_i = 1$.

We can combine random variables $(\mathbf{x_1},\ldots,\mathbf{x_M})$ and represent their conditional dependence structure as a network to construct a *probabilistic graphical model.* If the structure of the network is a directed acyclic graph, the joint distribution of the random variables is factorised as a product of conditionals and priors

$$p_\theta(\mathbf{x}_1,\ldots,\mathbf{x}_M) = \prod_{j=1}^{M} p_\theta\left(\mathbf{x}_j \mid \text{pa}(\mathbf{x}_j)\right), \tag{2.25}$$

where $\text{pa}(\mathbf{x}_j)$ are the parent nodes of node j in the directed graph. A graphical model with this structure is termed a *Bayesian network* and we can parameterise the conditional distributions in a Bayesian network using neural networks

$$\begin{aligned} \eta &= f(\text{pa}(\mathbf{x});\theta) \\ p_\theta(\mathbf{x} \mid \text{pa}(\mathbf{x})) &= p_\theta(\mathbf{x} \mid \eta). \end{aligned} \tag{2.26}$$

A model which only contains observable variables $\mathbf{x}_i$ is a *fully observed* model and is straightforward to learn. Given a dataset $\mathcal{D}$ of independently and identically distributed observations $\mathbf{x}_i$ we can perform *maximum likelihood estimation*, i.e. search for parameters θ such that the observed data is most probable, by simply estimating the gradients of the *log-likelihood*

$$\log p_\theta(\mathscr{D}) = \sum_{\mathbf{x}\in\mathscr{D}} \log p_\theta(\mathbf{x}). \tag{2.27}$$

Moreover, *maximum a posteriori estimation* of which maximum likelihood is a special case, can be performed and full Bayesian treatment of the posterior is also possible [25]. A complication arises when we include *latent* variables into the model, these are variables which are not observable and do not form a part of the dataset $\mathscr{D}$, usually denoted with $\mathbf{z}$. For unconditional model learning the graphical model now represents a joint probability distribution $p_\theta(\mathbf{x},\mathbf{z})$, which when marginalized gives the single point *model evidence* $p_\theta(\mathbf{x})$ as a function of the parameters θ. In the simplest deep latent variable models, the joint probability is factored into the prior distribution $p_\theta(\mathbf{z})$ and the conditional distribution, which is represented by a neural network $p_\theta(\mathbf{x}|\mathbf{z})$, as

$$p_\theta(\mathbf{x},\mathbf{z}) = p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}). \tag{2.28}$$

Alternatively, we can have a model with many layers of latent variables $\mathbf{z}_m$,

$$p_\theta(\mathbf{x},\mathbf{z}) = p\left(x \mid \mathbf{z}_1\right) p\left(\mathbf{z}_1 \mid \mathbf{z}_2\right) \cdots p\left(\mathbf{z}_{m-1} \mid \mathbf{z}_m\right) p\left(\mathbf{z}_m\right), \tag{2.29}$$

which are then able to learn hierarchical latent representation. In either case learning the posterior $p_\theta(\mathbf{x},\mathbf{z})$, i.e.

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x},\mathbf{z}) d\mathbf{z}, \tag{2.30}$$

is intractable, meaning it is (without tricks) impossible to differentiate the marginal likelihood w.r.t parameters θ and perform gradient-based optimisation. Advantage of DLVMs is their flexibility, even using very simple priors and conditionals they can express complicated distributions because of the nonlinearity of the neural network representations.

2.4.3 Learning in DLVMs

There are two main issues with learning in deep latent variable models, succinctly illustrated with two optimisation approaches. The first is the Expectation-Maximization (EM) [13] algorithm, it consists of two steps which are repeated until convergence.

- **E-step**: For each datapoint $\mathbf{x}_i$ compute the posterior $p_{\theta_t}(\mathbf{z}_i|\mathbf{x_i})$.
- **M-step**: Estimate new parameters by maximsing the expected log-likelihood $\theta_{t+1} = \arg\max_\theta \sum_{x\in D} \mathbb{E}_{z\sim p(z|x;\theta_t)} \log p(x,z;\theta)$.

