Topics in Approximate Inference

Yingzhen Li

current version: Dec 2020

Disclaimer

This monograph provides a high-level introduction on approximate inference and goes a bit more in depth for a selected list of related topics (with personal bias). The list here is non-exhaustive, even within each of the topics I have omitted many related works. The reason is either I have less expertise in the specific sub-area, or I have prioritised the explanation of the basic concepts. Readers might observe a mix of formal and informal writing styles, as most of the materials are adapted from my published papers, my reading notes, my PhD thesis, as well as the preparation materials from my NeurIPS 2020 tutorial on approximate inference. I expect to keep updating this list of topics to improve my explanations and include new techniques.

Contents

0 Front Matter

Approximate inference is key to modern probabilistic modelling, and since the start of my PhD there has been considerable progress on this subject. The literature becomes very diverse so that a new comer to the subject might find it difficult to learn the key techniques and identify important papers. Therefore in this document I share my reading and research notes on approximate inference, and I hope this would help people understand the general idea of this subject.

0.1 How to use this document

I assume you (as the reader) know some basic concepts in probability and machine learning. If you are completely new to approximate inference, then I would encourage you to start from § 0.3.

The notes are organised into the following categories:

- algorithms for fitting approximations;
- architecture designs of the approximation;

I should note that the topics included in the list are not "mutually independent", for example, a specific design of the approximate distribution might require special care of the fitting algorithm. Also this document is far from complete – basically inference computation is one of the most important research challenge in Bayesian statistics and probabilistic modelling, and I am certain that I have missed a lot of important work.

We will build from basics to advances for each topic in each category, so if you find sections x.1 and x.2 very confusing then please contact me. At the end of each topic section I will include a short list of (what I think are) "must read" papers, and all the citations can be found in the reference list.[1] If you want to suggest papers you are also welcome to contact me. Comments and extra examples are included as "remark" paragraphs: they often contain advanced stuff so they can be skipped if appropriate.

0.2 Math preparation

In this section we establish the mathematical notations and concepts that will be repeatedly used in the rest of the note.

- Observation variables $\boldsymbol{x}$, and in supervised learning case we also use $\mathbf{y}$ as the labels.
- Latent variable $\boldsymbol{z}$, it is unobserved and needs to be integrated out.
- Model parameter $\boldsymbol{\theta}$: for example it can be the set of neural network weights. Bayesian methods also consider it as a random variable.

[1] I am not an expert of sampling-based methods so I usually cite review papers and books for them, and I definitely have missed some important papers.

- Variational parameter $\boldsymbol{\phi}$: the parameters associate to the approximation.

- Probability density function:
  Denote the measurable space as (Θ, Σ), where Θ is the sample space of the random variable $\boldsymbol{\theta}$ of interest, and Σ is a pre-defined σ-algebra on Θ. A *probability distribution* P is a measure defined on Σ such that $P(\Theta) = 1$. Also we assume there exists a dominating measure (also called reference measure) μ on Σ such that, for any probability distribution P defined on Σ, we can define its *probability density function* p by $dP = pd\mu$.[2] For simplicity in the rest of the note we will work with the sample space $\Theta = \mathbb{R}^D$, the σ-algebra $\Sigma = \{S : S \subset \mathbb{R}^D\}$, and the dominating measure $d\mu = d\boldsymbol{\theta}$. Finally we write $\mathcal{P}$ the space of PDFs such that any probability distribution P defined on Σ has its PDF $p \in \mathcal{P}$.

- Divergence:
  Given a set of probability density functions $\mathcal{P}$ for a random variable $\boldsymbol{\theta}$, a divergence on $\mathcal{P}$ is defined as a function $\mathrm{D}[\cdot||\cdot] : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ such that $\mathrm{D}[p||q] \geq 0$ for all $p, q \in \mathcal{P}$, and $\mathrm{D}[p||q] = 0$ iff. $p = q$.

  The definition of divergence is much weaker than that for a *distance* such as the ℓ_2-norm, since it does not need to satisfy either symmetry in arguments or the triangle inequality. There exist many available divergences to use, where some of them are heavily used in approximate inference.

0.3 Inference, integration and optimisation

Probabilistic modelling starts by defining a distribution of data. For instance, in discriminative supervised learning, one would define a conditional distribution $p(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\theta})$, which is also called the *likelihood function* of $\boldsymbol{\theta}$. A concrete example for this would interpret $p(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\theta})$ as outputting the probability of a configuration of $\mathbf{y}$ (e.g. a label or a real value) by transforming the input $\boldsymbol{x}$ (an image, a sentence, etc.) through a neural network parameterised by $\boldsymbol{\theta}$. Before observing any real-world data, the parameters $\boldsymbol{\theta}$ are unknown, but we have a prior belief $p_0(\boldsymbol{\theta})$ about what value they might take, e.g. they should have small ℓ_2 norm if using a Gaussian prior centred at zero. Then we receive the observations $\mathcal{D} = \{(\boldsymbol{x}_n, \mathbf{y}_n)\}_{n=1}^N$, and based on data we want to answer questions on the unknown parameters $\boldsymbol{\theta}$, for example: given $\mathcal{D}$, what is the most probable value of $\boldsymbol{\theta}$, and how likely is $\boldsymbol{\theta}$ to be set to a given value? Answering these questions is precisely the procedure of *inference*: a procedure of deducing unknown properties (in our example the neural network weights) given the observed, or known information.

0.3.1 Exact Bayesian inference as integration

Bayesian statisticians are particularly interested in answering the latter question, by computing the *posterior distribution*, or the *posterior belief* of $\boldsymbol{\theta}$ given $\mathcal{D}$, using

[2] We can also define divergences without assuming a common reference measure, which is out of the scope of this note. In this case one should work with equalities up to zero measure.

Bayes' rule [Bayes and Price, 1763; Laplace, 1820]:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta})}{p(\mathcal{D})}, \tag{1}$$

with $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_n p(\mathbf{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta})$ following the i.i.d. assumption. The elegance of Bayes' rule is that *it separates inference from modelling*. The model – the prior distribution and the likelihood – completely determines the posterior distribution, and the only thing left is to *compute* the inference.

A closer look at Bayes' rule reveals that the core computation of Bayesian inference is *integration*. Using the sum rule of probability distributions we have the marginal distribution computed as[3]

$$p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta})d\boldsymbol{\theta},$$

and if this integral is tractable, then the posterior distribution can be easily computed by (1). Moreover, to predict the label $\mathbf{y}^*$ on unseen datum $\boldsymbol{x}^*$ a Bayesian would compute the *predictive* distribution

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}, \tag{2}$$

which again requires solving an integration problem. Even more, since it is hard to visualise the posterior distribution in high dimensions, one would instead look at the statistics of the posterior, for example

$$\text{posterior mean } \mu = \int \boldsymbol{\theta}p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta},$$

$$\text{posterior variance } \Sigma = \int (\boldsymbol{\theta} - \mu)(\boldsymbol{\theta} - \mu)^{\mathrm{T}}p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta},$$

both are integration tasks as well. In summary, many tasks in Bayesian computation can be framed as computing an integral of some function $F(\boldsymbol{\theta})$ against the posterior distribution:

$$\int F(\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}. \tag{3}$$

The goal of this document is to discuss how to perform this integration *pragmatically and efficiently*.

Remark (Inference in a more general sense)**.** More generally speaking, inference can be viewed as computing unknown quantities that is dependant on a given distribution $\pi(\boldsymbol{\theta})$:

$$\int F(\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}, \tag{4}$$

where in the Bayesian inference example $\pi(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D})$. Other examples of

[3] In discrete variable case the integral is calculated w.r.t. discrete measure, i.e. summation, which will also be referred as integration in the rest of the manuscript.

$\pi(\boldsymbol{\theta})$ include *implicit distributions* [Diggle and Gratton, 1984; Mohamed and Lakshminarayanan, 2016] which are defined by data simulation processes (e.g. physics simulator, neural network transform, etc.). In this document we focus on studying the following type of distribution $\pi(\boldsymbol{\theta}) = \pi^*(\boldsymbol{\theta})/Z$ with tractable unnormalised density $\pi^*(\boldsymbol{\theta})$, for example, we assume the joint model distribution $p(\boldsymbol{\theta}, \mathcal{D})$ is tractable, so the exact posterior $p(\boldsymbol{\theta}|\mathcal{D})$ is tractable up to a constant $p(\mathcal{D})$.

0.3.2 Approximate Bayesian inference as optimisation

Having an integration task at hand, the first action I would take is to check my college calculus book with the hope of finding an analytical solution. Unfortunately, for a vast number of integrands and distributions, the integral (3) does not exhibit an analytical form (or at least people have yet to discover it). This is particularly the case for neural networks: except for some limited special cases,[4] in general the marginal probability is intractable, let alone the posterior and the predictive distribution.

Instead of finding tractable forms of the integral, many mathematicians have their research careers dedicated to an alternative method: *numerical integration*. Because in a continuous space one could never compute $F(\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})$ at *all* locations then sum them up, instead methods such as discretisation and Monte Carlo are employed. The Monte Carlo idea is particularly interesting in our context: since the integral is computed against a probability distribution, a naive approach would first sample from the posterior $\boldsymbol{\theta}_k \sim p(\boldsymbol{\theta}_k|\mathcal{D})$ then calculate the integral as

$$\int F(\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}) \approx \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{\theta}_k). \tag{5}$$

However this simple Monte Carlo approach assumes that the posterior distribution is easy to draw samples from, which is again intractable in most scenarios. Statisticians have applied advanced sampling schemes to (approximately) draw samples from the posterior, including importance sampling, rejection sampling and Markov chain Monte Carlo [Gelman et al., 2014]. Unfortunately, in high dimensions these methods require a considerable number of samples, and the simulation time for MCMC can be prohibitively long.

Now comes the brilliant idea of (optimisation-based, or indirect) *approximate inference*: can we find another distribution $q(\boldsymbol{\theta})$ that makes the integral $\int F(\boldsymbol{\theta})q(\boldsymbol{\theta})d\boldsymbol{\theta}$ comparably easier, and at the same time has minimal *approximation error* to the exact integral we want? Concretely, using the knowledge of the functional form F one can come up with a class of candidate distributions $\mathcal{Q}$, in which integrating F w.r.t. any $q \in \mathcal{Q}$ has analytical form or can be evaluated quickly with numerical methods. Then the only task here is to obtain the *optimal* q distribution in $\mathcal{Q}$ such that the q integral is the most accurate approximation to

[4] e.g. the prior is Gaussian and the neural network only has one hidden layer with ReLU activation.

the exact one. So in short, *approximate inference* converts the integration problem of (Bayesian) inference into an *optimisation* task. For example, an indirect[5] approach for fitting the q distribution would minimise a distance/divergence/discrepancy measure from the approximation to the exact posterior

$$q^*(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{Q}} \mathrm{D}[q(\boldsymbol{\theta}) || p(\boldsymbol{\theta}|\mathcal{D})]. \tag{6}$$

Note here the measure $\mathrm{D}[\cdot||\cdot]$ might not be symmetric. A popular choice for the divergence measure is the *Kullback-Leibler divergence* [Kullback and Leibler, 1951; Kullback, 1959] which leads to the widely used *variational inference* algorithm [Jordan et al., 1999; Beal, 2003]. In general an optimisation objective function $\mathcal{F}$ is designed to allow an accurate approximation to be obtained:

$$q^*(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{Q}} \mathcal{F}(q(\boldsymbol{\theta}); p(\boldsymbol{\theta}|\mathcal{D})), \tag{7}$$

which might not reflect a specific choice of divergence/discrepancy. Often this objective function $\mathcal{F}$ is crafted such that at the optimum, $\mathcal{F}^*$ can serve as an accurate approximation to the (log) marginal distribution, or *model evidence* $\log p(\mathcal{D})$ as well. A prevalent approach in this category considers the *Bethe free energy* [Bethe, 1935] that was first studied in statistical physics, which has also been shown as the underlying objective of another popular approach called *belief propagation* [Pearl, 1982]. All these methods are thoroughly discussed in later sections, and once q is obtained, at prediction time the Bayesian predictive distribution (2) is approximated by

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) \approx \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) q(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{8}$$

Remark (a comparison to Sampling methods)**.** Many Bayesian statisticians prefer sampling methods – and in fact it is the emergence of sampling methods such as importance sampling (IS), sequential Monte Carlo (SMC) [Doucet et al., 2001] and Markov chain Monte Carlo (MCMC) that contributes to the rapid development of Bayesian statistics. They have very nice theoretical guarantees, for example, IS and SMC provide unbiased estimates of the integral and are asymptotically exact when the number of samples $K \to +\infty$. MCMC has similar asymptotic exactness guarantee but it also requires the number of transitions $T \to +\infty$. However, I view all these sampling methods as approximate inference algorithms, simply due to the fact that in practice one can never obtain an infinite number of samples, nor simulating the MCMC dynamics for an infinite amount of time. The "effective q distribution" in use is $q(\boldsymbol{\theta}) = \frac{1}{K}\sum_{k=1}^K \delta(\boldsymbol{\theta} = \boldsymbol{\theta}^k)$ in MC(MC) and $q(\boldsymbol{\theta}) = \sum_{k=1}^K \hat{w}_k \delta(\boldsymbol{\theta} = \boldsymbol{\theta}^k)$ in (re-sampled) IS/SMC. These methods are very carefully designed to achieve guarantees of unbiasedness and consistency.

[5] a direct method would consider minimising error$(\mathbb{E}_q[F], \mathbb{E}_p[F])$, however that involves the exact integral and is mostly intractable.

Remark (a comparison to Bayesian quadrature)**.** Another important technique for approximating integrals is Bayesian quadrature [O'Hagan, 1991; Kennedy and O'Hagan, 1996; Ghahramani and Rasmussen, 2003], which has attracted a lot of attention as well and has been expanded to form part of an emerging research field called probabilistic numerics.[a] Here we note that, Bayesian quadrature and the approximate inference methods discussed above, address different intractability issues in integration tasks. Typically, Bayesian quadrature assumes the analytical form of the function F is unknown or very expensive to evaluate, and builds a probabilistic model (e.g. Gaussian process) for F given samples from the target distribution p. Approximate inference, on the other hand, constructs approximate distributions to the intractable distribution p, and considers tractable functions F instead. In short, both approaches can be categorised as *model-based approximate integration*, with the only difference that they fit approximations to different components of the integrand. Readers are also referred to e.g. approximate Bayesian computation [Beaumont et al., 2002] for those integrands without tractable F and p, and in this document we only discuss approximate inference methods and assume F is analytic and cheap to compute for a given configuration.

[a] http://www.probabilistic-numerics.org/

Part I
Algorithms for fitting approximate distributions

1 Variational inference

Many approximate inference algorithms measure the approximation quality by considering the "closeness" between the target and the approximation. Then an approximate distribution can be obtained by minimising the selected "closeness" measure, and for *variational inference* (VI) this concept of "closeness" is established as the *Kullback-Leibler divergence*.

1.1 Kullback-Leibler (KL) divergence

Kullback-Leibler divergence [Kullback and Leibler, 1951; Kullback, 1959], or *KL divergence*, is arguably one of the most widely used divergence measures, not only in approximate inference but also in machine learning, statistics, and information theory.

Definition 1. *(Kullback-Leibler Divergence) The Kullback-Leibler (KL) divergence on* $\mathcal{P}$ *is defined as a function* $\mathrm{KL}[\cdot||\cdot] : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ *with the following form*

$$\mathrm{KL}[p||q] = \int p(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}, \quad p, q \in \mathcal{P}, \tag{9}$$

where log *is the natural logarithm (to base* e*).*

One can easily check that indeed the above definition is a valid divergence. By Jensen's inequality[6] we have (9) always non-negative, and it reaches zero iff. $p = q$. Also it is clear that the KL divergence is asymmetric, i.e. $\mathrm{KL}[p||q] \neq \mathrm{KL}[q||p]$. Historically, especially when used in approximate inference context, these two cases have been referred as the *inclusive* KL divergence for $\mathrm{KL}[p||q]$, and the *exclusive* KL divergence for $\mathrm{KL}[q||p]$. These names originate from the observation that fitting q to p by minimising these two KL divergences returns results of different behaviour, detailed as follows:

- Fitting q to p by minimising $\mathrm{KL}[q||p]$:
  This KL divergence would emphasise assignment of *low* probability mass of q to the location where p is very small, thus the name "exclusive" KL. Consider a region $S \in \Theta$ that has $q(\boldsymbol{\theta}) > 0$ but $p(\boldsymbol{\theta}) = 0$ for $\boldsymbol{\theta} \in S$, then this would make the integrand in (9) infinity, thus the KL divergence assigns an extremely high cost to q here. On the other hand, if $p(\boldsymbol{\theta}) > 0$ but $q(\boldsymbol{\theta}) = 0$, then the integrand restricted to the subset S is zero, meaning that the cost

[6] Jensen's inequality: for any convex function f and distribution p, $\mathbb{E}_p[f(x)] \geq f(\mathbb{E}_p[x])$.

for missing a region with positive p mass is much lower. We also refer this property as "zero-forcing", or "mode-seeking" when q is restricted to be uni-modal.

- Fitting q to p by minimising $\mathrm{KL}[p||q]$:
  Conversely, this KL divergence would emphasise assignment of *high* probability mass of q to the location where p has positive mass, thus the name "inclusive" KL. Consider the case that $q(\boldsymbol{\theta}) > 0$ but $p(\boldsymbol{\theta}) = 0$, then this would make the integrand in (9) zero. In contrast, if $p(\boldsymbol{\theta}) > 0$ but $q(\boldsymbol{\theta}) = 0$, then the integrand is infinity, meaning that the cost for missing a region with positive p mass is extremely high. We also refer this property as "mass-covering".

Remark (maximum likelihood estimation and KL divergences)**.** We will show that maximum likelihood estimation (MLE) is equivalent to minimising a KL divergence. For a given dataset $\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$, define the empirical distribution as $\hat{p}_{\mathcal{D}}(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_n)$ where $\delta(\cdot)$ denotes the Dirac delta function. Then we want to fit the data with a parametric probabilistic model $p(\boldsymbol{x}|\boldsymbol{\theta})$ using MLE:

$$\hat{\boldsymbol{\theta}}^{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}\in\Theta} \frac{1}{N}\sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}). \tag{10}$$

Simple calculation reveals that maximising the log-likelihood of $\boldsymbol{\theta}$ is equivalent to minimising the KL divergence

$$\hat{\boldsymbol{\theta}}^{\mathrm{ML}} = \arg\min_{\boldsymbol{\theta}\in\Theta} \mathrm{KL}[\hat{p}_{\mathcal{D}}(\boldsymbol{x})||p(\boldsymbol{x}|\boldsymbol{\theta})] = \arg\min_{\boldsymbol{\theta}\in\Theta} -\frac{1}{N}\sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) + \text{const.}$$

MLE is widely used in all types of machine learning tasks, e.g. learning generative models. We will come back to this topic in later sections.

1.2 Variational free-energy

Unfortunately, direct divergence minimisation is still intractable, since that involves evaluating the target distribution itself. For example, consider minimising the exclusive KL divergence $\mathrm{KL}[q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})]$ to obtain the approximate posterior. But we still need to compute $p(\boldsymbol{\theta}|\mathcal{D})$, and in particular the marginal likelihood $p(\mathcal{D})$ which is intractable. In this section we discuss variational inference (VI) – a widely used approximate inference algorithm – which incorporates divergence minimisation in a smart way. To emphasise that the algorithm is applicable to more general cases beyond posterior approximation, we now write the *target distribution* as

$$\pi(\boldsymbol{\theta}) = \frac{1}{Z}\pi^*(\boldsymbol{\theta}),$$

where $\pi^*(\boldsymbol{\theta})$ is the *unnormalised* target distribution and $Z = \int \pi^*(\boldsymbol{\theta})d\boldsymbol{\theta}$ is the *normalising constant* or *partition function*. In the posterior approximation context

$\pi^*(\boldsymbol{\theta}) = p(\boldsymbol{\theta}, \mathcal{D})$ and $Z = p(\mathcal{D})$.

As already discussed, the exclusive KL divergence minimisation problem is intractable. Fortunately the minimiser of the exclusive KL can also be obtained by an equivalent minimisation problem of the so called *variational free-energy* (VFE):

$$\begin{aligned} \min_q \mathcal{F}_{\text{VFE}}(q; p) &:= \min_q \text{KL}[q(\boldsymbol{\theta})||\pi(\boldsymbol{\theta})] - \log Z \\ &= \min_q \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{\pi^*(\boldsymbol{\theta})} d\boldsymbol{\theta}. \end{aligned} \tag{11}$$

This is because the normalising constant Z is independent with the approximation q, thus can be dropped in the exclusive KL. Historically the negative of the variational free-energy is also frequently discussed, which is named *variational lower-bound* or *evidence lower-bound (ELBO)* in the context of posterior approximation

$$\mathcal{L}_{\text{VI}}(q; p) := -\mathcal{F}_{\text{VFE}}(q; p) = \int q(\boldsymbol{\theta}) \log \frac{\pi^*(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}. \tag{12}$$

The lower-bound property comes from the fact that $\log Z \geq \mathcal{L}_{\text{VI}}(q; p)$, because of the non-negativity of KL divergence. Equivalently, this property can also be derived as follows:

$$\begin{aligned} \log Z &= \log \int \pi^*(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= \log \int q(\boldsymbol{\theta}) \frac{\pi^*(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} \\ &\geq \int q(\boldsymbol{\theta}) \log \frac{\pi^*(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}. \qquad \text{\# Jensen's inequality} \end{aligned}$$

Here Jensen's inequality is applied to the logarithm which is concave. When posterior approximation is considered we also denote the two quantities as $\mathcal{F}_{\text{VFE}}(q; \mathcal{D})$ and $\mathcal{L}_{\text{VI}}(q; \mathcal{D})$, respectively. In summary, variational inference finds an approximation to the posterior through an *optimisation* process, which is drastically different from sampling approaches that construct *empirical point mass* distributions to describe the posterior.

1.2.1 A brief history of variational inference

Variational inference can be viewed as an application of *variational methods* that mathematicians and physicists have studied for centuries. Historically, physicists mainly focused on mean-field theories for complex systems [Parisi, 1988], whereas Dempster et al. [1977] as statisticians proposed the famous *expectation maximisation* (EM) algorithm that also has a VI interpretation [Neal and Hinton, 1998]. Interestingly the pioneers of deep learning had also applied variational inference (though under other names) for Bayesian neural networks [Peterson and Anderson, 1987; Hinton and Van Camp, 1993] that will be surveyed later. Especially since the development of [Peterson and Anderson, 1987], mean-field approximations started to be an attractive alternative to sampling methods for probabilistic inference in graphical models [Ghahramani, 1995].

However it was until Saul et al. [1996] which introduced the generic form of the variational lower-bound to explain the mean-field approximation. The first papers that I can find which coined the term "variational inference" are Lawrence et al. [1998] and Jordan et al. [1999], where Jordan et al. [1999] provided a detailed summary of the previous work coming from the same group. Later on, researchers started to extend the variational principle to cases beyond graphical models, e.g. the *variational Bayes* (VB) algorithm [Attias, 1999, 2000; Sato, 2001; Beal, 2003] that is used to perform posterior approximations of the model parameters and even model selection.

1.3 A mean-field approximation example

As an example for the variational inference algorithm, here we present the variational mean-field approximation [Parisi, 1988] for Bayesian linear regression. Readers are also refer to [Bishop, 2006] for more details and here we would briefly cover the derivations presented there. Mean-field approximation, also known as the factorised approximation, assumes the approximate posterior to be the form of

$$q(\boldsymbol{\theta}) := \prod_{i=1}^{D} q_i(\theta_i). \tag{13}$$

In general one can partition the elements of $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_D)$ into disjoint groups and apply factorisations over groups. This general case is usually called *structured* mean-field approximation [Saul and Jordan, 1996], and for simplicity in the following example we only consider the fully factorised case (13). Also we emphasise that there's no further assumption/restriction that is made on the functional form of $q_i(\theta_i)$. As we shall see, the variational free-energy is still convex in $q_i(\theta_i)$ and thus the solution provided by the following is the global optimum.

To derive the best approximation in the mean-field distribution family, we first substitute (13) into (11) (and use $\boldsymbol{\theta}_{\neq j}$ to denote all the θ_i variables except θ_j):

$$\begin{aligned}
\mathcal{F}_{\text{VFE}}(q; p) &= \int \prod_i q_i(\theta_i) \left(\sum_i \log q_i(\theta_i) - \log \pi^*(\boldsymbol{\theta}) \right) d\boldsymbol{\theta} \\
&= \int q_j \log q_j(\theta_j) d\theta_j - \int q_j(\theta_j) \left(\int \prod_{i \neq j} q_i(\theta_i) \log \pi^*(\boldsymbol{\theta}) d\boldsymbol{\theta}_{\neq j} \right) d\theta_j + \text{const} \\
&:= \int q_j(\theta_j) \log q_j(\theta_j) d\theta_j - \int q_j(\theta_j) \log \tilde{p}(\theta_j) d\theta_j + \text{const},
\end{aligned}$$

where $\tilde{p}(\theta_j)$ denote the "marginal" distribution satisfying

$$\log \tilde{p}(\theta_j) = \int \prod_{i \neq j} q_i(\theta_i) \log \pi^*(\boldsymbol{\theta}) d\boldsymbol{\theta}_{\neq j} + \text{const}.$$

This means, by fixing the functional form of q_i for all $i \neq j$, VFE is reduced to the KL-divergence $\text{KL}[q_j(\theta_j) || \tilde{p}(\theta_j)]$ plus a constant that is independent to q_j.

Thus the free-energy is still convex in q_j, in which the unique global optimum is obtained by setting $q_j(\theta_j) = \tilde{p}(\theta_j)$. To be precise, we explicitly write down the optimal mean-field approximation as

$$q(\theta_j) = \frac{\exp\left[\int \prod_{i\neq j} q_i(\theta_i) \log \pi^*(\boldsymbol{\theta}) d\boldsymbol{\theta}_{\neq j}\right]}{\int \exp\left[\int \prod_{i\neq j} q_i(\theta_i) \log \pi^*(\boldsymbol{\theta}) d\boldsymbol{\theta}_{\neq j}\right] d\theta_j}. \tag{14}$$

Now as an example consider Bayesian linear regression with 2-D inputs $\boldsymbol{x}$ and 1-D output y:

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0^{-1}), \quad y|\boldsymbol{x} \sim \mathcal{N}(y; \boldsymbol{\theta}^T \boldsymbol{x}, \sigma^2).$$

Given the observations $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, the posterior distribution of $\boldsymbol{\theta}$ can be computed analytically as $p(\boldsymbol{\theta}|\mathcal{D}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$ with $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_0 + \frac{1}{\sigma^2}\sum_n \boldsymbol{x}_n \boldsymbol{x}_n^T$ and $\boldsymbol{\Lambda}\boldsymbol{\mu} = \boldsymbol{\Lambda}_0\boldsymbol{\mu}_0 + \frac{1}{\sigma^2}\sum_n y_n \boldsymbol{x}_n$. To see how the mean-field approach works, we explicitly write down the elements of the posterior parameters

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix}, \quad \Lambda_{12} = \Lambda_{21},$$

Then by explicitly expanding the mean-field solution (14):

$$\begin{aligned} \log q_1(\theta_1) &= \int q_2(\theta_2) \log p(\boldsymbol{\theta}, \mathcal{D}) d\theta_2 + \text{const} \\ &= \mathbb{E}_{q_2}\left[-\frac{1}{2}(\theta_1 - \mu_1)^2 \Lambda_{11} - (\theta_1 - \mu_1)\Lambda_{12}(\theta_2 - \mu_2)\right] + \text{const} \\ &= -\frac{1}{2}\theta_1^2 \Lambda_{11} + \theta_1 \mu_1 \Lambda_{11} - \theta_1 \Lambda_{12}(\mathbb{E}_{q_2}[\theta_2] - \mu_2) + \text{const} \\ &:= \log \mathcal{N}(\theta_1; m_1, \lambda^{-1}) + \text{const} \end{aligned} \tag{15}$$

where the new mean m_1 and the precision λ_1 satisfies

$$m_1 = \mu_1 - \Lambda_{11}^{-1}\Lambda_{12}(\mathbb{E}_{q_2}[\theta_2] - \mu_2), \quad \lambda_1 = \Lambda_{11}.$$

It is important to note here that we do not assume the approximation to be a Gaussian distribution in order to obtain the last equation in (15). Rather the Gaussian distribution solution came out from the derivation of the global optimum (14) and the completion of the square form. One can derive the terms $m_2 = \mu_2 - \Lambda_{22}^{-1}\Lambda_{21}(\mathbb{E}_{q_1}[\theta_1] - \mu_1)$ and $\lambda_2 = \Lambda_{22}$ for q_2 in the same way, and show that $\boldsymbol{m} = \boldsymbol{\mu}$ is the only stable fixed point of this iterative update. So we have $q_1 = \mathcal{N}(\theta_1; \mu_1, \Lambda_{11}^{-1})$, and similarly $q_2 = \mathcal{N}(\theta_1; \mu_2, \Lambda_{22}^{-1})$ as the unique global optimum of variational mean-field approximation. A visualisation of the mean-field approximation is provided in Figure 1. Note here the variance parameter of $q(\theta_1)$ also correspond to the variance of the conditional distribution $p(\theta_1|\theta_2, \mathcal{D})$, which is smaller than the variance of the marginal distribution $p(\theta_1|\mathcal{D})$, and therefore mean-field VI under-estimates the posterior uncertainty in this case.[7]

[7] We also have $\mathbb{H}[p] = -\frac{1}{2}\log|\Lambda_{11}\Lambda_{22} - \Lambda_{12}\Lambda_{21}| + \text{const} \geq \mathbb{H}[q] = -\frac{1}{2}\log|\Lambda_{11}\Lambda_{22}| + \text{const}$.

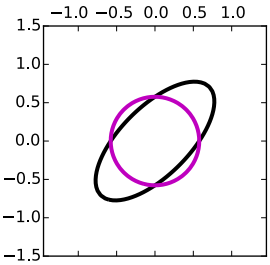

Figure 1: Mean-field approximation to the exact posterior distribution in the Bayesian linear regression example (with one-sigma contours). The exact posterior contour is shown in black and the variational approximation is in purple.

Remark (Does VI always fit to a local mode of the target distribution?)**.** VI is usually referred as a "mode seeking" method in that when a single-mode q distribution (such as Gaussian) is fitted to a multi-mode distribution, the optimal q solution often captures one of the modes. For example, consider the target distribution $p(\boldsymbol{\theta}) = \frac{1}{2}p_1(\boldsymbol{\theta}) + \frac{1}{2}p_2(\boldsymbol{\theta})$ where $supp(p_1) \cap supp(p_2) = \emptyset$. Due to the zero-forcing property of the exclusive KL-divergence (see § 1.1) q is forced to fit one of the modes of p. However, if the mixture component p_1 and p_2 has a significant amount of overlapping density mass, then q might not fit to one mode of p only. For example, Figure 2 from Turner and Sahani [2011] showed that, the optimal variational approximation q can even over-estimate the entropy of the target distribution p. Therefore it is a nice counter-example on the claim that variational inference approximation "always under-estimates the uncertainty".

1.4 Further reading

Jordan et al. [1999] presents VI in probabilistic graphical model context.

Wainwright and Jordan [2008]: a book-length paper that teaches both VI and message passing (and more!) in the context of probabilistic graphical models. I recommend reading chapters 2, 3 and 6 at this point, as they provide explanations on how variational methods relate to convex optimisation.

Beal [2003] is a highly-cited thesis mainly for variational inference. I recommend reading chapter 2 to start with, it shows how the EM algorithm relates to variational inference, and it also sketches variational EM as a fast variant.

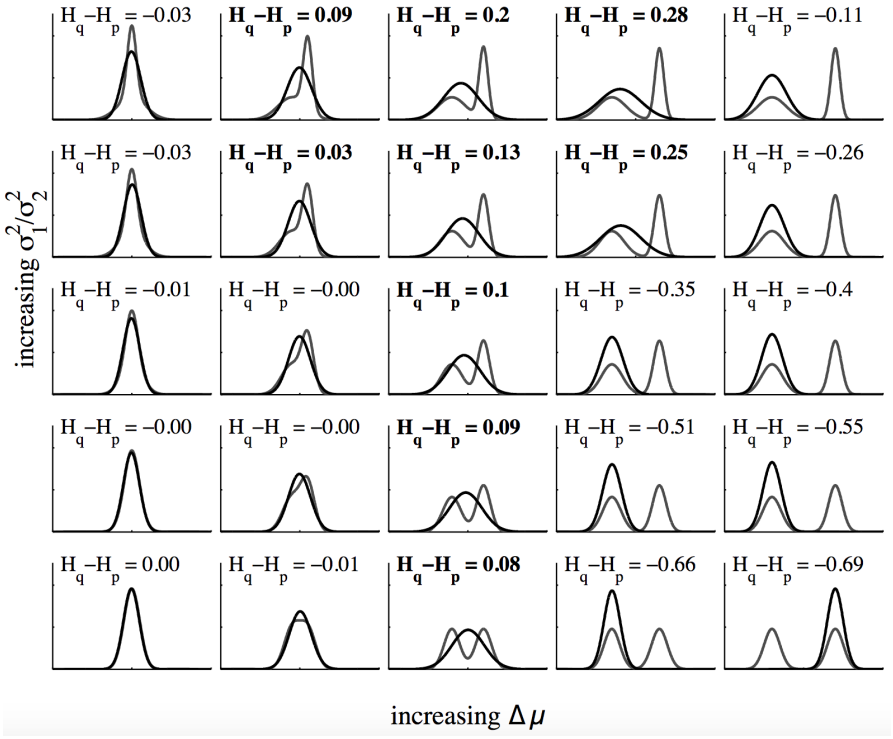

Figure 2: Variational approximation to the target distribution which is a mixture of two Gaussians: $\pi(\theta) = \frac{1}{2}\mathcal{N}(\theta; \mu_1, \sigma_1^2) + \frac{1}{2}\mathcal{N}(\theta; \mu_2, \sigma_2^2)$. The difference between means $\Delta\mu = \mu_1 - \mu_2$ increases from 0 for the left column panels to 10 for the right column panels. The ratio between variances σ_1^2/σ_2^2 (with fixed $\sigma_2^2 = 1$) also increases from 0 for the bottom row panels to 10 for the top row panels. Figure reproduced from Turner and Sahani [2011].

2 Monte Carlo variational inference

In the mean-field VI example we have discussed variational inference algorithms for linear regression. But real world problems are much more complicated. Often, the "reconstruction error" term $\mathbb{E}_q[\log \pi^*(\boldsymbol{\theta})]$ in the variational free energy lacks an analytical form. A prevalent example of such cases is variational inference for *large-scale data*: here the unnormalised distribution $\pi^*(\boldsymbol{\theta}) := p(\boldsymbol{\theta}, \mathcal{D})$ is proportional to the product of many likelihood functions, and evaluating $\mathbb{E}_q[\log \pi^*(\boldsymbol{\theta})]$ requires a pass of the whole dataset, which can be very expensive. On the other hand, insisting on having an analytical form of the entropy term $\mathbb{H}[q]$ (or $\mathrm{KL}[q||p_0]$) would restrict the selection of q distributions to simple distributions like Gaussians. Usually the exact posterior is very complicated, and these simple distributions are expected to be poor approximations to the target distribution. Hence a key challenge here is, can we design a variational algorithm that applies to complex models, and scales to big data?

One solution to the above request is to develop further approximation techniques *specific to the chosen variational approximation*. Indeed in the early days researchers attempted to do so, e.g. see Jaakkola and Jordan [1998]. However these solutions are applicable only to a handful of special cases, making them impractical in many other interesting scenarios. Instead in this section we will review another approach which can be quickly applied to many cases with little effort. It has also been referred as a "black-box"[8] approach [Ranganath et al., 2014] due to this feature, but in the rest of the thesis we will refer it as Monte Carlo VI (MC-VI or MCVI) [Paisley et al., 2012; Wingate and Weber, 2013].

2.1 Monte Carlo estimation of the variational lower-bound

To see how MCVI works, consider approximating the exact posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}) \propto \prod_n p(\boldsymbol{x}_n|\boldsymbol{\theta})p_0(\boldsymbol{\theta})$ by some simpler distribution $q(\boldsymbol{\theta})$. Rewriting the variational lower-bound:

$$\mathcal{L}_{\mathrm{VI}}(q; p) = \sum_{n=1}^{N} \mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})] + \mathbb{E}_q[\log p_0(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta})], \tag{16}$$

we see that it is the analytical tractability requirement of computing the expectations that restrict the q (and possibly the model p) distribution to be of simple form. This constraint can be removed by considering Monte Carlo (MC) approximation to the expectation, which estimates the expectation by, for example

$$\mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})] \approx \frac{1}{K} \sum_{k=1}^{K} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}^k), \quad \boldsymbol{\theta}^k \sim q(\boldsymbol{\theta}). \tag{17}$$

This forms an *unbiased* estimation, and under mild assumptions, the RHS term in (17) converges to the exact expectation value as $K \to +\infty$. The KL-divergence

[8] Here "black-box" means that the method can be applied to inference problems of many probabilistic models without specific modifications of the optimisation algorithm. It does not imply the model p or the q distribution is a black-box (detailed in later topics).

term in the variational lower-bound can also be estimated with Monte Carlo in a similar manner.

Also stochastic optimisation techniques can be extended here for scalability. Notice that we have made the i.i.d. assumption on data, so that any statistics on the dataset can be estimated by mini-batch sampling in an unbiased way. Specifically, we have:

$$\sum_{n=1}^{N} \mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})] = \mathbb{E}_{\mathcal{S}\sim\mathcal{D}^{|\mathcal{S}|}}\left[\frac{N}{|\mathcal{S}|}\sum_{n\in\mathcal{S}} \mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})]\right], \tag{18}$$

where $\mathcal{S}\sim\mathcal{D}^{|\mathcal{S}|}$ means sampling a mini-batch of datapoints from $\mathcal{D}$ with replacement. This approach is widely used in stochastic optimisation in general, due to the fact that

$$\begin{aligned}\nabla_{\boldsymbol{\phi}} \sum_{n=1}^{N} \mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})] &= \sum_{n=1}^{N} \nabla_{\boldsymbol{\phi}} \mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})] \\ &= \mathbb{E}_{\mathcal{S}\sim\mathcal{D}^{|\mathcal{S}|}}\left[\frac{N}{|\mathcal{S}|}\sum_{n\in\mathcal{S}} \nabla_{\boldsymbol{\phi}}\mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})]\right] \\ &= \nabla_{\boldsymbol{\phi}}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}^{|\mathcal{S}|}}\left[\frac{N}{|\mathcal{S}|}\sum_{n\in\mathcal{S}} \mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})]\right],\end{aligned} \tag{19}$$

which means we can directly differentiate the mini-batch estimate of the "reconstruction accuracy" part of the variational lower-bound in order to obtain a stochastic estimate of the full gradient.

In summary, with the "black-box" approach of Monte Carlo integration and the stochastic optimisation approach of mini-batch sampling, one can approximate the variational lower-bound as

$$\mathcal{L}_{\text{VI}}^{MC}(q;p) = \frac{N}{|\mathcal{S}|}\sum_{n\in\mathcal{S}}\frac{1}{K}\sum_{k}\log p(\boldsymbol{x}_n|\boldsymbol{\theta}^k) + \frac{1}{K}\sum_{k}[\log p_0(\boldsymbol{\theta}^k) - \log q(\boldsymbol{\theta}^k)], \quad \boldsymbol{\theta}^k \sim q(\boldsymbol{\theta}), \tag{20}$$

and compute stochastic gradient descent on the MC approximation (20) with mini-batch $\mathcal{S}\sim\mathcal{D}^{|\mathcal{S}|}$.

> **Remark** (MC samples for different observations)**.** In the MC approximation (20) we assumed using the same set of samples $\{\boldsymbol{\theta}^k\}$ to estimate all the expectation terms. In general we can use different sets of samples to do so, for example, for every datapoint $\boldsymbol{x}_n \in \mathcal{S}$, we can sample different sets of $\boldsymbol{\theta}^k$ to estimate the associated reconstruction term $\mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})]$. Prevalent examples of this approach include stochastic regularisation techniques (SRTs) such as dropout [Srivastava et al., 2014; Kingma et al., 2015; Gal, 2016].

2.2 Computing the MCVI gradients

Practitioners care a lot more about the stochastic optimisation process of the MCVI algorithm compared to computing the MCVI bound. As the training of

machine learning models relies on gradient descent based optimisation methods mainly, in this section we will detail some tricks of computing the gradient of the variational lower-bound using MC approximations.