The algorithm falls apart in two ways, firstly the E-step requires computation of the posterior distribution which is intractable in DLVMs. This can be solved by employing a sampling technique (MCMC EM algorithm) but at a high computational cost, requiring a per datapoint MCMC step. Secondly, the M-step optimises over all datapoints requiring vast amounts of memory, although it can be done in mini-batches (online EM [21]. Hence, the first issue with learning in DLVMs is computational efficiency. The second algorithm is the *wake-sleep* algorithm [20], it employs a recognition model that approximates the posterior distribution and is applicable to both discrete and continuous latent variable models. While it is not computationally expensive relative to EM, its optimisation objectives do not coincide with optimising the marginal likelihood, thus giving a bad estimate of the posterior. Therefore the second issue is inaccurate estimates of the posterior. Both these issues were addressed in two seminal papers that constitute the framework of the variational autoencoder [27, 37].

2.4.4 Variational autoencoder

The strength of variational methods lies in recasting inference as an optimisation problem, which scales better to large datasets than sampling methods. This is done by introducing an approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$, with parameters ϕ and optimising to maximise the similarity between $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z}|\mathbf{x})$. In the case of the VAE $q_\phi(\mathbf{z}|\mathbf{x})$ is also parameterised using a neural network. The similarity between the two distributions is measured by the Kullback-Leibler divergence

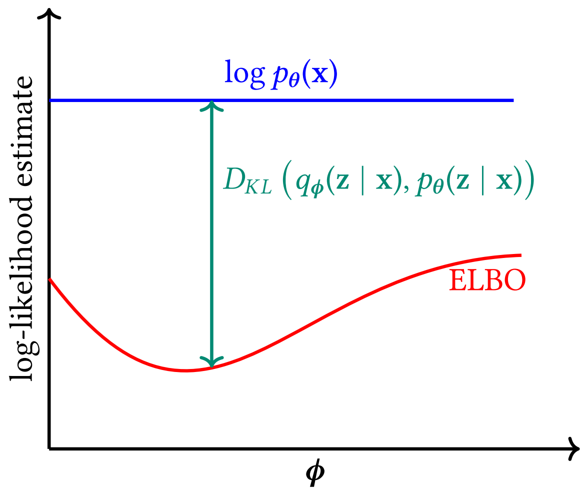
$$D_{KL}(q\|p) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})}, \tag{2.31}$$

which has two important properties. $D_{KL}(q\|p) \geq 0$ for all q and p and it is equal to zero only when $q = p$. It can be rewritten as

$$\begin{aligned} &D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \| p_\theta(\mathbf{z} \mid \mathbf{x})) = \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z},\mathbf{x})} + \log p_\theta(\mathbf{x}) \right] \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \left[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{z},\mathbf{x}) \right] + q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}), \end{aligned} \tag{2.32}$$

from which we can express the log likelihood as

$$\log p_\theta(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \| p_\theta(\mathbf{z} \mid \mathbf{x})) + \mathscr{L}_{\theta,\phi}(\mathbf{x}), \tag{2.33}$$



$\log p_\theta(\mathbf{x})$
$D_{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}), p_\theta(\mathbf{z} \mid \mathbf{x})\right)$
ELBO
log-likelihood estimate
ϕ

Fig. 2.1 ELBO is a lower bound of the log likelihood, the difference between the two is the KL divergence. The better the approximation of the posterior $q_\phi(\mathbf{z}|\mathbf{x})$, the smaller is KL and consequently the closer is ELBO to $\log p_\theta(\mathbf{x})$.

where $\mathscr{L}_{\theta,\phi}(\mathbf{x})$ is the *evidence lower bound* ELBO, defined as

$$\mathscr{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})\right]. \tag{2.34}$$

ELBO gives a lower bound on the log likelihood, because Kullback-Leiber divergence is non-negative. From equation (2.33) we see, that maximizing the ELBO with regards to both θ and ϕ does two things, it minimizes the KL divergence between the true and approximate posteriors and consequently better approximates the log evidence, see Figure 2.1. This makes ELBO a good optimisation objective for learning in VAEs.

2.4.5 Variance Loss

The evidence lower bound (ELBO) pathwise gradient estimator (SGVB), described in section B, has found great success but is limited to models with continuous latent variables. It would be more convenient to have a loss function that would deal with both discrete and continuous latent variables within the same framework. Recent work by Richter et al. [38] analysed a score-function gradient estimator for ELBO and showed that it can be obtained from a different divergence measure, which they termed the *log-variance loss*

$$\mathscr{L}_{\mathrm{LV}}\left(q_\phi(\mathbf{z}) \| p_\theta(\mathbf{z} \mid \mathbf{x})\right) = \frac{1}{2} \underset{\mathbf{z}\sim r(\mathbf{z})}{\mathbb{V}\mathrm{ar}}\left[\log\left(\frac{q_\phi(\mathbf{z})}{p_\theta(\mathbf{z} \mid \mathbf{x})}\right)\right], \tag{2.35}$$

where $r(\mathbf{z})$ is some arbitrary distribution. Moreover, they showed that the gradient of $\mathscr{L}_{\mathrm{LV}}$ w.r.t ϕ corresponds to that of the KL divergence if $r(\mathbf{z}) = q_\phi(\mathrm{z})$ and provided a gradient estimator of the log-variance, thus suggested an optimisation procedure that works for both discrete and continuous latent distributions. In this work we will proceed in a similar fashion but with a different loss function. Motivated by the fact that when the posterior $q(z|x) = p(z|x)$ is exact we have

$$\frac{p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})} = p(\mathbf{x}). \tag{2.36}$$

The variance of $p(\mathbf{x})$ with regards to samples drawn from an arbitrary probability distribution $r(\mathbf{z})$ is zero which suggests the variance of the left hand side

$$\mathscr{L}_{\phi,\theta}(\mathbf{x}) = \underset{\mathbf{z}\sim r(\mathbf{z})}{\mathbb{V}\mathrm{ar}}\left[\log\left(\frac{p_\theta(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\right)\right] \tag{2.37}$$

as an objective function. We can estimate the loss with sample variance using S Monte Carlo samples

$$\begin{gathered}\mathscr{L}_{\phi,\theta}(\mathbf{x}) \approx \frac{1}{(S-1)}\sum_{s=1}^{S}\left(f_{\phi,\theta}(\mathbf{z}^{(s)},\mathbf{x}) - \bar{f}_{\phi,\theta}(\mathbf{x})\right)^2, \\ \mathbf{z}^{(s)} \overset{\text{i.i.d.}}{\sim} r(\mathbf{z}),\end{gathered} \tag{2.38}$$

where

$$f_{\phi,\theta} = \log\left(\frac{p_\theta(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\right) \tag{2.39}$$

and $f_{\phi,\theta}$ is the average

$$\bar{f}_{\phi,\theta}(\mathbf{x}) = \frac{1}{S}\sum_{s=1}^{S} f_{\phi,\theta}\left(\mathbf{z}^{(s)},\mathbf{x}\right). \tag{2.40}$$

The gradient of expression (2.38) can be computed using automatic differentiation without any issue. The form of the gradient estimator of $\mathscr{L}_{\phi,\theta}(\mathbf{x})$ outlined above, is the same as the leave-one-out estimator described in [38], which means that it is an unbiased estimator of the gradient. The procedure of calculating the gradients for a given example $\mathbf{x}$ and parameters ϕ,θ is given in Algorithm 2. Importantly the samples $\mathbf{z}^{(s)}$, which will be for simplicity sampled from the model probability $\mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x})$, must be detached from the computational graph so that backpropagation is possible.

To test the applicability of the loss function we will test two variational autoencoders, one with continuous latent space and one with discrete latent space. The architecture of the neural networks that were used to parameterise the encoder and decoder will be discussed in chapter 3.