2.2.1 LOTUS/reparameterisation trick and path gradients

We start by assuming $\boldsymbol{\theta}$ a continuous variable, and the discrete case will be discussed later. Here we introduce a neat trick called the *law of the unconscious statistician* (LOTUS). It has been extended to the *reparameterisation trick* in variational inference context [Salimans and Knowles, 2013; Kingma and Welling, 2014; Rezende et al., 2014]. This trick, along with MC approximation, makes the variational lower-bound easy to handle. It comes from a very simple observation: given a distribution $p(\boldsymbol{\theta})$, if sampling $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$ is equivalent to first sampling a "noise" variable $\boldsymbol{\epsilon} \sim \pi(\boldsymbol{\epsilon})$ and then computing a mapping $\boldsymbol{\theta} = \boldsymbol{f}(\boldsymbol{\epsilon})$, then the expectation of some function $F(\boldsymbol{\theta})$ under distribution $p(\boldsymbol{\theta})$ can be rewritten as

$$\mathbb{E}_{p(\boldsymbol{\theta})}[F(\boldsymbol{\theta})] = \mathbb{E}_{\pi(\boldsymbol{\epsilon})}[F(\boldsymbol{f}(\boldsymbol{\epsilon}))].$$

This derives from the change of variable in PDFs. LOTUS applies to any transformation, but the reparamterisation trick specifically asks the transformation $\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})$ to be differentiable w.r.t. its parameters $\boldsymbol{\phi}$. In variational inference context, the sampling procedure is required to be

$$\boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \Leftrightarrow \boldsymbol{\epsilon} \sim \pi(\boldsymbol{\epsilon}), \quad \boldsymbol{\theta} = \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}).$$

Then using the LOTUS rule, the variational lower-bound is computed as (we ignore the entropy term for a moment):

$$\mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}; p) = \mathbb{E}_{\pi(\boldsymbol{\epsilon})}[\log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] + \mathbb{H}[q_{\boldsymbol{\phi}}], \tag{21}$$

and the gradient w.r.t. $\boldsymbol{\phi}$ is the following:

$$\begin{aligned} \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}; p) &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{\pi(\boldsymbol{\epsilon})}[\log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] + \nabla_{\boldsymbol{\phi}} \mathbb{H}[q_{\boldsymbol{\phi}}] \\ &= \mathbb{E}_{\pi(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} \log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] + \nabla_{\boldsymbol{\phi}} \mathbb{H}[q_{\boldsymbol{\phi}}] \quad \text{\# } \pi(\boldsymbol{\epsilon}) \text{ independent to } \boldsymbol{\phi} \\ &= \mathbb{E}_{\pi(\boldsymbol{\epsilon})}[\nabla_{\mathbf{f}} \log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})) \nabla_{\boldsymbol{\phi}} \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})] + \nabla_{\boldsymbol{\phi}} \mathbb{H}[q_{\boldsymbol{\phi}}]. \quad \text{\# chain rule} \end{aligned} \tag{22}$$

The gradient derived by the chain rule is also called the *path* gradient, and it relies on the assumption that $\mathbf{f}_{\boldsymbol{\phi}}$ is differentiable w.r.t. $\boldsymbol{\phi}$. With MC approximation, the gradient of the "error" term in (22) is further approximated as

$$\frac{1}{K} \sum_{k=1}^{K} \nabla_{\mathbf{f}} \log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k)) \nabla_{\boldsymbol{\phi}} \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k), \quad \boldsymbol{\epsilon}^k \sim \pi(\boldsymbol{\epsilon}). \tag{23}$$

Let us consider a simple but prevalent example, where the q distribution is designed to be Gaussian, i.e. $q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$. In practice the covariance matrix $\boldsymbol{\Sigma}$ is often parameterised using its Cholesky decomposition $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$, in this case

$\boldsymbol{\phi} = \{\boldsymbol{\mu}, \mathbf{L}\}$, and the transformation is written as $\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}) = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This means the MC gradient of the error term is

$$\begin{aligned} \nabla_{\boldsymbol{\mu}} &= \frac{1}{K}\sum_{k=1}^{K} \nabla_{\boldsymbol{\mu}} \log p(\mathcal{D}, \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}^k), \\ \nabla_{\mathbf{L}} &= \frac{1}{K}\sum_{k=1}^{K} \nabla_{\boldsymbol{\mu}} \log p(\mathcal{D}, \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}^k)\boldsymbol{\epsilon}^k, \quad \boldsymbol{\epsilon}^k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \end{aligned} \tag{24}$$

In practice, the above gradient expressions are not directly implemented. Instead, using the assumption that $\mathbf{f}_{\boldsymbol{\phi}}$ is differentiable w.r.t. $\boldsymbol{\phi}$, we can rewrite the MC approximation of the gradient as

$$\begin{aligned} \frac{1}{K}\sum_{k=1}^{K} \nabla_{\mathbf{f}} \log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k)) \nabla_{\boldsymbol{\phi}} \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k) &= \frac{1}{K}\sum_{k=1}^{K} \nabla_{\boldsymbol{\phi}} \log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k)) \\ &= \nabla_{\boldsymbol{\phi}} \frac{1}{K}\sum_{k=1}^{K} \log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k)) \\ &\approx \nabla_{\boldsymbol{\phi}} \mathbb{E}_{\pi(\boldsymbol{\epsilon})}[\log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))]. \end{aligned} \tag{25}$$

This means with the path gradient approach, we can directly differentiate the MC + LOTUS estimate of $\frac{1}{K}\sum_{k=1}^{K} \log p(\mathcal{D}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k)) \approx \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}[\log p(\mathcal{D}, \boldsymbol{\theta})]$ to get the gradient of the "reconstruction error" part of the variational lower-bound. The usages of MC estimates, LOTUS and path gradient are the central ideas of the reparameterisation trick.

Choosing the number of MC samples K in use requires a trade-off between speed and accuracy. In general, using small K (even $K = 1$ in many cases) leads to high variance of the MC gradient. Using large K, on the other hand, can significantly slow down the training process in wall-clock time.

2.2.2 Path gradient of the entropy term

The derivation of the path gradient for the entropy term is slightly involved. This is because, apart from the transformation $\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})$, the PDF $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ also depends on the variational parameters $\boldsymbol{\phi}$. Going back to the Gaussian example, we have the (log) PDF as

$$\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}) + \text{const},$$

in which the (log) normalising constant (or partition function) $\frac{1}{2}\log|\boldsymbol{\Sigma}| + \text{const}$ depends on $\boldsymbol{\phi}$ as well. Therefore, in general the gradient of the entropy term w.r.t.$\boldsymbol{\phi}$ is

$$\begin{aligned} \nabla_{\boldsymbol{\phi}} \mathbb{H}[q_{\boldsymbol{\phi}}] &= -\nabla_{\boldsymbol{\phi}} \mathbb{E}_{\pi(\boldsymbol{\epsilon})}[\log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] \\ &= -\mathbb{E}_{\pi(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] \\ &= -\mathbb{E}_{\pi(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})} + \nabla_{\mathbf{f}} \log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))\nabla_{\boldsymbol{\phi}}\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})] \\ &= -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}[\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})] - \mathbb{E}_{\pi(\boldsymbol{\epsilon})}[\nabla_{\mathbf{f}} \log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))\nabla_{\boldsymbol{\phi}}\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})]. \end{aligned} \tag{26}$$

We see that due to the dependance of the PDF to $\boldsymbol{\phi}$ we have the gradient splitting into two terms. Interestingly, we can show that the first term (non-path gradient, which is also the expectation of the *score function*) in (26) eventually vanishes:

$$\begin{aligned}\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}[\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})] &= \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) q_{\boldsymbol{\phi}}(\boldsymbol{\theta})^{-1} \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) d\boldsymbol{\theta} \quad \text{\# chain rule of log gradient} \\ &= \int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) d\boldsymbol{\theta} = 0. \quad \text{\# } q_{\boldsymbol{\phi}} \text{ always integrates to one}\end{aligned} \tag{27}$$

This means using the reparameterisation trick, the gradient of the entropy term w.r.t. the variational parameter $\boldsymbol{\phi}$ can also be derived as a path derivative:

$$\nabla_{\boldsymbol{\phi}} \mathbb{H}[q_{\boldsymbol{\phi}}] \approx -\frac{1}{K} \sum_{k=1}^{K} \nabla_{\mathbf{f}} \log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k)) \nabla_{\boldsymbol{\phi}} \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k), \quad \boldsymbol{\epsilon}^k \sim \pi(\boldsymbol{\epsilon}). \tag{28}$$

Notice that if you implement the entropy term in a naive way like $\mathbb{H}[q_{\boldsymbol{\phi}}] \approx -\frac{1}{K}\sum_{k=1}^{K} \log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k))$, and ask automatic differentiation to handle the gradient, the software will still include the (MC estimate of the) non-path gradient without knowing that it is actually zero in expectation. Roeder et al. [2017] empirically demonstrate that this can lead to high variance, and instead they suggest implement the MC approximated entropy in the following way:

$$\mathbb{H}[q_{\boldsymbol{\phi}}] \approx -\frac{1}{K} \sum_{k=1}^{K} \log q_{\boldsymbol{\phi}'}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^k)), \quad \boldsymbol{\phi}' = \text{stop\_gradient}(\boldsymbol{\phi}).$$

2.2.3 Log derivative trick and REINFORCE

The reparameterisation trick only works when there exists a differentiable transformation $\mathbf{f}_{\boldsymbol{\phi}}$ and the input noise variable $\boldsymbol{\epsilon}$ is independent to the variational parameter $\boldsymbol{\phi}$. This does not apply to discrete variables, although recent work has tried continuous relaxation techniques to enable path gradients [Maddison et al., 2017b; Jang et al., 2017]. So without any assumption on the random variable $\boldsymbol{\theta}$, the gradient of the variational lower-bound w.r.t. $\boldsymbol{\phi}$ reads

$$\begin{aligned}\nabla_{\boldsymbol{\phi}} \mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}; p) &= \int \nabla_{\boldsymbol{\phi}} \left(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \right) d\boldsymbol{\theta} \\ &= \int \log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) d\boldsymbol{\theta} + \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\phi}} \log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} d\boldsymbol{\theta} \\ &= \int \log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) d\boldsymbol{\theta}. \quad \text{\# the second term is zero, see (27)}\end{aligned} \tag{29}$$

Now we apply the *log derivative trick* which is also named as the REINFORCE trick in reinforcement learning literature [Williams, 1992]. This trick is also closely related to the likelihood ratio method in statistics literature, e.g. see Glynn [1990]. It states that for any function F,

$$\begin{aligned}&\int F(\boldsymbol{\theta})\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta})d\boldsymbol{\theta}\\ =&\int F(\boldsymbol{\theta})\frac{\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{\theta})d\boldsymbol{\theta}\\ =&\int q_{\boldsymbol{\phi}}(\boldsymbol{\theta})F(\boldsymbol{\theta})\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})d\boldsymbol{\theta}.\end{aligned}$$

Therefore applying the log derivative trick to (29), we can see the MC approximated gradient is

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}; p) \approx \frac{1}{K}\sum_{k=1}^{K}\log\frac{p(\mathcal{D}, \boldsymbol{\theta}^k)}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}^k)}\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}^k), \tag{30}$$

which is an unbiased, but high variance estimator of the exact gradient.

2.3 Variance reduction for MCVI gradients

It is now clear that in practice the optimisation problem of variational inference is often solved using MC approximated gradients. Therefore, the variance of the MC gradients is key to the performance, since if the variance is too high, then the MC gradient can point to wrong directions and thus slow down the convergence. In fact, variance reduction has become a major research topic not only in approximate inference but also in reinforcement learning and optimisation.[9] In this section we will briefly cover some important techniques for variance reduction in the context of variational inference.

2.3.1 Rao-Blackwellization

Let us start from a very simple example. We assume for now $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ and we would like to estimate $\mathbb{E}_{q(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}[F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)]$ with Monte Carlo. Then the Rao-Blackwell theorem [Rao et al., 1973; Blackwell, 1947; Kolmogorov, 1950] states that, the variance of the estimates can be reduced by conditioning on either $\boldsymbol{\theta}_1$ or $\boldsymbol{\theta}_2$. Mathematically, if we write $F_2(\boldsymbol{\theta}_2) = \mathbb{E}_{q(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2)}[F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)]$, then the variance is

$$\begin{aligned}\mathbb{V}_{q(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}[F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)] =&\mathbb{E}_{q(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}[(F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) - \mathbb{E}_{q(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}[F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)])^2]\\ =&\mathbb{E}_{q(\boldsymbol{\theta}_2)}\mathbb{E}_{q(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2)}[(F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) - F_2(\boldsymbol{\theta}_2) + F_2(\boldsymbol{\theta}_2) - \mathbb{E}_{q(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}[F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)])^2]\\ =&\mathbb{E}_{q(\boldsymbol{\theta}_2)}[(F_2(\boldsymbol{\theta}_2) - \mathbb{E}_{q(\boldsymbol{\theta}_2)}[F_2(\boldsymbol{\theta}_2)])^2] + \mathbb{E}_{q(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}[(F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) - F_2(\boldsymbol{\theta}_2))^2]\\ =&\mathbb{V}_{q(\boldsymbol{\theta}_2)}[F_2(\boldsymbol{\theta}_2)] + \mathbb{E}_{q(\boldsymbol{\theta}_2)}[\mathbb{V}_{q(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2)}[F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)]]\\ \geq&\mathbb{V}_{q(\boldsymbol{\theta}_2)}[F_2(\boldsymbol{\theta}_2)].\end{aligned} \tag{31}$$

[9] In NeurIPS 2016 three tutorials on these topics had spent considerate amount of time discussing variance reduction techniques.

How does Rao-Blackwellization apply to variance reduction for MCVI gradients? If we assume the approximate posterior factorises as $q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)q_{\boldsymbol{\phi}_2}(\boldsymbol{\theta}_2)$, then the MCVI gradient for $\boldsymbol{\phi}_1$ (and similarly for $\boldsymbol{\phi}_2$) reads

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}_1}\mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}; p) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\left[\log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \nabla_{\boldsymbol{\phi}_1} \log q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)}\left[\mathbb{E}_{q_{\boldsymbol{\phi}_2}(\boldsymbol{\theta}_2)}\left[\log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\right] \nabla_{\boldsymbol{\phi}_1} \log q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)}\left[\left[\mathbb{E}_{q_{\boldsymbol{\phi}_2}(\boldsymbol{\theta}_2)}[\log p(\mathcal{D}, \boldsymbol{\theta})] - \log q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1) + \mathbb{H}[q_{\boldsymbol{\phi}_2}(\boldsymbol{\theta}_2)]\right] \nabla_{\boldsymbol{\phi}_1} \log q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)}\left[\left[\mathbb{E}_{q_{\boldsymbol{\phi}_2}(\boldsymbol{\theta}_2)}[\log p(\mathcal{D}, \boldsymbol{\theta})] - \log q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)\right] \nabla_{\boldsymbol{\phi}_1} \log q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)\right],
\end{aligned} \tag{32}$$

where in the last line derivation we used the fact that $\mathbb{H}[q_{\boldsymbol{\phi}_2}(\boldsymbol{\theta}_2)]$ is a constant w.r.t. $q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)$ and the same trick as in (27). This means, if we can compute $\mathbb{E}_{q_{\boldsymbol{\phi}_2}(\boldsymbol{\theta}_2)}[\log p(\mathcal{D}, \boldsymbol{\theta})]$ (or at least approximate it with many samples in a fast way), then the following MCVI gradient

$$\nabla_{\boldsymbol{\phi}_1}\mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}; p) \approx \frac{1}{K}\sum_{k=1}^{K}\left[\mathbb{E}_{q_{\boldsymbol{\phi}_2}(\boldsymbol{\theta}_2)}[\log p(\mathcal{D}, \boldsymbol{\theta}_1^k, \boldsymbol{\theta}_2)] - \log q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1^k)\right] \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1^k) \tag{33}$$

with $\boldsymbol{\theta}_1^k \sim q_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}_1)$ will have smaller variance than the original version (30).

Remark (Local expectation gradient as Rao-blackwellization)**.** What if $q(\boldsymbol{\theta})$ represents some *structured* approximation to the exact posterior? In this case q is specified by a *directed graphical model*

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = \prod_i q_{\boldsymbol{\phi}_i}(\theta_i|\text{pa}_i),$$

and pa_i denotes the parents of node θ_i. We also use $\boldsymbol{\theta}_{-i}$ to collect all the other entries $\theta_j, j \neq i$. Then, going through similar derivations as (32), we have

$$\nabla_{\boldsymbol{\phi}_i}\mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}; p) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{-i})}\left[\mathbb{E}_{q_{\boldsymbol{\phi}_i}(\theta_i|\text{mb}_i)}\left[\log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\right] \nabla_{\boldsymbol{\phi}_i} \log q_{\boldsymbol{\phi}_i}(\theta_i|\text{pa}_i)\right],$$

in which mb_i denotes the Markov blanket of random variable θ_i. Then the local expectation gradient method [Titsias and Lázaro-Gredilla, 2015] developed for discrete variables applies similar Rao-blackwellization trick as in above: given a sample $\boldsymbol{\theta}_{-i} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{-i})$, the MCVI gradient can be computed as

$$\nabla_{\boldsymbol{\phi}_i}\mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}; p) \approx \sum_{\theta_i} q_{\boldsymbol{\phi}_i}(\theta_i|\text{mb}_i)\left[\log \frac{p(\mathcal{D}, \theta_i, \boldsymbol{\theta}_{-i})}{q_{\boldsymbol{\phi}}(\theta_i, \boldsymbol{\theta}_{-i})}\right] \nabla_{\boldsymbol{\phi}_i} \log q_{\boldsymbol{\phi}_i}(\theta_i|\text{pa}_i). \tag{34}$$

Notice here that the log-derivative trick is non-applicable because we never simulate samples from $q_{\boldsymbol{\phi}_i}(\theta_i|\text{mb}_i)$.

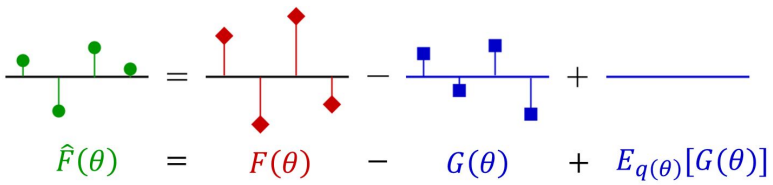

Figure 3: Visualising the idea of control variates. With $G(\boldsymbol{\theta})$ strongly and postively correlated with $F(\boldsymbol{\theta})$, the variance of $\hat{F}(\boldsymbol{\theta})$ is significantly reduced.

2.3.2 Control variate: general idea

Another important idea for variance reduction is control variate [Hammersley and Handscomb, 1966; Boyle, 1977]. Again we describe it for the general case, and the specific application of it to MCVI gradients is discussed in the next section. Assume we are interested in estimating $\mathbb{E}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta})]$ with Monte Carlo. Then one can easily show that for any function $G(\boldsymbol{\theta})$ that has finite mean $\mathbb{E}_{q(\boldsymbol{\theta})}[G(\boldsymbol{\theta})]$, we have

$$\mathbb{E}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta})] = \mathbb{E}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta}) - G(\boldsymbol{\theta}) + \mathbb{E}_{q(\boldsymbol{\theta})}[G(\boldsymbol{\theta})]]. \tag{35}$$

Denote $\hat{F}(\boldsymbol{\theta}) = F(\boldsymbol{\theta}) - G(\boldsymbol{\theta}) + \mathbb{E}_{q(\boldsymbol{\theta})}[G(\boldsymbol{\theta})]$ and assume $\mathbb{V}_{q(\boldsymbol{\theta})}[G(\boldsymbol{\theta})] < +\infty$. Then the variance of $\hat{F}$ under q is

$$\mathbb{V}_{q(\boldsymbol{\theta})}[\hat{F}(\boldsymbol{\theta})] = \mathbb{V}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta})] + \mathbb{V}_{q(\boldsymbol{\theta})}[G(\boldsymbol{\theta})] - 2\mathrm{Cov}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta}), G(\boldsymbol{\theta})]. \tag{36}$$

This means, a clever choice of the G function would make

$$\mathbb{V}_{q(\boldsymbol{\theta})}[G(\boldsymbol{\theta})] - 2\mathrm{Cov}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta}), G(\boldsymbol{\theta})] < 0,$$

and therefore it leads to a lower-variance estimator of $\mathbb{E}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta})] \approx \hat{F}(\boldsymbol{\theta}), \boldsymbol{\theta} \sim q(\boldsymbol{\theta})$ (see Figure 3). Such choice of the G function is called a control variate of F.

2.3.3 Some notable control variate methods for MCVI gradients

Now let us consider some notable examples of control variate that researchers has developed for MCVI. In this case the target function we consider is

$$F(\boldsymbol{\theta}) = \log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}).$$

In some papers $f(\boldsymbol{\theta}) = \log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}$ is also referred as the learning signal for the variational parameters $\boldsymbol{\phi}$ [Mnih and Gregor, 2014]. Many of the techniques described in below also applies to variance reduction for policy gradient in reinforcement learning.

- **Optimal scaling for score functions**:
  If we further define $G(\boldsymbol{\theta}) = \lambda g(\boldsymbol{\theta})$, then by (36), the reduction of variance is

$$\mathbb{V}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta})] - \mathbb{V}_{q(\boldsymbol{\theta})}[\hat{F}(\boldsymbol{\theta})] = 2\lambda \mathrm{Cov}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta}), g(\boldsymbol{\theta})] - \lambda^2 \mathbb{V}_{q(\boldsymbol{\theta})}[g(\boldsymbol{\theta})].$$

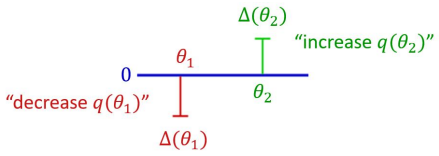

Figure 4: The baseline approach for control variates, with $\Delta(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) - b$. Here $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \sim q(\boldsymbol{\theta})$ are the MC samples from the approximate posterior $q(\boldsymbol{\theta})$.

Therefore, the optimal scaling λ can be derived by maximising the variance reduction, which gives

$$\lambda^* = \frac{\mathrm{Cov}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta}), g(\boldsymbol{\theta})]}{\mathbb{V}_{q(\boldsymbol{\theta})}[g(\boldsymbol{\theta})]}.$$

Ranganath et al. [2014] considers using the score function as a control variate, i.e. $g(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$, where from (27) we have $\mathbb{E}_{q(\boldsymbol{\theta})}[G(\boldsymbol{\theta})] = \mathbf{0}$.[10] This implies

$$\hat{F}_{\text{opt}}(\boldsymbol{\theta}) = [f(\boldsymbol{\theta}) - \lambda^*] \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}).$$

However in practice it is impossible to compute the exact optimal scaling λ^*, and instead we typically form an (MC) estimate $\hat{\lambda} \approx \lambda^*$, which also introduces extra variances. Still empirically, Ranganath et al. [2014] shows that this control variate works well in some practical cases.

- **Neural variational inference and learning (NVIL)**:
  Mnih and Gregor [2014] also considers $G(\boldsymbol{\theta}) = \lambda \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$. However, instead of estimating the optimal scaling, the authors notice that λ can be any function that is independent to $\boldsymbol{\theta}$. Therefore, they parameterise $\lambda = C_{\boldsymbol{\psi}}(\mathcal{D}) - c$ where $C_{\boldsymbol{\psi}}(\mathcal{D})$ is a neural network with parameters $\boldsymbol{\psi}$,[11] and train the neural network by minimising the ℓ_2 error

$$\min_{\boldsymbol{\psi}} \quad \mathbb{E}_{q_{\boldsymbol{\phi}}} \left[(f(\boldsymbol{\theta}) - C_{\boldsymbol{\psi}}(\mathcal{D}) - c)^2 \right].$$

  Therefore $C_{\boldsymbol{\psi}}(\mathcal{D}) + c$ is also called "data dependent baselines". Although the optimal solution in this case does not correspond to the optimal scaling, Mnih and Gregor [2014] argued that this alternative objective is much easier to minimise, and empirically, the resulting control variate performed quite well in their experiments. Another way to interpret this idea is visualised in Figure 4 with the baseline $b = C_{\boldsymbol{\psi}}(\mathcal{D}) + c$. Essentially, the gradient estimator would encourage allocating more probability mass to the regions where the reward $f(\boldsymbol{\theta})$ is improved over a baseline b, i.e. $\Delta(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) - b > 0$. Similarly it would avoid decreasing the reward to be below the baseline, by allocating less probability mass to those regions.

[10] Interestingly as we have mentioned, Roeder et al. [2017] has shown that in path gradient settings including the score function term can lead to high variance. However the authors didn't consider optimal scaling for the control variate.

[11] In the applications that Mnih and Gregor [2014] considered, $\boldsymbol{\theta}$ corresponds to the latent variables and $\mathcal{D} = \boldsymbol{x}$.

To further reduce variance in the scaling, NVIL also normalises the difference term with an estimate of its standard deviation (unless when $\boldsymbol{\phi}$ is close to a local optimum), making the final gradient as

$$\hat{F}_{\text{NVIL}}(\boldsymbol{\theta}) = \frac{f(\boldsymbol{\theta}) - C_{\boldsymbol{\psi}}(\mathcal{D}) - c}{\hat{\sigma}[f(\boldsymbol{\theta}) - C_{\boldsymbol{\psi}}(\mathcal{D})]} \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}).$$

I would doubt whether this normalisation step is necessary, if scale-invariant gradient descent methods like Adagrad [Duchi et al., 2011], RMSprop [Tieleman and Hinton, 2012] and Adam [Kingma and Ba, 2015] is in use. In their experiments Mnih and Gregor [2014] applied stochastic gradient descent, and in that case the normalisation technique can be of great help. In fact they estimated the standard-deviation using exponential moving average, which makes the resulting algorithm closely related to RMSprop [Tieleman and Hinton, 2012].

- **Taylor expansion for the learning signal**:
  The control variate for MCVI is not limited to the score function up to constant scaling. Indeed, if we define $G(\boldsymbol{\theta}) = [h(\boldsymbol{\theta}) + b]\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$, then (35) expands to

  $$\mathbb{E}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta})] = \mathbb{E}_{q(\boldsymbol{\theta})}[(f(\boldsymbol{\theta}) - b - h(\boldsymbol{\theta}))\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})] + \mathbb{E}_{q(\boldsymbol{\theta})}[h(\boldsymbol{\theta})\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})].$$

  Now we assume $f(\boldsymbol{\theta})$ is differentiable w.r.t. $\boldsymbol{\theta}$ which is often true for the density ratio in the VI case. Then we can use Taylor expansion at some location $\boldsymbol{\theta}_0$ that is independent to $\boldsymbol{\theta}$ to define the control variate, for example $b + h(\boldsymbol{\theta}) = f(\boldsymbol{\theta}_0) + \nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$. By rearranging terms and applying the identity (27), we have

  $$\mathbb{E}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta})] = \mathbb{E}_{q(\boldsymbol{\theta})}[\epsilon_f(\boldsymbol{\theta}, \boldsymbol{\theta}_0)\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})] + \mathbb{E}_{q(\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0)\boldsymbol{\theta}\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})],$$

  with $\epsilon_f(\boldsymbol{\theta}, \boldsymbol{\theta}_0) = f(\boldsymbol{\theta}) - f(\boldsymbol{\theta}_0) - \nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$ as the Taylor expansion error. One can further use the log derivative trick to show that the second term reduces to $\mathbb{E}_{q(\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0)\boldsymbol{\theta}\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})] = \nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0)\nabla_{\boldsymbol{\phi}}\mathbb{E}_{q(\boldsymbol{\theta})}[\boldsymbol{\theta}]$. Higher order Taylor expansion has also been explored, see Paisley et al. [2012]; Gu et al. [2016].

2.4 Further reading

Ranganath et al. [2014] described the black-box VI (BBVI) algorithm that is referred as MCVI in this note. Paisley et al. [2012]; Wingate and Weber [2013] also described very similar approaches, however Ranganath et al. [2014] has further detailed discussions on variance reduction techniques.

Should definitely read [Kingma and Welling, 2014] for the reparameterisation trick. Around the same time the trick was also described in Salimans and Knowles [2013]; Rezende et al. [2014], which unfortunately get less citations. I will also suggest a come-back reading on these papers when we discuss applications of approximate inference to generative models.

Opper and Archambeau [2009] describes another gradient estimator for the variational parameters of Gaussian approximations. The idea is less well-known but still very interesting.

We can construct "reparameterisations" for discrete variables as well if we can compute the inverse CDF. For general invertible transforms, Ruiz et al. [2016] constructed a generalised reparameterisation gradient that contains both path gradient terms and REINFORCE like terms. Similar ideas have also been explored in e.g. Tucker et al. [2017].

In general, gradient estimation is a hot topic of research in approximate inference and reinforcement learning. For the progress up to late 2019, see Mohamed et al. [2019] for a review.

In practice many of these MCVI methods are implemented using automatic differentiation, therefore it might be useful to understand how automatic differentiation works. For example see Baydin et al. [2015] for a survey.

I briefly discussed control variate methods applied to MCVI (and policy gradients). If you are interested in variance reduction methods for stochastic/distributed optimisation, then I would suggest reading papers for the following algorithms to start with: SAG [Le Roux et al., 2012; Schmidt et al., 2013], SVRG [Johnson and Zhang, 2013], and SAGA [Defazio et al., 2014].

3 Amortised inference

So far we have demonstrated how to apply VI to approximate the posterior distribution. However it can still be very slow for running VI on a probabilistic model with lots of unobserved variables. For example, a well-known probabilistic model for text data – latent Dirichlet allocation (LDA) [Blei et al., 2003], could involve millions of latent variables when applied to a large corpus. Thus it brings in prohibitive computational burden since each latent variable must have its approximate posterior iteratively refined. Furthermore, for models whose hyper-parameters are updated constantly, the inference procedure is also repeatedly required as a subroutine. These issues had restricted the extensions of VI to many interesting cases, until the introduction of *amortised inference* that is detailed in below.

3.1 Inference dependencies on observations

Let us start from the mean-field approximation example we had in section 1.3 but with a slightly different set-up. In this case we are interested in learning a latent variable model

$$\boldsymbol{z}_n \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}), \quad y_n|\boldsymbol{x}_n \sim \mathcal{N}(\mathbf{y}; \boldsymbol{z}_n^T \boldsymbol{x}_n, \sigma^2).$$

which is closely related to factor analysis [Harman, 1976]. In this case the model parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Lambda}, \sigma\}$ are to be learned by approximate maximum likelihood or variational EM, where the variational lower-bound is used as the surrogate loss. In this case we assume for each latent variable $\boldsymbol{z}_n \in \mathbb{R}^D$ we compute a mean-field approximate posterior $q_n(\boldsymbol{z}_n) = \prod_{i=1}^D q_n(z_{ni})$, in which we define each factorisation as $q_n(z_{ni}) = \mathcal{N}(z_{ni}; m_{ni}, \lambda_{ni}^{-1})$.

One strategy to learn these q distributions is to do gradient descent w.r.t. all the variational parameters $\{m_{ni}, \lambda_{ni}\}$ until reaching a local optimum. However this approach is inefficient for large-scale data. First, variational parameters must then be maintained for every pair of observations $(\boldsymbol{x}_n, y_n)$, meaning a space complexity of $\mathcal{O}(ND)$ if N is the total number of observations. Furthermore, when the model parameters are updated, the previously optimal variational parameters are no longer optimal thus requiring loops of gradient descent again. Depending on the changes of the model parameters, the previous optimal solution for the variational parameters might not always be a good initialisation for the current round's optimisation.

Fortunately, observe that in § 1.3 we have the optimal solutions satisfying

$$\lambda_{ni} = \boldsymbol{\Lambda}_{ii} + \frac{1}{\sigma^2} x_{ni}^2, \quad \boldsymbol{m}_n = (\boldsymbol{\Lambda} + \frac{1}{\sigma^2} \boldsymbol{x}_n \boldsymbol{x}_n^T)^{-1} (\boldsymbol{\Lambda}\boldsymbol{\mu} + \frac{1}{\sigma^2} y_n \boldsymbol{x}_n),$$

meaning that these optimal variational parameters are functions of the observations $\boldsymbol{x}_n$ and y_n. Hence if we explicitly define the variational parameters as a function of the observations, e.g. by parameterising $\lambda_{ni} = a_i x_{ni}^2 + b_i$, and optimise the parameters of these mappings (in our example a_i and b_i), then we can drastically reduce the memory cost to $\mathcal{O}(D)$ which is scalable to big data. Furthermore

the previous round's solution of a_i, b_i is more likely to be a good initialiser for the current round's optimisation, as the "local structure" of q (in our example the quadratic term x_{ni}^2) is already encoded in the mapping. In general we will explicitly define the approximate posterior as $q(\boldsymbol{z}_n|\boldsymbol{x}_n, y_n)$ to emphasise the dependency on the observations, and only parameterise the "global structure" that is shared across all q distributions.

The above method is termed as *amortised inference* for VI [Salimans and Knowles, 2013; Kingma and Welling, 2014; Rezende et al., 2014], which is:

- memory efficient, as we only learn the shared information across the approximate posterior distributions for different $(\boldsymbol{x}_n, y_n, \boldsymbol{z}_n)$ tuples;
- faster for training, as the previous round's solution is more likely to initialise the current step well;
- a good initialisation of q for unseen data, which will then be refined.[12]

Obviously it also has disadvantages: as the "global structure" is typically unknown to the user, a careless design of such amortisation would return distributions with restrictive representation power. One might suggest using neural networks in a way that leads to very flexible q distributions, however the computation of the (MC approximation of the) variational lower-bound requires $\log q(\boldsymbol{\theta})$ to be tractable, which is again a very restrictive constraint. Indeed currently neural networks are mostly used to parameterise simple distributions (for example the mean and variance of a Gaussian q distribution), or distributions that are carefully designed using invertible transform [Rezende and Mohamed, 2015; Kingma et al., 2016]. It is also possible to use implicit distributions as the approximate posterior, which is further discussed in § 9.

3.2 A popular approach: variational auto-encoder

Generating realistic images, sound and text has always been one of the main theme in AI research. One approach towards this goal that is very popular now is building a *deep generative model* to transform some random noise to generate desired objects. Similar to the latent variable model we discussed in § 3.1, the model starts from a latent variable $\boldsymbol{z}$ sampled from a prior distribution $p_0(\boldsymbol{z})$, and then samples the observations $\boldsymbol{x}$ from a conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ parameterised by $\boldsymbol{\theta}$.[13] Unlike the linear case discussed before, here deep neural networks are applied to form the the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, usually by determining the parameters of the distribution by neural networks taking $\boldsymbol{z}$ as their input.[14] As a concrete example, let us consider the following model:

$$\boldsymbol{z}_n \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I}), \quad \boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}_n), \mathrm{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\boldsymbol{z}_n))),$$

[12] See later discussions in § 3.4.

[13] It is possible to have a hierarchy of latent variable models, but here we will stick to the simplest case.

[14] As a comparison, in many GAN approaches, $p(\boldsymbol{x}|\boldsymbol{z})$ is implicitly defined by neural network transforms.

where $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\sigma}_{\boldsymbol{\theta}}$ are defined by deep neural network transforms of $\boldsymbol{z}$. With the observed dataset $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ in hand, we are interested in finding the most likely configuration of the neural network parameters $\boldsymbol{\theta}$ by maximum likelihood:

$$\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \tag{37}$$

which involves integrating out all the latent variables $\boldsymbol{z}_n$ and which is thus analytically intractable. Traditionally, the expectation maximisation (EM) algorithm [Dempster et al., 1977] is used here to train the parameters $\boldsymbol{\theta}$. More specifically, in variational EM [Beal and Ghahramani, 2003], variational inference is deployed in the E-step, which corresponds to maximising the variational distributions $\{q_n(\boldsymbol{z}_n)\}$ using the following objective:

$$\max_{\boldsymbol{\theta}, q_n} \sum_{n=1}^{N} \mathcal{L}_{\text{VI}}(q_n, \boldsymbol{\theta}; \boldsymbol{x}_n), \quad \mathcal{L}_{\text{VI}}(q_n, \boldsymbol{\theta}; \boldsymbol{x}_n) = \mathbb{E}_{q_n(\boldsymbol{z}_n)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{z}_n)}{q_n(\boldsymbol{z}_n)}\right]. \tag{38}$$

A common choice of the q_n distributions is factorised Gaussian distribution $q_n(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{z}_n; \boldsymbol{\mu}_n, \text{diag}(\boldsymbol{\sigma}_n^2))$, which, as argued in § 3.1, costs $\mathcal{O}(ND)$ memory thus inefficient.

The idea of the popular *variational auto-encoder* approach [Kingma and Welling, 2014; Rezende et al., 2014] is two fold. First VAE applies the amortised inference idea to VI: instead of having q_n attached to each latent variable $\boldsymbol{z}_n$, the authors constructed a data-dependent posterior approximation $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ with variational parameters $\boldsymbol{\phi}$. Therefore the corresponding variational lower-bound is:

$$\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \sum_{n=1}^{N} \mathcal{L}_{\text{VI}}(\boldsymbol{\phi}, \boldsymbol{\theta}; \boldsymbol{x}_n), \quad \mathcal{L}_{\text{VI}}(\boldsymbol{\phi}, \boldsymbol{\theta}; \boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right]. \tag{39}$$

In factorised Gaussian case, the $q_{\boldsymbol{\phi}}$ distribution is

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \text{diag}(\boldsymbol{\sigma}_{\boldsymbol{\phi}}^2(\boldsymbol{x}))),$$

and again $\boldsymbol{\mu}_{\boldsymbol{\phi}}$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}$ can be mappings parameterised by deep neural networks with parameter $\boldsymbol{\phi}$. In this context the q distribution is also called the *recognition model* or the *inference network*.

The second idea is the deployment of MC-VI with the reparameterisation trick, which is detailed in § 2. One can easily notice that, drawing samples from the above $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ distribution is done by the following procedure

$$\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \Leftrightarrow \boldsymbol{\epsilon} \sim \pi(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}; \boldsymbol{0}, \boldsymbol{I}), \boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}, \tag{40}$$

and essentially this performs a change-of-variable operation. Following the LOTUS rule, the variational lower-bound can be rewritten as

$$\mathcal{L}_{\text{VI}}(\boldsymbol{\phi}, \boldsymbol{\theta}; \boldsymbol{x}) = \mathbb{E}_{\pi(\boldsymbol{\epsilon})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon})}{q_{\boldsymbol{\phi}}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}|\boldsymbol{x})}\right], \tag{41}$$

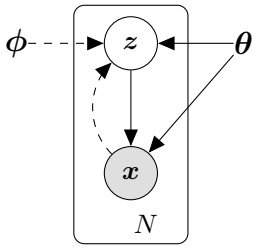

Figure 5: The graphical model of VAE, showing the generative model and the inference network. Dash arrows imply dependencies in the q distribution. Reproduced from Kingma and Welling [2014].

and it can be further approximated using simple Monte Carlo (MC):

$$\mathcal{L}_{\text{VI}}^{\text{MC}}(\boldsymbol{\phi}, \boldsymbol{\theta}; \boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}_k)}{q_{\boldsymbol{\phi}}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}_k | \boldsymbol{x})}, \quad \boldsymbol{\epsilon}_k \sim q(\boldsymbol{\epsilon}). \tag{42}$$

In practice the MC estimate is computed with very few samples, and in the case of drawing only one sample ($K = 1$), the resulting algorithm is very similar to training a standard auto-encoder with a (learnable) noise-injected encoding operation, thus the name *variational auto-encoder* [Kingma and Welling, 2014; Rezende et al., 2014]. Its graphical model is also visualised In Figure 5.

3.3 More examples beyond applications to VI

Amortised inference is sometimes misunderstood as being equivalent to the variational auto-encoder approach due to the huge popularity of the latter. In fact the general idea goes far beyond: amortisation can be applied to *any* inference technique as long as the optimal solution for it can be described by a mapping from the observed data.

Historically, amortised inference was first developed for non-Bayesian inference schemes. Hinton et al. [1995] developed the *wake-sleep* algorithm to train the *Helmholtz machine* [Dayan et al., 1995], where the sleep step trains the $q(\boldsymbol{z}|\boldsymbol{x})$ distribution using samples from the model, i.e. $\boldsymbol{z}, \boldsymbol{x} \sim p(\boldsymbol{z}, \boldsymbol{x})$. Morris [2001] further applied the sleep step update to learn an approximation to the conditional distributions of a directed graphical model. In Gaussian process (GP) literature, amortised inference has been applied to GP latent variable models (GPLVM) to infer the latent variables with amortised MLE, under the name "back constraints" [Lawrence and Quiñonero-Candela, 2006].