Algorithm 2: Gradients of $\mathscr{L}_{\phi,\theta}(\mathbf{x})$

```
Input: Encoder parameters $\phi$, decoder parameters $\theta$, example $\mathbf{x}$
Result: Gradients w.r.t $\phi, \theta$ of the objective function 1
for $s = 1, \ldots, S$ do
    $z^{(s)} \leftarrow \text{sample}\left(q_\phi(\cdot \mid \mathbf{x})\right)$
    $z^{(s)} \leftarrow \text{stop\_gradient}\left(z^{(s)}\right)$
    $f^{(s)}_{\phi,\theta} \leftarrow \log p_\theta(x|z^{(s)})p(z^{(s)}) - \log q_\phi(\mathbf{z}^{(s)}|\mathbf{x})$
end
$\widehat{\mathscr{L}} \leftarrow$ Variance $\left(\left\{f^{(s)}_{\phi,\theta}\right\}_{s=1}^{S}\right)$
return grad $\left(\widehat{\mathscr{L}}\right)$
```

2.4.6 Convolutional Neural Networks

Chapter 3

My third chapter

3.1 Convolutions

Chapter 4

Methodology

4.1 Neural network ansatzes

Chapter 5

Results

5.1 something

References

[1] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169.

[2] Anderson, J. B. (1975). A random-walk simulation of the schrödinger equation: H+ 3. *The Journal of Chemical Physics*, 63(4):1499–1503.

[3] Assaraf, R. and Caffarel, M. (1999). Zero-variance principle for monte carlo algorithms. *Physical review letters*, 83(23):4682.

[4] Austin, B. M., Zubarev, D. Y., and Lester Jr, W. A. (2012). Quantum monte carlo and related approaches. *Chemical reviews*, 112(1):263–288.

[5] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

[6] Carleo, G., Becca, F., Sanchez-Palencia, L., Sorella, S., and Fabrizio, M. (2014). Light-cone effect and supersonic correlations in one-and two-dimensional bosonic superfluids. *Physical Review A*, 89(3):031602.

[7] Carleo, G., Becca, F., Schiró, M., and Fabrizio, M. (2012). Localization and glassy dynamics of many-body quantum systems. *Scientific reports*, 2(1):1–6.

[8] Ceperley, D., Chester, G. V., and Kalos, M. H. (1977). Monte carlo simulation of a many-fermion study. *Physical Review B*, 16(7):3081.

[9] Ceperley, D. M. (1995). Path integrals in the theory of condensed helium. *Reviews of Modern Physics*, 67(2):279.

[10] Ceperley, D. M. and Alder, B. J. (1980). Ground state of the electron gas by a stochastic method. *Physical review letters*, 45(7):566.

[11] Chriss, N. A. and Chriss, N. (1997). *Black Scholes and beyond: option pricing models*. McGraw-Hill.

[12] Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The helmholtz machine. *Neural computation*, 7(5):889–904.

[13] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

[14] Everitt, B. S. (2014). Finite mixture distributions. *Wiley StatsRef: Statistics Reference Online*.

[15] Foulkes, W., Mitas, L., Needs, R., and Rajagopal, G. (2001). Quantum monte carlo simulations of solids. *Reviews of Modern Physics*, 73(1):33.

[16] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

[17] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*. MIT press Cambridge.

[18] Gubernatis, J., Kawashima, N., and Werner, P. (2016). *Quantum Monte Carlo Methods: Algorithms for Lattice Models*. Cambridge University Press.

[19] Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*.

[20] Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995). The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161.

[21] Hoffman, M., Bach, F. R., and Blei, D. M. (2010). Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864. Citeseer.

[22] Ido, K., Ohgoe, T., and Imada, M. (2015). Time-dependent many-variable variational monte carlo method for nonequilibrium strongly correlated electron systems. *Physical Review B*, 92(24):245106.

[23] Kalos, M. (1962). Monte carlo calculations of the ground state of three-and four-body nuclei. *Physical Review*, 128(4):1791.

[24] Kalos, M. (1966). Stochastic wave function for atomic helium. *Journal of Computational Physics*, 1(2):257–276.

[25] Kingma, D. P. (2017). Variational inference & deep learning: A new synthesis.