Recent progress on amortising (approximate) Bayesian inference includes applications to MAP inference [Sonderby et al., 2017], importance sampling [Burda et al., 2016], sequential Monte Carlo [Le et al., 2017; Maddison et al., 2017a; Naesseth et al., 2017] and MCMC [Li et al., 2017]. In the rest of this section I will briefly discuss some of these approaches, and for MCMC readers are referred to § 9.3.4.

3.3.1 Wake-sleep algorithm

Deep learning community also refers the $q(\boldsymbol{z}|\boldsymbol{x})$ distribution as *recognition model/network* or *inference network*. The name "recognition model" is at least dated back to the development of Helmholtz machine [Dayan et al., 1995] which is trained using the wake-sleep algorithm [Hinton et al., 1995]:

- Wake-step: train the generative model $p_{\boldsymbol{\phi}}(\boldsymbol{x})$ by VI-approximated MLE (with MC approximation):

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}[\mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}; p_{\boldsymbol{\theta}})].$$

  This is called "wake step" since the latent variables are inferred using observations from data (like what you will see when you're awake). Usually we only perform 1-step gradient update in this step.

- Sleep-step: train the recognition network $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ by minimising inclusive KL (with MC approximation):

$$\min_{\boldsymbol{\phi}} \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}[\text{KL}[p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})||q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})]].$$

  This is called "sleep step" because q is trained using "dreamed" samples from $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. Again usually we only perform 1-step gradient update in this step.

Interestingly, the wake-sleep method uses exactly the MC-VI objective (42) to learn the generative model, but optimises the "recognition weights" $\boldsymbol{\phi}$ using another KL. We note that the inference network never observes real-world data in the sleep step, rather, it is trained using samples $\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$. The rational is, if $\text{supp}(p_{\mathcal{D}}) \subset \text{supp}(p_{\boldsymbol{\theta}})$ for any model parameter setting $\boldsymbol{\theta}$ and the inference network $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is flexible enough, then in the sleep-step, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ will approach to the conditional distribution[15] $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ for any $\boldsymbol{x} \in \text{supp}(p_{\boldsymbol{\theta}})$, and thus for any $\boldsymbol{x} \in \text{supp}(p_{\mathcal{D}})$. This will also makes the variational lower-bound objective in the wake-step approach to the exact marginal likelihood. However, since the KL divergence is asymmetric, it is difficult to translate a bound on $\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}[\text{KL}[p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})||q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})]]$ to the tightness result of the wake-step objective, and it is also hard to describe the approximation quality of q when evaluated on $\boldsymbol{x} \sim \mathcal{D}$ (unless $p_{\boldsymbol{\theta}} = \hat{p}_{\mathcal{D}}$). Therefore it has no theoretical guarantee to return a good approximation to $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ for $\boldsymbol{x} \sim \mathcal{D}$. Nevertheless the wake-sleep algorithm performs well in practice, and it has been revisited by researchers recently. Recent work on improving the wake-sleep algorithm [Bornschein and Bengio, 2015] proposed an adjusted sleep-step in order to incorporate the observations $\boldsymbol{x} \sim \mathcal{D}$. In short, this proposal replaces the objective function for $\boldsymbol{\phi}$ to

$$\min_{\boldsymbol{\phi}} \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}[\text{KL}[p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})||q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})]],$$

[15] Notice that here I do not use the terminology "posterior".

and the gradient of the KL inside the expectation of $p_{\mathcal{D}}(\boldsymbol{x})$ this objective is approximated by self-normalised importance sampling:

$$\begin{aligned}\nabla_{\boldsymbol{\phi}}\mathrm{KL}[p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})||q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})] &= -\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}[\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})] \\ &\approx -\mathbb{E}_{\boldsymbol{z}^1,\dots,\boldsymbol{z}^K\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\sum_{k=1}^K \hat{w}_k \nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\boldsymbol{z}^k|\boldsymbol{x})\right], \\ w_k = \log\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}^k)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}^k|\boldsymbol{x})}, \quad \hat{w}_k &= \frac{w_k}{\sum_{k'=1}^K w_{k'}}.\end{aligned} \tag{43}$$

This makes the improved algorithm closely related to the importance weighted auto-encoder (IWAE) algorithm [Burda et al., 2016] that will be detailed in the following.

3.3.2 Importance weighted auto-encoder

Importance sampling (IS) is a simple but popular inference technique for estimating expectations. Notice that for any distribution $q(\boldsymbol{z}|\boldsymbol{x})$ satisfying $supp(p) \subset supp(q)$:

$$\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})] = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[F(\boldsymbol{z})\frac{p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z}|\boldsymbol{x})}\right] \approx \frac{1}{K}\sum_{k=1}^K F(\boldsymbol{z}^k)\frac{p(\boldsymbol{z}^k|\boldsymbol{x})}{q(\boldsymbol{z}^k|\boldsymbol{x})}, \quad \boldsymbol{z}^k \sim q.$$

It is easy to show that the IS estimate is consistent, with almost surely convergence if $F(\boldsymbol{z}) > 0$. Burda et al. [2016] constructed a lower-bound of the marginal likelihood using the IS estimate and Jensen's inequality:

$$\begin{aligned}\log p(\boldsymbol{x}) &= \log \mathbb{E}_{\boldsymbol{z}^1,\dots,\boldsymbol{z}^K\sim q}\left[\frac{1}{K}\sum_{k=1}^K \frac{p(\boldsymbol{x},\boldsymbol{z}^k)}{q(\boldsymbol{z}^k|\boldsymbol{x})}\right] \\ &\geq \mathbb{E}_{\boldsymbol{z}^1,\dots,\boldsymbol{z}^K\sim q}\log\left[\frac{1}{K}\sum_{k=1}^K \frac{p(\boldsymbol{x},\boldsymbol{z}^k)}{q(\boldsymbol{z}^k|\boldsymbol{x})}\right] := \mathcal{L}_K.\end{aligned} \tag{44}$$

Using Jensen's inequality, the authors also proved that the lower-bound $\mathcal{L}_K$ is non-decreasing in K, with $\mathcal{L}_1 = \mathcal{L}_{\mathrm{VI}}$. The lower-bound $\mathcal{L}_K$ is optimised jointly w.r.t. both p and q. One can obtain the gradient by automatic differentiation, but if the reparameterisation trick is applicable, simple derivation shows that

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_K = \mathbb{E}_{\boldsymbol{\epsilon}^1,\dots,\boldsymbol{\epsilon}^K\sim\pi(\boldsymbol{\epsilon})}[\hat{w}_k\nabla_{\boldsymbol{\phi}}\log w_k], \; w_k = \log\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{\epsilon}^k))}{q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{\epsilon}^k)|\boldsymbol{x})}, \; \hat{w}_k = \frac{w_k}{\sum_{k'=1}^K w_{k'}}. \tag{45}$$

Therefore the approach is named importance weighted auto-encoder (IWAE), and it is straightforward to see that the above gradient reduces to the VI case when $K = 1$. Also when compared to the adjusted sleep-step update (43), we see that the learning of $\boldsymbol{\phi}$ also takes the gradient information of $\log p(\boldsymbol{x},\boldsymbol{z})$ into account.[16]

[16] For discrete variables, $\nabla_{\boldsymbol{\phi}}\mathcal{L}_K$ adds an extra term to the adjusted sleep-step gradient (43), and that term is $\mathbb{E}_{\boldsymbol{z}^1,\dots,\boldsymbol{z}^K\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[(\log\frac{1}{K}\sum_{k=1}^K w_k)(\sum_{k=1}^K \nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\boldsymbol{z}^k|\boldsymbol{x}))\right]$.

IWAE often returns better performance in terms of test-LL when compared to VAE, since the lower-bound is tighter, so that less bias is introduced to the optimisation of p. But as $K \to +\infty$ the lower-bound becomes exact regardless of the choice of the q distribution (as long as $supp(p) \subset supp(q)$). This means $\mathcal{L}_K$ with large K is less informative to learning q, and indeed Rainforth et al. [2018]; Nowozin [2018] provided some theoretical analyses, showing that tighter lower-bounds are not necessarily better for posterior approximation. But this is not a problem for wake-sleep, and indeed increasing the number of samples K helps reducing both the bias and variance of the gradient (43) [Le et al., 2018].

3.3.3 Amortising proposal distributions for sequential Monte Carlo

Sequential Monte Carlo (SMC) is a class of sampling algorithms that are appealing for simulating posterior samples for sequential models. For a good introduction I recommend reading Doucet et al. [2001]; Doucet and Johansen [2009], and here I will very briefly cover a simple example. In this case the probabilistic model is a *hidden Markov model* (HMM):

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t|\boldsymbol{z}_{<t})p(\boldsymbol{x}_t|\boldsymbol{z}_t), \quad p(\boldsymbol{z}_1|\boldsymbol{z}_{<1}) = p(\boldsymbol{z}_1). \tag{46}$$

The goal is to compute the expectation $\mathbb{E}_{p(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})}[F(\boldsymbol{z}_{1:T})]$, which can be done by importance sampling:

$$\mathbb{E}_{p(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})}[F(\boldsymbol{z}_{1:T})] = \mathbb{E}_{q(\boldsymbol{z}_{1:T})}\left[F(\boldsymbol{z}_{1:T})\frac{p(\boldsymbol{z}_{1:T}, \boldsymbol{x}_{1:T})}{q(\boldsymbol{z}_{1:T})p(\boldsymbol{x}_{1:T})}\right]. \tag{47}$$

Furthermore we know $p(\boldsymbol{x}_{1:T}) = \mathbb{E}_{q(\boldsymbol{z}_{1:T})}\left[\frac{p(\boldsymbol{z}_{1:T},\boldsymbol{x}_{1:T})}{q(\boldsymbol{z}_{1:T})}\right]$ when $supp(p) \subset supp(q)$. Therefore one can use Monte Carlo samples $\boldsymbol{z}^k_{1:T} \sim q(\boldsymbol{z}_{1:T}), k = 1, ..., K$ to estimate the expectations, and have the following approximation:

$$\mathbb{E}_{p(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})}[F(\boldsymbol{z}_{1:T})] \sum_{k=1}^{K} \hat{w}^k_T F(\boldsymbol{z}^k_{1:T}), \quad \hat{w}^k_T = \frac{w^k_T}{\sum_{j=1}^K w^j_T}, \; w^k_T = \frac{p(\boldsymbol{z}^k_{1:T}, \boldsymbol{x}_{1:T})}{q(\boldsymbol{z}^k_{1:T})}. \tag{48}$$

Now it remains to define the proposal distribution $q(\boldsymbol{z}_{1:T})$. Notice that if we define the proposal as follows:

$$q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{1:t}), \quad q(\boldsymbol{z}_1|\boldsymbol{z}_{<1}, \boldsymbol{x}_1) = p(\boldsymbol{z}_1|\boldsymbol{x}_1), \tag{49}$$

then the importance weight is computed as

$$w^k_T = \prod_{t=1}^{T} \frac{p(\boldsymbol{z}^k_t|\boldsymbol{z}^k_{<t})p(\boldsymbol{x}_t|\boldsymbol{z}^k_t)}{q(\boldsymbol{z}^k_t|\boldsymbol{z}^k_{<t}, \boldsymbol{x}_{1:t})} = w^k_{T-1}\frac{p(\boldsymbol{z}^k_T|\boldsymbol{z}^k_{<T})p(\boldsymbol{x}_T|\boldsymbol{z}^k_T)}{q(\boldsymbol{z}^k_T|\boldsymbol{z}^k_{<T}, \boldsymbol{x}_{1:T})}, \quad w^k_0 = 1, \tag{50}$$

and similarly

$$\hat{w}^k_T \propto \hat{w}^k_{T-1}\frac{p(\boldsymbol{z}^k_T|\boldsymbol{z}^k_{<T})p(\boldsymbol{x}_T|\boldsymbol{z}^k_T)}{q(\boldsymbol{z}^k_T|\boldsymbol{z}^k_{<T}, \boldsymbol{x}_{1:T})}, \quad \hat{w}^k_0 = \frac{1}{K}. \tag{51}$$

It is important to adapt the proposal $q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$ to data, and again in amortised inference set-up, the proposal is parameterised as $q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$, where the variational parameters $\boldsymbol{\phi}$ is optimised using some objective. Since naive SMC is just importance sampling applied to HMMs, the IWAE objective also applies here:

$$\mathcal{L}_K(q) = \mathbb{E}_{\boldsymbol{z}^1_{1:T},...,\boldsymbol{z}^K_{1:T}\sim q}\left[\log \frac{1}{K}\sum_{k=1}^{K} w_T^k\right]. \tag{52}$$

However, since the unnormalised importance weight w_T^k is computed recursively, as T increases the lower-bound (52) will be dominated by $\max_k \log w_T^k$ which is undesirable. To address this, a re-sampling scheme can be introduced at some time τ:

$$i_k \sim \text{Categorical}(\hat{w}_\tau^1, ..., \hat{w}_\tau^K), \quad (\boldsymbol{z}_{1:\tau}^k, w_\tau^k, \hat{w}_\tau^k) \leftarrow (\boldsymbol{z}_{1:\tau}^{i_k}, 1, \frac{1}{K}), k = 1, ..., K. \tag{53}$$

Therefore, if we denote the lower-bound in (52) as $\mathcal{L}_k(q, 1:T)$ and select re-sampling time $\boldsymbol{\tau} = \{\tau_0, ..., \tau_M\}, 1 = \tau_0 < \tau_1 < ... < \tau_M \leq T$, then the new variational lower-bound induced by this resampling importance estimate is

$$\mathcal{L}_{K,\boldsymbol{\tau}}(q) = \sum_{m=1}^{M} \mathcal{L}_k(q, \tau_{m-1}:\tau_m). \tag{54}$$

This idea has been proposed concurrently by Le et al. [2017]; Maddison et al. [2017a]; Naesseth et al. [2017], with different focuses on applications in generative modelling.

3.4 Adding refinements to amortised inference

As discussed in § 3.1 the design of the distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is crucial for the performance of amortised inference. While interested readers can check out later sections in part II for some popular choices, in general there still exists non-negligible approximation error due to the flexibility of the distributions in $\mathcal{Q}$ and/or sub-optimality issues of stochastic optimisation.

For instance, consider $\mathcal{Q}$ as the set of mean-field Gaussian distributions on $\boldsymbol{z}$, and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x})))$. For each input $\boldsymbol{x}$, we find the optimal approximation of $p(\boldsymbol{z}|\boldsymbol{x})$ in $\mathcal{Q}$ by running variational inference, which returns $q^*(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_*(\boldsymbol{x}), \text{diag}(\boldsymbol{\sigma}^2_*(\boldsymbol{x})))$. However, if the neural network used to compute $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x}))$ is not flexible enough, then the optimal amortised inference solution also exhibits some approximation error to the optimal mean-field VI solution $q^*(\boldsymbol{z}|\boldsymbol{x}) \in \mathcal{Q}$. As a results, in general $\log p(\boldsymbol{x}) \geq \mathcal{L}_{\text{VI}}(q^*) \geq \mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}^*})$ when $\mathcal{Q}_\Phi = \{q_{\boldsymbol{\phi}}|\boldsymbol{\phi} \in \Phi\} \subset \mathcal{Q}$. Therefore the optimal amortised variational lower-bound $\mathcal{L}_{\text{VI}}$ has higher bias compared to the optimal variational lower-bound (as depicted in Figure 6), which can be harmful for maximum likelihood training. See Cremer et al. [2018] for a quantitative analysis of this amortisation gap.

Some recent techniques aim to reduce the amortisation gap, by refining the amortised posterior distribution with extra optimisation steps:

- Given an input $\boldsymbol{x}$, initialise the approximate posterior with the amortised distribution $q_0(\boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$;
- Refine the approximate posterior by e.g. running T-step optimisation starting from $q_0(\boldsymbol{z}|\boldsymbol{x})$ using some objective function. This returns an improved approximate posterior $q_T(\boldsymbol{z}|\boldsymbol{x})$;
- Optimise p using an objective that is dependent on q_T, e.g. $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{q_T}[\log p(\boldsymbol{x}, \boldsymbol{z})]$, and optimise $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ according to some criteria.

Table 1 presents the detailed choices of algorithms in recent papers, and they all used $\mathbb{E}_{\mathcal{D}}\mathbb{E}_{q_T}[\log p(\boldsymbol{x}, \boldsymbol{z})]$ as the objective to train p.[17] We see that in general the refinement algorithm does not necessarily need to be the same as the algorithm to train $q_{\boldsymbol{\phi}}$. In more detail, Marino et al. [2018]; Kim et al. [2018] fix the $\mathcal{Q}$ family as exponential family distributions in order to obtain tractable VI gradient steps as well as (a Monte Carlo estimate of) $\mathcal{L}_{\text{VI}}(q_T)$. For MCMC refinement $q_T(\boldsymbol{z}|\boldsymbol{x})$ is typically intractable, and the proposal by Hoffman [2017] does not take into account the convergence of MCMC. The algorithm by Li et al. [2017] encourages fast convergence of MCMC by finding better initialisations, which also pushes $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ towards the exact posterior. Unfortunately for many divergence measure $\mathrm{D}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||q_T(\boldsymbol{z}|\boldsymbol{x})]$ is intractable, thus it requires further approximations. Still these refinement approaches have been shown to be beneficial, e.g. better sharpness of the generated images, and faster mixing of a pseudo Gibbs-sampling for missing data imputation.

Remark (Bias of VAE training)**.** It is fairly straight-forward to show that VAE training is equivalent to minimising the following objective:

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \ \mathrm{KL}[\hat{p}_{\mathcal{D}}(\boldsymbol{x})q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})]. \tag{55}$$

This means for a fixed variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, the learning of p will bias the exact posterior $p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ towards $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. This shows that over-simplified variational approximation can be harmful as it also biases p towards simpler models.
While in the main text we argued that refinement is beneficial for preventing under-fitting, Shu et al. [2018] argued that it is also possible to use this bias to introduce regularisations for approximate maximum likelihood training. This "implicit regularisation" touches the philosophical question that whether the approximate posterior should also be treated as part of the *model* (as then q is not selected purely for approximating the exact posterior), and I prefer not to discuss it in this remark. But indeed by utilising the bias introduced by q, we can implant some nice property, e.g. smoothness, to the learned model, which might be beneficial for generalisation.

[17] It is straight-forward to extend the amortised SVGD approach [Wang and Liu, 2016; Feng et al., 2017] to refine amortised inference.

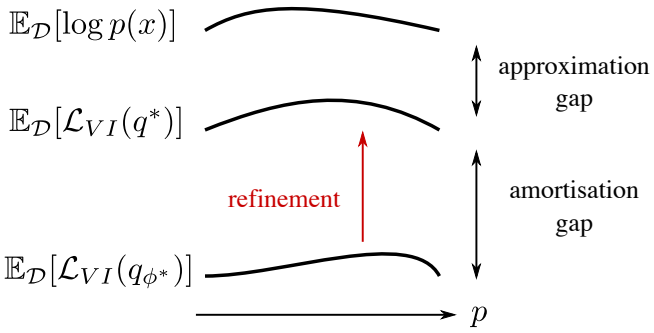

Figure 6: Decomposition of approximation error for amortised variational inference. In general the gap is not equal at every possible p, which introduces extra bias for approximate maximum likelihood training. The amortisation gap can be reduced by refinement techniques that is discussed in the main text.

Table 1: Recent approaches on refining amortised inference.

method	refinement alg.	$q_{\boldsymbol{\phi}}$ objective
Hoffman [2017]	T-step MCMC	$\mathbb{E}_{\mathcal{D}}\mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}})$
Li et al. [2017]	T-step MCMC	$\mathbb{E}_{\mathcal{D}}[\text{D}[q_{\boldsymbol{\phi}}(\boldsymbol{z}\vert\boldsymbol{x})\vert\vert q_T(\boldsymbol{z}\vert\boldsymbol{x})]]$ ($\text{D}[\cdot\vert\vert\cdot]$ is a valid divergence)
Marino et al. [2018] Kim et al. [2018]	T-gradient-step of VI	$\mathbb{E}_{\mathcal{D}}\mathcal{L}_{\text{VI}}(q_T)$

3.5 Further reading

Kingma and Welling [2014] is a nice paper, well-known for its introduction of amortisation to variational inference. So I would suggest Kingma and Welling [2014] as a must-read, but also note that concurrently the idea was also discussed in Rezende et al. [2014]; Salimans and Knowles [2013]. It is quite unfortunate that these two papers have got much less citations than Kingma and Welling [2014], let alone other papers that also discuss similar ideas [Stuhlmüller et al., 2013]. A tutorial on VAEs can also be found in Kingma et al. [2019].

The usage of neural networks in the amortised distribution dates back to the Helmholtz machine [Dayan et al., 1995], which is trained by the wake-sleep algorithm [Hinton et al., 1995] as discussed in previous sections. Kingma and Welling [2014] demonstrated that the VAE algorithm works better than wake-sleep in terms of training continuous latent variable models. However, recently researchers start to revisit wake-sleep, and they have shown some good results in terms of learning discrete latent variable models [Bornschein and Bengio, 2015; Le et al., 2018].

We have yet to discuss the choice of the amortised posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, which will be detailed in the second part of this topic list. But to summarise, the development has been focused on making the distribution family $\mathcal{Q}$ richer towards the goal that $p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \in \mathcal{Q}$ for $\boldsymbol{x} \sim \mathcal{D}$.

4 Alternative divergence objectives

One of the most important tasks in approximate Bayesian inference is to approximate the intractable posterior $p(\boldsymbol{\theta}|\mathcal{D})$. Specifically, we wish to construct an approximation $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$ such that the "approximation error" is minimised. This "approximation error" is measured by a *divergence* $\mathrm{D}[\cdot||\cdot] : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ such that $\mathrm{D}[p||q] \geq 0$ for all distributions $p, q \in \mathcal{P}$, and $\mathrm{D}[p||q] = 0$ iff. $p = q$. In other words, the best approximate posterior can be obtained by first selecting a suitable divergence, then minimising this divergence to obtain a (local) optimum:

$$q^*(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{Q}} \mathrm{D}[p(\boldsymbol{\theta}|\mathcal{D})||q(\boldsymbol{\theta})]. \tag{56}$$

In many cases directly minimising this divergence is intractable. However, there might exist an equivalent objective, such that minimising/maximising this objective is also equivalent to minimising the selected divergence. Consider variational inference (VI) as a simple example: VI optimises the variational lower-bound as the actual objective, which is equivalent to minimising the KL-divergence $\mathrm{KL}[q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})]$.

In practice, due to the zero-forcing property of $\mathrm{KL}[q||p]$ (see § 1.1), the resulting approximation to the exact posterior is often over-confident (although not always the case). This might be sub-optimal to tasks that require conservative uncertainty estimates, therefore alternative divergence objectives have been proposed for fitting the approximate posterior distribution. In this section we discuss some representative approaches in this paradigm, where we present the divergence definition first, then introduce the algorithm for minimising this divergence, perhaps via optimising an equivalent objective.

4.1 Alpha-divergences

4.1.1 Alpha-divergence definitions

There exist multiple different definitions of α-divergences, listed as follows.

- Rényi's α-divergence [Rényi, 1961] (defined on $\alpha \neq 1, \alpha > 0$):

$$\mathrm{D}_\alpha^R[p||q] = \frac{1}{\alpha - 1} \log \int p(\boldsymbol{\theta})^\alpha q(\boldsymbol{\theta})^{1-\alpha} d\boldsymbol{\theta}. \tag{57}$$

  By continuity in α we can show that $\lim_{\alpha \to 1} \mathrm{D}_\alpha^R[p||q] = \mathrm{KL}[p||q]$. Van Erven and Harremoës [2014] further extends Rényi's alpha divergence to negative α values, and we will further discuss this aspect in later sections.

- Tsallis's α-divergence [Tsallis, 1988] (defined on $\alpha \neq 1$):

$$\mathrm{D}_\alpha^T[p||q] = \frac{1}{\alpha - 1} \left(\int p(\boldsymbol{\theta})^\alpha q(\boldsymbol{\theta})^{1-\alpha} d\boldsymbol{\theta} - 1 \right).$$

  Again by continuity in α we can show that $\lim_{\alpha \to 1} \mathrm{D}_\alpha^T[p||q] = \mathrm{KL}[p||q]$.

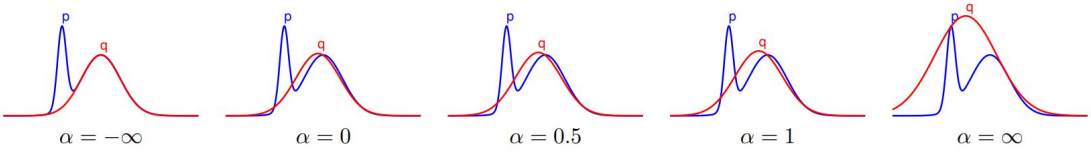

Figure 7: Visualising the mass-covering/mode-seeking property of α-divergence as an objective to fit a Gaussian q to a mixture of Gaussian distribution p. Note here the unnormalised density for q is shown. Figure reproduced from Minka [2005].

Table 2: Special cases in the Rényi divergence family. Table taken from Li and Turner [2016].

α	Definition	Notes
$\alpha \to 1$	$\int p(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}$	*Kullback-Leibler (KL) divergence*, used in VI (KL$[q\|\|p]$) and EP (KL$[p\|\|q]$)
$\alpha = 0.5$	$-2\log(1 - \text{Hel}^2[p\|\|q])$	function of the square *Hellinger distance*
$\alpha \to 0$	$-\log \int_{p(\boldsymbol{\theta})>0} q(\boldsymbol{\theta}) d\boldsymbol{\theta}$	zero when $\text{supp}(q) \subseteq \text{supp}(p)$ (not a divergence)
$\alpha = 2$	$-\log(1 - \chi^2[p\|\|q])$	proportional to the χ^2-divergence
$\alpha \to +\infty$	$\log \max_{\boldsymbol{\theta}\in\Theta} \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}$	*worst-case regret* in *minimum description length* principle [Grünwald, 2007]

- Amari's α-divergence [Amari, 1982, 1985] (defined on $\alpha \neq \pm 1$):

$$\text{D}_\alpha^A[p||q] = \frac{4}{1-\alpha^2}\left(1 - \int p(\boldsymbol{\theta})^{\frac{1+\alpha}{2}} q(\boldsymbol{\theta})^{\frac{1-\alpha}{2}} d\boldsymbol{\theta}\right).$$

  By continuity in α, we can show that $\lim_{\alpha\to 1} \text{D}_\alpha^A[p||q] = \text{KL}[p||q]$, and $\lim_{\alpha\to -1} \text{D}_\alpha^A[p||q] = \text{KL}[q||p]$.

We see that KL divergences (in both directions) can be viewed as special cases in the α-divergence family. Indeed this is a rich divergence family, with a few more special cases presented in Table 2. Recall that minimising KL$[q||p]$ w.r.t. q tends to have the zero-forcing behaviour, while for KL$[p||q]$ the resulting q tends to be mass-covering. Therefore it is reasonable to conjecture that choosing different α values for α-divergence minimisation would return q distributions that balances mode-seeking and mass-covering behaviours. This is further visualised in Figure 7 as reproduced from Minka [2005], where the α-divergence in that figure is from Zhu and Rohwer [1995] which is equivalent to Amari's α-divergence but with different α scales. This form of α-divergence will be further discussed in § 6.2.2.

The different formulations of α-divergence are equivalent in their discriminative powers, since they all contain the integral $\int p(\boldsymbol{\theta})^\alpha q(\boldsymbol{\theta})^{1-\alpha} d\boldsymbol{\theta}$. But notice that evaluating the α-divergence $\text{D}_\alpha^R[q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})]$ requires computing the exact posterior $p(\boldsymbol{\theta}|\mathcal{D})$ which is intractable (and similarly for D_α^T and D_α^A). In the following we describe the alternative objectives for minimising α-divergences.

Remark (History of α-divergences)**.** Readers might have noticed that all the different definitions of α-divergences are equivalent, since they all include the term that is called *Chernoff* α-coefficient

$$\int p(\boldsymbol{\theta})^{\alpha} q(\boldsymbol{\theta})^{1-\alpha} d\boldsymbol{\theta}, \quad \alpha \in (0,1).$$

Just after a year of the proposal of the KL-divergence, statistician Herman Chernoff introduced a test statistic for the likelihood-ratio test [Chernoff, 1952], and at the end of the paper, he linked the proposed technique to a divergence measure that is computed by the infimum of the above *Chernoff* α-coefficient w.r.t. $q(\boldsymbol{\theta})$.
In 1961, mathematician Alfréd Rényi argued that, by removing the additivity requirement, Shannon entropy can be further generalised to many interesting cases [Rényi, 1961]. He proposed one of such entropy definitions, and then characterised the induced mutual information and relative entropy measures using his version of α-divergence.[a] These two quantities are now referred to *Rényi entropy* and *Rényi divergence*, respectively. Perhaps surprisingly, Rényi's definition of α-divergence also contains the Chernoff α-coefficient, although these two developments are rather independent.
In the 70s-80s of the 20th century, differential geometry was introduced to statistics, e.g. see Efron [1975, 1978]; Amari [1985], which studies the geometric properties of the manifold obtained by mapping $\mathcal{P}$ to the parameter space Θ. In particular, researchers were interested in the geometrical properties of *exponential family* distributions (introduced later) and the corresponding divergences that reflect these features. In this context, mathematician Shun-ichi Amari introduced his version of α-divergence [Amari, 1982, 1985], by generalising the application of Chernoff α-coefficient to $\alpha \in \mathbb{R}$.

[a] The KL divergence characterises the corresponding mutual information and relative entropy measures for Shannon entropy.

4.1.2 Rényi divergence variational inference (RDVI)

Now consider approximating the exact posterior $p(\boldsymbol{\theta}|\mathcal{D})$ by minimizing Rényi's α-divergence $\mathrm{D}_{\alpha}^{R}[q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})]$ for some selected $\alpha > 0$. We can formulate an equivalent optimisation problem

$$\max_{q \in \mathbb{Q}} \quad \log p(\mathcal{D}) - \mathrm{D}_{\alpha}^{R}[q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})],$$

when $\alpha \neq 1$, the objective can be rewritten as

$$\begin{aligned} \mathcal{L}_{\alpha}(q;\mathcal{D}) :=& \log p(\mathcal{D}) - \mathrm{D}_{\alpha}^{R}[q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})] \\ =& \log p(\mathcal{D}) - \frac{1}{\alpha - 1} \log \mathbb{E}_q \left[\left(\frac{q(\boldsymbol{\theta}) p(\mathcal{D})}{p(\boldsymbol{\theta}, \mathcal{D})} \right)^{\alpha - 1} \right] \\ =& \frac{1}{1-\alpha} \log \mathbb{E}_q \left[\left(\frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta})} \right)^{1-\alpha} \right]. \end{aligned} \tag{58}$$

This equivalent objective is named *variational Rényi bound* (VR-bound) in Li and Turner [2016]. In particular, with Monte Carlo approximation techniques, we can formulate the following consistent but *biased* estimator of the VR-bound:

$$\frac{1}{1-\alpha}\log \mathbb{E}_q\left[\left(\frac{p(\boldsymbol{\theta},\mathcal{D})}{q(\boldsymbol{\theta})}\right)^{1-\alpha}\right] \approx \frac{1}{1-\alpha}\log\frac{1}{K}\sum_{k=1}^{K}\left(\frac{p(\boldsymbol{\theta}^k,\mathcal{D})}{q(\boldsymbol{\theta})}\right)^{1-\alpha}, \quad \boldsymbol{\theta}^k \sim q(\boldsymbol{\theta}). \tag{59}$$

Since in many cases we can assume the joint distribution $p(\boldsymbol{\theta},\mathcal{D})$ can be evaluated, this MC estimate of the VR-bound is tractable. Li and Turner [2016] further discussed the reparameterisation trick for gradient estimates of the VR-bound, while for discrete variables, the gradient estimator (with variance reduction) of the VR-bound is discussed in Webb and Teh [2016].

The above VR-bound objective can also be used as an equivalent objective for $\mathrm{D}_\alpha^R[p(\boldsymbol{\theta}|\mathcal{D})||q(\boldsymbol{\theta})]$, due to the skew-symmetry property [Van Erven and Harremoës, 2014].

Proposition 1. *(Skew symmetry) For* $\alpha \notin \{0,1\}$, $\mathrm{D}_\alpha^R[p||q] = \frac{\alpha}{1-\alpha}\mathrm{D}_{1-\alpha}^R[q||p]$. *This implies* $\mathrm{D}_\alpha^R[p||q] \leq 0$ *for* $\alpha < 0$. *For the limiting case* $\mathrm{D}_{-\infty}^R[p||q] = -\mathrm{D}_{+\infty}^R[q||p]$.

Therefore we have for $\alpha < 0$,

$$\mathcal{L}_\alpha(q;\mathcal{D}) = \log p(\mathcal{D}) + \frac{\alpha}{1-\alpha}\mathrm{D}_\alpha^R[p(\boldsymbol{\theta}|\mathcal{D})||q(\boldsymbol{\theta})],$$

which means *minimising* the VR-bound for $\alpha < 0$ is equivalent to minimising $\mathrm{D}_\alpha^R[p(\boldsymbol{\theta}|\mathcal{D})||q(\boldsymbol{\theta})]$. In practice, with MC estimation techniques, optimising $\mathcal{L}_\alpha(q;\mathcal{D})$ with $\alpha < 0$ is much more involved, see Li and Turner [2016] for further discussions.

Let us revisit the mean-field approximation example, where the goal is to approximate the target distribution – the exact posterior distribution of a Bayesian linear regression model – with a fully factorised Gaussian distribution. The analytic solution of the optimal q^* obtained by minimising the Rényi divergence is visualised in Figure 8. We see that by choosing $\alpha \in (-\infty, +\infty)$, the resulting approximation interpolates between "mass-covering" and "zero-forcing" behaviour, and in particular, for $\alpha = +\infty$ the resulting approximation (in cyan) still captures uncertainty in the posterior (although over-confidently so) rather than returning a point-estimate such as MAP.

Remark (Local minimisation of α-divergences)**.** The RDVI approach discussed in this section is a *global divergence minimisation* approach towards approaximate inference, in the sense that the approximate posterior is obtained by minimising (an equivalent objective to) a divergence between the exact and the approximate posterior. However, historically α-divergences are first employed in approximate inference methods based on *local divergence minimisations* [Minka, 2001b, 2004, 2005]. Readers are referred to discussions in message passing methods (§ 6) for further details; the high-level idea is to decompose the exact posterior into product of *factors*, then approximating the complicated factors (especially those contributing to the intractability of the exact posterior)

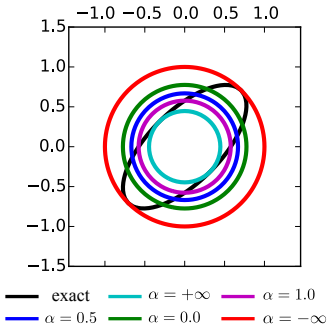

Figure 8: Mean-field approximation to the target distribution (in black) using different α-divergences.

> with simpler ones. Importantly, these simpler factors are obtained by minimising a set of KL/alpha-divergences defined for each of them. Therefore this approach performs "local approximations" in the sense that each divergence minimisation focuses on modifying a single factor in the approximate posterior. Each step of local divergence minimisation does not necessarily guarantee an improved approximate posterior when compared with the exact posterior. But in practice message passing approaches converge faster than gradient-based variational inference, and they are preferred choices of approximate inference methods on a handful of probabilistic models [Yedidia et al., 2005; Kuss and Rasmussen, 2005].

4.2 f-divergences

The previously discussed α-divergence allows the resulting approximate posterior to interpolate between zero-forcing and mass-covering behaviour (at least in theory when global optimum is obtained). As an even broader family of divergences, the f-divergences were introduce by Csiszár [1963], Morimoto [1963] and Ali and Silvey [1966], and are sometimes referred as Csiszár's f-divergences, Csiszár-Morimoto divergences or Ali-Silvey distances. This family of divergences is defined as follows.

Definition 2. *(f-divergence) Given a convex function $f : \mathbb{R}^+ \to \mathbb{R}$ satisfying $f(1) = 0$, the corresponding f-divergence on $\mathcal{P}$ is defined as a function $\mathrm{D}_f[\cdot||\cdot] : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ with the following form*

$$\mathrm{D}_f[p||q] = \int q(\boldsymbol{\theta}) f\left(\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right) d\boldsymbol{\theta}, \quad p, q \in \mathcal{P}. \tag{60}$$

By taking $f(x) = -\log x$ and $f(x) = x \log x$, in which both are convex in x, we recover the two KL divergence $\mathrm{KL}[q||p]$ and $\mathrm{KL}[p||q]$, respectively. The convexity of function f is required by Jensen's inequality in order to prove the conditions of a valid divergence.

Amari's α-divergence is a special instance of f-divergence, by taking $f(x) = \frac{4}{1-\alpha^2}(1 - x^{\frac{1+\alpha}{2}}) - \frac{2}{1-\alpha}(x - 1)$. Since different forms of α-divergences are more

or less equivalent, we can see that f-divergences are indeed more general. But more interestingly, if f is smooth, then one can show with Taylor expansion that the corresponding f-divergence can be represented by a series of chi-divergences, which are essentially special cases of α-divergences (up to scaling constant) with integer α values.

Direct minimisation of f-divergence $\mathrm{D}_f[p(\boldsymbol{\theta}|\mathcal{D})||q(\boldsymbol{\theta})]$ in approximate Bayesian inference context is more involved. Specifically, the trick of subtracting the f-divergence from the model evidence $\log p(\mathcal{D})$ does not lead to a tractable objective, i.e. we do not know how to compute $\log p(\mathcal{D}) - \mathrm{D}_f[p(\boldsymbol{\theta}|\mathcal{D})||q(\boldsymbol{\theta})]$ without evaluating the exact posterior $p(\boldsymbol{\theta}|\mathcal{D})$. On the other hand, the Jensen's inequality trick might still be applied in a way as

$$f(p(\mathcal{D})) = f\left(\mathbb{E}_q\left[\frac{p(\boldsymbol{\theta},\mathcal{D})}{q(\boldsymbol{\theta})}\right]\right) \leq \mathbb{E}_q\left[f\left(\frac{p(\boldsymbol{\theta},\mathcal{D})}{q(\boldsymbol{\theta})}\right)\right] = \mathbb{E}_q\left[f\left(\frac{p(\boldsymbol{\theta}|\mathcal{D})}{q(\boldsymbol{\theta})}p(\mathcal{D})\right)\right]. \tag{61}$$

However, the resulting objective on the RHS of the above equation does not directly correspond to an equivalent objective of an f-divergence. The KL-divergence $\mathrm{KL}[q||p]$ with $f(x) = -\log x$ is an exception, since $\log \frac{p(\boldsymbol{\theta}|\mathcal{D})}{q(\boldsymbol{\theta})}p(\mathcal{D}) = \log \frac{p(\boldsymbol{\theta}|\mathcal{D})}{q(\boldsymbol{\theta})} + \log p(\mathcal{D})$ so that $\log p(\mathcal{D})$ can be ignored as a constant in optimisation. Still previous tricks in variational inference with KL-divergence and α-divergences do not directly translate to the application of f-divergence minimisation.

4.3 Integral probability metrics

Is f-divergence the ultimate form for flexible distance/divergence/discrepancy between distributions? The answer is negative: there is another class of distance/divergence/discrepancy called *integral probability metrics* (IPMs) which are typically not in the form of f-divergence. They are widely used in statistics and machine learning such as two-sample tests [Gretton et al., 2012; Gorham and Mackey, 2015; Liu et al., 2016; Chwialkowski et al., 2016] and generative model learning [Arjovsky et al., 2017; Gulrajani et al., 2017]. Recent research has also employed IPMs in approximate inference context, e.g. see Liu and Wang [2016]; Ranganath et al. [2016a]; Ambrogioni et al. [2018]. Before diving into the details for a particular example, let us formally define an integral probability metric.

Definition 3. *(Integral probability metric (IPM)) Given a set of test functions $\mathcal{F}$, consider the following quantity:*

$$\mathrm{D}[p,q] = \sup_{f\in\mathcal{F}} |\mathbb{E}_p[f(\boldsymbol{\theta})] - \mathbb{E}_q[f(\boldsymbol{\theta})]|, \tag{62}$$

where $|\cdot|$ denotes a norm in the output space of f. If $\mathcal{F}$ is sufficiently large such that $\mathrm{D}[p,q] = 0$ iff. $p = q$, then $\mathrm{D}[p,q]$ is said to be an integral probability metric defined by the test functions in $\mathcal{F}$.

The interpretation of IPMs might be unclear to the readers at first sight. To provide an intuition, consider a strategy of comparing distributions by comparing

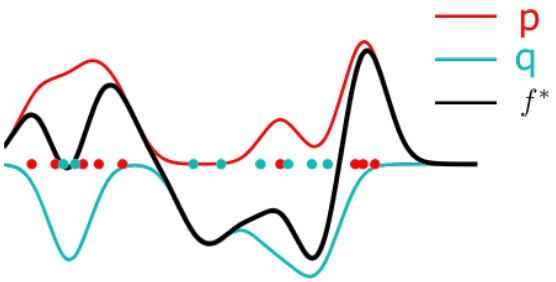

Figure 9: A visualisation of the optimal test function in an IPM. The dots are samples from the two distributions, and for visualisation ease the q distribution density is shown in upside down.

their *moments*, e.g. mean, variance, kurtosis, etc. Loosely speaking, if two distributions p and q have the same moments for all orders then p and q should be identical.[18] Therefore, to check whether p and q are identical or not, one can find the best moment, or in a broader sense the best test function f that can distinguish p from q the most, and if such optimal test function still fails to distinguish between p and q, then the two distributions p and q are identical.[19] This intuition is further visualised in Figure 9.[20] We see from the visualisation that the optimal test function f^* takes positive values in the region where $p(\boldsymbol{\theta}) > q(\boldsymbol{\theta})$ and vise versa. In other words, the optimal test function tells us more than whether $p = q$ or not; it also provides information on *how* p and q differ from each other. This is a useful property for IPMs for applications in adversarial learning: as f^* describes in detail the difference between p and q, we can optimise the q distribution in a guided way towards approximating the target distribution p.

Examples of IPMs include:

- (Kernelised) Maximum mean discrepancy (MMD): given a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ equipped with a generating kernel $\mathcal{K}(\cdot,\cdot)$,the test function set is defined as $\mathcal{F} = \{f \in \mathcal{H}, ||f||_{\mathcal{H}} \leq 1\}$. In this case the supremum problem in Definition 3 has a closed-form solution, resulting in the following discrepancy [Gretton et al., 2012]:

$$\text{MMD}[p,q] = \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\theta}'\sim p}[\mathcal{K}(\boldsymbol{\theta},\boldsymbol{\theta}')] + \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\theta}'\sim q}[\mathcal{K}(\boldsymbol{\theta},\boldsymbol{\theta}')] - 2\mathbb{E}_{\boldsymbol{\theta}\sim p,\boldsymbol{\theta}'\sim q}[\mathcal{K}(\boldsymbol{\theta},\boldsymbol{\theta}')]. \tag{63}$$

- Wasserstein distance: Wasserstein distance is a key concept developed in optimal transport which aims at finding the lowest cost approach to transform a distribution to another [Villani, 2008]. It is an IPM since its dual form is defined by taking the test functions from $\mathcal{F} = \{f : ||f||_L \leq 1\}$, the set of 1-Lipschitz functions:

$$W_2[p,q] = \sup_{||f||_L \leq 1} |\mathbb{E}_p[f(\boldsymbol{\theta})] - \mathbb{E}_q[f(\boldsymbol{\theta})]| \tag{64}$$

[18] This is subject to some conditions, e.g. the moment generating functions exist for p & q.

[19] Again under some assumptions of the form for p & q.

[20] Figure adapted from Dougal Sutherland's slides: http://www.gatsby.ucl.ac.uk/~dougals/slides/dali/#/38

So far the definition of IPMs requires samples from both p and q which might be intractable in the case of posterior inference, i.e. efficient ways of sampling from the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ might be unavailable. In such case we wish to define a set of test functions where for the target distribution p we have $\mathbb{E}_p[f(\boldsymbol{\theta})] = 0$ for all $f \in \mathcal{F}$, so that we can write the IPM as $\mathrm{D}[p, q] = \sup_{f\in\mathcal{F}} |\mathbb{E}_q[f(\boldsymbol{\theta})]|$ which only requires samples from the q distribution. Stein's method [Stein, 1972, 1981] provides a solution to this problem. Roughly speaking, given a distribution $p(\boldsymbol{\theta})$ whose density vanishes sufficiently fast at the boundary of the support, we have the following *Stein's identity*:

$$\mathbb{E}_{p(\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) g(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta})] = \mathbf{0}. \tag{65}$$

This identity can be shown using integration-by-parts for functions in $\mathcal{G}_p = \{g : p(\boldsymbol{\theta})g(\boldsymbol{\theta})|_{\partial\Theta} = 0\}$. Therefore we can use the test functions in the form of $f(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) g(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta})$, which leads to the following Stein discrepancy [Gorham and Mackey, 2015; Liu et al., 2016; Chwialkowski et al., 2016]:

$$\mathrm{S}[p, q] = \sup_{g\in\mathcal{G}_p} |\mathbb{E}_{q(\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) g(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta})]|. \tag{66}$$

This IPM only requires samples from q and the score function of $p(\boldsymbol{\theta})$, which is particularly suited for approximate inference. For example, the posterior density is intractable, however, the score function of the exact posterior can be evaluated as $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D}) = \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \mathcal{D})$. Assuming a tractable joint distribution, this means one can minimise the Stein discrepancy $\mathrm{S}[p, q]$ w.r.t. q to obtain the approximate posterior. There are several notable applications of Stein discrepancy to approximate inference:

- Ranganath et al. [2016a] proposed the operator variational inference (OVI) approach based on Stein discrepancy. OVI defines the set of g functions with neural networks: $\mathcal{G}_p = \{g_\eta(\boldsymbol{\theta}) = \mathrm{NN}_\eta(\boldsymbol{\theta})\}$. Therefore the optimisation procedure of OVI includes optimising the varatiaonal parameters attached to the q distribution as well as the neural network parameters η for the test function $g_\eta(\boldsymbol{\theta})$. This approach is later revisited in a recent attempt in the context of learning energy-based models [Grathwohl et al., 2020].

- One might want to avoid using optimisation-based approach to find the best $g(\boldsymbol{\theta})$ function. In this case we can use the kernel trick again and define $\mathcal{G}_p = \{g \in \mathcal{H} : ||g||_{\mathcal{H}} \leq 1\}$. This makes the optimal g function analytic, and we have the following *kernelised Stein discrepancy* (KSD):

$$\begin{aligned}\mathrm{S}^2[p, q] = \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\theta}'\sim q}[&s_p(\boldsymbol{\theta})^\top \mathcal{K}(\boldsymbol{\theta}, \boldsymbol{\theta}') s_p(\boldsymbol{\theta}') + s_p(\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\theta}'} \mathcal{K}(\boldsymbol{\theta}, \boldsymbol{\theta}') \\ &+ s_p(\boldsymbol{\theta}')^\top \nabla_{\boldsymbol{\theta}} \mathcal{K}(\boldsymbol{\theta}, \boldsymbol{\theta}') + \mathrm{Tr}(\nabla_{\boldsymbol{\theta},\boldsymbol{\theta}'} \mathcal{K}(\boldsymbol{\theta}, \boldsymbol{\theta}'))].\end{aligned} \tag{67}$$

  Though one can directly minimise KSD to obtain an approximation to the exact posterior, a better-known use of KSD in posterior inference context is the *Stein variational gradient descent* (SVGD) algorithm [Liu and Wang, 2016].

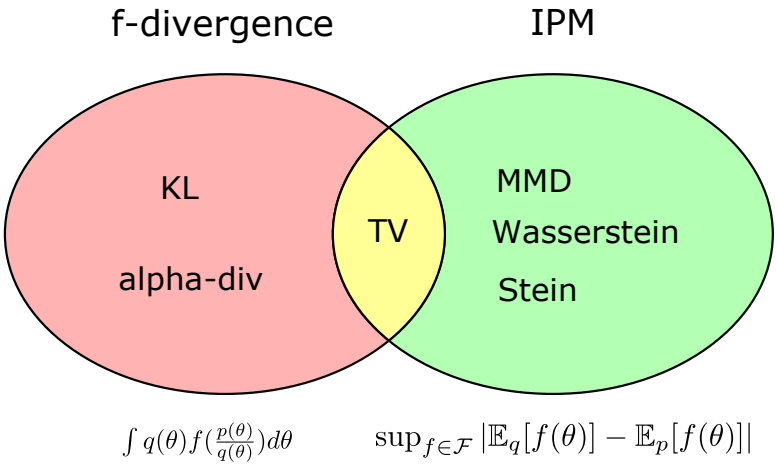

Figure 10: Comparing f-divergences and IPMs. The total variation (TV) distance is a common special case

- Another way to use Stein's identity and KSD in approximate inference is to approximate the *gradient* of the variational parameters for variational inference with *implicit posterior approximations* [Li and Turner, 2018]. This will be further discussed in § 9.

4.4 Connections between f-divergences and IPMs

As two major classes of divergences/discrepancies, both f-divergences and IPMs contain a rich set of divergences/discrepancies. By defining the convex f function for f-divergences and the test function family $\mathcal{F}$ in IPMs, both divergences can focus on different aspects in terms of describing the differences between two distributions. E.g. as discussed, choosing different α values in α-divergence minimisation results in different zero-forcing/mass-covering behaviour in approximate inference. Therefore, choosing the f functions that is suited for the task at hand is crucial for the success of approximate inference with f-divergences or IPMs.

f-divergence and IPMs, although largely different, have a common special case – the *total variation* (TV) distance (see Figure 10):

$$\mathrm{TV}[P, Q] = \sup_{A \in \Sigma} |P(A) - Q(A)|, \tag{68}$$

where Σ is the sigma algebra of the sample space where the distribution p and q are defined. Simplifying the sigma algebra to a Borel set on $\mathbb{R}^D$ and assuming the Lebesgue measure on $\mathbb{R}^D$ as the common dominating measure of P and Q, we can write the TV distance in an IPM form:

$$\mathrm{TV}[p, q] = \sup_{f \in \{\delta(\boldsymbol{\theta} \in A) : A \subset \mathbb{R}^D\}} |\mathbb{E}_p[f(\boldsymbol{\theta})] - \mathbb{E}_q[f(\boldsymbol{\theta})]|. \tag{69}$$

On the other hand, TV distance can be defined using f-divergence, by using

$f(t) = |t-1|$:

$$\begin{aligned}
2\mathrm{TV}[p,q] &= \int q(\boldsymbol{\theta})|\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} - 1|d\boldsymbol{\theta} \\
&= \int_{p(\boldsymbol{\theta})>q(\boldsymbol{\theta})} q(\boldsymbol{\theta})\left(\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} - 1\right)d\boldsymbol{\theta} + \int_{p(\boldsymbol{\theta})\leq q(\boldsymbol{\theta})} q(\boldsymbol{\theta})\left(1 - \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right)d\boldsymbol{\theta} \\
&= \int_{p(\boldsymbol{\theta})>q(\boldsymbol{\theta})} (p(\boldsymbol{\theta}) - q(\boldsymbol{\theta}))d\boldsymbol{\theta} + \int_{p(\boldsymbol{\theta})\leq q(\boldsymbol{\theta})} (q(\boldsymbol{\theta}) - p(\boldsymbol{\theta}))d\boldsymbol{\theta} \\
&= P(A^*) - Q(A^*) + [(1 - Q(A^*)) - (1 - P(A^*))] \\
&= 2|P(A^*) - Q(A^*)|.
\end{aligned} \tag{70}$$

where it can be shown that $A^* := \{\boldsymbol{\theta} \in \mathbb{R}^D : p(\boldsymbol{\theta}) > q(\boldsymbol{\theta})\}$ is the optimal solution of the supremum problem in (69).

Another interesting observation is that f-divergences can also be written in a "supremum form" that is related to IPMs. This is because a convex function f can also be defined using its *dual* form:

$$f(t) = \sup_{u \in dom(f^*)} ut - f^*(u), \tag{71}$$

where $f^*(u)$ is the convex dual of $f(t)$. Observing this, Nguyen et al. [2010] writes the f-divergence as

$$\begin{aligned}
\mathrm{D}_f[p||q] &= \int q(\boldsymbol{\theta}) \sup_{u \in dom(f^*)} \{u\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} - f^*(u)\}d\boldsymbol{\theta} \\
&\geq \sup_{u(\boldsymbol{\theta})} \mathbb{E}_p[u(\boldsymbol{\theta})] - \mathbb{E}_q[f^*(u(\boldsymbol{\theta}))],
\end{aligned} \tag{72}$$

where $u(\boldsymbol{\theta}) : \Theta \to \mathbb{R}$ is an arbitrary function to be optimised, and the second line of equation comes from Jensen's inequality. This gives a variational lower-bound to the f-divergence that has similar form as an IPM (though not exactly in the form of comparing moments between distributions), but at the same time it also shows that in general f-divergence is different from IPMs except for some special cases (e.g. TV). A notable application of the variational form (72) is f-GAN [Nowozin et al., 2016] which parameterises $u(\boldsymbol{\theta})$ with a neural network and performed gradient ascent to solve the supremum optimisation problem.

4.5 Further reading

Minka [2005] is a good reference published in the 2000s for the applications of α-divergence in approximate inference, although before reading it I would suggest another read on the basic concepts of factor graph [Kschischang and Frey, 1998] and the sum-product algorithm (e.g. see Bishop [2006]).

In general for an in-depth understanding of α-divergence I would recommend Shun-ichi Amari's papers on α-divergence, e.g. [Amari et al., 2001; Amari, 2009]. Amari has used α-divergences minimisation and related approaches substantially

in his research *information geometry*. Also Van Erven and Harremoës [2014] is a good reference for properties of Rényi divergence.

As a side note, Rényi divergence has also been used in *differential privacy* as a tool to describe the privacy guarantees. This line of research starts from Mironov [2017], interested readers can also read papers citing this work.

Wang et al. [2018a] discussed the limitations of α-divergence especially in terms of the high variance of their empirical estimators. Motivating by this observation, they proposed a variational objective that corresponds to an adaptive f-divergence to bring the benefit of α-divergence while reducing the variance in the empirical estimates. Rainforth et al. [2018]'s analysis of the high variance issue in empirical gradients also applies to RDVI, and a very recent analysis of gradient variance for α-divergence minimisation is provided in Geffner and Domke [2020].

Stein's method has recently attracted attention in the machine learning community, with the applications in goodness-of-fit tests, learning energy-based models, and approximate inference as mentioned before. A good start to understand this line of research is to first read Stein [1972] to see the original derivation for Gaussian distributions, then Ranganath et al. [2016a] for an introduction to its application in approximate inference. Gorham and Mackey [2015] is also a good read if readers are interested in the usage of Stein's method in statistical testing.

Sriperumbudur et al. [2009] provided a discussion on the connections between f-divergences and IPMs (the ϕ-divergence in that paper is the same as the f-divergence). Also I would recommend watching (parts of) the following tutorials:

- Arthur Gretton, Dougal Sutherland and Wittawat Jitkrittum. NeurIPS tutorial on "Interpretable comparison of distributions and models", part I: http://www.gatsby.ucl.ac.uk/~gretton/papers/neurips19_1.pdf
- Sebastian Nowozin. MLSS Madrid 2018 tutorial on "Generative Adversarial Networks". http://www.nowozin.net/sebastian/blog/mlss-2018-in-madrid.html

5 Estimating the marginal likelihood

5.1 Why estimating the marginal likelihood

As described in § 0.3, the goal of approximate inference is to compute the expectation of some function $F(\boldsymbol{z})$ under the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$. However the posterior distribution is intractable, mainly due to the difficulty of computing the denominator in Bayes' rule:

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})}, \quad p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}. \tag{73}$$

The denominator is also called the *marginal likelihood*, and in general, for a normalised density $\pi(\boldsymbol{z}) = \frac{1}{Z}\pi^*(\boldsymbol{z})$, Z is also called the *normalising constant* or the *partition function*. Usually this distribution has some hyper-parameters that Z and/or $\pi^*(\boldsymbol{x})$ are dependent on. Therefore estimating the partition function is central to distribution fitting that determines the best value of these hyper-parameters.

For example, consider fitting to the observations $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ an *energy-based model* (EBM) $p(\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\theta}}}\exp[-E(\boldsymbol{x};\boldsymbol{\theta})]$ using maximum likelihood estimation (MLE). In this case the partition function $Z_{\boldsymbol{\theta}} = \int \exp[-E(\boldsymbol{x};\boldsymbol{\theta})]d\boldsymbol{x}$ is also a function of the parameters $\boldsymbol{\theta}$, therefore gradient-based optimisation of MLE also requires computing $\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}$. In this case if we have an approximation to the partition function $\log Z_{\boldsymbol{\theta}} \approx \log \tilde{Z}_{\boldsymbol{\theta}}$, then we can replace $Z_{\boldsymbol{\theta}}$ with $\tilde{Z}_{\boldsymbol{\theta}}$ in the MLE objective, and perform gradient descent for training.

Another prevalent example is model selection. Consider the case that we want to select the best neural network architecture for classification tasks. Given N different neural network architectures $\mathcal{M}_1, ..., \mathcal{M}_N$, Bayesian model selection methods pick the best architecture by maximising the model evidence $\mathcal{M}^* = \arg\max_n \log p(\mathcal{D}|\mathcal{M}_n)$, and again $p(\mathcal{D}|\mathcal{M}_n)$ is the marginal likelihood of the model $\mathcal{M}_n$ as well as the partition function of the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}_n)$.

We see from the above two examples the importance of having an accurate estimation/approximation to the partition function/marginal likelihood. In this section we will discuss sampling-based, and variational inference approaches to estimate the marginal log-likelihood $\log p(\boldsymbol{x}) = \log \int p(\boldsymbol{x}, \boldsymbol{z})d\boldsymbol{z}$ (assuming tractable $p(\boldsymbol{x}, \boldsymbol{z})$), with a focus on importance sampling and its advanced variants. Other methods, e.g. message passing, can also provide accurate estimation of the marginal likelihood, and we also provide a brief introduction to them in later sections (§ 6).

5.2 Constructing stochastic lower-bounds through sampling

An attractive way to construct an estimation of the marginal log-likelihood is sampling. In short, given a distribution $q(\mathbf{h})$ to draw sample from, under some mild conditions it is fairly simple to construct a function $F(\mathbf{h}, \boldsymbol{x})$ such that

$$p(\boldsymbol{x}) = \mathbb{E}_{q(\mathbf{h})}[F(\mathbf{h}, \boldsymbol{x})].$$

For example, assume $\mathbf{h} = \boldsymbol{z}$, $p(\boldsymbol{x}, \boldsymbol{z})$ is tractable for any given pair $(\boldsymbol{x}, \boldsymbol{z})$ and consider $q(\boldsymbol{z})$ as some arbitrary distribution with the same or larger support $\text{supp}(p(\boldsymbol{z}|\boldsymbol{x})) \subset \text{supp}(q(\boldsymbol{z}))$. Then with the density ratio $F(\boldsymbol{z}, \boldsymbol{x}) = \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})}$ as the F function, $F(\boldsymbol{z}, \boldsymbol{x}), \boldsymbol{z} \sim q(\boldsymbol{z})$ returns an unbiased estimator of the marginal likelihood $p(\boldsymbol{x})$. This is exactly the idea of *importance sampling*, and in principle $q(\boldsymbol{z})$ does not even need to be close to $p(\boldsymbol{z}|\boldsymbol{x})$ in order to compute an unbiased estimate of $p(\boldsymbol{x})$ (although it might result in high variance in practice).

Given a sampling algorithm to estimate $p(\boldsymbol{x})$, it is straight-forward to obtain a lower-bound on $\log p(\boldsymbol{x})$ using Jensen's inequality:

$$\log p(\boldsymbol{x}) = \log \mathbb{E}_{q(\mathbf{h})}[F(\mathbf{h}, \boldsymbol{x})] \geq \mathbb{E}_{q(\mathbf{h})}[\log F(\mathbf{h}, \boldsymbol{x})]. \tag{74}$$

This means $\log F(\mathbf{h}, \boldsymbol{x}), \mathbf{h} \sim q(\mathbf{h})$ also forms a stochastic lower-bound for $\log p(\boldsymbol{x})$. The lower-bound of the importance weighted auto-encoder (IWAE) approach presented in § 3.3.2 can also be derived from the above approach by defining the *augmented* random variable $\mathbf{h} = \{\boldsymbol{z}^1, ..., \boldsymbol{z}^K\}$, $q(\mathbf{h}) = \prod_{k=1}^K q(\boldsymbol{z}^k)$ and $F(\mathbf{h}, \boldsymbol{x}) = \frac{1}{K}\sum_{k=1}^K \frac{p(\boldsymbol{x},\boldsymbol{z}^k)}{q(\boldsymbol{z}^k)}$.

The quality of the stochastic lower-bound is highly dependent on the quality of the proposal distribution $q(\mathbf{h})$, for example importance sampling with $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x})$ will return an exact estimate with zero variance. Therefore, advanced techniques to improve importance sampling has been proposed, such as *annealed importance sampling* (AIS) [Neal, 2001] that will be introduced below.

Assume we have two distributions $p_0(\boldsymbol{z}) = \frac{1}{Z_0}p_0^*(\boldsymbol{z})$ and $p_T(\boldsymbol{z}) = \frac{1}{Z_T}p^*(\boldsymbol{z})$, where $p_0(\boldsymbol{z})$ is relatively simple, and $p_T(\boldsymbol{z})$ is some complicated target distribution that we aim to sample from. AIS constructs a "bridge" between these two distributions so that by a series of transitions samples from $p_0(\boldsymbol{z})$ can be (stochastically) transformed into samples from $p_T(\boldsymbol{z})$. Concretely, AIS constructs *intermediate distributions* $p_t, t = 1, ..., T-1$ to "bridge" between two distributions p_0 and p_T:

$$p_t(\boldsymbol{z}) = \frac{1}{Z_t}p_0^*(\boldsymbol{z})^{1-\beta_t}p_T^*(\boldsymbol{z})^{\beta_t}, \quad 0 = \beta_0 < \beta_1 < ... < \beta_T = 1.$$

Now assume $\mathcal{T}_t(\boldsymbol{z}|\boldsymbol{z}')$ is an MCMC transition kernel that leaves $p_t(\boldsymbol{z})$ invariant. Then we consider an augmented state space $\mathbf{h} = \{\boldsymbol{z}_0, ..., \boldsymbol{z}_T\}$ and define the "model" and proposal distribution as

$$\begin{aligned} \tilde{p}(\mathbf{h}) = p_T(\boldsymbol{z}_T)\prod_{t=1}^T \tilde{\mathcal{T}}_t(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t), \quad \tilde{\mathcal{T}}_t(\boldsymbol{z}|\boldsymbol{z}') = \frac{\mathcal{T}_t(\boldsymbol{z}'|\boldsymbol{z})p_t(\boldsymbol{z})}{p_t(\boldsymbol{z}')}, \\ q(\mathbf{h}) = p_0(\boldsymbol{z}_0)\prod_{t=1}^T \mathcal{T}_t(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}). \end{aligned} \tag{75}$$

This returns the density ratio as

$$\begin{aligned}\frac{\tilde{p}(\mathbf{h})}{q(\mathbf{h})} &= \frac{p_T(\boldsymbol{z}_T)}{p_0(\boldsymbol{z}_0)}\prod_{t=1}^{T}\frac{\tilde{\mathcal{T}}_t(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{\mathcal{T}_t(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})} \\ &= \frac{Z_0}{Z_T}\frac{p_T^*(\boldsymbol{z}_T)}{p_0^*(\boldsymbol{z}_0)}\prod_{t=1}^{T}\frac{p_t^*(\boldsymbol{z}_{t-1})}{p_t^*(\boldsymbol{z}_t)} \\ &= \frac{Z_0}{Z_T}\prod_{t=1}^{T}\frac{p_t^*(\boldsymbol{z}_{t-1})}{p_{t-1}^*(\boldsymbol{z}_{t-1})} := \frac{Z_0}{Z_T}\prod_{t=1}^{T} w_t(\boldsymbol{z}_{t-1}), \quad w_t(\boldsymbol{z}) := \frac{p_t^*(\boldsymbol{z})}{p_{t-1}^*(\boldsymbol{z})}.\end{aligned} \tag{76}$$

Since $\mathbb{E}_{q(\mathbf{h})}[\tilde{p}(\mathbf{h})/q(\mathbf{h})] = 1$, this means $Z_0\mathbb{E}_{q(\mathbf{h})}[\prod_{t=1}^{T} w_t(\boldsymbol{z}_{t-1})] = Z_T$. Therefore $F(\mathbf{h}, \boldsymbol{x}) = Z_0\prod_{t=1}^{T} w_t(\boldsymbol{z}_{t-1}), \mathbf{h} \sim q(\mathbf{h})$ is an unbiased estimator of Z_T, which then equals to $\log p(\boldsymbol{x})$ if we set $p_T(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x})$. Thus by further assuming $p_0(\boldsymbol{z}) = p(\boldsymbol{z})$ it is straight-forward to show that $p_t(\boldsymbol{z}) = \frac{1}{Z_t}p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})^{\beta_t}$, $Z_0 = 1, Z_T = p(\boldsymbol{x})$, and

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_q\left[\sum_{t=1}^{T}\log w_t(\boldsymbol{z}_{t-1})\right] = \mathbb{E}_q\left[\sum_{t=1}^{T}\log p(\boldsymbol{x}|\boldsymbol{z}_{t-1})^{\beta_t-\beta_{t-1}}\right]. \tag{77}$$

AIS is a special case of *sequential importance sampling* that augmentes the state-space and constructes a sequence of importance sampling procedure. In general the proposal $\mathcal{T}_t(\boldsymbol{z}|\boldsymbol{z}')$ does not necessarily need to be an MCMC transition kernel, and the "reverse model" $\tilde{T}_t$ can be relaxed to arbitrary distribution $r_t(\boldsymbol{z}'|\boldsymbol{z})$. In this case, with notations $\mathcal{T}_t(\boldsymbol{z}|\boldsymbol{z}') = q_t(\boldsymbol{z}|\boldsymbol{z}'), p_0(\boldsymbol{z}) = q(\boldsymbol{z})$ the density ratio is

$$\frac{\tilde{p}(\mathbf{h})}{q(\mathbf{h})} = \frac{p_T^*(\boldsymbol{z}_T)}{Z_T q(\boldsymbol{z}_0)}\prod_{t=1}^{T}\frac{r_t(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{q_t(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}, \tag{78}$$

therefore using similar reasoning we see that $F(\mathbf{h}, \boldsymbol{x}) = \frac{p(\boldsymbol{x},\boldsymbol{z}_T)}{q(\boldsymbol{z}_0)}\prod_{t=1}^{T}\frac{r_t(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{q_t(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}, \mathbf{h} \sim q(\mathbf{h})$ is also an unbiased estimator of $p(\boldsymbol{x})$ when $p_T(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x})$. The corresponding lower-bound $\mathbb{E}_q[\log F(\mathbf{h}, \boldsymbol{x})]$ coincides with the auxiliary variational inference objective presented in Agakov and Barber [2004]; Salimans et al. [2015]; Ranganath et al. [2016b], and the lower-bounds presented in Maaløe et al. [2016]; Tran et al. [2016] correspond to the special cast that $T = 1$. All these papers suggested fitting both q_t and r_t by maximising this lower-bound in order to improve the quality of the proposal distribution $q(\mathbf{h})$.

Remark (IWAE as VAE in an augmented space)**.** Notably Cremer et al. [2017] and Domke and Sheldon [2018] proposed re-interpretations of IWAE so that with a specific design of $\mathbf{h}$ variables (and the corresponding $q(\mathbf{h})$), the corresponding F function can be formulated as $F(\mathbf{h}, \boldsymbol{x}) = \frac{\tilde{p}(\boldsymbol{x},\mathbf{h})}{q(\mathbf{h})}$ for some "model" $\tilde{p}(\boldsymbol{x}, \mathbf{h})$. Therefore IWAE can also be viewed as performing KL-divergence based variational inference just like VAE (but with a different model in an augmented space). Readers are refer to the two papers for details, and here we briefly present a third approach which considers $\mathbf{h} = \{\boldsymbol{z}^1, ..., \boldsymbol{z}^K, i\}$ with $i \in \{1, ..., K\}$. Define $q(\mathbf{h}) = \prod_{k=1}^{K} q(\boldsymbol{z}^k)q(i|\boldsymbol{z}^{1:K})$

and $\tilde{p}(\boldsymbol{x}, \mathbf{h}) = \prod \tilde{p}(\boldsymbol{x}|i, \boldsymbol{z}^{1:K}) \prod_{k=1}^{K} \tilde{p}(\boldsymbol{z}^k|i)\tilde{p}(i)$, with $q(\boldsymbol{z}^k)$ the sampling proposal that has a shared density form across k, and

$$q(i|\boldsymbol{z}^{1:K}) = \frac{p(\boldsymbol{x}, \boldsymbol{z}^i)/q(\boldsymbol{z}^i)}{\sum_{k=1}^{K} p(\boldsymbol{x}, \boldsymbol{z}^k)/q(\boldsymbol{z}^k)} \quad \text{// resampling,}$$

$$\tilde{p}(i) = \frac{1}{K} \quad \text{// uniform distribution,}$$

$$\tilde{p}(\boldsymbol{z}^k|i) = \mathbb{1}(k = i)p(\boldsymbol{z}^k) + \mathbb{1}(k \neq i)q(\boldsymbol{z}^k), \quad \tilde{p}(\boldsymbol{x}|i, \boldsymbol{z}^{1:K}) = p(\boldsymbol{x}|\boldsymbol{z}^i),$$

readers can verify that using the above definition we have $p(\boldsymbol{x}) = \mathbb{E}_{q(\mathbf{h})}[F(\mathbf{h}, \boldsymbol{x})]$, and at the same time,

$$\log \tilde{p}(\boldsymbol{x}) = \log p(\boldsymbol{x}) = \mathbb{E}_q \left[\log \frac{\tilde{p}(\boldsymbol{x}, \boldsymbol{z}^{1:K}, i)}{q(\boldsymbol{z}^{1:K}, i)} \right] + \mathrm{KL}[q(\boldsymbol{z}^{1:K}, i)||\tilde{p}(\boldsymbol{z}^{1:K}, i|\boldsymbol{x})].$$

Therefore we clearly see that IWAE is a VAE-style inference method for the augmented model $\tilde{p}(\boldsymbol{x}, \boldsymbol{z}^{1:K}, i)$. Although not rigorously proved, I believe this type of understanding can also be applied to other advanced inference methods such as variational SMC [Maddison et al., 2017a; Le et al., 2017; Naesseth et al., 2017]. This reasoning of variational inference in an augmented space is further expanded by Domke and Sheldon [2019] to the lower-bound of the form in (74).

5.3 Stochastic upper-bounds of the marginal log-likelihood

We have briefly discussed how to construct stochastic lower-bounds via sampling. However, as lower-bounds only give conservative estimates of the marginal log-likelihood, they might not reflect the true model evidence in model selection, and thus bias the selection procedure towards less powerful models. In this section I will briefly present existing *upper-bounds* for the marginal log-likelihood, which, combined with the lower-bounds, provides more reliable estimates of $\log Z$ for downstream tasks.

5.3.1 Variational Rényi bound/χ upper-bound

Recall the introduction of the *variational Rényi bound* VR-bound in § 4.1.2 where the bound writes as (with $\alpha \neq 1$)

$$\mathcal{L}_\alpha(q; \boldsymbol{x}) := \frac{1}{1-\alpha} \log \mathbb{E}_q \left[\left(\frac{p(\boldsymbol{z}, \boldsymbol{x})}{q(\boldsymbol{z})} \right)^{1-\alpha} \right]. \tag{79}$$

Furthermore, we stated in § 4.1.2 becomes an upper-bound of $\log p(\boldsymbol{x})$ for $\alpha < 0$ due to the skew symmetry property of Rényi divergence. Van Erven and Harremoës [2014] also observed the following monotonicity of Rényi divergence.

Proposition 2. *(Monotonicity) Rényi's α-divergence definition (57), extended to negative α, is* ***continuous*** *and* ***non-decreasing*** *on $\alpha \in \{\alpha : -\infty < \mathrm{D}_\alpha^R < +\infty\}$.*

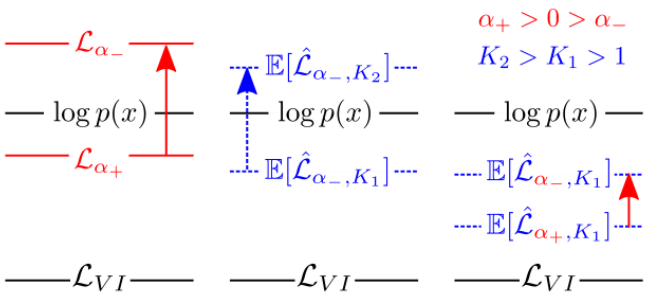

Figure 11: An intuition of the quality of MC estimates for the VR-bounds with different α and K values. Reproduced from Li and Turner [2016].

This immediately leads to the following theorem regarding the VR-bound.

Theorem 1. *The objective* $\mathcal{L}_\alpha(q;\boldsymbol{x})$ *is* ***continuous*** *and* ***non-increasing*** *on* $\alpha \in \{\alpha : |\mathcal{L}_\alpha| < +\infty\}$. *Especially for all* $0 < \alpha_+ < 1$ *and* $\alpha_- < 0$,

$$\mathcal{L}_{VI}(q;\boldsymbol{x}) = \lim_{\alpha\to 1} \mathcal{L}_\alpha(q;\boldsymbol{x}) \leq \mathcal{L}_{\alpha_+}(q;\boldsymbol{x}) \leq \mathcal{L}_0(q;\boldsymbol{x}) \leq \mathcal{L}_{\alpha_-}(q;\boldsymbol{x})$$

Also $\mathcal{L}_0(q;\boldsymbol{x}) = \log p(\boldsymbol{x})$ *if and only if the support* $\text{supp}(p(\boldsymbol{z}|\boldsymbol{x})) \subseteq \text{supp}(q(\boldsymbol{z}))$.

When α has negative integer values the bound is also referred as the χ upper bound (CUBO) in Dieng et al. [2017].

Theorem 1 indicates that the VR bound can be useful for model selection by sandwiching the marginal log-likelihood with bounds computed using positive and negative α values. In particular $\mathcal{L}_0 = \log p(\boldsymbol{x})$ under the mild assumption that q is supported where the exact posterior is supported. This assumption holds for many commonly used distributions, e.g. Gaussians are supported on the entire space.

Unfortunately the VR bound is usually just as intractable as the marginal likelihood for many useful models when $\alpha \neq 1$. Li and Turner [2016] further applies MC approximation to (58), and IWAE is a special case of that MC approximation method when $\alpha = 0$. Therefore, MC-VR can be applied to precisely the same set of models as MC-VI [Paisley et al., 2012; Salimans and Knowles, 2013; Ranganath et al., 2014; Kucukelbir et al., 2015].

Regarding the estimation quality of the marginal log-likelihood, Li and Turner [2016] also characterised the bias introduced by MC approximation in their Theorem 2 and Corollary 1. The idea behind the theorems is visualised in Figure 11. In short the observation is that for a given $\alpha < 0$ and fixed proposal $q(\boldsymbol{z})$ and target $p(\boldsymbol{z}|\boldsymbol{x})$, we can select a sufficiently large number of samples K to obtain a stochastic upper-bound of the marginal log-likelihood. Furthermore, this required K number becomes smaller when α approaches to negative infinity. Unfortunately there is no detailed guidance on selecting this K number for a given α, and in practice, depending on the quality of the q distribution (in terms of being used as an importance sampling proposal), this K can be very large.

5.3.2 Harmonic mean estimator

Newton and Raftery [1994] proposed the harmonic mean estimator which can potentially be used to construct a stochastic upper-bound of the marginal log-likelihood. It is based on the observation that

$$\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{p(\boldsymbol{z})}{p(\boldsymbol{z},\boldsymbol{x})}\right] = \int \frac{p(\boldsymbol{z},\boldsymbol{x})p(\boldsymbol{z})}{p(\boldsymbol{x})p(\boldsymbol{z},\boldsymbol{x})}d\boldsymbol{z} = \frac{1}{p(\boldsymbol{x})}\int p(\boldsymbol{z})d\boldsymbol{z} = \frac{1}{p(\boldsymbol{x})}. \tag{80}$$

This means if one can get samples from the exact posterior $p(\boldsymbol{z}|\boldsymbol{x})$, then these samples can be used to construct a stochastic upper-bound of the marginal log-likelihood (again by Jensen's inequality):

$$-\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\log\frac{p(\boldsymbol{z})}{p(\boldsymbol{z},\boldsymbol{x})}\right] \geq -\log\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{p(\boldsymbol{z})}{p(\boldsymbol{z},\boldsymbol{x})}\right] = \log p(\boldsymbol{x}). \tag{81}$$

Unfortunately the samples from the exact posterior are difficult to obtain; even when we do have these samples, the above stochastic upper-bound can be very loose. On the other hand, a native application of importance sampling does not address the issue since (1) the importance weight is unavailable due to intractability of $p(\boldsymbol{z}|\boldsymbol{x})$, and (2) self-normalised importance sampling does not maintain a stochastic upper-bound.

Grosse et al. [2015] provides an interesting solution to the above issue by leveraging the AIS technique, but in the reverse direction. Recall in the derivation of the AIS estimator where used the "augmented model" and the proposal distribution defined in (75). Similar to the derivation of (76), we can also write

$$\frac{q(\mathbf{h})}{\tilde{p}(\mathbf{h})} = \frac{p_0(\boldsymbol{z}_0)}{p_T(\boldsymbol{z}_T)}\prod_{t=1}^{T}\frac{\mathcal{T}_t(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}{\tilde{\mathcal{T}}_t(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)} := \frac{Z_T}{Z_0}\prod_{t=1}^{T}\tilde{w}_t(\boldsymbol{z}_{t-1}), \quad \tilde{w}_t(\boldsymbol{z}) := \frac{p^*_{t-1}(\boldsymbol{z})}{p^*_t(\boldsymbol{z})}. \tag{82}$$

Since $\mathbb{E}_{\tilde{p}(\boldsymbol{h})}[\frac{q(\mathbf{h})}{\tilde{p}(\mathbf{h})}] = 1$ this effectively says $\mathbb{E}_{\tilde{p}(\boldsymbol{h})}[\prod_{t=1}^{T}\tilde{w}_t(\boldsymbol{z}_{t-1})] = \frac{Z_0}{Z_T}$. Thus by making similar assumptions as before ($p_0(\boldsymbol{z}) = p(\boldsymbol{z})$, $p_t(\boldsymbol{z}) = \frac{1}{Z_t}p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})^{\beta_t}$, $0 = \beta_0 < ... < \beta_T = 1$), we have $Z_0 = 1, Z_T = p(\boldsymbol{x})$, and the following stochastic upper-bound

$$\log p(\boldsymbol{x}) \leq -\mathbb{E}_{\tilde{p}}\left[\sum_{t=1}^{T}\log\tilde{w}_t(\boldsymbol{z}_{t-1})\right] = \mathbb{E}_{\tilde{p}}\left[\sum_{t=1}^{T}\log p(\boldsymbol{x}|\boldsymbol{z}_{t-1})^{\beta_t-\beta_{t-1}}\right]. \tag{83}$$

Comparing the bounds obtained in equations (77) and (83), we see that the estimators differs only in the sampling distribution, where the lower-bound (77) is obtained by a "forward" AIS from $p_0(\boldsymbol{z})$ to $p_T(\boldsymbol{z})$, while the upper-bound (83) is obtained by a "backward" AIS from $p_T(\boldsymbol{z})$ to $p_0(\boldsymbol{z})$. Since starting from sampling $p_0(\boldsymbol{z})$ is considerably easier, Grosse et al. [2015] suggested a "bi-directional Monte Carlo" procedure, by first running the forward AIS to compute the stochastic lower-bound (77) and get samples from $p_T(\boldsymbol{z}|\boldsymbol{x})$, then using these last-step samples from the forward AIS to initialise the backward AIS, and running the backward AIS to obtain the stochastic upper-bound (83). By doing so, one can obtain a

"sandwiched estimate" of the marginal log-likelihood, which provides a more robust tool for model selection. Still the bias and variance in these AIS-based bounds depends on the proposal distribution (e.g. the choice of $p_0(\boldsymbol{z})$ and the temperature schedule) as well as the intermediate MCMC steps to (approximately) get samples from $p_t(\boldsymbol{z})$.

5.4 Further reading

Hamiltonian annealed importance sampling (HAIS) is one of the state-of-the-art approaches for estimating the marginal likelihood. This method is based on AIS [Neal, 2001], and the samples from each of the intermediate distribution $p_t(\boldsymbol{z})$ are obtained using Hamiltonian Monte Carlo (HMC) [Duane et al., 1987; Neal, 2011]. Wu et al. [2017] has shown that for evaluating the quality of deep generative models in terms of their marginal log-likelihood, HAIS might provide better estimates than naive importance sampling. The tempering path idea in AIS is also used in Masrani et al. [2019] to construct lower-bounds and upper-bounds of the marginal log likelihood.