[26] Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. (2014). Semi-supervised learning with deep generative models. *arXiv preprint arXiv:1406.5298*.

[27] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[28] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

[29] McMillan, W. L. (1965). Ground state of liquid he 4. *Physical Review*, 138(2A):A442.

[30] Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020). Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62.

[31] Needs, R., Towler, M., Drummond, N., Lopez Rios, P., and Trail, J. (2020). Variational and diffusion quantum monte carlo calculations with the casino code. *The Journal of chemical physics*, 152(15):154106.

[32] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[33] Prokof'Ev, N., Svistunov, B., and Tupitsyn, I. (1998). Exact, complete, and universal continuous-time worldline monte carlo approach to the statistics of discrete quantum systems. *Journal of Experimental and Theoretical Physics*, 87(2):310–321.

[34] Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16.

[35] Reynolds, P. J., Tobochnik, J., and Gould, H. (1990). Diffusion quantum monte carlo. *Computers in Physics*, 4(6):662–668.

[36] Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR.

[37] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR.

[38] Richter, L., Boustati, A., Nüsken, N., Ruiz, F. J., and Akyildiz, Ö. D. (2020). Vargrad: A low-variance gradient estimator for variational inference. *arXiv preprint arXiv:2010.10436*.

[39] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

[40] Sorella, S. (1998). Green function monte carlo with stochastic reconfiguration. *Physical review letters*, 80(20):4558.

[41] Umrigar, C., Wilson, K., and Wilkins, J. (1988). Optimized trial wave functions for quantum monte carlo calculations. *Physical Review Letters*, 60(17):1719.

[42] Vosko, S. H., Wilk, L., and Nusair, M. (1980). Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis. *Canadian Journal of physics*, 58(8):1200–1211.

Appendix A

Jax ecosystem

Appendix B

Gradient estimation

In order to perform gradient descent on the ELBO objective, we need to be able to evaluate its gradients with respect to parameters θ and ϕ. Taking the gradient w.r.t generative parameters θ is straightforward, because we can change the order of the expectation operator and the gradient, leaving us with

$$\begin{aligned} \nabla_\theta \mathscr{L}_{\theta,\phi}(\mathbf{x}) &= \nabla_\theta \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\ &\simeq \nabla_\theta \left(\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right) \\ &= \nabla_\theta \left(\log p_\theta(\mathbf{x},\mathbf{z})\right), \end{aligned} \tag{B.1}$$

where $\simeq$ denotes an unbiased estimator. This reversing of the order of operations is not possible when taking gradients w.r.t variational parameters ϕ because the expectation $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}$ is performed w.r.t the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$. The gradient could be estimated with a vanilla Monte Carlo estimator, but it has very high variance and is not practical [27].

The problem of stochastic gradient estimation of an expectation of a function is a well studied problem that transcends machine learning and has a variety of applications [11, 39]. Different estimators differ in from and their properties, variance being one of the most important. In their review [30] Mohamed et al. categorise MC gradient estimators into three categories

1. *Score-function estimator*: The score function is a logarithm of a probability distribution w.r.t to distributional parameters. It can be used as a gradient estimator

$$\begin{aligned} \nabla_\theta \mathbb{E}_{p_\theta(\mathbf{x})}[f(\mathbf{x})] &= \nabla_\theta \int p_\theta(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \\ &= \mathbb{E}_{p_\theta(\mathbf{x})}\left[f(\mathbf{x}) \nabla_\theta \log p_\theta(\mathbf{x})\right]. \end{aligned} \tag{B.2}$$