Many sampling-based methods for marginal likelihood estimation have been proposed, e.g. see Chib [1995]; Skilling et al. [2006]. Also recent studies have also introduced multi-level Monte Carlo techniques [Giles, 2015] to marginal log-likelihood estimation. The idea is to construct a better estimator of the target by combining a series of MC estimators that require different number of MC samples and have different precision. For an application in deep generative models, see e.g. Nowozin [2018]; Luo et al. [2020].

For estimating the partition function of an EBM based on a graph (e.g. Ising model and conditional random field), message passing based approaches have been investigated for long, see § 6 for further discussions. On the sampling side, the Swendsen-Wang algorithm [Swendsen and Wang, 1987] is a popular approach for simulating samples from Ising models and Potts models. For an application to computer vision tasks (graph cut and image segmentation), see e.g. Barbu and Zhu [2005].

In general for model selection, I would recommend reading materials related to Akaike information criterion (AIC) [Akaike, 1974], Bayesian information criterion (BIC) [Schwarz, 1978] and minimum description length (MDL) [Grünwald, 2007] principles. The connections of AIC/BIC to approximate posterior inference are presented in e.g. Burnham and Anderson [2004]; Konishi and Kitagawa [2008], and Hinton and Van Camp [1993] demonstrated the connections between VI and MDL.

6 Message passing algorithms

In § 1.2 the variational lower-bound is introduced as an lower-bound of $\log Z$:

$$\mathcal{L}_{\text{VI}}(q) := \int q(\boldsymbol{z}) \log \frac{p^*(\boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z} \leq \log Z. \tag{84}$$

Variational lower-bound is a conservative estimate of the log marginal likelihood, therefore it is "safe" to optimise it w.r.t. parameters of $p(\boldsymbol{z})$. However it is also less accurate if we are interested in an estimation of $\log Z$ only (even after optimising q in a constrained set $\mathcal{Q}$). In this section I will briefly describe another approximation approach to the log marginal likelihood called *Bethe free-energy*, which has been shown to be more accurate after optimising the q distribution. Note that the derivation of the (constrained) Bethe free-energy optimisation problem is similar to the derivation of the Expectation Propagation (EP) optimisation problem, thus in this section we also refer the primal form of EP energy to Bethe free-energy.

6.1 The sum rule and the product rule

6.1.1 Factor graph

One way to represent a distribution is to draw a *factor graph* [Kschischang and Frey, 1998; Kschischang et al., 2001]. A simple example would be a joint distribution $P(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{x}_3)$ which can be factorised into $P(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3) = f_1(\boldsymbol{z}_1) f_2(\boldsymbol{z}_2) f_3(\boldsymbol{z}_3)$, illustrated in Figure 12. To give a formal definition, a factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$ is a bipartite graph between variable nodes (circles) $i \in \mathcal{V}$ and factor nodes (squares) $f \in \mathcal{F}$, where a factor node f is connected to a variable node $\boldsymbol{z}_i$ iff. $\boldsymbol{z}_i$ belongs to the function f's domain. A distribution may be represented by different factor graphs, since we can merge factors to a single one. In particular to our example, we may also write $P(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3) = f_1'(\boldsymbol{z}_1) f_2'(\boldsymbol{z}_2, \boldsymbol{z}_3)$, where $f_1'(\boldsymbol{z}_1) = f_1(\boldsymbol{z}_1)$ and $f_2'(\boldsymbol{z}_2, \boldsymbol{z}_3) = f_2(\boldsymbol{z}_2) f_3(\boldsymbol{z}_3)$. In the rest of this section we use $\boldsymbol{z}_a$ to denote a subset of random variables $\{\boldsymbol{z}_i\}$ connected to a factor node f_a. Therefore given a factor graph $\mathcal{G}$, the distribution of random variables is defined as ($\boldsymbol{z} = \{\boldsymbol{z}_1, ..., \boldsymbol{z}_d\}$)

$$p(\boldsymbol{z}) = \frac{1}{Z} \prod_{f_a \in \mathcal{F}} f_a(\boldsymbol{z}_a). \tag{85}$$

In the following we give two examples of such distributions:

- Undirected graphical model (UGM) with a bipartite graph: consider a UGM with graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and distribution as

$$p(\boldsymbol{z}) = \frac{1}{Z} \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j) \prod_{i \in \mathcal{V}} \psi_i(\boldsymbol{z}_i). \tag{86}$$

  In the literature $\psi_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ and $\psi_i(\boldsymbol{z}_i)$ are often referred to *pairwise potentials* and *singleton potentials*, respectively. An alternative way to write down the

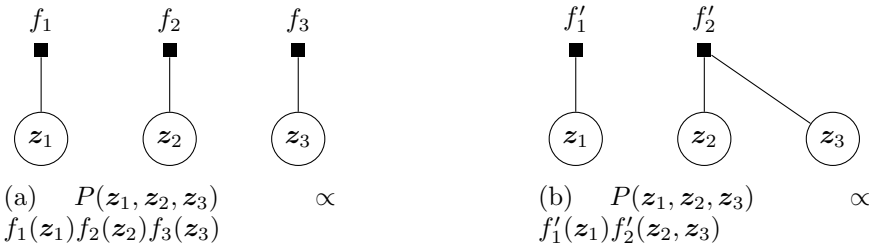

Figure 12: Two different factor graph representations for the same probability distribution, if defining $f'_1 = f_1$, $f'_2 = f_2 f_3$.

UGM distribution is as follows. Denote n_i the number of neighbours of a node $i \in \mathcal{V}$. Then we define a set of factors as

$$\mathcal{F} = \{f_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \psi_i(\boldsymbol{z}_i)\psi_j(\boldsymbol{z}_j)\psi_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j) \mid (i,j) \in \mathcal{E}\} \cup \{f_i(\boldsymbol{z}_i) = \psi_i(\boldsymbol{z}_i)^{1-n_i} \mid i \in \mathcal{V}\},$$

and it is straightforward to show that $p(\boldsymbol{z}) = \frac{1}{Z}\prod_{f_a \in \mathcal{F}} f_a(\boldsymbol{z}_a)$. In this case $\boldsymbol{z}_a = \boldsymbol{z}_i$ for factor f_i and $\boldsymbol{z}_a = (\boldsymbol{z}_i, \boldsymbol{z}_j)$ for factor f_{ij}.

- Bayesian posterior inference: for a given dataset $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$, consider the following posterior distribution

$$p(\boldsymbol{z}|\mathcal{D}) = \frac{1}{Z} p_0(\boldsymbol{z}) \prod_{n=1}^N p(\boldsymbol{x}_n|\boldsymbol{z}). \tag{87}$$

Then we can define a set of factors as

$$\mathcal{F} = \{f_0(\boldsymbol{z}) = p_0(\boldsymbol{z})\} \cup \{f_n(\boldsymbol{z}) = p(\boldsymbol{x}_n|\boldsymbol{z}) \mid n = 1, ..., N\},$$

and it is straightforward to show that $p(\boldsymbol{z}|\mathcal{D}) = \frac{1}{Z}\prod_{f_a \in \mathcal{F}} f_a(\boldsymbol{z}_a)$. In this case $\boldsymbol{z}_a = \boldsymbol{z}$ for all the factors in $\mathcal{F}$.

6.1.2 The sum-product algorithm

Many inference tasks require computing the marginal distribution on a subset of variables in $\boldsymbol{z}$. More generally speaking, computing the *moments* or statistics of a given distribution represents many of the inference tasks in practice. For example, consider computing the marginal distribution $p(\boldsymbol{z}_1)$ for the following distribution:

$$p(\boldsymbol{z}) = f_a(\boldsymbol{z}_1)f_b(\boldsymbol{z}_2)f_c(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3)f_d(\boldsymbol{z}_3, \boldsymbol{z}_4)f_e(\boldsymbol{z}_3, \boldsymbol{z}_5). \tag{88}$$

The factor graph representation of the above $p(\boldsymbol{z})$ is shown in Figure 13. It is easy to show that

$$\begin{aligned} p(\boldsymbol{z}_1) &= \int f_a(\boldsymbol{z}_1)f_b(\boldsymbol{z}_2)f_c(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3)f_d(\boldsymbol{z}_3, \boldsymbol{z}_4)f_e(\boldsymbol{z}_3, \boldsymbol{z}_5)d\boldsymbol{z}_{2:5} \\ &= f_a(\boldsymbol{z}_1) \int f_b(\boldsymbol{z}_2) \left(\int \left[f_c(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3) \int f_d(\boldsymbol{z}_3, \boldsymbol{z}_4)d\boldsymbol{z}_4 \int f_e(\boldsymbol{z}_3, \boldsymbol{z}_5)d\boldsymbol{z}_5 \right] d\boldsymbol{z}_3 \right) d\boldsymbol{z}_2, \end{aligned} \tag{89}$$

where the second line of the equation is obtained by grouping the factors according to their inputs. This means we can define

$$p(\boldsymbol{z}_1) = m_{f_a\to\boldsymbol{z}_1}(\boldsymbol{z}_1)m_{f_c\to\boldsymbol{z}_1}(\boldsymbol{z}_1), \quad m_{f_a\to\boldsymbol{z}_1}(\boldsymbol{z}_1) := f_a(\boldsymbol{z}_1)$$
$$m_{f_c\to\boldsymbol{z}_1}(\boldsymbol{z}_1) := \int f_b(\boldsymbol{z}_2)\left(\int\left[f_c(\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3)\int f_d(\boldsymbol{z}_3,\boldsymbol{z}_4)d\boldsymbol{z}_4\int f_e(\boldsymbol{z}_3,\boldsymbol{z}_5)d\boldsymbol{z}_5\right]d\boldsymbol{z}_3\right)d\boldsymbol{z}_2. \tag{90}$$

It remains to compute $m_{f_c\to\boldsymbol{z}_1}(\boldsymbol{z}_1)$ as a "message" from the factor f_c to the variable $\boldsymbol{z}_1$. As shown in the equation, the integration can be done in a local way. Define

$$m_{f_b\to\boldsymbol{z}_2}(\boldsymbol{z}_2) := f_b(\boldsymbol{z}_2),$$
$$m_{f_e\to\boldsymbol{z}_3}(\boldsymbol{z}_3) := \int f_e(\boldsymbol{z}_3,\boldsymbol{z}_5)d\boldsymbol{z}_5, \quad m_{f_d\to\boldsymbol{z}_3}(\boldsymbol{z}_3) := \int f_d(\boldsymbol{z}_3,\boldsymbol{z}_4)d\boldsymbol{z}_4, \tag{91}$$
$$m_{\boldsymbol{z}_3\to f_c}(\boldsymbol{z}_3) := m_{f_e\to\boldsymbol{z}_3}(\boldsymbol{z}_3)m_{f_d\to\boldsymbol{z}_3}(\boldsymbol{z}_3), \quad m_{\boldsymbol{z}_2\to f_c}(\boldsymbol{z}_2) := m_{f_b\to\boldsymbol{z}_2}(\boldsymbol{z}_2)$$

then

$$\begin{aligned} m_{f_c\to\boldsymbol{z}_1}(\boldsymbol{z}_1) &= \int m_{f_b\to\boldsymbol{z}_2}(\boldsymbol{z}_2)\left(\int f_c(\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3)m_{f_d\to\boldsymbol{z}_3}(\boldsymbol{z}_3)m_{f_e\to\boldsymbol{z}_3}(\boldsymbol{z}_3)d\boldsymbol{z}_3\right)d\boldsymbol{z}_2 \\ &= \int f_c(\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3)m_{\boldsymbol{z}_2\to f_b}(\boldsymbol{z}_2)m_{\boldsymbol{z}_3\to f_c}(\boldsymbol{z}_3)d\boldsymbol{z}_{2:3} \\ &:= \int f_c(\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3)\prod_{\boldsymbol{z}_i\in N(f_c),i\neq 1} m_{\boldsymbol{z}_i\to f_c}(\boldsymbol{z}_i)d\boldsymbol{z}_{2:3}, \end{aligned} \tag{92}$$

where $N(f_c) = \{\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3\}$ denotes the variable that are connected to the factor node in the factor graph in Figure 13. The above "message passing" procedure provides us a convenient way to compute the marginal distribution given a joint distribution with its associated factor graph. This algorithm is also called the sum-product algorithm since the two type of messages – variable-to-factor message and factor-to-variable message – are based on the sum rule and product rule of probability distributions:

$$m_{f_i\to\boldsymbol{z}_j}(\boldsymbol{z}_j) := \int f_i(\boldsymbol{z}_{N(f_i)})\prod_{\boldsymbol{z}_k\in N(f_i),k\neq j} m_{\boldsymbol{z}_k\to f_i}(\boldsymbol{z}_k)d\boldsymbol{z}_{N(f_i)\backslash\{i\}}, \quad \text{\# sum-product} \tag{93}$$

$$m_{\boldsymbol{z}_j\to f_i}(\boldsymbol{z}_j) := \prod_{f_k\in N(\boldsymbol{z}_j),k\neq i} m_{f_k\to\boldsymbol{z}_j}(\boldsymbol{z}_j), \quad \text{\# product} \tag{94}$$

$$p(\boldsymbol{z}_j) = \prod_{f_i\in N(\boldsymbol{z}_j)} m_{f_i\to\boldsymbol{z}_j}(\boldsymbol{z}_j). \quad \text{\# product} \tag{95}$$

The message passing procedure is also visualised in Figure 13.

6.2 Expectation Propagation

Now we consider a distribution $\pi(\boldsymbol{z}) \propto \prod_a \tilde{f}_a(\boldsymbol{z}_a)$, where $\tilde{f}_a$ are the factors in the corresponding factor graph with associated variables $\boldsymbol{z}_a$. As mentioned above

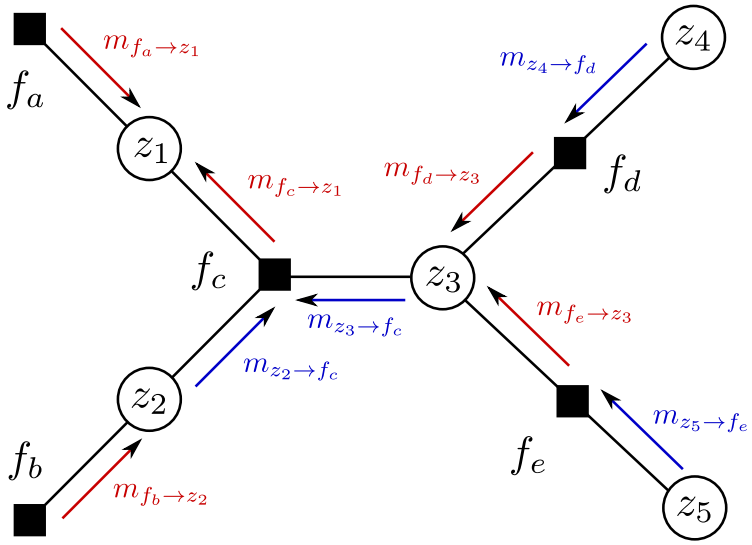

Figure 13: The message passing algorithm (i.e. the sum-product algorithm) to compute the marginal distribution $p(\boldsymbol{z}_1)$.

the exact posterior is a good example here, where $\pi(\boldsymbol{z}) \propto p_0(\boldsymbol{z}) \prod_n p(\boldsymbol{x}_n|\boldsymbol{z})$, $\tilde{f}_a$ represents either the prior distribution $p_0(\boldsymbol{z})$ or the likelihood function $p(\boldsymbol{x}_n|\boldsymbol{z})$ associated with a datum. In this section we introduce the Expectation Propagation (EP) [Minka, 2001b] algorithm which approximates $\pi(\boldsymbol{z})$ by another factor graph with a collection of simpler functions $q(\boldsymbol{z}) \propto \prod_a f_a(\boldsymbol{z}_a)$.

To summarise the algorithm, we first define the "leave-one-out", or *cavity distribution*[21]

$$q_{-a}(\boldsymbol{z}) \propto \prod_{b \neq a} f_b(\boldsymbol{z}_b) \propto q(\boldsymbol{z}) / f_a(\boldsymbol{z}_a), \tag{96}$$

which is computed by multiplying all the other factors except the selected one. The second step is to compute the *tilted distribution* by inserting back the true factor $\tilde{f}_a$ that f_a approximates:

$$\tilde{p}_a(\boldsymbol{z}) \propto q_{-a}(\boldsymbol{z}) \tilde{f}_a(\boldsymbol{z}_a), \tag{97}$$

and update the selected factor f_a by minimising the KL divergence from $\tilde{p}_a$ to the approximation q (with f_a included), with the restriction that the new q belongs to the same family of previous approximation. For example, this can be done by minimising the following KL divergence:

$$f_a(\boldsymbol{z}) \leftarrow \arg\min_{f_a} \mathrm{KL}[\tilde{p}_a(\boldsymbol{z}) || q(\boldsymbol{z})], \quad q(\boldsymbol{z}) \propto q_{-a}(\boldsymbol{z}) f_a(\boldsymbol{z}_a). \tag{98}$$

It is important to note that the above optimisation task is solved by optimising $f_a(\boldsymbol{z}_a)$ only, and the other factors in $q_{-a}(\boldsymbol{z})$ are all fixed. Therefore EP is regarded

[21] The name "cavity" comes from the *cavity method* that is used to study Ising models [Mézard et al., 1987], showing the deep connections between EP and belief propagation.

as a "local" approximation algorithm since at each step the update is performed locally on a selected factor.

For exponential family approximations, we solve the optimisation problem (98) by zeroing the gradients of the KL w.r.t. the natural parameters of q. This is equivalent to matching the moments of the arguments between the two distributions. More precisely, assume the factors f_a belong to the same exponential family with feature function $\boldsymbol{\Phi}(\boldsymbol{z}) = (\Phi_1(\boldsymbol{z}), ..., \Phi_d(\boldsymbol{z}))$, i.e. $\boldsymbol{z}_a = \boldsymbol{z}$ and $f_a(\boldsymbol{z}) = \exp(\boldsymbol{\lambda}_a^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z}))$. Then we write the q distribution as

$$q(\boldsymbol{z}) \propto \exp(\boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z})), \quad \boldsymbol{\lambda} = \sum_a \boldsymbol{\lambda}_a. \tag{99}$$

Zeroing the gradient of $\mathrm{KL}[\tilde{p}_a||q]$ wrt. $\boldsymbol{\lambda}_a$ returns

$$\mathbb{E}_q \left[\boldsymbol{\Phi}(\boldsymbol{z})\right] = \mathbb{E}_{\tilde{p}_a} \left[\boldsymbol{\Phi}(\boldsymbol{z})\right], \tag{100}$$

which gives the name of moment matching [Seeger, 2005]. We denote this optimisation computation with proj operator

$$\mathrm{proj}[p] = \arg\min_q \mathrm{KL}[p||q], \tag{101}$$

which returns the minimiser of the KL-divergence $\mathrm{KL}[\tilde{p}_a||q]$ by passing the moments of $\tilde{p}_a$. This operator is also called M-projection [Cover and Thomas, 1991]. After the computation of $q^* = \mathrm{proj}[\tilde{p}_a]$ we recover the update of f_a by

$$f_a(\boldsymbol{z}_a) \leftarrow q^*(\boldsymbol{z}) / q_{-a}(\boldsymbol{z}), \tag{102}$$

which returns in the exponential family case

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}_* - \boldsymbol{\lambda}_{-a}, \quad \boldsymbol{\lambda}_{-a} = \boldsymbol{\lambda} - \boldsymbol{\lambda}_a, \quad q^*(\boldsymbol{z}) \propto \exp(\boldsymbol{\lambda}_*^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z})) \tag{103}$$

In summary, EP updates the approximating factors $\tilde{f}_a$ iteratively through the "exclusion, moment matching, inclusion" procedure, which is detailed in Algorithm 1. To improve convergence damping updates can be applied to step 4 in Algorithm 1 as

$$f_a(\boldsymbol{z}_a) \leftarrow f_a(\boldsymbol{z}_a)^{1-\epsilon} f_a^*(\boldsymbol{z}_a)^{\epsilon}, \quad f_a^*(\boldsymbol{z}_a) = \mathrm{proj}[\tilde{p}_a] / q_{-a}(\boldsymbol{z}), \tag{104}$$

where ϵ denotes the step-size. The last step, corresponded to the inclusion step in Algorithm 1, is to incorporate the updated factor back to the approximation:

$$q(\boldsymbol{z}) \leftarrow q_{-a}(\boldsymbol{z}) f_a(\boldsymbol{z}_a). \tag{105}$$

The reader may find that in Algorithm 1 the updated distribution q equals to q^* in the moment matching step, while this yields EP without damping only.

Remark (misconceptions about EP)**.** A misleading interpretation states that EP is the counterpart algorithm of VI which minimises the inclusive KL. The correct answer is more than "yes or no" and it strongly depends on the factor

Algorithm 1 Expectation Propagation (without damping)

while not converged **do**
  choose a factor $f_a(\boldsymbol{z}_a)$ to refine (according to a schedule):
  exclusion: $q_{-a}(\boldsymbol{z}) \propto q(\boldsymbol{z})/f_a(\boldsymbol{z}_a)$
  moment matching: $f_a(\boldsymbol{z}_a) \leftarrow \text{proj}[q_{-a}(\boldsymbol{z})\tilde{f}_a(\boldsymbol{z}_a)]/q_{-a}(\boldsymbol{z})$
  inclusion: $q(\boldsymbol{z}) \leftarrow q_{-a}(\boldsymbol{z})f_a(\boldsymbol{z}_a)$
end while

graph structure. If the factor graph only contains a single factor node, then EP is minimising the inclusive KL globally. However, computing moments on this factor graph often requires further approximations, thus it is rarely considered in EP literature. Otherwise, EP does *not* minimise the inclusive KL divergence to the target distribution. Again we refer EP as a *local approximation* algorithm since it works with the components of the target distribution directly. Conversely, VI does minimise the exclusive KL divergence *globally*, regardless of the factor graph structure [Winn and Bishop, 2005].

6.2.1 A message passing view of EP

In this section we explain the EP update steps as message passing with projections. As an illustrative example, for now we assume the target distribution $\pi(\boldsymbol{z})$ as

$$\pi(\boldsymbol{z}) \propto \tilde{f}_a(\boldsymbol{z}_1, \boldsymbol{z}_2)\tilde{f}_b(\boldsymbol{z}_2, \boldsymbol{z}_3)\tilde{f}_c(\boldsymbol{z}_1), \tag{106}$$

which means $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3)$ and $\boldsymbol{z}_a = (\boldsymbol{z}_1, \boldsymbol{z}_2)$. Our goal is to approximate $\pi(\boldsymbol{z})$ with $q(\boldsymbol{z}) \propto f_a(\boldsymbol{z}_1, \boldsymbol{z}_2)f_b(\boldsymbol{z}_2, \boldsymbol{z}_3)f_c(\boldsymbol{z}_1)$ with simple factors f_a, f_b, f_c so that $q(\boldsymbol{z})$ becomes tractable. To simplify the intuition, we further assume the q distribution to follow an exponential form:

$$q(\boldsymbol{z}) = \frac{1}{Z}\exp\left(\sum_{i\in\{a,b,c\}} \boldsymbol{\lambda}_i^{\mathrm{T}}\Phi_i(\boldsymbol{z}_{N(f_i)})\right), \tag{107}$$

in other words $\boldsymbol{z}_{N(f_a)} = \boldsymbol{z}_a, f_a = \exp(\boldsymbol{\lambda}_a^{\mathrm{T}}\Phi_i(\boldsymbol{z}_a))$. The factor graph of $q(\boldsymbol{z})$ is visualised in Figure 14 (left). Now consider the EP steps for updating $f_a(\boldsymbol{z}_a)$. Note that we can also consider messages from a factor to *multiple* variable nodes, i.e. we can define the factor-to-node messages on the group level:

$$m_{f_i\to\boldsymbol{z}_{N(f_i)}}(\boldsymbol{z}_{N(f_i)}) := f_i(\boldsymbol{z}_{N(f_i)}). \tag{108}$$

Compared with (93), we see that there is no integration computation here since the variables $\boldsymbol{z}_{N(f_i)}$ in consideration contains all the variable nodes that the factor f_i is connected to. Then we have the cavity distribution as

$$q_{-a}(\boldsymbol{z}) \propto f_b(\boldsymbol{z}_2, \boldsymbol{z}_3)f_c(\boldsymbol{z}_1) = \prod_{i\neq a} m_{f_i\to\boldsymbol{z}_{N(f_i)}}(\boldsymbol{z}_{N(f_i)}). \tag{109}$$

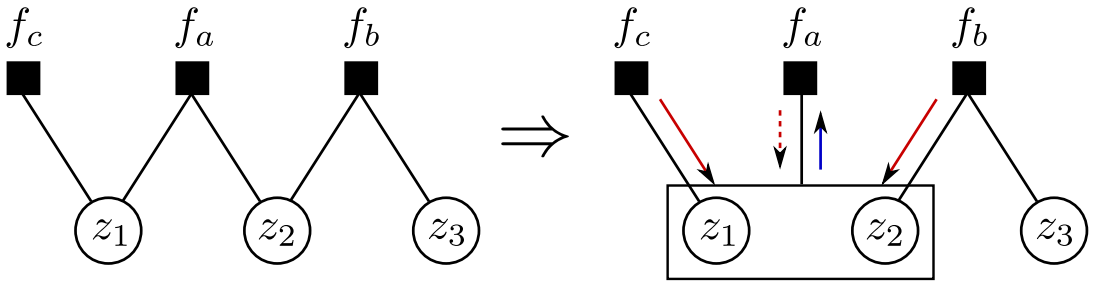

Figure 14: Visualising the message passing interpretation of EP, by merging $(\boldsymbol{z}_1, \boldsymbol{z}_2)$ as a "supernode" $\boldsymbol{z}_a$. Note here the message from f_a to $\boldsymbol{z}_a$ is highlighted with a dash line since approximation might make the message inexact.

Now we need to compute the message from $\boldsymbol{z}_a$ to f_a. In the moment matching step, this is done by solving the optimisation problem (98). Zeroing the gradient of the KL divergence w.r.t. $\boldsymbol{\lambda}_a$ (with $\tilde{p}_a(\boldsymbol{z})$ treated as a constant) returns the updated q distribution $q \to q^*$ such that

$$\mathbb{E}_{q^*}[\Phi_i(\boldsymbol{z}_a)] = \mathbb{E}_{\tilde{p}_a}[\Phi_i(\boldsymbol{z}_a)] \tag{110}$$

This means the updated factor $f_a(\boldsymbol{z}_a) \to f_a^*(\boldsymbol{z}_a)$ (i.e. $m_{f_a \to \boldsymbol{z}_a}(\boldsymbol{z}_a) \to m^*_{f_a \to \boldsymbol{z}_a}(\boldsymbol{z}_a)$ needs to satisfy:

$$\begin{aligned} &\text{Moments-of-}\boldsymbol{z}_a \left(q(\boldsymbol{z}) \propto m^*_{f_a \to \boldsymbol{z}_a}(\boldsymbol{z}_a) q_{-a}(\boldsymbol{z})\right) \\ =&\text{Moments-of-}\boldsymbol{z}_a \left(\tilde{p}_a(\boldsymbol{z}) \propto \tilde{f}_a(\boldsymbol{z}_a) q_{-a}(\boldsymbol{z})\right). \end{aligned} \tag{111}$$

Since we only match the moments of $\boldsymbol{z}_a$, this means we can first integrate out the variables $\boldsymbol{z}_{-a} = \boldsymbol{z} \backslash \boldsymbol{z}_a$. Since $\boldsymbol{z}_{-a}$ only appears in $q_{-a}(\boldsymbol{z})$, this means $\boldsymbol{z}_{-a}$ can be marginalised out already in the cavity distribution computation step: $q_{-a}(\boldsymbol{z}_a) = \int q_{-a}(\boldsymbol{z}) d\boldsymbol{z}_{-a}$. Compare the cavity marginal $q_{-a}(\boldsymbol{z}_a)$ to eq. (94), we see that the cavity distribution can be interpreted as a variable-to-factor message:

$$q_{-a}(\boldsymbol{z}_a) \propto m_{\boldsymbol{z}_a \to f_a}(\boldsymbol{z}_a) := \int \prod_{i \neq a} m_{f_i \to \boldsymbol{z}_{N(f_i)}}(\boldsymbol{z}_{N(f_i)}) d\boldsymbol{z}_{-a} = \prod_{i \neq a} m_{f_i \to \boldsymbol{z}_{N(a,i)}}(\boldsymbol{z}_{N(a,i)}), \tag{112}$$

with $\boldsymbol{z}_{N(a,i)} = \boldsymbol{z}_{N(f_i)} \cap \boldsymbol{z}_a$ denotes the variables in $\boldsymbol{z}_a$ that is also connected to factor f_i. Therefore the moment matching constraint becomes

$$\begin{aligned} &\text{Moments-of-}\boldsymbol{z}_a \left(q(\boldsymbol{z}_a) \propto m^*_{f_a \to \boldsymbol{z}_a}(\boldsymbol{z}_a) m_{\boldsymbol{z}_a \to f_a}(\boldsymbol{z}_a)\right) \\ =&\text{Moments-of-}\boldsymbol{z}_a \left(\tilde{p}_a(\boldsymbol{z}_a) \propto \tilde{f}_a(\boldsymbol{z}_a) m_{\boldsymbol{z}_a \to f_a}(\boldsymbol{z}_a)\right). \end{aligned} \tag{113}$$

In other words, the update rule of the new factor (or the new factor-to-variable message) is

$$f_a(\boldsymbol{z}_a) := m_{f_a \to \boldsymbol{z}_a}(\boldsymbol{z}_a), \quad m_{f_a \to \boldsymbol{z}_a}(\boldsymbol{z}_a) \leftarrow \frac{1}{Z' m_{\boldsymbol{z}_a \to f_a}(\boldsymbol{z}_a)} \text{proj}[\tilde{f}_a(\boldsymbol{z}_a) m_{\boldsymbol{z}_a \to f_a}(\boldsymbol{z}_a)], \tag{114}$$

Algorithm 2 Power EP with fraction α

while not converged **do**
  choose a factor $f_a(\boldsymbol{z}_a)$ to refine:
  exclusion: $q_{-a}(\boldsymbol{z}) = q(\boldsymbol{z})/f_a(\boldsymbol{z}_a)^\alpha$
  moment matching: $f_a(\boldsymbol{z}_a)^\alpha \leftarrow \text{proj}[q_{-a}(\boldsymbol{z})\tilde{f}_a(\boldsymbol{z}_a)^\alpha]/q_{-a}(\boldsymbol{z})$
  inclusion: $q(\boldsymbol{z}) \leftarrow q(\boldsymbol{z})f_a(\boldsymbol{z}_a)/f_a(\boldsymbol{z}_a)^{old}$
end while

with proj$[\cdot]$ the M-projection as discussed before (w.r.t. the moments of $\boldsymbol{z}_a$) and Z' a scaling constant.[22] This essentially means the two major steps in EP (see Algorithm 1 and Figure 14 (right)) can be summarised as

- Choose a factor f_a to refine;
- Exclusion step: compute the variable-to-factor message $m_{\boldsymbol{z}_a \to f_a}(\boldsymbol{z}_a)$;
- Moment Matching: compute/update the factor-to-variable message $m_{f_a \to \boldsymbol{z}_a}(\boldsymbol{z}_a)$.

One can see that if f_a and $\tilde{f}_a$ belongs to the same family (in our example $\tilde{f}_a$ will also have an exponential family form), then the moment-matching update will immediately set $f_a(\boldsymbol{z}_a) \leftarrow \tilde{f}_a(\boldsymbol{z}_a)$, the M-projection in (114) becomes an identity mapping, and the variable-to-factor message $m_{\boldsymbol{z}_a \to f_a}(\boldsymbol{z}_a)$ in both the numerator and denominator of (114) are cancelled out. This makes the newly updated message consistent with the definition of factor-to-multiple-variable message in (108). From this view we can see EP as a generalised sum-product algorithm where additional approximation steps are added to the summation steps of message computations.

6.2.2 Linking power EP and α-divergence

An extension of EP is power EP [Minka, 2004], which excludes a fraction of the approximation (i.e. $f_a(\boldsymbol{z})^\alpha$) and includes the same fraction of the true factor for updates (see Algorithm 2). Since we exclude/include only a fraction of the factors, damping should be applied to the natural parameters of $f_a(\boldsymbol{z})^\alpha$ directly before recovering the update for $f_a(\boldsymbol{z})$.

Power EP with fraction α corresponds to minimising the α-divergence from $\tilde{p}_a(\boldsymbol{z}) \propto q(\boldsymbol{z})\tilde{f}_a(\boldsymbol{z})/f_a(\boldsymbol{z})$ to $q(\boldsymbol{z})$. To see this, we first adapt Amari's definition [Amari, 1985] of α-divergence D_α^A to the following form [Zhu and Rohwer, 1995; Minka, 2005], which we refer to as Minka's α-divergence:

$$\mathrm{D}_\alpha^M[p||q] = \frac{1}{\alpha(1-\alpha)}\left(1 - \int p(\boldsymbol{z})^\alpha q(\boldsymbol{z})^{1-\alpha} d\boldsymbol{z}\right) \tag{115}$$

which is equivalent to the original definition by setting $\alpha' = 2\alpha - 1$ in Amari's notation $\mathrm{D}_{\alpha'}^A = \mathrm{D}_\alpha^M$. So similarly the exclusive KL divergence $\mathrm{KL}[q||p] = \lim_{\alpha \to 0} \mathrm{D}_\alpha^M[p||q]$

[22] In practical implementations the messages (from both directions) need not to be normalised, the normalisation constant comes in only when a density evaluation is required.

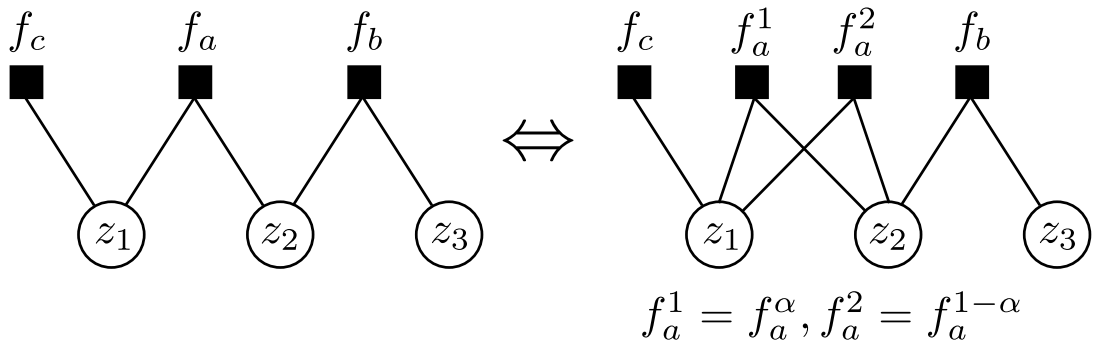

Figure 15: Two equivalent factor graphs for the same distribution, with the RHS factor graph used in power EP (with damping).

and the inclusive one $\mathrm{KL}[p||q] = \lim_{\alpha\to 1} \mathrm{D}_\alpha^M[p||q]$ can be recovered. Now we investigate the case which replaces the moment matching step in the EP algorithm (Algorithm 1) by $q^* = \arg\min D_\alpha^M[\tilde{p}_a||q]$, which is also called α-projection [Amari and Nagaoka, 2000]. Similarly we assume q has an exponential family form (99), and we fix $\tilde{p}_a(\boldsymbol{z})$ as the target (so it is treated as a constant). Then when $\alpha \neq 0, 1$,

$$\nabla_{\boldsymbol{\lambda}} D_\alpha^M[\tilde{p}_a||q] = \frac{1}{\alpha}\left(\mathbb{E}_q[\boldsymbol{\Phi}(\boldsymbol{z})] - \mathbb{E}_{\tilde{p}_a^{(\alpha)}}[\boldsymbol{\Phi}(\boldsymbol{z})]\right),$$
$$\tilde{p}_a^{(\alpha)} \propto \tilde{p}_a^\alpha q^{1-\alpha}.$$

In this case the solution of $\nabla_{\boldsymbol{\lambda}} D_\alpha^M[\tilde{p}_a||q]$ is non-trivial since now $\tilde{p}_a^{(\alpha)}$ contains contributions from q. Instead power EP proposes a fixed point iterative update procedure, by initialising q with last iteration's solution (which makes $\tilde{p}_a^{(\alpha)} \propto q\tilde{f}_a^\alpha / f_a^\alpha$, see the moment matching step in Algorithm 2), fixing $\tilde{p}_a^{(\alpha)}$ as the target, and computing the next update for the natural parameter $\boldsymbol{\lambda}$ such that the moments of q and $\tilde{p}_a^{(\alpha)}$ are matched. Again there is no guarantee for convergence here, but if power EP does converge, this means the fixed point iterative updates are also converged, thus the final solution q does minimises Minka's α-divergence (or Amari's α-divergence but with a different α value) locally. Minka [2004] also sketched an algorithm called α-EP which assumes the α-projection is tractable.

As discussed in § 4.1, different α-divergences show different characteristics. Often the exclusive KL-divergence $\mathrm{KL}[q||p]$ is prefered if a mode-seeking solution is preferred. Furthermore, the exclusive KL-divergence has an unique advantage in that local optimisation can return optima of global approximation [Winn and Bishop, 2005]. On the other hand, standard EP with the inclusive KL-divergence $\mathrm{KL}[p||q]$ is preferred if the inference task require computing some sufficient statistics locally at $\boldsymbol{z}_a$.

We can also view power EP with power α and a specific damping rate as running EP on an equivalent factor graph of the target distribution. We can write

the target distribution as

$$\pi(\boldsymbol{z}) \propto \prod_{i \neq a} \tilde{f}_i(\boldsymbol{z}_{N(f_i)}) \tilde{f}_a^1(\boldsymbol{z}_a) \tilde{f}_a^2(\boldsymbol{z}_a), \quad \tilde{f}_a^1(\boldsymbol{z}_a) := \tilde{f}_a(\boldsymbol{z}_a)^\alpha, \tilde{f}_a^2(\boldsymbol{z}_a) := \tilde{f}_a(\boldsymbol{z}_a)^{1-\alpha}. \tag{116}$$

Similarly we also split the $f_a(\boldsymbol{z}_a)$ factor in $q(\boldsymbol{z})$ as $f_a(\boldsymbol{z}_a) = f_a^1(\boldsymbol{z}_a) f_a^2(\boldsymbol{z}_a)$, $f_a^1(\boldsymbol{z}_a) := f_a(\boldsymbol{z}_a)^\alpha$, $f_a^2(\boldsymbol{z}_a) := f_a(\boldsymbol{z}_a)^{1-\alpha}$. Now we can run the EP procedure to update $f_a^1(\boldsymbol{z}_a)$, one can show that $\tilde{p}_a^{(\alpha)} \propto q \tilde{f}_a^1 / f_a^1$ and the new update becomes

$$f_a^1(\boldsymbol{z}_a) \leftarrow \text{proj}[\tilde{p}_a^{(\alpha)}] / q_{-a1}(\boldsymbol{z}), \quad q_{-a1}(\boldsymbol{z}) := q_{-a}(\boldsymbol{z}) \propto q(\boldsymbol{z}) / f_a^1(\boldsymbol{z}_a), \tag{117}$$

which corresponds to the moment matching step in Algorithm 2 (if we treat the update of $f_a^\alpha(\boldsymbol{z}_a)$ as the update of $f_a^1(\boldsymbol{z}_a)$). Now instead of directly running the inclusion step in 2, we can also follow similar damping strategy as (104) to define the update for $f_a(\boldsymbol{z}_a)$:

$$f_a(\boldsymbol{z}_a) \leftarrow f_a^{1,new}(\boldsymbol{z}_a) f_a^2(\boldsymbol{z}_a) \Leftrightarrow f_a(\boldsymbol{z}_a) \leftarrow f_a(\boldsymbol{z}_a)^{1-\alpha} \text{proj}[\tilde{p}_a^{(\alpha)}] / q_{-a1}(\boldsymbol{z}). \tag{118}$$

Compare with the moment matching and inclusive steps in Algorithm 2 we see that power EP with power α and damping rate $\epsilon = \alpha$ can be viewed as running EP on an alternative factor graph (Figure 15).

6.3 Bethe free-energy

6.3.1 From variational free-energy to Bethe free-energy

This section derives the constrained Bethe free-energy optimisation problem by a relaxation of variational inference. In this case, we write the target distribution as $\pi(\boldsymbol{z}) \propto \prod_{f_a \in \mathcal{F}} f_a(\boldsymbol{z}_a)$, and define the variational free-energy as

$$\mathcal{F}_{\text{VFE}}(q) := \int q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{\pi^*(\boldsymbol{z})} d\boldsymbol{z} = -\mathbb{H}[q] - \sum_{f_a \in \mathcal{F}} \mathbb{E}_q[\log f_a(\boldsymbol{z}_a)], \tag{119}$$

with $\mathbb{H}[q]$ the entropy of q. First we make use of the additivity of logarithm to rewrite an equivalent energy function using some auxiliary functions $\{g_a(\boldsymbol{z}_a)\}$:

$$\begin{aligned} \mathcal{F}_{\text{VFE}}(q) &= -\mathbb{H}[q] + \sum_{f_a \in \mathcal{F}} \mathbb{E}_q \left[\log \frac{g_a(\boldsymbol{z}_a)}{f_a(\boldsymbol{z}_a)} + \log g_a(\boldsymbol{z}_a) \right] \\ &= \sum_{f_a \in \mathcal{F}} \mathbb{E}_q \left[\log \frac{g_a(\boldsymbol{z}_a)}{f_a(\boldsymbol{z}_a)} \right] - \mathbb{H}[q] - \sum_{f_a \in \mathcal{F}} \mathbb{E}_q \left[\log g_a(\boldsymbol{z}_a) \right]. \end{aligned} \tag{120}$$

For different factor graphs one would choose different auxiliary functions. We provide two examples here for the UGM and posterior distribution defined in § 6.1.1.

Bethe free-energy: the UGM case We consider mean-field approximation in this case, i.e. $q(\boldsymbol{z}) = \prod_{i\in\mathcal{V}} q(\boldsymbol{z}_i)$. Then we define

$$g_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j) = q(\boldsymbol{z}_i)q(\boldsymbol{z}_j), \quad g_i(\boldsymbol{z}_i) = q(\boldsymbol{z}_i)^{1-n_i}.$$

It can be shown that eq. (120) reduces to

$$\mathcal{F}_{\text{VFE}}(q) = \sum_{(i,j)\in\mathcal{E}} \mathbb{E}_{q(\boldsymbol{z}_i)q(\boldsymbol{z}_j)}\left[\log \frac{q(\boldsymbol{z}_i)q(\boldsymbol{z}_j)}{f_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)}\right] + \sum_{i\in\mathcal{V}}(1-n_i)\mathbb{E}_{q(\boldsymbol{z}_i)}\left[\log \frac{q(\boldsymbol{z}_i)}{f_i(\boldsymbol{z}_i)}\right]. \quad (121)$$

This means mean-field variational inference on a UGM is equivalent to minimise the energy (120) w.r.t. $q(\boldsymbol{z}) = \prod_{i\in\mathcal{V}} q(\boldsymbol{z}_i)$. Now consider the following equivalent minimisation problem:

$$\begin{aligned} &\arg\min_{\{b_i\},\{b_{ij}\}} \sum_{(i,j)\in\mathcal{E}} \mathbb{E}_{b_{ij}(\boldsymbol{z}_i,\boldsymbol{z}_j)}\left[\log \frac{b_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)}{f_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)}\right] + \sum_{i\in\mathcal{V}}(1-n_i)\mathbb{E}_{b_i(\boldsymbol{z}_i)}\left[\log \frac{b_i(\boldsymbol{z}_i)}{f_i(\boldsymbol{z}_i)}\right], \\ &\text{subject to } b_i(\boldsymbol{z}_i) = q(\boldsymbol{z}_i), \forall i \in \mathcal{V}, \quad b_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j) = b_i(\boldsymbol{z}_i)b_j(\boldsymbol{z}_j), \forall (i,j) \in \mathcal{E}, \end{aligned} \quad (122)$$

which has the same solution as the variational inference problem (121). Relaxing the constraints to marginal matching returns the *constrained Bethe free-energy problem* [Bethe, 1935; Yedidia et al., 2001]:

$$\begin{aligned} &\arg\min_{\{b_i\},\{b_{ij}\}} \mathcal{F}_{\text{Bethe}}(\{b_i\}, \{b_{ij}\}) \\ &\text{subject to } \int b_i(\boldsymbol{z}_i)d\boldsymbol{z}_i = 1, \forall i \in \mathcal{V}, \int b_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)d\boldsymbol{z}_j = b_i(\boldsymbol{z}_i), \\ &\int b_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)d\boldsymbol{z}_i = b_j(\boldsymbol{z}_j), \forall (i,j) \in \mathcal{E}, \\ \mathcal{F}_{\text{Bethe}}(\{b_i\}, \{b_{ij}\}) &:= \sum_{(i,j)\in\mathcal{E}} \mathbb{E}_{b_{ij}(\boldsymbol{z}_i,\boldsymbol{z}_j)}\left[\log \frac{b_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)}{f_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)}\right] + \sum_{i\in\mathcal{V}}(1-n_i)\mathbb{E}_{b_i(\boldsymbol{z}_i)}\left[\log \frac{b_i(\boldsymbol{z}_i)}{f_i(\boldsymbol{z}_i)}\right]. \end{aligned} \quad (123)$$

Bethe free-energy: posterior approximation case Recall that $f_0(\boldsymbol{z}) = p_0(\boldsymbol{z})$ and $f_n(\boldsymbol{z}) = p(\boldsymbol{x}_n|\boldsymbol{z})$, also we assume the prior $p_0(\boldsymbol{z})$ is of simple form so that $p_0(\boldsymbol{z}) \in \mathcal{Q}$. Consider the following auxiliary functions

$$g_0(\boldsymbol{z}) = 1, \quad g_n(\boldsymbol{z}) = \frac{q(\boldsymbol{z})}{p_0(\boldsymbol{z})}, n = 1, ..., N.$$

It can be shown that eq. (120) reduces to

$$\mathcal{F}_{\text{VFE}}(q) = (1-N)\text{KL}[q||p_0] + \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{z})}\left[\frac{q(\boldsymbol{z})}{p(\boldsymbol{z})f_n(\boldsymbol{z})}\right] \quad (124)$$

Similarly we then consider the following equivalent minimisation problem:

$$\begin{aligned} &\min_{q,\{\tilde{p}_n\}} (1-N)\text{KL}[q||p_0] + \sum_{n=1}^{N} \mathbb{E}_{\tilde{p}_n(\boldsymbol{z})}\left[\frac{\tilde{p}_n(\boldsymbol{z})}{p(\boldsymbol{z})f_n(\boldsymbol{z})}\right], \\ &\text{subject to } \tilde{p}_n(\boldsymbol{z}) = q(\boldsymbol{z}), n = 1, ..., N. \end{aligned} \quad (125)$$

The constraints can be *relaxed* to matching all the moments $\mathbb{E}_{\tilde{p}_n}[\boldsymbol{z}^k] = \mathbb{E}_q[\boldsymbol{z}^k]$ for $k \in \mathbb{N}$,[23] and a further crude relaxation suggests moment matching just for the first K moments[24] $\mathbb{E}_{\tilde{p}_n}[\boldsymbol{z}^k] = \mathbb{E}_q[\boldsymbol{z}^k], k = 1, 2, ..., K$. In the following we use a vectorial function $\boldsymbol{\Phi}(\boldsymbol{z})$ to summarise these constraints as $\mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}] = \mathbb{E}_q[\boldsymbol{\Phi}]$, where as an example for Gaussian EP: $\boldsymbol{\Phi}(\boldsymbol{z}) = [\boldsymbol{z}, \boldsymbol{z}\boldsymbol{z}^T]$. In general $\boldsymbol{\Phi}$ can contain any polynomial terms or other basis functions. This relaxation returns the following *constrained Bethe free-energy (or EP energy) optimisation* problem:

$$\begin{aligned} &\min_{q, \{\tilde{p}_n\}} \mathcal{F}_{\text{Bethe}}(q, \{\tilde{p}_n\}) \quad \text{subject to } \mathbb{E}_{q(\boldsymbol{z})}[\boldsymbol{\Phi}(\boldsymbol{z})] = \mathbb{E}_{\tilde{p}_n(\boldsymbol{z})}[\boldsymbol{\Phi}(\boldsymbol{z})] \\ &\mathcal{F}_{\text{Bethe}}(q, \{\tilde{p}_n\}) := (1 - N)\text{KL}[q||p_0] + \sum_{n=1}^{N} \mathbb{E}_{\tilde{p}_n(\boldsymbol{z})}\left[\frac{\tilde{p}_n(\boldsymbol{z})}{p(\boldsymbol{z})f_n(\boldsymbol{z})}\right]. \end{aligned} \tag{126}$$

Remark (Another way to derive the constrained Bethe free-energy optimisation for UGMs)**.** The variational lower-bound (119) assumes $q(\boldsymbol{z})$ is in the *marginal polytope*:

$$\mathbb{M}(\mathcal{G}) = \{\{q_i(\boldsymbol{z}_i), q_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)\} | \ \exists q(\boldsymbol{z}) \text{ s.t. } q_i(\boldsymbol{z}_i) = \int q(\boldsymbol{z}) d\boldsymbol{z}_{-i}, \forall i \in \mathcal{V}, \\ q_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \int q(\boldsymbol{z}) d\boldsymbol{z}_{-ij}, \forall (i,j) \in \mathcal{E}\}.$$

Essentially this means the marginal distributions $q_i(\boldsymbol{z}_i)$ and the pairwise joint distributions $q_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ is *globally consistent*, in the sense that there exists a joint distribution $q(\boldsymbol{z})$ with its marginals as q_i and q_{ij}.
We make two approximations/relaxations to derive the constrained Bethe free-energy optimisation problem. First since the entropy is computed on the whole graph which can be intractable, the first approximation applies to the entropy term and break it down into entropies on marginal distributions:

$$\mathbb{H}[q(\boldsymbol{z})] \approx \mathbb{H}_{\text{Bethe}}[\{q_i, q_{ij}\}] = \sum_{i \in \mathcal{V}} \mathbb{H}[q_i(\boldsymbol{z}_i)] - \sum_{(i,j) \in \mathcal{E}} \mathbb{I}[q_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)],$$

where $\mathbb{I}[q(\boldsymbol{z}_i, \boldsymbol{z}_j)]$ denotes the mutual information between $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ under distribution $q(\boldsymbol{z}_i, \boldsymbol{z}_j)$. Intuitively this approximation replaces the joint entropy by a sum of marginal entropy, and it counters for the correlations between variables by subtracting mutual information between neighbouring nodes. The second step is to relax the candidate set from marginal polytope to *local polytope*:

$$\mathbb{L}(\mathcal{G}) = \{\{q_i(\boldsymbol{z}_i), q_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)\} | \int q_i(\boldsymbol{z}_i) d\boldsymbol{z}_i = 1, \forall i \in \mathcal{V}, \\ \int q_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j) d\boldsymbol{z}_i = q_i(\boldsymbol{z}_i), \int q_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j) d\boldsymbol{z}_i = q_i(\boldsymbol{z}_i), \forall (i,j) \in \mathcal{E}\}.$$

[23] Having the same moments for p and q does not imply having the same *moment generating function*.

[24] The zeroth moment matching constraint is replaced by the constraint that $\tilde{p}_n$ integrates to 1.

This means instead of enforcing global consistency, here it only requires *local consistency* in the sense that the pairwise and singleton marginals are consistent. Also it is straightforward to see $\mathbb{M}(\mathcal{G}) \subset \mathbb{L}(\mathcal{G})$ as global consistency naturally implies local consistency. Combining the two approximations together we obtain the constrained Bethe free-energy optimisation problem as

$$\min_{\{b_i, b_{ij}\} \in \mathbb{L}(\mathcal{G})} -\mathbb{H}_{\text{Bethe}}[\{b_i, b_{ij}\}] - \sum_{(i,j)\in\mathcal{E}} \mathbb{E}_{b_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)} \left[\log f_{ij}(\boldsymbol{z}_i, \boldsymbol{z}_j)\right] - \sum_{i\in\mathcal{V}} (1 - n_i) \mathbb{E}_{b_i(\boldsymbol{z}_i)} \left[\log f_i(\boldsymbol{z}_i)\right],$$

and readers can verify that it is the same optimisation problem as (123). It can be shown that $\mathbb{M}(\mathcal{G}) = \mathbb{L}(\mathcal{G})$ when $\mathcal{G}$ is a tree, and the optimum of this constrained Bethe free-energy optimisation problem is exactly the log partition function $\log Z$ [Wainwright and Jordan, 2008; Yedidia et al., 2001].

6.3.2 Message passing: dual form optimisation of Bethe free-energy

In this section we briefly show that the constrained Bethe free-energy optimisation problem can be solved by a message passing algorithm. In below I only include a derivation for the posterior approximation case, which leads to the expectation propagation (EP) [Minka, 2001b] algorithm. Similar derivations for the UGM case result in belief propagation (BP).

We provide a derivation in a similar way as Heskes [2002], starting from a note on the KL duality[25]

$$-\text{KL}[q||p_0] = \min_{\boldsymbol{\lambda}_q(\boldsymbol{z})} -\mathbb{E}_q[\lambda_q(\boldsymbol{z})] + \log \mathbb{E}_{p_0}\left[\exp[\lambda_q(\boldsymbol{z})]\right], \tag{127}$$

with $\lambda_q(\boldsymbol{z})$ a function to be specified later on. This duality is in the same spirit as deriving convex conjugate function for the log partition function for an exponential family [see e.g. Wainwright and Jordan, 2008], if viewing $p_0(\boldsymbol{z})$ as the base measure. The equality is achieved by $q(\boldsymbol{z}) \propto p_0(\boldsymbol{z}) \exp[\lambda_q(\boldsymbol{z})]$. Substitution into (126) then yields a transformed energy that we denoted as $\mathcal{F}_{\text{Bethe}}(q, \{\tilde{p}_n\}, \lambda_q(\boldsymbol{z}))$.

$$\mathcal{F}_{\text{Bethe}}(q, \{\tilde{p}_n\}, \lambda_q(\boldsymbol{z})) = (1 - N)\mathbb{E}_q[\lambda_q(\boldsymbol{z})] + (N - 1) \log \mathbb{E}_{p_0}\left[\exp[\lambda_q(\boldsymbol{z})]\right] - \sum_n \mathbb{E}_{\tilde{p}_n}\left[\log \frac{p_0(\boldsymbol{z}) f_n(\boldsymbol{z})}{\tilde{p}_n(\boldsymbol{z})}\right]. \tag{128}$$

Denote $\boldsymbol{\lambda}_{-n}$ as the Lagrange multiplier for moment matching and ν, ν_n for the normalisation constraints of q and $\tilde{p}_n$, respectively. This returns the following Lagrangian

$$\min_{q, \{\tilde{p}_n\}, \lambda_q(\boldsymbol{z})} \max_{\{\boldsymbol{\lambda}_{-n}, \nu_n, \nu\}} \mathcal{F}_{\text{Bethe}}(\{\tilde{p}_n\}, q, \lambda_q(\boldsymbol{z})) + \sum_n \boldsymbol{\lambda}_{-n}^{\mathrm{T}} (\mathbb{E}_q[\boldsymbol{\Phi}] - \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}]) + \sum_n \nu_n \left(\int \tilde{p}_n(\boldsymbol{z}) d\boldsymbol{z} - 1\right) + \nu \left(\int q(\boldsymbol{z}) d\boldsymbol{z} - 1\right). \tag{129}$$

[25] We include this step in order to connect to the EP energy with optimisation arguments all in the dual space.

Solving the fixed points for $\tilde{p}_n$ and ν_n returns

$$\tilde{p}_n(\boldsymbol{z}) = \frac{1}{Z_n} p_0(\boldsymbol{z}) f_n(\boldsymbol{z}) \exp\left[\boldsymbol{\lambda}_{-n}^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z})\right],$$

where the normalising constant is

$$Z_n = \int p_0(\boldsymbol{z}) f_n(\boldsymbol{z}) \exp\left[\boldsymbol{\lambda}_{-n}^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z})\right] d\boldsymbol{z}.$$

Also it is straight-forward to evaluate the fixed point condition for q:

$$(N-1)\lambda_q(\boldsymbol{z}) = \sum_n \boldsymbol{\lambda}_{-n}^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z}) + \nu.$$

We explicitly specify $\lambda_q(\boldsymbol{z}) = \boldsymbol{\lambda}_q^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z}) + \nu$ w.l.o.g., also the constant ν can be dropped since exponential family distributions are translation invariant to constants. Importantly, substituting $\tilde{p}_n$ back to (129) and enforcing the fixed point condition for q yields the *EP energy* [Minka, 2001a]:

$$\begin{aligned} \min_{\boldsymbol{\lambda}_q} \max_{\{\boldsymbol{\lambda}_{-n}\}} \mathcal{F}_{\mathrm{EP}}(\boldsymbol{\lambda}_q, \{\boldsymbol{\lambda}_{-n}\}) = (N-1) \log \mathbb{E}_{p_0}\left[\exp[\boldsymbol{\lambda}_q^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z})]\right] - \sum_n \log Z_n, \\ \text{subject to } (N-1)\boldsymbol{\lambda}_q = \sum_n \boldsymbol{\lambda}_{-n}. \end{aligned} \tag{130}$$

Notice now the optimisation problem over q is dropped since (130) does not depend on it. To obtain the approximate posterior back, we make use of the tightness of the KL duality, and define

$$q(\boldsymbol{z}) = \frac{1}{Z_q} p_0(\boldsymbol{z}) \exp\left[\boldsymbol{\lambda}_q^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z})\right], \quad \log Z_q = \log \mathbb{E}_{p_0}\left[\exp[\boldsymbol{\lambda}_q^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z})]\right].$$

The expectation consistent approximate inference (EC) algorithm [Opper and Winther, 2005] is a special case with $p_0(\boldsymbol{z}) \propto 1$ and $N = 2$.

EP [Minka, 2001b] proposes parametrising the (natural parameters of) local approximating factors to remove the constraints in problem (130). To be concrete, EP defines

$$\boldsymbol{\lambda}_n = \boldsymbol{\lambda}_q - \boldsymbol{\lambda}_{-n}, \quad \Rightarrow \quad \sum_n \boldsymbol{\lambda}_n = N\boldsymbol{\lambda}_q - \sum_n \boldsymbol{\lambda}_{-n} = \boldsymbol{\lambda}_q. \tag{131}$$

Then EP runs a fixed point iteration algorithm to find the stationary points of the following objective for $\{\boldsymbol{\lambda}_n\}_{n=1}^N$:

$$\mathcal{F}_{\mathrm{EP}}(\{\boldsymbol{\lambda}_n\}) = (N-1) \log \mathbb{E}_{p_0}\left[\exp[\boldsymbol{\lambda}_q^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z})]\right] - \sum_n \log \mathbb{E}_{p_0}\left[\exp[(\boldsymbol{\lambda}_q - \boldsymbol{\lambda}_n)^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{z})] f_n(\boldsymbol{z})\right]. \tag{132}$$

Simple calculus shows that the gradient of (132) w.r.t. the local parameter $\boldsymbol{\lambda}_n$ is

$$\nabla_{\boldsymbol{\lambda}_n} \mathcal{F}_{\mathrm{EP}} = (N-1)\mathbb{E}_q[\boldsymbol{\Phi}(\boldsymbol{z})] - \sum_{m \neq n} \mathbb{E}_{\tilde{p}_m}[\boldsymbol{\Phi}(\boldsymbol{z})]. \tag{133}$$

Zeroing the above gradient for all $\boldsymbol{\lambda}_n$ results in the fixed point condition

$$\mathbb{E}_q[\boldsymbol{\Phi}(\boldsymbol{z})] = \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}(\boldsymbol{z})], \quad \forall n,$$

which gives the moment matching update in EP.

6.4 Further reading

Wainwright and Jordan [2008] is a comprehensive tutorial-style monograph that covers the basics and advances in message passing. Apart from the BP algorithm which corresponds to finding fixed points of the Bethe free energy, other approaches such as the tree-based approximations [Wainwright et al., 2002, 2005] and the junction tree algorithm (CITATION). Jin et al. [2018] applied the junction tree algorithm to perform inference on a graph-based VAE, with the end application aiming at molecule graph generation.

The EP algorithm [Minka, 2001b] minimises the KL divergence between the q distribution and the tilted distribution locally. Other divergences can be employed here, e.g. α-divergences [Minka, 2004]. Similar idea has been studied in BP context, e.g. see [Wiegerinck and Heskes, 2003]. I would recommend Minka [2005] for disucssion on divergences, connections between EP and BP, and generalisations of message passing.

Although EP is often more accurate, one wrinkle that hindered its wide application in large-scale machine learning is the $\mathcal{O}(N)$ memory requirement if the factor graph contains N factors, since it needs to store every approximating factor. This issue is particularly severe for large-scale data since with i.i.d. data assumption, having observed N datapoints means a datapoint level factor graph construction of the exact posterior contains at least N factors. There are two lines of solutions available in the literature. First one can group the datapoints into clusters and construct the factor graph on the cluster level (rather than on datapoint level), and the moment matching steps are approximated by MCMC [Gelman et al., 2014; Barthelmé and Chopin, 2014; Hasenclever et al., 2017]. The stochastic EP algorithm [Li et al., 2015; Dehaene and Barthelmé, 2015; Dehaene and Barthelmé, 2018], as a representative of the second line of solutions, addressed the memory challenge by sharing the approximating factor, by doing so the memory requirement reduced to $\mathcal{O}(1)$ which is independent with the number of factors in the factor graph. The black-box alpha approach [Hernández-Lobato et al., 2016] uses the similar idea of factor sharing, but instead directly optimise the corresponding energy function with gradient-based approaches, which is different from stochastic EP that is based on fixed-point iterative algorithms.

Since 2016, graph neural networks has become a hot topic of research, with many applications such as molecule design, social network analysis and program synthesis. See e.g. Zhou et al. [2018] for a survey. The message passing algorithm applied to GNNs is deeply connected to the message passing algorithms in probabilistic models. In fact, for probabilistic graphical models, earlier work [Song et al., 2011; Dai et al., 2016b] has explored message passing algorithms that are performed on the embedding space of the local beliefs. The convergence properties of GNNs are largely unknown, so perhaps the theoretical analysis of BP on the probabilistic graphical model side (e.g. see Ihler et al. [2005] and related papers) can provide potential inspirations on better understandings of GNNs.

Part II
Approximate distribution design

7 Invertible transformations and normalising flows

Gaussian distributions might not be flexible enough to form an accurate approximation to the target distribution. Here we introduce the idea of *normalising flows* which uses *invertible transformations* to transform a simple distribution (like a Gaussian) to a flexible one.

7.1 Change of random variable under invertible transformations

Now consider adding the invertibility constraint to the non-linear mapping $\mathbf{f}$ which acts on the latent variables $\boldsymbol{z} \in \mathbb{R}^d$. This means:

$$\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d, \quad \exists \boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^d \quad \text{s.t.} \quad \boldsymbol{g} \circ \mathbf{f}(\boldsymbol{z}) = \mathbf{f} \circ \boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{z}.$$

In the following we also write $\mathbf{f}^{-1} = \boldsymbol{g}$, and assume the mapping is smooth, i.e. $\nabla_{\boldsymbol{z}} \mathbf{f}(\boldsymbol{z})$ exists everywhere. Therefore, given a distribution $q(\boldsymbol{z})$ and by temporarily denoting $\mathbf{y} = \mathbf{f}(\boldsymbol{z})$, we have

$$q(\mathbf{y}) = q(\boldsymbol{z}) |\det(\nabla_{\boldsymbol{z}} \mathbf{f}(\boldsymbol{z}))|^{-1}. \tag{134}$$

Then when density evaluation is required on a given $\mathbf{y}$, one would first compute the inverse mapping $\boldsymbol{z} = \mathbf{f}^{-1}(\mathbf{y})$, compute the density value $q(\boldsymbol{z})$, and obtain the determinant of the Jacobian $\det(\nabla_{\boldsymbol{z}} \mathbf{f}(\boldsymbol{z}))$.

Why? Recall that the probability density function $q(\boldsymbol{z})$ is defined given the probability measure $Q_{\boldsymbol{z}}(\boldsymbol{z})$ and a *reference measure* $d\mu(d\boldsymbol{z})$. Usually we use the *Lebesgue measure* $\mu(d\boldsymbol{z}) = \lambda(d\boldsymbol{z})$ as the reference measure,[26] and the following proposition can be found in any measure theory book:

Proposition 3. *If a linear mapping* $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ *is one-to-one, then*

$$\lambda(d\mathbf{y}) = \lambda(\mathbf{f}(d\boldsymbol{z})) = |det(\nabla_{\boldsymbol{z}} \mathbf{f}(\boldsymbol{z}))| \lambda(d\boldsymbol{z}).$$

The idea of this proposition is visualised in Figure 16. The idea of the determinant term is to represent volume changes, by comparing the best hypercube approximation of the twisted region in $\boldsymbol{y}$ space to the original $d\boldsymbol{z}$ region. This proposition can be extended to invertible mappings in general. Now for a measurable set $A \subset \mathbb{R}^d$, an invertible mapping does not change the *probability measure*:

$$Q_{\boldsymbol{z}}(A) = Q_{\mathbf{y}}(\mathbf{f}(A)).$$

Recall the definition of the probability density function. This means

$$q(\mathbf{y}) \lambda(d\mathbf{y}) = Q_{\mathbf{y}}(d\mathbf{y}) = Q_{\mathbf{y}}(\mathbf{f}(d\boldsymbol{z})) = Q_{\boldsymbol{z}}(d\boldsymbol{z}) = q(\boldsymbol{z}) \lambda(d\boldsymbol{z}).$$

Combined with the proposition, we have the density $q(\mathbf{y})$ defined as in (134).

[26] Usually we also write the Lebesgue measure as $d\boldsymbol{z} := \lambda(d\boldsymbol{z})$.

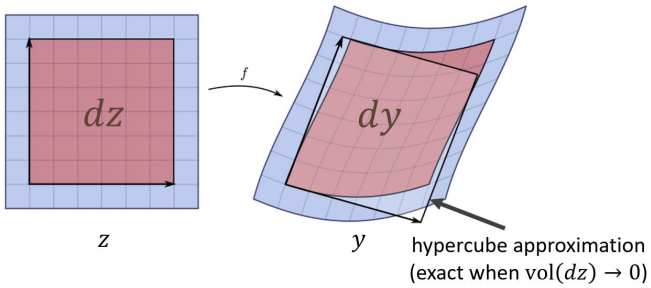

Figure 16: Visualising the idea of the change-of-variable rule.

7.2 Defining normalising flows

We can further construct a highly non-linear invertible mapping by composing invertible transforms, i.e.

$$\boldsymbol{z}_T = \mathbf{f}_T \circ \mathbf{f}_{T-1} \circ \cdots \circ \mathbf{f}_1(\boldsymbol{z}_0),$$

which induces a random variable $\boldsymbol{z}_T$ that is deterministic given $\boldsymbol{z}_0$:

$$q(\boldsymbol{z}_T) = q(\boldsymbol{z}_0) \prod_{t=1}^{T} |\det(\nabla_{\boldsymbol{z}_{t-1}} \mathbf{f}_t(\boldsymbol{z}_{t-1}))|^{-1}. \tag{135}$$

This type of distributions is named *normalising flow* distributions, and in principle one can construct *arbitrarily* complex distributions by increasing T or having expressive mappings $\mathbf{f}_t$. However, in practice the representation power of these distributions is largely restrictive due to the *constraints introduced by algorithms to fit them*. We provide two application scenarios to explain why.

- Maximum likelihood training for generative models.
  Consider a generative model defined as the following:

  $$\boldsymbol{z}_0 \sim p(\boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I}), \quad \boldsymbol{x} = \boldsymbol{z}_T = \mathbf{f}_T \circ \mathbf{f}_{T-1} \circ \cdots \circ \mathbf{f}_1(\boldsymbol{z}_0),$$

  in which the parameters of the mappings $\mathbf{f}_t$ are collected into $\boldsymbol{\theta}$. Then given a dataset $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ a maximum likelihood estimate of $\boldsymbol{\theta}$ requires solving the following optimisation problem:

  $$\boldsymbol{\theta}^{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \quad \sum_{n=1}^{N} \{\log p(\boldsymbol{z}_0^n) - \sum_{t=1}^{T} \log |\det(\nabla_{\boldsymbol{z}_{t-1}^n} \mathbf{f}_t(\boldsymbol{z}_{t-1}^n))|\}|_{\boldsymbol{z}_T^n = \boldsymbol{x}_n}. \tag{136}$$

  Although now the density $\log p(\boldsymbol{x})$ is tractable thanks to the invertible transform rules, for high dimensional observations $\boldsymbol{x}$ the above optimisation task can still be very challenging. First computing the determinant of the Jacobian $\det(\nabla_{\boldsymbol{z}_{t-1}^n} \mathbf{f}_t(\boldsymbol{z}_{t-1}^n))$ normally requires $\mathcal{O}(d^3)$ time if the Jacobian matrix is dense, which can be very expensive for e.g. image data with hundreds of dimensions. Second, one would typically expect to use more invertible transformations when modelling high dimensional data, indicating that T also increases with d.

- Variational inference with normalising flows.
  One can also use a normalising flow to construct the approximate posterior distribution
  $$\boldsymbol{z}_0 \sim q(\boldsymbol{z}_0), \quad \boldsymbol{z}_T = \mathbf{f}_T \circ \mathbf{f}_{T-1} \circ \cdots \circ \mathbf{f}_1(\boldsymbol{z}_0),$$
  and fit the variational parameters $\boldsymbol{\phi}$ by maximising the variational lower-bound:
  $$\mathcal{L}_{VI}(q; \boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}_T)}[\log p(\boldsymbol{x}, \boldsymbol{z}_T)] - \mathbb{E}_{q(\boldsymbol{z}_T)}[\log q(\boldsymbol{z}_T)]. \tag{137}$$
  The reconstruction term can be easily calculated following the LOTUS rule:
  $$\mathbb{E}_{q(\boldsymbol{z}_T)}[\log p(\boldsymbol{x}, \boldsymbol{z}_T)] = \mathbb{E}_{q(\boldsymbol{z}_0)}[\log p(\boldsymbol{x}, \mathbf{f}_T \circ \mathbf{f}_{T-1} \circ \cdots \circ \mathbf{f}_1(\boldsymbol{z}_0))],$$
  however, the second entropy term still involves evaluating the determinant of the Jacobian matrices:
  $$\mathbb{H}[q(\boldsymbol{z}_T)] = \mathbb{E}_{q(\boldsymbol{z}_0)}\left[-\log q(\boldsymbol{z}_0^n) + \sum_{t=1}^{T} \log |\det(\nabla_{\boldsymbol{z}_{t-1}^n} \mathbf{f}_t(\boldsymbol{z}_{t-1}^n))|\right],$$
  which, for similar reasons as in the MLE example, can still be very expensive.

Importantly, in both examples, one needs to compute a sequence of Jacobian matrices as a sub-routine of the optimisation procedure. So to make them practical for real-world problems, researchers have designed a number of invertible mappings $\mathbf{f}$ whose Jacobian is diagonal/low-rank/triangular. Then the induced normalising flow distribution allows faster density evaluation that scales almost linearly to the dimension d.

7.2.1 Some examples of normalising flow

As discussed, the design of normalising flow requires a balance between computational cost and representation power. In below I provide some existing examples of normalising flow.

Element/group-wise mapping Denoting $z_t[i]$ as the i^{th} element of the vector $\boldsymbol{z}_t$. We can construct an *element-wise* invertible mapping
$$z_T[i] = f_T \circ f_{T-1} \circ \cdots \circ f_1(z_0[i]),$$
and obviously the Jacobian matrix is diagonal, and the determinant can be computed in linear time. This idea can also be generalised to group-wise mappings, which construct invertible transformations on a subset of the random variables. The resulting Jacobian matrix is block-diagonal whose determinant can still be evaluated in a fast way.

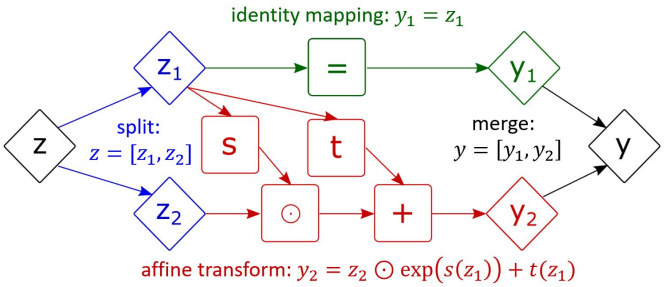

Figure 17: A visualisation of the computation in a RealNVP layer.

Linear time invertible transform Rezende and Mohamed [2015] proposed a linear time invertible mapping as the following

$$\mathbf{f}(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{z} + b),$$

where the free parameters are $\boldsymbol{u} \in \mathbb{R}^d$, $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, and $\sigma(\cdot)$ denotes a smooth invertible non-linearity. Then one can evaluate the log-det of the Jacobian in linear time as

$$\log|\det(\nabla_{\boldsymbol{z}}\mathbf{f}(\boldsymbol{z}))| = |1 + \boldsymbol{u}^{\mathrm{T}}\sigma'(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{z} + b)\boldsymbol{w}|.$$

NICE and realNVP: introducing coupling mappings Dinh et al. [2014] introduced an invertible mapping which is composed by transformations with auto-regressive structure. The algorithm starts from splitting the variables into disjoint sets $\boldsymbol{z} = [\boldsymbol{z}^1, \boldsymbol{z}^2]$, $\boldsymbol{z}^1 \in \mathbb{R}^e$, then define the transform:

$$\mathbf{y} = [\mathbf{y}^1, \mathbf{y}^2] = \mathbf{f}(\boldsymbol{z}) \quad \Leftrightarrow \quad \mathbf{y}^1 = \boldsymbol{z}^1, \mathbf{y}^2 = \boldsymbol{g}(\boldsymbol{z}^2; m(\boldsymbol{z}^1)),$$

where $m : \mathbb{R}^e \to \mathbb{R}^c$ (no need to be invertible) and $\boldsymbol{g} : \mathbb{R}^{d-e+c} \to \mathbb{R}^{d-e}$ is a "conditional" invertible mapping, i.e. one can compute $\boldsymbol{z}^2$ given $m(\boldsymbol{z}^1)$ and $\mathbf{y}^2$. See Figure 17 for a visualisation of the computation graph. For this mapping the Jacobian matrix and its determinant are

$$\nabla_{\boldsymbol{z}}\mathbf{f}(\boldsymbol{z}) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \nabla_{\boldsymbol{z}^1}\boldsymbol{g} & \nabla_{\boldsymbol{z}^2}\boldsymbol{g} \end{bmatrix}, \quad |\det(\nabla_{\boldsymbol{z}}\mathbf{f}(\boldsymbol{z}))|_{\mathbf{y}=\mathbf{f}(\boldsymbol{z})} = |\det(\nabla_{\boldsymbol{z}^2}\boldsymbol{g})|_{\boldsymbol{z}^2=\boldsymbol{g}^{-1}(\mathbf{y}^2;m(\mathbf{y}^1))}.$$

The authors named this type of mappings as *non-linear independent component estimation* (NICE). One can then stack the NICE mappings to construct a deep non-linear transformation, and in particular to allow non-linear behaviour of the output variables, define

$$\boldsymbol{x} = [\boldsymbol{x}^1, \boldsymbol{x}^2] = \mathbf{f}_2(\mathbf{y}) \quad \Leftrightarrow \quad \boldsymbol{x}^2 = \mathbf{y}^2, \boldsymbol{x}^1 = \boldsymbol{g}_2(\mathbf{y}^1; m_2(\mathbf{y}^2)).$$

In their following work, Dinh et al. [2017] further extended NICE to *real-valued non-volume preserving flow* (realNVP), by defining

$$\boldsymbol{g}(\boldsymbol{z}^2, m(\boldsymbol{z}^1)) = \boldsymbol{z}^2 \odot \exp(s(\boldsymbol{z}^1)) + t(\boldsymbol{z}^1), \quad m(\boldsymbol{z}^1) = [s(\boldsymbol{z}^1), t(\boldsymbol{z}^1)],$$

and constructing the splitting $\boldsymbol{z} = [\boldsymbol{z}^1, \boldsymbol{z}^2]$ using check-board masks and channel masks.

Inverse auto-regressive flow Kingma et al. [2016] introduced the *inverse auto-regressive flow* (IAF) which defines the invertible transform as

$$\boldsymbol{z}_t = \text{sigmoid}(\boldsymbol{s}_t)\odot\boldsymbol{z}_{t-1}+(1-\text{sigmoid}(\boldsymbol{s}_t))\odot\boldsymbol{m}_t, \quad [\boldsymbol{s}_t, \boldsymbol{m}_t] = \text{Auto-regressiveNN}[t](\boldsymbol{z}_{t-1}).$$

Here the Auto-regressiveNN$[t](\boldsymbol{z}_{t-1})$ is a network acting on the previous random variable $\boldsymbol{z}_t$, such that the i^th element of $\boldsymbol{s}_t$ (and $\boldsymbol{m}_t$) only depends on $\boldsymbol{z}_{t-1}[1:(i-1)]$. Therefore, the Jacobian $\nabla_{\boldsymbol{z}_{t-1}}\boldsymbol{s}_t$ (and $\nabla_{\boldsymbol{z}_{t-1}}\boldsymbol{m}_t$) is a lower-triangular matrix with zeros on the diagonal, so that the determinant of the Jacobian $\nabla_{\boldsymbol{z}_{t-1}}\boldsymbol{z}_t$ is

$$\det(\nabla_{\boldsymbol{z}_{t-1}}\boldsymbol{z}_t) = \prod_{i=1}^{d}\text{sigmoid}(\boldsymbol{s}_t)[i].$$

7.3 Connecting normalising flows to MCMC methods

As a side note, one can also construct *continuous time* normalising flow using stochastic differential equations. Further, one can add auxiliary variables and apply invertible mappings to the augmented space, and then use the marginal distribution of $\boldsymbol{z}$ as the induced probability density. A prevalent example is the Hamiltonian Monte Carlo (HMC) method [Duane et al., 1987; Neal, 2011], which essentially deploys a *deterministic* dynamics in the augmented space of $\{\boldsymbol{z}, \boldsymbol{p}\}$. Here $\boldsymbol{p}$ is often called *momentum* and is usually assumed to be Gaussian distributed. Therefore if defining the joint distribution $\pi(\boldsymbol{z}, \boldsymbol{p}) = \pi(\boldsymbol{z})\mathcal{N}(\boldsymbol{p}; \boldsymbol{0}, \mathbf{M})$, then sampling from $\pi(\boldsymbol{z})$ can be done by simulating HMC with target distribution $q(\boldsymbol{z}, \boldsymbol{p})$ and throw away the momentum samples.

To be more specific, the Hamiltonian dynamics is governed by the following differential equation:

$$\frac{d\boldsymbol{z}}{dt} = \mathbf{M}^{-1}\boldsymbol{p}, \quad \frac{d\boldsymbol{p}}{dt} = \nabla_{\boldsymbol{z}}\log\pi(\boldsymbol{z}). \tag{138}$$

Neal [2011] showed that this continuous-time flow is *volume preserving*, i.e. the determinant of the Jacobian is 1. We refer to the original paper for proof details, and here we briefly introduce the *Leapfrog* integration scheme [Neal, 2011] that is often used in practice. The Leapfrog integrator simulates the dynamics as the following: with discretisation step-size η

$$\begin{aligned}
\boldsymbol{p}_{t+\frac{1}{2}} &= \boldsymbol{p}_t + \frac{\eta}{2}\nabla_{\boldsymbol{z}}\log\pi(\boldsymbol{z}_t), \\
\boldsymbol{z}_{t+1} &= \boldsymbol{z}_t + \eta\mathbf{M}^{-1}\boldsymbol{p}_{t+\frac{1}{2}}, \\
\boldsymbol{p}_{t+1} &= \boldsymbol{p}_{t+\frac{1}{2}} + \frac{\eta}{2}\nabla_{\boldsymbol{z}}\log\pi(\boldsymbol{z}_{t+1}).
\end{aligned} \tag{139}$$

Importantly this Leapfrog transformation is volume-preserving. To see this, consider three mappings $F_1(\boldsymbol{z}_t, \boldsymbol{p}_t) = (\boldsymbol{z}_t, \boldsymbol{p}_{t+\frac{1}{2}})$, $F_2(\boldsymbol{z}_t, \boldsymbol{p}_{t+\frac{1}{2}}) = (\boldsymbol{z}_{t+1}, \boldsymbol{p}_{t+\frac{1}{2}})$ and $F_3(\boldsymbol{z}_{t+1}, \boldsymbol{p}_{t+1}) = (\boldsymbol{z}_{t+1}, \boldsymbol{p}_{t+1})$. Also denote $\boldsymbol{Z} = (\boldsymbol{z}, \boldsymbol{p})$ Then we have $\boldsymbol{Z}_{t+1} = F_3 \circ F_2 \circ F_1(\boldsymbol{Z}_t)$, and the determinant of the Jacobian of the transformation is the

product $\det(\nabla_{\boldsymbol{Z}_t}\boldsymbol{Z}_{t+1}) = \det(\nabla F_3)\det(\nabla F_2)\det(\nabla F_1)$. Writing down the Jacobian of F_1 explicitly

$$\nabla_{\boldsymbol{Z}} F_1(\boldsymbol{Z}) = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \frac{\eta}{2}\nabla^2_{\boldsymbol{z}} \log \pi(\boldsymbol{z}) & \mathbf{I} \end{pmatrix}, \tag{140}$$

it is straight-forward to show $\det(\nabla_{\boldsymbol{Z}} F_1(\boldsymbol{Z})) = 1$. Similarly one can also show that $\det(\nabla_{\boldsymbol{Z}} F_2(\boldsymbol{Z})) = \det(\nabla_{\boldsymbol{Z}} F_3(\boldsymbol{Z})) = 1$, indicating the volume preserving property of this Leapfrog integrator.