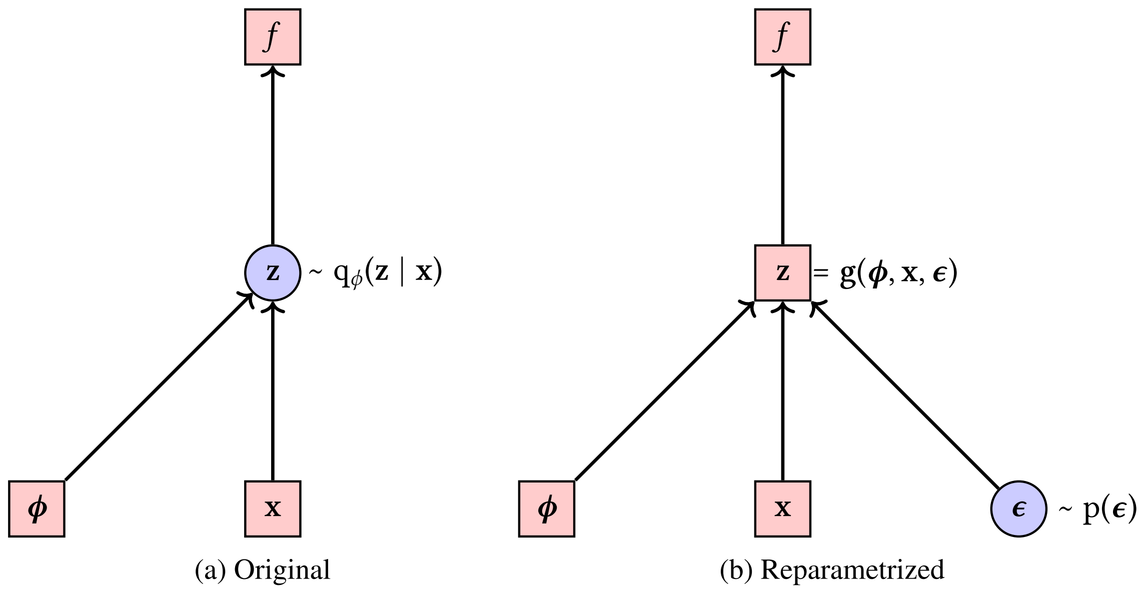


Fig. B.1 The reparametrization trick, adapted from [25]. The stochasticity of the **z** node is pushed out into a separate input to the same node, resulting in deterministic gradients w.r.t ϕ through the node.

The score-function estimator is compatible with any cost function, it requires that the measure $p_\theta(\mathbf{x})$ is differentiable and easy to sample. Very importantly it is applicable to both discrete and continuous distribution, but has a drawback of having high variance.

2. *Pathwise estimator*: Continuous distributions can be sampled either directly by generating samples from the distribution $p_\theta(\mathbf{x})$ or indirectly, by sampling from a simpler base distribution $p(\varepsilon)$ and transforming the variate through a deterministic path $g_\theta(\varepsilon)$. Using this, it is possible to move the source of randomness in such a way that the objective is differentiable. In essence this approach pushes the parameters of the measure into the cost function which is then differentiated. The estimator is

$$\begin{aligned}\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{x})}[f(\mathbf{x})] &= \nabla_\theta \int p_\theta(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \\ &= \nabla_\theta \int p(\varepsilon) f(g_\theta(\varepsilon)) d\varepsilon \\ &= \mathbb{E}_{p(\varepsilon)}\left[\nabla_\theta f(g_\theta(\varepsilon))\right].\end{aligned} \tag{B.3}$$

This was the gradient estimator originally used in the VAE implementation [27] there named as the *reparametrization trick*, see also Figure B.1. In many cases the transformation paths are so simple they can be implemented in one line of code, referred to as *one-liners*. The pathwise-estimator can only be used on differentiable cost functions, but is easy to implement and crucially has lower variance than the score-function estimator.

3. *Measure-valued gradient estimator*, which exploits the properties of signed-measures, is beyond the scope of this report.

Unbiased gradient estimation makes training on the ELBO objective possible. During training the VAE learns a mapping between a complicated observed space of $\mathbf{x}$ and the latent space of $\mathbf{z}$, which is usually defined to be relatively simple. The *decoder* network or the *generative* part of the VAE $p_\theta(\mathbf{x}|\mathbf{z})$ learns to map from the latent space to the data space, while the *inference* or *encoder* network $q_\phi(\mathbf{z}|\mathbf{x})$ approximates the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ of this process, see Figure.

Appendix C

Probability distributions