This idea has also been investigated in Salimans et al. [2015] where the authors discretised the HMC dynamics and learned the HMC parameters with VI. It selects $\pi(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x})$, $q(\boldsymbol{p}) = \mathcal{N}(\boldsymbol{p}; \mathbf{0}, \mathbf{M})$, and uses the idea that will be detailed in **??** to introduce an auxiliary lower-bound

$$\mathcal{L}(q; \boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{p})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}')r(\boldsymbol{p}'|\boldsymbol{x}, \boldsymbol{z}')}{q(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{p})}\right], \quad (\boldsymbol{z}', \boldsymbol{p}') = T\text{-step Leapfrog}(\boldsymbol{z}, \boldsymbol{p}), \tag{141}$$

with an auxiliary distribution $r(\boldsymbol{p}'|\boldsymbol{x}, \boldsymbol{z}')$ that will also be learned during training.

7.4 Further reading

Before reading on the latest research papers on this topic, readers should be comfortable with change-of-variable operation in calculus. For a reminder of how invertible transformation is applied to approximate inference, I would recommend Rezende and Mohamed [2015] to start with.

Papamakarios et al. [2019] is an excellent survey that summarises the status of normalising flow related research up to mid 2019. To me, the milestone papers to read in this field include NICE [Dinh et al., 2014], RealNVP [Dinh et al., 2017], IAF [Kingma et al., 2016] and Glow [Kingma and Dhariwal, 2018]. Some other ideas that I found interesting include Sylvester normalising flows that make use of Sylvester's determinant identity [van den Berg et al., 2018], neural spline flows [Durkan et al., 2019] which define the invertible transformations as monotonic spline functions, and Gaussianisation flows [Meng et al., 2020] based on the idea of iterative Gaussianisation [Laparra et al., 2011].

So far we only discussed normalising flows for transforming continuous random variables. For discrete variables, Tran et al. [2019] and Hoogeboom et al. [2019] independently proposed invertible transformations for discrete variables, and the gradients required by back-propagation are approximated by bias estimators such as the straight-through estimator [Bengio et al., 2013] and the Gumbel-softmax trick [Jang et al., 2017; Maddison et al., 2017b]. Note that unlike the continuous case, invertible transforms on discrete variables do NOT change the entropy of the distribution. This means one needs to be careful on the choice of the initial distribution $q(\boldsymbol{z}_0)$ in that the entropy of $q(\boldsymbol{z}_0)$ should be close to the entropy of the target distribution.

Another observation on normalising flows is that the invertible transformations do NOT reduce the dimension of the support of the random variables. Since a common practice is to set $q(\boldsymbol{z}_0)$ to be Gaussian which has its support as $\mathbb{R}^{\dim(\boldsymbol{z}_0)}$, this means the target variable $\boldsymbol{z}_T$ has the same dimensionality as $\boldsymbol{z}_0$, therefore

no dimension reduction is performed. As dimension reduction is a widely-used technique for representation learning, here is an interesting question: what is the representation learned by normalising flow techniques? In the RealNVP paper [Dinh et al., 2017] the $\boldsymbol{z}_0$ variables are split into chunks, with each chunk going through different numbers of invertible transformations. This means $\boldsymbol{z}_0$ might contain a multi-scaled representation of the target variable, but depending on how the split is performed, the meaning of "local" and "global" representation can be very different. Another line of work observes that the support of the data distribution (or target distribution) often lies on a low-dimensional manifold. Based on this observation, recent developments have extended normalising flows to random variables defined on low-dimensional manifolds, e.g. see Gemici et al. [2016]; Rezende et al. [2020].

We can also make the number of layers continuous like in the HMC case. This results in continuous-time flows, although in practice they will be approximated by numerical integration methods which go back to the discrete number of layers scheme. For example, the HMC example in § 7.3) uses Leapfrog integration. This is an active research field which aims to combine the best from normalising flow and differential equations, to start on reading related work, see Grathwohl et al. [2019] and the references therein.

8 Stochastic regularisation techniques

As the neural networks in use become bigger and deeper, researchers start to think about better regularisation techniques (beyond ℓ_2) to prevent overfitting. Since the proposal of *dropout* [Srivastava et al., 2014], *stochastic regularisation techniques* (SRTs) has become increasingly popular for this purpose. Roughly speaking, the idea of SRTs is to randomly modify the inputs/hidden units/outputs and/or the network architecture during training. After training the network's parameters will be fixed according to some correction rules, which is later used to produce predictions in a deterministic way.

Unlike the above approach, Gal and Ghahramani [2016] argued that dropout performs approximate Bayesian inference, therefore dropout is also a natural thing to do in test time according to this Bayesian interpretation. From approximate accuracy point of view, the flexibility of the induced approximate distribution from SRTs might not be superb. But from computational complexity point of view, SRTs improve space complexity (and potentially time complexity) substantially, and they are by far one of the most successful approximate inference techniques for Bayesian *convolutional/recurrent* neural networks. As SRTs are much easier to implement comparing to other Bayesian neural network techniques, this Bayesian interpretation has made quite a substantial impact to the deep learning community.

8.1 MC-dropout as variational inference

Here we summarise the idea of SRTs as an approximate inference method, with Bernoulli dropout as an example. In this case $\boldsymbol{z} = \{\mathbf{W}^l\}$ are the network weight matrices, and recall in Bayesian neural networks they are treated as latent variables. We assume the variational parameters $\boldsymbol{\phi} = \{\mathbf{M}^l\}$ where $\mathbf{M}^l$ have the same shapes and dimensions as $\mathbf{W}^l$ (readers will soon realise why we define $\boldsymbol{\phi}$ in such way). Then in the forward pass, we will use $\{\mathbf{M}^l\}$ to parameterise the network, and compute the output of each layer (with dropout) as

$$\boldsymbol{h}^l = \sigma(\boldsymbol{\epsilon} \odot (\mathbf{M}^l \boldsymbol{h}^{l-1})), \quad \boldsymbol{\epsilon} \sim \text{Bern}(\rho), \quad \rho \in [0, 1]. \tag{142}$$

In general $\sigma(\cdot)$ is an activation function and $\boldsymbol{\epsilon}$ is a random noise variable applied on pre-activations. Gal and Ghahramani [2016] showed that dropout is equivalent to the following process:

$$\boldsymbol{h}^l = \sigma(\mathbf{W}^l \boldsymbol{h}^{l-1}), \quad \mathbf{W}^l \sim q_{\boldsymbol{\phi}}(\mathbf{W}^l) = \prod_{\text{rows } i} \rho \mathcal{N}(\mathbf{M}_i^l, \eta \mathbf{I}) + (1-\rho)\mathcal{N}(\mathbf{0}, \eta \mathbf{I}), \quad \eta \to 0, \tag{143}$$

meaning that dropout can be viewed as an *implicit* sampling technique to evaluate the output of the neural network with an MC sample $\mathbf{W} \sim q_{\boldsymbol{\phi}}(\mathbf{W})$. See Figure 18 for a visual proof.

From this mixture of Gaussian view of Bernoulli dropout, the MC approxima-

tion of the variational lower-bound is

$$\mathcal{L}_{\text{VI}}^{\text{MC}}(q) = \frac{1}{K}\sum_{k=1}^{K}\sum_{n=1}^{N}\log p(\mathbf{y}_n|\boldsymbol{x}_n, \mathbf{W}^{n,k}) - \text{KL}[q(\mathbf{W})|p(\mathbf{W})]. \tag{144}$$

Notice that here we implicitly sample NK set of weights, $\{\mathbf{W}^{n,k}\}_{n=1,k=1}^{N,K}$ for N data-points $\mathcal{D} = \{(\boldsymbol{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$. It remains to compute the KL term, and here we present a brief derivation for Gaussian prior case $p(\mathbf{W}) = \prod_{l=1}^{L} p(\mathbf{W}^l), p(\mathbf{W}^l) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$:

$$-\text{KL}[q(\mathbf{W})|p(\mathbf{W})] = \sum_{l=1}^{L} -\frac{\lambda}{2}\mathbb{E}_q[||\mathbf{W}^l||_2^2] + \mathbb{H}[q(\mathbf{W}^l)] + \text{const.}$$

It is easy to show that the first term involves computing the second moment of q:

$$\mathbb{E}_q[||\mathbf{W}^l||_2^2] = \frac{1}{\lambda}\mathbf{I} + (1-\rho)||\mathbf{M}^l||_2^2.$$

It remains to approximate the entropy term. For a sample $\mathbf{W}^l \sim \mathcal{N}(\mathbf{0}, \eta\mathbf{I})$, the ℓ_2 distance $||\mathbf{W}^l - \mathbf{M}^l||_2^2 \to +\infty$ as the dimensionality of $\mathbf{W}^l$ goes to infinity. Therefore for this sample, $q(\mathbf{W}^l) \approx \rho\mathcal{N}(\mathbf{W}^l; \mathbf{0}, \eta\mathbf{I})$ when the product of the layer's input and output dimensions is large, which is usually the case. Similarly, for a sample $\mathbf{W}^l \sim \mathcal{N}(\mathbf{M}^l, \eta\mathbf{I})$, the density value $q(\mathbf{W}^l)$ is dominated by the term $(1-\rho)\mathcal{N}(\mathbf{W}^l; \mathbf{M}^l, \eta\mathbf{I})$. Therefore the entropy can be approximated as

$$\mathbb{H}[q(\mathbf{W}^l)] \approx \rho\mathbb{H}[\mathcal{N}(\mathbf{0}, \eta\mathbf{I})] + (1-\rho)\mathbb{H}[\mathcal{N}(\mathbf{0}, \mathbf{M}^l\mathbf{I})] + \mathbb{H}[\rho] = \mathbb{H}[\mathcal{N}(\mathbf{0}, \eta\mathbf{I})] + \mathbb{H}[\rho].$$

Then the entropy term can be dropped if the dropout rate ρ is not optimised, as now the entropy doesn't depend on the variational parameters.[27] Dropping other constant terms, the resulting variational lower-bound is

$$\mathcal{L}_{\text{VI}}^{\text{MC}}(q) \approx \frac{1}{K}\sum_{k=1}^{K}\sum_{n=1}^{N}\log p(\mathbf{y}_n|\boldsymbol{x}_n, \mathbf{W}^{n,k}) - \sum_{l=1}^{L}\frac{(1-\rho)\lambda}{2}||\mathbf{M}^l||_2^2. \tag{145}$$

In practice only $K = 1$ sample of $\mathbf{W}$ is used for each data-point. Also for many loss functions $\ell(\cdot,\cdot)$ the log-likelihood terms can be defined as $\log p(\mathbf{y}|\boldsymbol{x}, \mathbf{W}) = \ell(\mathbf{y}, \text{NN}_{\mathbf{W}}(\boldsymbol{x}))$. Therefore, training a deep neural network with dropout and an ℓ_2 regulariser is equivalent to training a Bayesian neural network using VI and an unusual approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{W})$ as a "mixture of delta mass functions".

Remark (Some theoretical issues)**.** Recall that in the derivations we dropped the entropy term $\mathbb{H}[q(\mathbf{W})]$. When $\eta \to 0$ this is problematic since now one cannot define the entropy using density values. In fact in this case traditional variational inference is unlikely to be applicable. Since the Gaussian prior $p(\mathbf{W})$ has full support, with likelihood functions that are non-zero almost everywhere,

[27] There are some theoretical issues for this approach when limiting $\eta \to 0$, see later remarks.

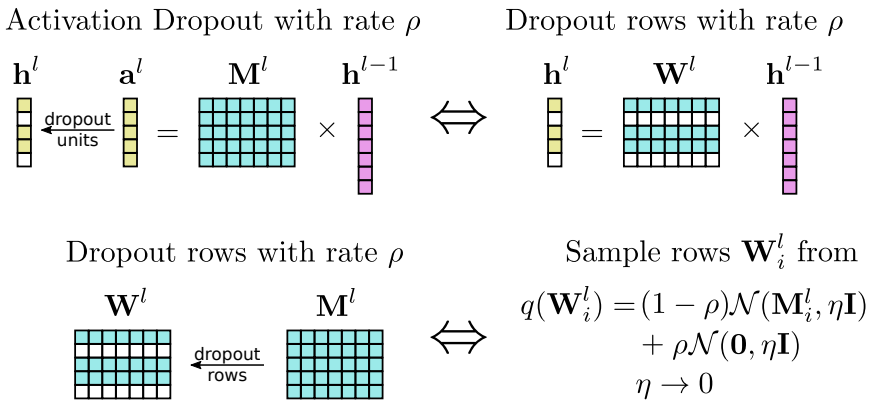

Figure 18: An illustration of Bernoulli dropout (with rate ρ of keeping a unit) as an approximate inference procedure. See main text for more discussions.

the posterior will also have non-zero density almost everywhere. But when limiting $\eta \to 0$ one cannot define density $q(\mathbf{W})$ as now the corresponding probability measure $Q(\mathbf{W})$ is not dominated by the Lebesgue measure. Also one can show that Q is not dominated by the posterior as its support is a zero-measure set under the posterior probability measure. Thus the KL between Q and the posterior is always infinity and there is no way to reduce it as long as this support mismatch exists!

We can think of several possible fixes of this issue. A naive approach is to set η arbitrarily close to zero but not exactly zero, so that the KL term is still well-defined, and the entropy term can be ignored during optimisation. But since Bernoulli dropout always samples from the "mixture of delta mass", not the relaxed mixture of Gaussian version, technically speaking this makes the objective function a biased estimate of the variational lower-bound. The second solution is to "discretise" the distributions and work with very fine grids in high dimensions. This approach prevents the pathological case when $\eta \to 0$, but the approximation error is very sensitive to the choice of the grid. A third proposal is called Quasi-KL divergence [Hron et al., 2018] although no empirical evaluation is available on Bayesian neural networks.

8.2 Gaussian dropout and the local reparameterisation trick

The predictive distribution of a Bayesian neural network with Gaussian approximate posterior can also be done in a similar fashion as MC-dropout.

Consider a row of the weight matrix $\boldsymbol{w} \sim q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Then the corresponding pre-activation $a = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$ is distributed as $q(a|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x}, \boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{x})$. This means if we use mean-field approximations – meaning that $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma}^2)$ is a diagonal matrix, then we have the following equivalence:

$$a = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}, \boldsymbol{w} \sim q(\boldsymbol{w}) \quad \Leftrightarrow \quad a \sim \mathcal{N}(\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x}, (\boldsymbol{\sigma}^2)^{\mathrm{T}}\boldsymbol{x}^2), \tag{146}$$

with $\boldsymbol{\sigma}^2 = (\sigma_1^2, ..., \sigma_{\dim(\boldsymbol{x})}^2)$ and $\boldsymbol{x}^2 = (x_1^2, ..., x_{\dim(\boldsymbol{x})}^2)$. Applying the reparameteri-

sation trick to the Gaussian distribution $q(a|\boldsymbol{x})$, we have

$$a \sim q(a|\boldsymbol{x}) \quad \Leftrightarrow \quad \epsilon \sim \mathcal{N}(0,1), a = \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x} + \epsilon(\boldsymbol{\sigma}^2)^{\mathrm{T}}\boldsymbol{x}^2. \tag{147}$$

This trick is named the *local reparameterisation trick* in Kingma et al. [2015].

Here we briefly show that Gaussian dropout also corresponds to approximate Bayesian inference for neural networks. Following the discussions on Gaussian approximate posteriors, if we assume $\boldsymbol{\Sigma} = \alpha\boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}$, then it is straight-forward to show that $q(a|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x}; \alpha(\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x})^2)$, and the sampling process of a is equivalently

$$a \sim q(a|\boldsymbol{x}) \Leftrightarrow \epsilon \sim \mathcal{N}(1, \alpha), a = \epsilon\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x},$$

which recovers Gaussian multiplicative dropout as presented in Srivastava et al. [2014]; Wang and Manning [2013]. In summary, with Gaussian dropout, the effective approximate posterior distribution is

$$q(\mathbf{W}) = \prod q(\mathbf{W}^l), \quad q(\mathbf{W}^l) = \prod_{\text{rows i}} \mathcal{N}(\mathbf{M}_i^l; \alpha\mathbf{M}_i^l \otimes \mathbf{M}_i^l), \tag{148}$$

and we can directly plug-in this definition to the variational lower-bound to define the optimisation objective.

Noticing that the variance parameter of $q(\mathbf{W})$ depends on $\mathbf{M}$, Kingma et al. [2015] proposed an improper prior $p(\mathbf{W}_{ij}^l) \propto \frac{C}{|\mathbf{W}_{ij}^l|}$ in order to make $\mathrm{KL}[q(\mathbf{W})||p(\mathbf{W})]$ independent with $\mathbf{M}$. Hron et al. [2018] showed that this is problematic because the improper prior is likely to make the posterior improper if the likelihood functions do not decay fast enough around $\mathbf{W}_{ij}^l = 0$. To avoid this pathology a truncation of $p(\mathbf{W}_{ij}^l)$ into an interval $[-e^\beta, e^\beta]$ might be preferred, however this means $p(\mathbf{W}_{ij}^l)$ does not have full support (therefore same for the posterior), and fitting a Gaussian approximate posterior with variational inference is less well justified.

8.3 Revisiting variance reduction for Monte Carlo VI

We have discussed some variance reduction techniques for MC-VI in § 2.3, and in this section we revisit this topic again but in the context of Bayesian neural networks, and from this discussion we will see one of the advantages of SRTs in terms of space complexity.

Recall that the variational lower-bound for a Bayesian neural network is the following:

$$\mathcal{L}_{\mathrm{VI}}(q) = \mathbb{E}_{q(\mathbf{W})}\left[\sum_{n=1}^{N} \log p(\mathbf{y}_n|\boldsymbol{x}_n, \mathbf{W})\right] - \mathrm{KL}[q(\mathbf{W})||p(\mathbf{W})], \tag{149}$$

and for simplicity we assume $q(\mathbf{W})$ is Gaussian distributed with mean $\mathbf{M}$. Therefore the gradient of the first "error" term w.r.t. $\mathbf{M}$ is

$$\nabla_{\mathbf{M}}\mathbb{E}_{q(\mathbf{W})}\left[\frac{1}{N}\sum_{n=1}^{N} \log p(\mathbf{y}_n|\boldsymbol{x}_n, \mathbf{W})\right] = \mathbb{E}_{q(\mathbf{W})}\mathbb{E}_{(\boldsymbol{x},\mathbf{y})\sim\mathcal{D}}\left[\sum_{n=1}^{N} \nabla_{\mathbf{W}} \log p(\mathbf{y}|\boldsymbol{x}, \mathbf{W})\right]$$

Denote $\nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y}) = \nabla_{\mathbf{W}} \log p(\mathbf{y}|\boldsymbol{x}, \mathbf{W})$. In practice stochastic optimisation and Monte Carlo approximation is applied, i.e. a mini-batch $\mathcal{S} = \{(\boldsymbol{x}_m, \mathbf{y}_m)\}_{m=1}^M \sim \mathcal{D}^M$ is sampled from data, and the variational lower-bound is approximated with MC samples $\mathbf{W} \sim q(\mathbf{W})$. Often one MC sample $\mathbf{W} \sim q(\mathbf{W})$ is used to evaluate $\nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y})$ for all $(\boldsymbol{x}, \mathbf{y}) \in \mathcal{S}$, and the resulting stochastic gradient $\hat{\nabla}_{\mathbf{W}}(\mathcal{S}) = \frac{1}{M} \sum_{(\boldsymbol{x},\mathbf{y})\in\mathcal{S}} \nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y})$ is an unbiased estimate of $\mathbb{E}_{q(\mathbf{W})}\mathbb{E}_{\mathcal{D}}[\nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y})]$ Therefore by the decomposition rule of total variance,

$$
\begin{aligned}
\mathbb{V}[\hat{\nabla}_{\mathbf{W}}(\mathcal{S})] =& \mathbb{E}_{\mathcal{S}\sim\mathcal{D}^M}[\mathbb{V}_{q(\mathbf{W})}[\hat{\nabla}_{\mathbf{W}}(\mathcal{S})]] + \mathbb{V}_{\mathcal{S}\sim\mathcal{D}^M}[\mathbb{E}_{q(\mathbf{W})}[\hat{\nabla}_{\mathbf{W}}(\mathcal{S})]], \\
\mathbb{E}_{\mathcal{S}\sim\mathcal{D}^M}[\mathbb{V}_{q(\mathbf{W})}[\hat{\nabla}_{\mathbf{W}}(\mathcal{S})]] =& \frac{1}{M}\mathbb{E}_{\mathcal{D}}[\mathbb{V}_{q(\mathbf{W})}[\nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y})]] \\
&+ \frac{M-1}{M}\mathbb{E}_{(\boldsymbol{x},\mathbf{y}),(\boldsymbol{x}'.\mathbf{y}')\sim\mathcal{D}}[\mathrm{Cov}_{q(\mathbf{W})}[\nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y}), \nabla_{\mathbf{W}}(\boldsymbol{x}', \mathbf{y}')]], \\
\mathbb{V}_{\mathcal{S}\sim\mathcal{D}^M}[\mathbb{E}_{q(\mathbf{W})}[\hat{\nabla}_{\mathbf{W}}(\mathcal{S})]] =& \frac{1}{M}\mathbb{V}_{\mathcal{D}}[\mathbb{E}_{q(\mathbf{W})}[\nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y})]] \\
&+ \frac{M-1}{M}\mathrm{Cov}_{(\boldsymbol{x},\mathbf{y}),(\boldsymbol{x}'.\mathbf{y}')\sim\mathcal{D}}[\mathbb{E}_{q(\mathbf{W})}[\nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y})], \mathbb{E}_{q(\mathbf{W})}[\nabla_{\mathbf{W}}(\boldsymbol{x}', \mathbf{y}')]] \\
=& \frac{1}{M}\mathbb{V}_{\mathcal{D}}[\mathbb{E}_{q(\mathbf{W})}[\nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y})]]. \quad \text{\# i.i.d. assumption on data}
\end{aligned}
\tag{150}
$$

We see that the covariance term $\frac{M-1}{M}\mathbb{E}_{(\boldsymbol{x},\mathbf{y}),(\boldsymbol{x}'.\mathbf{y}')\sim\mathcal{D}}[\mathrm{Cov}_{q(\mathbf{W})}[\nabla_{\mathbf{W}}(\boldsymbol{x}, \mathbf{y}), \nabla_{\mathbf{W}}(\boldsymbol{x}', \mathbf{y}')]]$ appears in the variance of the stochastic gradient, and it would be come dominant as M increases. It means with this simple Monte Carlo estimate there is a certain amount of variance that is irreducible, which can slow down the convergence of stochastic gradient descent.

To remove the covariance term, a simple strategy would independently sample $\mathbf{W}^m \sim q(\mathbf{W})$ for each $(\boldsymbol{x}_m, \mathbf{y}_m) \in \mathcal{S}$, and estimate the gradient as $\tilde{\nabla}_{\mathbf{W}}(\mathcal{S}) = \frac{1}{M} \sum_{m=1}^M \nabla_{\mathbf{W}^m}(\boldsymbol{x}_m, \mathbf{y}_m)$. However there is a wrinkle of this approach: explicit sampling of M weight matrices requires $\mathcal{O}(Md)$ time for sequential computation or $\mathcal{O}(Md)$ space for parallel computing, where d is the dimension of the neural network parameters. As a deep neural network typically has millions (if not billions) of parameters, this means $d \gg 0$ and the $\mathcal{O}(Md)$ cost becomes very expensive even for a reasonable value of the mini-batch size M (say $M = 64$). So it would be preferred if the sampling process can be done in an *implicit* way.

Now recall the Bernoulli/Gaussian dropout approaches discussed before. These methods perform sampling in the (pre-)activation space, rather than directly sampling M set of weights. Therefore there is no need to pay the extra $\mathcal{O}(Md)$ cost either in time or space complexity. More importantly, as the random noises attached to the (pre-)activations differ for different datapoint, this (pre-)activation space sampling approach implicitly samples different $\mathbf{W}^m$ matrices for different input $(\boldsymbol{x}_m, \mathbf{y}_m)$.

8.4 Further reading

The MC-dropout technique applies to other dropout techniques as well, e.g. DropConnect [Wan et al., 2013], SwapOut [Singh et al., 2016] and ZoneOut [Krueger

et al., 2017]. The the Bayesian interpretation can also be used to guide the design of new SRTs [Kingma et al., 2015; Louizos and Welling, 2017; Molchanov et al., 2017]. E.g. Atanov et al. [2018] modified the batch normalisation method [Ioffe and Szegedy, 2015] to enable uncertainty estimation.

One can also design better sampling approaches for a given $q(\mathbf{W})$ distribution, such that the sampling is done efficiently (e.g. in the activation space), and the variance of the gradient estimate is reduced. For example Wen et al. [2018] observed that for Gaussian approximate posteriors one can perform antithetic sampling to further reduce the variance of the gradient estimate, and they proposed an efficient implementation which avoids direct sampling of the weight matrices.

9 Implicit distributions

So far we have discussed a various of approximate distribution designs to enable better approximations to the posterior. These distributions have one thing in common: they have tractable densities (or at least tractable densities in the joint form for mixture distributions). This section will cover another type of distributions that has been recently introduced to the approximate inference community: implicit distributions [Diggle and Gratton, 1984; Mohamed and Lakshminarayanan, 2016].

9.1 Revisiting tractability issues in approximate inference

First we would like to explain why introducing implicit distributions to approximate inference may be a good idea. To do so, I would invite the readers to consider the following question:

What does tractability mean for an approximate inference algorithm?

To answer the above question, we first start by revisiting the definition of *approximate inference* (see § 0.3), with Bayesian posterior inference as an illustrating example. Assume a model with prior distribution $p(\boldsymbol{z})$ and likelihood function $p(\boldsymbol{x}|\boldsymbol{z})$. Then *inference* means computing the expectation of some function $F(\boldsymbol{z})$ under the exact posterior, which is $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})]$. Examples of such F functions include:

- $F(\boldsymbol{z}) = \boldsymbol{z}^k$, i.e. computing the moments of p;
- $F(\boldsymbol{z}) = p(\mathbf{y}^*|\boldsymbol{z}, \boldsymbol{x}^*)$ if in supervised learning and $\boldsymbol{z}$ represents the model parameters;
- $F(\boldsymbol{z}) = \delta_A$ if one wishes to evaluate $p(\boldsymbol{z} \in A|\boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[\delta_A]$.

For simplicity in the rest of the discussion we assume the evaluation of $F(\boldsymbol{z})$ can be done using available computational resources, otherwise it needs more approximations, see remarks in § 0.3.

The core idea of (optimisation based) approximate inference is to fit an approximate posterior distribution $q(\boldsymbol{z}|\boldsymbol{x})$ in a "tractable" distribution family $\mathcal{Q}$ to the exact posterior $p(\boldsymbol{z}|\boldsymbol{x})$, such that $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})]$ can be well approximated by

$$\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})] \approx \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})]. \tag{151}$$

Critically, the primary tractability requirement here for the approximate posterior is the *fast computation* of the *approximate expectation* $\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})]$ given the function F.

Historically, approximate distributions of simple forms, such as mean-field approximations and factorised Gaussians [Jordan et al., 1999], have been proposed to obtain analytical solutions of the approximated expectation. These approaches often require the probabilistic model to comprise conjugate exponential families,

which excludes a broad range of powerful models, e.g. those who warp noise variables through non-linear mappings. Instead, modern approximate inference introduces Monte Carlo (MC) estimation techniques to approximate the predictive likelihood [Paisley et al., 2012; Ranganath et al., 2014] that we reviewed in § 2. The MC method enables a wider class of models to be amendable to VI (the requirement is that the log-joint can be computed point-wise), and is key to modern training methods of generative models such as the VAE [Kingma and Welling, 2014; Rezende et al., 2014].

Precisely, at inference time, the MC approximation method samples $\{\boldsymbol{z}^1, ..., \boldsymbol{z}^K\}$ from the approximate posterior q, and estimates the required quantity by

$$\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})] \approx \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{z}^k), \quad \boldsymbol{z}^k \sim q(\boldsymbol{z}|\boldsymbol{x}). \tag{152}$$

Consequently, this converts the *fast expectation computation* requirement to *fast sampling* from the approximate posterior, as the expectation is further approximated by the empirical average. Fast sampling is arguably a stronger condition compared to fast expectation computation, as for the latter, one would normally expect fast calculations for a *given* function F. The latter is typically the case for traditional numerical integration methods since for different functions one would select different quadrature rules. On the other hand, once we have obtained the samples from the approximate posterior, we can use them to compute an empirical estimate of the expectation for *any* function. Hence methods that entail fast sampling might be preferred for tasks that require estimating expectations of a set of functions.

Unfortunately, the approximate inference algorithms that we have discussed so far impose further constraints to the design of q. For example, recall the MC-VI objective in § 2:

$$\mathcal{L}_{\text{VI}}^{\text{MC}}(q; p) = \frac{1}{K}\sum_{k=1}^{K} \log p(\boldsymbol{x}, \boldsymbol{z}^k) - \log q(\boldsymbol{z}^k|\boldsymbol{x}), \quad \boldsymbol{z}^k \sim q(\boldsymbol{z}^k|\boldsymbol{x}). \tag{153}$$

Then it is clear that the training procedure requires *fast density evaluation*, or at leat *fast log-density gradient evaluation* for $q(\boldsymbol{z}|\boldsymbol{x})$ given a configuration of $\boldsymbol{z}$, if only the optimisation process rather than the bound value itself is in concern. Indeed, this requirement is only presented in the VI optimisation procedure to seek for the best fit of q: once obtain a (local) optimum, MC inference only requires evaluating the empirical expectation thus no need to compute the density of q point-wise.

9.2 Examples of implicit distributions

The observations in § 9.1 raise an outstanding question: can we design efficient approximate inference algorithms to train flexible approximate posterior distributions which are *implicit*, i.e. without access to an explicit density function? As defined in Diggle and Gratton [1984], implicit distributions are those distributions

where one can sample from it fairly easily, but cannot evaluate its density pointwise. As presented in this section, implicit distributions cover a very wide range of distributions that can potentially provide very accurate approximations to the exact posterior. Therefore it is worth considering them as approximate posterior distributions and design approximate inference algorithms to fit them.

- Neural network transform with noise inputs.
  For a 1-dimensional random variable, the sampling process can be described by deterministically transforming a uniform noise variable with the inverse *cumulative density function* (CDF) or quantile function:

$$z \sim p(z) \quad \Leftrightarrow \quad u \sim \text{Uniform}(0,1), \quad z = \text{CDF}_p^{-1}(u). \tag{154}$$

  For multivariate distributions, a similar process would return a set of possible configurations $\text{CDF}_p^{-1}(u) = \{\inf\ \boldsymbol{z} : \text{CDF}_p(\boldsymbol{z}) \geq u\}$, and one could then uniformly sample from it. However, computing the (inverse) CDF can be even harder than evaluating the density $p(\boldsymbol{z})$ at a given configuration. Therefore normally one would not use inverse CDF mappings to define approximate posteriors.

  Still the observation above inspires us to define the q distribution by transforming a random noise variable $\boldsymbol{\epsilon}$ through a deterministic mapping $\mathbf{f}_{\boldsymbol{\phi}}$:

$$\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \quad \Leftrightarrow \quad \boldsymbol{\epsilon} \sim \pi(\boldsymbol{\epsilon}), \quad \boldsymbol{z} = \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}), \tag{155}$$

  with $\pi(\boldsymbol{\epsilon})$ a simple distribution such as a standard Gaussian. The deterministic mapping can be parameterised by a neural network, in other words $\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}) = \text{NN}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$. Since neural networks are well known to be universal functional approximators [Hornik et al., 1989], the hope is that the constructed neural network is expressive enough to learn how to return a point in the set of $\text{CDF}_p^{-1} \circ \text{CDF}_\pi$. This type of distributions is also called *variational programs* in [Ranganath et al., 2016a], or *implicit generative models* in the generative model context [Mohamed and Lakshminarayanan, 2016]. The density network [Mackay and Gibbs, 1999] further generalises this idea using "Bayesian" neural networks, by putting a prior distribution on the neural network parameters. In this case the sampling procedure changes to

$$\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x}) \quad \Leftrightarrow \quad \boldsymbol{\epsilon} \sim \pi(\boldsymbol{\epsilon}), \boldsymbol{W} \sim \pi(\boldsymbol{W}), \quad \boldsymbol{z} = \text{NN}_{\boldsymbol{W}}(\boldsymbol{\epsilon}, \boldsymbol{x}). \tag{156}$$

- Hierarchical mixture distributions with implicit conditionals.
  Recall the hierarchical approximate posterior with auxiliary variables that is discussed in **??**:

$$q(\boldsymbol{z}|\boldsymbol{x}) = \int \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}) d\boldsymbol{z}_{0:T-1}, \quad \boldsymbol{z}_T := \boldsymbol{z}. \tag{157}$$

  In **??** the conditionals $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x})$ have tractable density, and an auxiliary variational lower-bound is derived to train this hierarchical approximate posterior. What if $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x})$ is implicit? Following the discussions of neural

network approximations to the inverse CDF function, generally one can implicitly define the conditional distribution using a neural network taking $\boldsymbol{z}_{t-1}$ and an extra "nuisance" noise variable $\boldsymbol{\epsilon}_t$ as the input:

$$\boldsymbol{z}_t \sim q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}) \quad \Leftrightarrow \quad \boldsymbol{z}_t = \mathbf{f}_t(\boldsymbol{z}_{t-1}, \boldsymbol{\epsilon}_t, \boldsymbol{x}). \quad \boldsymbol{\epsilon}_t \sim \pi(\boldsymbol{\epsilon}_t). \tag{158}$$

The joint distribution $q(\boldsymbol{z}_{0:T}|\boldsymbol{x})$ is then parameterised by a stochastic deep neural network. Again the marginal distribution $q(\boldsymbol{z}_T|\boldsymbol{x})$ does not have a tractable density, but sampling from this distribution can be done by a forward pass of the stochastic neural network.

- Dynamics based on stochastic differential equations. MCMC algorithms are considered as "gold standard" for posterior inference. Stochastic gradient MCMC (SG-MCMC) methods are a type of popular sampling-based inference algorithms for Bayesian neural networks. Readers can read related papers for the detail applications, but in a nutshell, an SG-MCMC algorithm obtains posterior samples by simulating a stochastic dynamic defined by a stochastic differential equation (SDE) whose stationary distribution is the exact posterior. Under some mild conditions, Ma et al. [2015] showed that a stochastic gradient MCMC (SG-MCMC) algorithm within the Itô diffusion framework follows the SDE below:

$$d\boldsymbol{z}_t = [(\boldsymbol{D}(\boldsymbol{z}_t)+\boldsymbol{Q}(\boldsymbol{z}_t))\nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z} = \boldsymbol{z}_t|\boldsymbol{x})+\Gamma(\boldsymbol{z}_t)]dt+\sqrt{2\boldsymbol{D}(\boldsymbol{z}_t)}dW_t, \tag{159}$$

with W_t a Wiener process (or Brownian motion process). In practice one cannot simulate the diffusion process above for infinite amount of time; rather one would choose an integrator and numerically compute the integration/simulation until a given time limit T. Later on one will use $\boldsymbol{z}_T$ as the samples from the approximate posterior, therefore by setting $\boldsymbol{z} = \boldsymbol{z}_T$ this simulation process also implicitly define a $q(\boldsymbol{z}|\boldsymbol{x})$ distribution.

The quality of the $\boldsymbol{z}_T$ samples depends largely on the SDE configurations, where $\boldsymbol{D}(\boldsymbol{z}_t)$ and $\boldsymbol{Q}(\boldsymbol{z}_t)$ control the drift and diffusion of the dynamics, and $\Gamma(\boldsymbol{z}_t)$ is a correction term to ensure asymptotic exactness (with infinitesimal discretisation step-size). Therefore it would be desirable if one can optimise these matrix-valued functions to improve the sample quality of the SG-MCMC algorithm.

9.3 Algorithmic options

Implicit distributions cannot be fitted using traditional approximate inference methods such as MC-VI. In this section, we discuss algorithmic options for training these approximations to the posterior.[28] In short, I will cover:

- **energy approximation:** methods to approximate the variational lower-bound given an implicit q distribution;

[28] We note here that the discussed options are applicable to tractable q distributions as well.

- **gradient approximation:** methods to approximate the gradient of the variational lower-bound;
- **alternative divergence minimisation:** two methods using the Stein's discrepancy;
- **amortising deterministic/stochastic dynamics:** training a fast sampler that can (approximately) produce samples from MCMC or other dynamics.

Most of the approaches covered here assume q to be reparameterisable, i.e. $\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x}) \Leftrightarrow \boldsymbol{\epsilon} \sim \pi(\boldsymbol{\epsilon}), \boldsymbol{z} = \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$. It remains a main challenge to extend implicit distribution fitting to discrete distributions in general.

9.3.1 Energy approximation

Assume q is reparameterisable, then by the chain rule, the gradient $\nabla_{\boldsymbol{\phi}}\mathcal{L}$ is computed as $\nabla_{\mathbf{f}}\mathcal{L}\nabla_{\boldsymbol{\phi}}\mathbf{f}$. Therefore, if we have an approximation $\hat{\mathcal{L}}$ to the objective function, then we can approximate the gradient as $\nabla_{\boldsymbol{\phi}}\mathcal{L} \approx \nabla_{\mathbf{f}}\hat{\mathcal{L}}\nabla_{\boldsymbol{\phi}}\mathbf{f}$. We refer this approach as *energy/objective approximation*.

A popular idea considers density ratio estimation methods [Qin, 1998; Sugiyama et al., 2009, 2012] for energy approximation, which is concurrently considered in Li and Liu [2016]; Karaletsos [2016]; Mescheder et al. [2017]; Huszár [2017]; Tran et al. [2017] and later in Shi et al. [2018a]. This is done by introducing an auxiliary distribution $\tilde{q}$ and rewrite the variational lower-bound:

$$\mathcal{L}_{\text{VI}}(\boldsymbol{\theta}, q; \boldsymbol{x}) = \mathbb{E}_q \left[\log \frac{p_0(\boldsymbol{z}) p(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta})}{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})} + \log \frac{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z}|\boldsymbol{x})} \right]. \tag{160}$$

The auxiliary distribution $\tilde{q}$ is required to have tractable density and is easy to sample. Then one can use sample-based density ratio estimation methods to fit a model $\tilde{R}$ for the ratio between $\tilde{q}$ and q. The gradient approximation for general $\tilde{q}$ distributions can be derived similarly as

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\text{VI}} = \mathbb{E}_q \left[\nabla_{\boldsymbol{\phi}} \log \frac{p_0(\boldsymbol{z}) p(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta})}{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})} + \nabla_{\boldsymbol{z}} \tilde{R}(\boldsymbol{z}, \boldsymbol{x}) \nabla_{\boldsymbol{\phi}} \mathbf{f} \right]. \tag{161}$$

In the following we briefly show that the original GAN approach [Goodfellow et al., 2014] can be applied as a density ratio estimator. Consider a discriminator $D(\boldsymbol{z}, \boldsymbol{x})$ which outputs the probability of a sample $\boldsymbol{z}$ coming from the auxiliary distribution $\tilde{q}(\boldsymbol{z}|\boldsymbol{x})$. Using the same calculation in Goodfellow et al. [2014] one can easily show that the optimal discriminator is

$$D^*(\boldsymbol{z}, \boldsymbol{x}) = \frac{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x}) + \tilde{q}(\boldsymbol{z}|\boldsymbol{x})} = \frac{1}{1 + \exp(-\log \frac{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z}|\boldsymbol{x})})} = \text{sigmoid}\left(\log \frac{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z}|\boldsymbol{x})} \right). \tag{162}$$

As in practice the discriminator is often parameterised using a neural network with sigmoid activation, we can define $D(\boldsymbol{z}, \boldsymbol{x}) = \text{sigmoid}(\tilde{R}(\boldsymbol{z}, \boldsymbol{x}))$ and use $\tilde{R}(\boldsymbol{z}, \boldsymbol{x})$ as the density-ratio estimator. Using the GAN training this density-ratio estimator

will get improved towards the exact ratio. Note that this density-ratio estimator can also be obtained using f-GAN [Nowozin et al., 2016].

An alternative approach in Shi et al. [2018a] applies kernel methods for density ratio estimation. In short, the authors considers minimising the following objective to fit an approximation $R(\boldsymbol{z}, \boldsymbol{x}) \approx \frac{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z}|\boldsymbol{x})}$:

$$\begin{aligned} \mathcal{L}(R) &= \mathbb{E}_q \left[\left(R(\boldsymbol{z}, \boldsymbol{x}) - \frac{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z}|\boldsymbol{x})} \right)^2 \right] \\ &= \mathbb{E}_q \left[R(\boldsymbol{z}, \boldsymbol{x})^2 \right] - 2\mathbb{E}_{\tilde{q}} \left[R(\boldsymbol{z}, \boldsymbol{x}) \right] + C \end{aligned} \tag{163}$$

Then they follow the kernel density ratio estimator method (e.g. see Sugiyama et al. [2009]), which parameterises R with a kernel machine, and obtains analytical solutions for the linear coefficients. By doing so, neither discriminator nor double loop training are required.

Remark (the choice of the auxiliary $\tilde{q}$)**.** A simple example considers $\tilde{q} = p_0$ and the classification approach for ratio estimation [Karaletsos, 2016; Huszár, 2017; Tran et al., 2017]. However in practice, density ratio estimation works poorly if the two distribution in comparison are very different, especially in the case that the regions contains most of the probability mass have little overlap. Instead Mescheder et al. [2017] discussed an advanced technique termed as *adaptive contrast*, which takes $\tilde{q}$ as a Gaussian approximation to the wild distribution q. In this case the estimation of $\tilde{q}$ requires many samples from q which can significantly slow down training. To address this issue, the authors further constructed a specific type of implicit q distributions, which allows sharing randomness between different $q(\boldsymbol{z}_n|\boldsymbol{x}_n)$ distributions and thus reducing the total number of MC samples computed on a mini-batch of data. This trick improves the approximation accuracy by a significant margin as density ratio estimation is accurate when the two distributions are similar to each other.

9.3.2 Direct gradient approximation

The recent development of machine learning algorithms, including VI and SG-MCMC, rely on advanced optimisation tools such as stochastic gradient descent with adaptive learning rates. Informally the optimisation procedure works as the following: given the current mini-batch of data, we first compute the gradients, then feed them to the optimiser to construct the final update of the training parameters. In the above energy approximation example, this gradient computation is done by first approximating the original objective function $\hat{\mathcal{L}} \approx \mathcal{L}$, then differentiating this approximate energy to obtain an approximate descending direction. However, even when $\hat{\mathcal{L}}$ approximates $\mathcal{L}$ very well at the points from the gradient descent trajectory, the approximate gradient $\nabla_{\boldsymbol{\phi}} \hat{\mathcal{L}}$ can still be a poor estimator for the exact gradient $\nabla_{\boldsymbol{\phi}} \mathcal{L}$. We depict this phenomenon in Figure 19.

We see that the energy approximation approach can be problematic if not done in a correct way, therefore a *direct gradient approximation* to the exact gradient might be preferred. To see how the gradient approximation idea applies to the

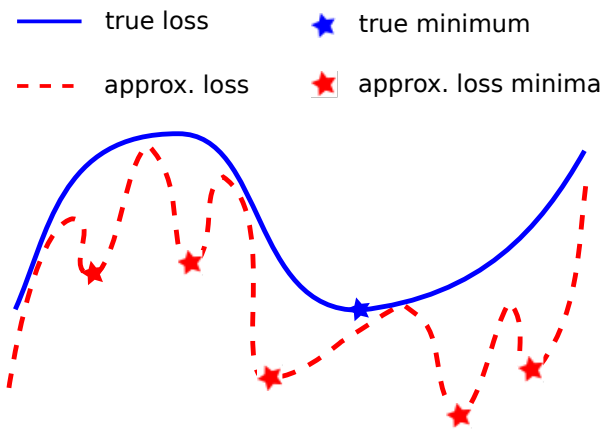

Figure 19: A visualisation of the exact/approximate loss. As one would typically use a deep neural network to help approximate the exact loss function, without careful control of its representation power, the deep net can potentially return a loss function that is accurate at the evaluated points, but has over-complicated shapes in general. It could result in strongly biased gradient updates of the training parameters and bad local optimum.

VI case, consider the gradient $\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\text{VI}}$ using the reparameterisation trick (also see § 2.2.1 and § 2.2.2):

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\text{VI}} = \nabla_{\boldsymbol{\phi}}\mathbb{E}_{\pi(\boldsymbol{\epsilon})}[(\nabla_{\mathbf{f}} \log p(\boldsymbol{x}, \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})) - \nabla_{\mathbf{f}} \log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})|\boldsymbol{x}))\nabla_{\boldsymbol{\phi}}\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})]. \tag{164}$$

Therefore to perform gradient based optimisation, it remains to approximate $\nabla_{\boldsymbol{z}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, as $\nabla_{\boldsymbol{z}} \log p(\boldsymbol{x}, \boldsymbol{z})$ and $\nabla_{\boldsymbol{\phi}}\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$ are tractable by assumption. However traditional methods, e.g. Stone [1985]; Zhou and Wolfe [2000]; Ruppert and Wand [1994]; Fan and Gijbels [1996]; De Brabanter et al. [2013], do not apply because they require at least a noisy version of $\nabla_{\boldsymbol{z}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, which is intractable in our case. Instead, we will discuss three gradient approximation methods based on kernel methods as follows.

- KDE plug-in estimator.
  A naive idea would again fit another approximation $\hat{q}$ using samples from q, then use $\nabla_z \log \hat{q}(\boldsymbol{z}|\boldsymbol{x})$ to approximate the gradient. In kernel methods context, Singh [1977] applied a kernel estimator directly to the first and higher order derivatives. However this method still approximates the target gradient function in an indirect way, and depending on the bandwidth selection, the fitted KDE density can be less smooth or too smooth, making the gradient approximation error high.

- Score matching gradient estimator.
  As motivated, it is preferred to directly minimising the approximation error of the gradient function. Here Sasaki et al. [2014]; Strathmann et al. [2015] considered the ℓ_2 error between the true gradient $\nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})$ and

the approximation $\hat{\boldsymbol{g}}(\boldsymbol{z}) = (\hat{g}_1(\boldsymbol{z}), ..., \hat{g}_d(\boldsymbol{z}))^{\mathrm{T}}$:

$$\mathcal{F}(\hat{\boldsymbol{g}}) = \mathbb{E}_q \left[||\hat{\boldsymbol{g}}(\boldsymbol{z}) - \nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})||_2^2 \right]. \tag{165}$$

Although the ℓ_2 error still contains the exact gradient $\nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})$, Hyvärinen [2005] showed that by using *integration by parts* and assuming the *boundary condition* $\lim_{\boldsymbol{z}\to\infty} \hat{\boldsymbol{g}}(\boldsymbol{z}) q(\boldsymbol{z}|\boldsymbol{x}) = \mathbf{0}$, the ℓ_2 error can be rewritten as

$$\mathcal{F}(\hat{\boldsymbol{g}}) = \mathbb{E}_q \left[||\hat{\boldsymbol{g}}(\boldsymbol{z})||_2^2 + 2\langle \nabla, \hat{\boldsymbol{g}}(\boldsymbol{z}) \rangle \right], \quad \langle \nabla, \hat{\boldsymbol{g}}(\boldsymbol{z}) \rangle = \sum_{i=1}^{d} \nabla_{\boldsymbol{z}_i} \hat{g}_i(\boldsymbol{z}). \tag{166}$$

The above loss is also referred as the *score matching* objective, and therefore the optimum of $\hat{\boldsymbol{g}}$ is also called the score matching gradient estimator. Since (166) requires computing the gradient of $\hat{\boldsymbol{g}}$, Sasaki et al. [2014]; Strathmann et al. [2015] designed a *parametric model*

$$\hat{\boldsymbol{g}}(\boldsymbol{z}) = \sum_{k=1}^{K} a_k \nabla_{\boldsymbol{z}} \mathcal{K}(\boldsymbol{z}, \boldsymbol{z}^k), \quad \boldsymbol{z}^k \sim q(\boldsymbol{z}|\boldsymbol{x}),$$

and proposed fitting the linear coefficients a_k by minimising (166).

- Stein gradient estimator.
  Using the same trick as to derive (166) Stein's identity [Stein, 1981; Gorham and Mackey, 2015; Liu et al., 2016] can also be derived (also see § 4.3). Given a *test function* $\mathbf{h}(\boldsymbol{z}) : \mathbb{R}^d \to \mathbb{R}^{d'}$ and assuming the *boundary condition* again $\lim_{\boldsymbol{z}\to\infty} \mathbf{h}(\boldsymbol{z}) q(\boldsymbol{z}|\boldsymbol{x}) = \mathbf{0}$, Stein's identity is the following:

  $$\mathbb{E}_q \left[\mathbf{h}(z) \nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})^{\mathrm{T}} + \nabla_{\boldsymbol{z}} \mathbf{h}(\boldsymbol{z}) \right] = \mathbf{0}. \tag{167}$$

  Observing this, Li and Turner [2018] proposed the *Stein gradient estimator*, by inverting Stein's identity to obtain an estimator of $\nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})$. They first approximated (167) with Monte Carlo, and then performed Ridge regression to obtain a *non-parametric* estimate of the gradient (by noticing that the MC approximation to (167) is linear in $\nabla_{z^k} \log q(\boldsymbol{z}^k|\boldsymbol{x})$). The Stein gradient estimator idea is further improved by Shi et al. [2018b]; Zhou et al. [2020], with an application to Bayesian neural networks [Sun et al., 2018].

Remark (denoising auto-encoder as a score function estimator)**.** It has been shown in [Särelä and Valpola, 2005; Alain and Bengio, 2014] that denoising auto-encoders (DAEs) [Vincent et al., 2008], once trained, can be used to compute the score function approximately. Briefly speaking, a DAE learns to reconstruct a datum $\boldsymbol{x}$ from a corrupted input $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \sigma\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ by minimising the mean square error. Then the optimal DAE can be used to approximate the score function as $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) \approx \frac{1}{\sigma^2}(\mathrm{DAE}^*(\boldsymbol{x}) - \boldsymbol{x})$. Sonderby et al. [2017] deployed this idea to train an implicit model for image super-resolutions, providing some promising results in some metrics. However applying similar ideas to vari-

ational inference can be very expensive, because the estimation of $\nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})$ is a sub-routine for VI which is repeatedly required.

9.3.3 Alternative optimisation objectives

§ 4 discussed alternative divergences for approximate inference. Among many of these alternatives, one promising direction is to replace the KL divergence with Stein discrepancy [Stein, 1972; Barbour, 1988; Gorham and Mackey, 2015]:

$$\mathrm{S}[p, q] = \sup_{g \in \mathcal{G}_p} |\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}, \boldsymbol{x}) g(\boldsymbol{z}) + \nabla_{\boldsymbol{z}} g(\boldsymbol{z})]|, \tag{168}$$

which only requires samples from q and the score function $\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}, \boldsymbol{x})$ thus indeed tractable. When applid to approximate inference, Ranganath et al. [2016a] defined $\mathcal{G}_p$ as parametric functions represented by neural networks, and approximate the minimax optimisation with gradient descent in an analogous way to GAN training [Goodfellow et al., 2014]. This idea is revisited by Grathwohl et al. [2020] in the context of energy-based models. In contrast, analytic solution of the supremum exists if $\mathcal{G}_p$ is defined as the unit ball in an RKHS, where Liu et al. [2016] and Chwialkowski et al. [2016] termed the corresponding measure as the kernelised Stein discrepancy (KSD). Liu and Feng [2016] further developed an approximate inference algorithm by directly minimising the KSD between the exact and approximate posterior distributions.

9.3.4 Amortising dynamics

MCMC and particle-based approximate inference methods [Dai et al., 2016a; Liu and Wang, 2016], though very accurate, become inefficient when inference from multiple different distributions is repeatedly required. As an example consider learning a (deep) generative model, where fast (approximate) marginalisation of latent variables is desirable. Here we consider amortised inference to learn an inference network to mimic a selected stochastic dynamics. More precisely, the algorithm works in three steps:

1. We sample $\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x})$;
2. We improve the sample $\boldsymbol{z}$ to $\boldsymbol{z}_T$ by running T-step stochastic/deterministic dynamics;
3. We use the improved samples $\boldsymbol{z}_T$ as targets to improve $q(\boldsymbol{z}|\boldsymbol{x})$.

Li et al. [2017] considered an MCMC sampler as such a dynamics that we want to amortise. The theoretical intuition behind this approach is illustrated in Figure 20. Since the MCMC "oracle" always improves the sample quality in terms of approximating the target distribution,[29] by following the MCMC dynamics, the q distribution will also get improved, until the stage when $\boldsymbol{z}_T$ has the same distribution as $\boldsymbol{z}$ which means $q = p$. Similar intuition also applies to other deterministic

[29] We have $\mathrm{KL}[q_t||p] \geq \mathrm{KL}[q_{t+1}||p]$ iff $q_t \to p$ for any $q_0 = q$ [Cover and Thomas, 1991].

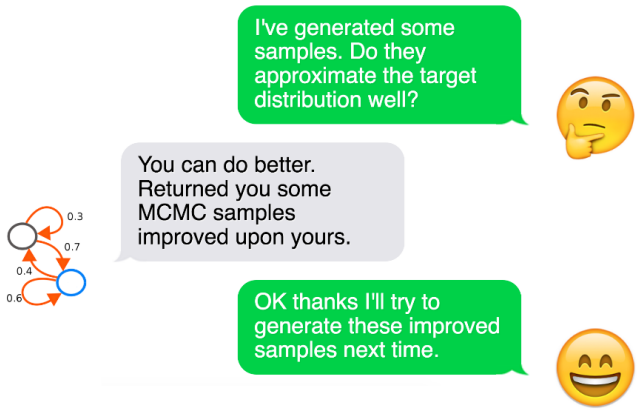

Figure 20: A cartoon illustration of the amortised MCMC idea in Li et al. [2017].

dynamics as long as they generate particles that are always approaching to the target distribution. For example, Wang and Liu [2016] used this idea to amortised a deterministic dynamics called Stein variational gradient descent (SVGD). The "catch-up" step for $q(\boldsymbol{z}|\boldsymbol{x})$ can be defined as

$$\boldsymbol{\phi}^{\text{new}} = \arg\min_{\boldsymbol{\phi}} \mathrm{D}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||q_T(\boldsymbol{z}|\boldsymbol{x})], \quad q_T(\boldsymbol{z}|\boldsymbol{x}) = \int \mathcal{T}_T(\boldsymbol{z}|\boldsymbol{z}')q_{\boldsymbol{\phi}}(\boldsymbol{z}'|\boldsymbol{x})d\boldsymbol{z}', \quad (169)$$

where D denotes any divergence/discrepancy/distance and q_T is fixed as the target and does not get differentiated through. In practice only one gradient step is performed, i.e.

$$\boldsymbol{\phi}^{\text{new}} = \boldsymbol{\phi} - \eta\nabla_{\boldsymbol{\phi}}\mathrm{D}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||q_T(\boldsymbol{z}|\boldsymbol{x})]. \quad (170)$$

Liu and Wang [2016] used ℓ_2 distance in sample space $||\boldsymbol{z} - \boldsymbol{z}_T||_2^2$ and therefore the "catch-up" step is defined by deliberately chaining the gradients $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \eta\mathbb{E}_q[\nabla_{\boldsymbol{\phi}}\boldsymbol{z}(\boldsymbol{z}_T - \boldsymbol{z})]$. However for stochastic dynamics the ℓ_2 distance in sample space does not work well, and instead Li et al. [2017] proposed using any GAN idea to match the q distribution (as fake data) to the target q_T (as real data).

9.4 Further reading

The energy approximation method depends on adversarial training, therefore I would recommend reading papers on stable training for GANs, e.g. Salimans et al. [2016]; Miyato et al. [2018].

MCMC methods implicitly constructed an approximate posterior distribution which also lacks a tractable density form. Recent work has introduced meta-learning based approaches to optimise the parameters/configurations of an MCMC method, see e.g. Wang et al. [2018b]; Gong et al. [2019].

Titsias [2017] proposed a hybrid approach combining invertible transformations and MCMC to construct an implicit approximate posterior. The key observation is that, MCMC algorithms often converge much faster on simpler distributions such as Gaussians. Therefore, if there exists an invertible mapping $\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$ which transforms the complicated exact posterior $p(\boldsymbol{z}|\boldsymbol{x})$ to a considerably simpler distribution

$p(\boldsymbol{\epsilon}|\boldsymbol{x}) = p(\mathbf{f}(\boldsymbol{\epsilon}, \boldsymbol{x})|\boldsymbol{x})|\nabla_{\boldsymbol{\epsilon}}\mathbf{f}|$, then one can perform MCMC to obtain $\boldsymbol{\epsilon}^k \sim p(\boldsymbol{\epsilon}|\boldsymbol{x})$ (approximately) then map them back to $\boldsymbol{z}^k = \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}) \sim p(\boldsymbol{z}|\boldsymbol{x})$ (approximately). Similar observations have been made to improve gradient-based MCMC methods, e.g. see Song et al. [2017]; Levy et al. [2018]; Hoffman et al. [2019].

Score-matching based approach, as a way to train generative models, has recently attracted (returned) research interest. This round of resurgence is largely brought by Song and Ermon [2019] who showed that deep generative models trained by score-matching based approach can generate realistic looking images. Interested readers can read this paper and the follow-up work in this line.

References

Agakov, F. V. and Barber, D. (2004). An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723.

Alain, G. and Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593.

Ali, S. M. and Silvey, S. D. (1966). A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 131–142.

Amari, S.-i. (1982). Differential geometry of curved exponential families-curvatures and information loss. *The Annals of Statistics*, pages 357–385.

Amari, S.-I. (1985). *Differential-geometrical methods in statistics*. Springer.

Amari, S.-I. (2009). α-divergence is unique, belonging to both f-divergence and Bregman divergence classes. *IEEE Transactions on Information Theory*, 55(11):4925–4931.

Amari, S.-i., Ikeda, S., and Shimokawa, H. (2001). Information geometry of α-projection in mean field approximation. *Advanced Mean Field Methods*, pages 241–257.

Amari, S.-I. and Nagaoka, H. (2000). *Methods of information geometry*. Oxford University Press.

Ambrogioni, L., Güçlü, U., Güçlütürk, Y., Hinne, M., van Gerven, M. A., and Maris, E. (2018). Wasserstein variational inference. In *Advances in Neural Information Processing Systems*, pages 2473–2482.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223.

Atanov, A., Ashukha, A., Molchanov, D., Neklyudov, K., and Vetrov, D. (2018). Uncertainty estimation via stochastic batch normalization. *arXiv preprint arXiv:1802.04893*.

Attias, H. (1999). Inferring parameters and structure of latent variable models by variational Bayes. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 21–30. Morgan Kaufmann Publishers Inc.

Attias, H. (2000). A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems*, pages 209–215.

Barbour, A. D. (1988). Stein's method and Poisson process convergence. *Journal of Applied Probability*, pages 175–184.

Barbu, A. and Zhu, S.-C. (2005). Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1239–1253.

Barthelmé, S. and Chopin, N. (2014). Expectation propagation for likelihood-free inference. *Journal of the American Statistical Association*, 109(505):315–333.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2015). Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767*.

Bayes, T. and Price, R. (1763). An essay towards solving a problem in the doctrine of chances. by the late rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. *Philosophical Transactions (1683-1775)*, 53:370–418.

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London.

Beal, M. J. and Ghahramani, Z. (2003). The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian Statistics*, 7:453–463.

Beaumont, M. A., Zhang, W., and Balding, D. J. (2002). Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035.

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Bethe, H. (1935). Statistical theory of superlattices. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 150(871):552–575.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blackwell, D. (1947). Conditional expectation and unbiased sequential estimation. *The Annals of Mathematical Statistics*, pages 105–110.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Bornschein, J. and Bengio, Y. (2015). Reweighted wake-sleep. In *International Conference on Learning Representations*.

Boyle, P. P. (1977). Options: A monte carlo approach. *Journal of financial economics*, 4(3):323–338.

Burda, Y., Grosse, R., and Salakhutdinov, R. (2016). Importance weighted autoencoders. In *International Conference on Learning Representations*.

Burnham, K. P. and Anderson, D. R. (2004). Multimodel inference: understanding aic and bic in model selection. *Sociological methods & research*, 33(2):261–304.

Chernoff, H. (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507.

Chib, S. (1995). Marginal likelihood from the gibbs output. *Journal of the american statistical association*, 90(432):1313–1321.

Chwialkowski, K., Strathmann, H., and Gretton, A. (2016). A kernel test of goodness of fit. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2606–2615.

Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. John Wiley & Sons.

Cremer, C., Li, X., and Duvenaud, D. (2018). Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*.

Cremer, C., Morris, Q., and Duvenaud, D. (2017). Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*.

Csiszár, I. (1963). Eine informationstheoretische ungleichung und ihre anwendung auf den beweis der ergodizitat von markoffschen ketten. *Magyar. Tud. Akad. Mat. Kutató Int. Közl*, 8:85–108.

Dai, B., He, N., Dai, H., and Song, L. (2016a). Provable Bayesian inference via particle mirror descent. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics*, pages 985–994.

Dai, H., Dai, B., and Song, L. (2016b). Discriminative embeddings of latent variable models for structured data. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2702–2711.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, 7(5):889–904.

De Brabanter, K., De Brabanter, J., De Moor, B., and Gijbels, I. (2013). Derivative estimation with local polynomial fitting. *The Journal of Machine Learning Research*, 14(1):281–301.

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654.

Dehaene, G. and Barthelmé, S. (2015). Expectation propagation in the large-data limit. *arXiv:1503.08060*.

Dehaene, G. and Barthelmé, S. (2018). Expectation propagation in the large data limit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(1):199–217.

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)*, pages 1–38.

Dieng, A. B., Tran, D., Ranganath, R., Paisley, J., and Blei, D. (2017). Variational inference via χ upper bound minimization. In *Advances in Neural Information Processing Systems*, pages 2729–2738.

Diggle, P. J. and Gratton, R. J. (1984). Monte Carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 193–227.

Dinh, L., Krueger, D., and Bengio, Y. (2014). NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real NVP. In *International Conference on Learning Representations*.

Domke, J. and Sheldon, D. (2018). Importance weighting and varational inference. In *Advances in Neural Information Processing Systems*.

Domke, J. and Sheldon, D. R. (2019). Divide and couple: Using monte carlo variational objectives for posterior approximation. In *Advances in Neural Information Processing Systems*.

Doucet, A., De Freitas, N., and Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer.

Doucet, A. and Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216 – 222.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7511–7522.

Efron, B. (1975). Defining the curvature of a statistical problem (with applications to second order efficiency). *The Annals of Statistics*, pages 1189–1242.

Efron, B. (1978). The geometry of exponential families. *The Annals of Statistics*, 6(2):362–376.

Fan, J. and Gijbels, I. (1996). *Local polynomial modelling and its applications*. Chapman & Hall.

Feng, Y., Wang, D., and Liu, Q. (2017). Learning to draw samples with amortised stein variational gradient descent. In *Uncertainty in Artificial Intelligence*.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1050–1059.

Geffner, T. and Domke, J. (2020). On the difficulty of unbiased alpha divergence minimization. *arXiv preprint arXiv:2010.09541*.

Gelman, A., Vehtari, A., Jylänki, P., Sivula, T., Tran, D., Sahai, S., Blomstedt, P., Cunningham, J. P., Schiminovich, D., and Robert, C. (2014). Expectation propagation as a way of life: A framework for Bayesian inference on partitioned data. *arXiv preprint arXiv:1412.4869*.

Gemici, M. C., Rezende, D., and Mohamed, S. (2016). Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*.

Ghahramani, Z. (1995). Factorial learning and the EM algorithm. In *Advances in Neural Information Processing Systems*, pages 617–624.

Ghahramani, Z. and Rasmussen, C. E. (2003). Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 505–512.

Giles, M. B. (2015). Multilevel monte carlo methods. *Acta Numerica*, 24:259.

Glynn, P. W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

Gong, W., Li, Y., and Hernández-Lobato, J. M. (2019). Meta-learning for stochastic gradient mcmc. In *International Conference on Learning Representations*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Gorham, J. and Mackey, L. (2015). Measuring sample quality with Stein's method. In *Advances in Neural Information Processing Systems*, pages 226–234.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., and Duvenaud, I. S. D. (2019). Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., and Zemel, R. (2020). Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9485–9499.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

Grosse, R. B., Ghahramani, Z., and Adams, R. P. (2015). Sandwiching the marginal likelihood using bidirectional Monte Carlo. *arXiv preprint arXiv:1511.02543*.

Grünwald, P. (2007). *Minimum Description Length Principle*. MIT press, Cambridge, MA.

Gu, S., Levine, S., Sutskever, I., and Mnih, A. (2016). MuProp: Unbiased backpropagation for stochastic neural networks. In *International Conference on Learning Representations*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5769–5779.

Hammersley, J. M. and Handscomb, D. C. (1966). *Monte Carlo Methods*. Wiley.

Harman, H. H. (1976). *Modern factor analysis*. University of Chicago Press.

Hasenclever, L., Webb, S., Lienart, T., Vollmer, S., Lakshminarayanan, B., Blundell, C., and Teh, Y. W. (2017). Distributed bayesian learning with stochastic natural gradient expectation propagation and the posterior server. *The Journal of Machine Learning Research*, 18(1):3744–3780.

Hernández-Lobato, J. M., Li, Y., Rowland, M., Hernández-Lobato, D., Bui, T. D., and Turner, R. E. (2016). Black-box α-divergence minimization. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1511–1520.

Heskes, T. (2002). Stable fixed points of loopy belief propagation are local minima of the Bethe free energy. In *Advances in Neural Information Processing Systems*, pages 343–350.

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.

Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM.

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. (2019). Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*.

Hoffman, M. D. (2017). Learning deep latent Gaussian models with Markov chain Monte Carlo. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1510–1519.

Hoogeboom, E., Peters, J., van den Berg, R., and Welling, M. (2019). Integer discrete flows and lossless compression. In *Advances in Neural Information Processing Systems*, pages 12134–12144.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Hron, J., Matthews, A., and Ghahramani, Z. (2018). Variational bayesian dropout: pitfalls and fixes. In *International Conference on Machine Learning*, pages 2019–2028.

Huszár, F. (2017). Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709.

Ihler, A. T., Willsky, A. S., et al. (2005). Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6(May):905–936.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Jaakkola, T. S. and Jordan, M. I. (1998). Improving the mean field approximation via the use of mixture distributions. *Nato ASI Series D Behavioural and Social Sciences*, 89:163–174.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with Gumbel-softmax. In *International Conference on Learning Representations*.

Jin, W., Barzilay, R., and Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pages 2323–2332.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Karaletsos, T. (2016). Adversarial message passing for graphical models. *arXiv preprint arXiv:1612.05048*.

Kennedy, M. and O'Hagan, A. (1996). Iterative rescaling for Bayesian quadrature. *Bayesian Statistics*, 5:639–645.

Kim, Y., Wiseman, S., Miller, A. C., Sontag, D., and Rush, A. M. (2018). Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

Kingma, D. P., Welling, M., et al. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Kolmogorov, A. N. (1950). Unbiased estimates. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 14(4):303–326.

Konishi, S. and Kitagawa, G. (2008). *Information criteria and statistical modeling*. Springer Science & Business Media.

Krueger, D., Maharaj, T., Kramár, J., Pezeshki, M., Ballas, N., Ke, N. R., Goyal, A., Bengio, Y., Larochelle, H., and Courville, A. (2017). Zoneout: Regularizing rnns by randomly preserving hidden activations. In *International Conference on Learning Representations*.

Kschischang, F. R. and Frey, B. J. (1998). Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Communications*, 16(2):219–230.

Kschischang, F. R., Frey, B. J., and Loeliger, H.-a. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519.

Kucukelbir, A., Ranganath, R., Gelman, A., and Blei, D. (2015). Automatic variational inference in Stan. In *Advances in Neural Information Processing Systems*, pages 568–576.

Kullback, S. (1959). *Information theory and statistics*. John Wiley & Sons.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Kuss, M. and Rasmussen, C. E. (2005). Assessing approximate inference for binary Gaussian process classification. *The Journal of Machine Learning Research*, 6:1679–1704.

Laparra, V., Camps-Valls, G., and Malo, J. (2011). Iterative gaussianization: from ica to random rotations. *IEEE transactions on neural networks*, 22(4):537–549.

Laplace, P. S. (1820). *Théorie analytique des probabilités*. Courcier.

Lawrence, N. D., Bishop, C. M., and Jordan, M. I. (1998). Mixture representations for inference and learning in Boltzmann machines. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 320–327. Morgan Kaufmann Publishers Inc.

Lawrence, N. D. and Quiñonero-Candela, J. (2006). Local distance preservation in the GP-LVM through back constraints. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 513–520.

Le, T. A., Igl, M., Jin, T., Rainforth, T., and Wood, F. (2017). Auto-encoding sequential Monte Carlo. *arXiv preprint arXiv:1705.10306*.

Le, T. A., Kosiorek, A. R., Siddharth, N., Teh, Y. W., and Wood, F. (2018). Revisiting reweighted wake-sleep. *arXiv preprint arXiv:1805.10469*.

Le Roux, N., Schmidt, M., and Bach, F. R. (2012). A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671.

Levy, D., Hoffman, M. D., and Sohl-Dickstein, J. (2018). Generalizing hamiltonian monte carlo with neural networks. In *International Conference on Learning Representations*.

Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2015). Stochastic expectation propagation. In *Advances in neural information processing systems*, pages 2323–2331.

Li, Y. and Liu, Q. (2016). Wild variational approximations. *NIPS 2016 approximate inference workshop*.

Li, Y. and Turner, R. E. (2016). Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, pages 1073–1081.

Li, Y. and Turner, R. E. (2018). Gradient estimators for implicit models. In *International Conference on Learning Representations*.

Li, Y., Turner, R. E., and Liu, Q. (2017). Approximate inference with amortised MCMC. *arXiv preprint arXiv:1702.08343*.

Liu, Q. and Feng, Y. (2016). Two methods for wild variational inference. *arXiv preprint arXiv:1612.00081*.

Liu, Q., Lee, J., and Jordan, M. (2016). A kernelized Stein discrepancy for goodness-of-fit tests. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 276–284.

Liu, Q. and Wang, D. (2016). Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2370–2378.

Louizos, C. and Welling, M. (2017). Multiplicative normalizing flows for variational Bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2218–2227.

Luo, Y., Beatson, A., Norouzi, M., Zhu, J., Duvenaud, D., Adams, R. P., and Chen, R. T. Q. (2020). Sumo: Unbiased estimation of log marginal probability for latent variable models. In *International Conference on Learning Representations*.

Ma, Y.-A., Chen, T., and Fox, E. (2015). A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, pages 2917–2925.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. (2016). Auxiliary deep generative models. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1445–1453.

Mackay, D. J. and Gibbs, M. N. (1999). Density networks. In *Statistics and neural networks*, pages 129–144. Oxford University Press, Inc.

Maddison, C. J., Lawson, J., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. (2017a). Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pages 6576–6586.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017b). The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Marino, J., Yue, Y., and Mandt, S. (2018). Learning to infer.

Masrani, V., Le, T. A., and Wood, F. (2019). The thermodynamic variational objective. In *Advances in Neural Information Processing Systems*, pages 11525–11534.

Meng, C., Song, Y., Song, J., and Ermon, S. (2020). Gaussianization flows. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Mescheder, L., Nowozin, S., and Geiger, A. (2017). Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2391–2400.

Mézard, M., Parisi, G., and Virasoro, M. (1987). *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, volume 9. World Scientific Publishing Co Inc.

Minka, T. P. (2001a). The EP energy function and minimization schemes. Technical report, Technical report.

Minka, T. P. (2001b). Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, volume 17, pages 362–369.

Minka, T. P. (2004). Power EP. Technical Report MSR-TR-2004-149, Microsoft Research, Cambridge.

Minka, T. P. (2005). Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research, Cambridge.

Mironov, I. (2017). Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*.

Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1791–1799.

Mohamed, S. and Lakshminarayanan, B. (2016). Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2019). Monte carlo gradient estimation in machine learning. *arXiv preprint arXiv:1906.10652*.

Molchanov, D., Ashukha, A., and Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2498–2507.

Morimoto, T. (1963). Markov processes and the H-theorem. *Journal of the Physical Society of Japan*, 18(3):328–331.

Morris, Q. (2001). Recognition networks for approximate inference in BN20 networks. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 370–377. Morgan Kaufmann Publishers Inc.

Naesseth, C. A., Linderman, S. W., Ranganath, R., and Blei, D. M. (2017). Variational sequential Monte Carlo. *arXiv preprint arXiv:1705.11140*.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing*, 11(2):125–139.

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162.

Neal, R. M. and Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Newton, M. A. and Raftery, A. E. (1994). Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(1):3–26.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2010). Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861.

Nowozin, S. (2018). Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In *International Conference on Learning Representations*.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279.

O'Hagan, A. (1991). Bayes-Hermite quadrature. *Journal of statistical planning and inference*, 29(3):245–260.

Opper, M. and Archambeau, C. (2009). The variational gaussian approximation revisited. *Neural computation*, 21(3):786–792.

Opper, M. and Winther, O. (2005). Expectation consistent approximate inference. *The Journal of Machine Learning Research*, 6:2177–2204.

Paisley, J., Blei, D., and Jordan, M. (2012). Variational Bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1363–1370.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*.

Parisi, G. (1988). *Statistical field theory*. Addison-Wesley.

Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In *The Second National Conference on Artificial Intelligence (AAAI-82)*.

Peterson, C. and Anderson, J. R. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019.

Qin, J. (1998). Inferences for case-control and semiparametric two-sample density ratio models. *Biometrika*, 85(3):619–630.

Rainforth, T., Kosiorek, A. R., Le, T. A., Maddison, C. J., Igl, M., Wood, F., and Teh, Y. W. (2018). Tighter variational bounds are not necessarily better. *arXiv preprint arXiv:1802.04537*.

Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 814–822.

Ranganath, R., Tran, D., Altosaar, J., and Blei, D. (2016a). Operator variational inference. In *Advances in Neural Information Processing Systems*, pages 496–504.

Ranganath, R., Tran, D., and Blei, D. (2016b). Hierarchical variational models. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 324–333.

Rao, C. R., Rao, C. R., Statistiker, M., Rao, C. R., and Rao, C. R. (1973). *Linear statistical inference and its applications*, volume 2. Wiley New York.

Rényi, A. (1961). On measures of entropy and information. *Fourth Berkeley symposium on mathematical statistics and probability*, 1.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1530–1538.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286.

Rezende, D. J., Papamakarios, G., Racanière, S., Albergo, M. S., Kanwar, G., Shanahan, P. E., and Cranmer, K. (2020). Normalizing flows on tori and spheres. *arXiv preprint arXiv:2002.02428*.

Roeder, G., Wu, Y., and Duvenaud, D. K. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems*, pages 6928–6937.

Ruiz, F. R., AUEB, M. T. R., and Blei, D. (2016). The generalized reparameterization gradient. In *Advances in Neural Information Processing Systems*, pages 460–468.

Ruppert, D. and Wand, M. P. (1994). Multivariate locally weighted least squares regression. *The Annals of Statistics*, pages 1346–1370.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242.

Salimans, T., Kingma, D., and Welling, M. (2015). Markov chain Monte Carlo and variational inference: Bridging the gap. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1218–1226.

Salimans, T. and Knowles, D. A. (2013). Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882.

Särelä, J. and Valpola, H. (2005). Denoising source separation. *Journal of machine learning research*, 6(Mar):233–272.

Sasaki, H., Hyvärinen, A., and Sugiyama, M. (2014). Clustering via mode seeking by direct estimation of the gradient of a log-density. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 19–34. Springer.

Sato, M.-A. (2001). Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681.

Saul, L. K., Jaakkola, T., and Jordan, M. I. (1996). Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4:61–76.

Saul, L. K. and Jordan, M. I. (1996). Exploiting tractable substructures in intractable networks. In *Advances in Neural Information Processing Systems*, pages 486–492.

Schmidt, M., Roux, N. L., and Bach, F. (2013). Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Seeger, M. (2005). Expectation propagation for exponential families. Technical report, EPFL-REPORT-161464.

Shi, J., Sun, S., and Zhu, J. (2018a). Kernel implicit variational inference. In *International Conference on Learning Representations*.

Shi, J., Sun, S., and Zhu, J. (2018b). A spectral approach to gradient estimation for implicit distributions. In *International Conference on Machine Learning*, pages 4644–4653.

Shu, R., Bui, H. H., Zhao, S., Kochenderfer, M. J., and Ermon, S. (2018). Amortized inference regularization. *arXiv preprint arXiv:1805.08913*.

Singh, R. S. (1977). Improvement on some known nonparametric uniformly consistent estimators of derivatives of a density. *The Annals of Statistics*, pages 394–399.

Singh, S., Hoiem, D., and Forsyth, D. (2016). Swapout: Learning an ensemble of deep architectures. In *Advances in Neural Information Processing Systems*, pages 28–36.

Skilling, J. et al. (2006). Nested sampling for general bayesian computation. *Bayesian analysis*, 1(4):833–859.

Sonderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F. (2017). Amortised MAP inference for image super-resolution. In *International Conference on Learning Representations*.

Song, J., Zhao, S., and Ermon, S. (2017). A-nice-mc: Adversarial training for mcmc. In *Advances in Neural Information Processing Systems*, pages 5140–5150.

Song, L., Gretton, A., Bickson, D., Low, Y., and Guestrin, C. (2011). Kernel belief propagation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 707–715.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11918–11930.

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., and Lanckriet, G. R. (2009). On integral probability metrics,\phi-divergences and binary classification. *arXiv preprint arXiv:0901.2698*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Stein, C. M. (1972). A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*, pages 583–602.

Stein, C. M. (1981). Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, pages 1135–1151.

Stone, C. J. (1985). Additive regression and other nonparametric models. *The Annals of Statistics*, pages 689–705.

Strathmann, H., Sejdinovic, D., Livingstone, S., Szabo, Z., and Gretton, A. (2015). Gradient-free Hamiltonian Monte Carlo with efficient kernel exponential families. In *Advances in Neural Information Processing Systems*, pages 955–963.

Stuhlmüller, A., Taylor, J., and Goodman, N. (2013). Learning stochastic inverses. In *Advances in Neural Information Processing Systems*, pages 3048–3056.

Sugiyama, M., Kanamori, T., Suzuki, T., Hido, S., Sese, J., Takeuchi, I., and Wang, L. (2009). A density-ratio framework for statistical data processing. *Information and Media Technologies*, 4(4):962–987.

Sugiyama, M., Suzuki, T., and Kanamori, T. (2012). Density-ratio matching under the Bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044.

Sun, S., Zhang, G., Shi, J., and Grosse, R. (2018). Functional variational bayesian neural networks. In *International Conference on Learning Representations*.

Swendsen, R. H. and Wang, J.-S. (1987). Nonuniversal critical dynamics in monte carlo simulations. *Physical review letters*, 58(2):86.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Titsias, M. and Lázaro-Gredilla, M. (2015). Local expectation gradients for black box variational inference. In *Advances in Neural Information Processing Systems*, pages 2638–2646.

Titsias, M. K. (2017). Learning model reparametrizations: Implicit variational inference by fitting mcmc distributions. *arXiv preprint arXiv:1708.01529*.

Tran, D., Ranganath, R., and Blei, D. (2017). Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pages 5529–5539.

Tran, D., Ranganath, R., and Blei, D. M. (2016). The variational Gaussian process. In *International Conference on Learning Representations*.

Tran, D., Vafa, K., Agrawal, K., Dinh, L., and Poole, B. (2019). Discrete flows: Invertible generative models of discrete data. In *Advances in Neural Information Processing Systems*, pages 14719–14728.

Tsallis, C. (1988). Possible generalization of Boltzmann-Gibbs statistics. *Journal of statistical physics*, 52(1-2):479–487.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., and Sohl-Dickstein, J. (2017). Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pages 2624–2633.

Turner, R. E. and Sahani, M. (2011). Two problems with variational expectation maximisation for time-series models. In Barber, D., Cemgil, T., and Chiappa, S., editors, *Bayesian Time series models*, chapter 5, pages 109–130. Cambridge University Press.

van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. (2018). Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*.

Van Erven, T. and Harremoës, P. (2014). Rényi divergence and Kullback-Leibler divergence. *Information Theory, IEEE Transactions on*, 60(7):3797–3820.

Villani, C. (2008). *Optimal transport: old and new*. Springer Science & Business Media.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103.

Wainwright, M. J., Jaakkola, T., and Willsky, A. S. (2002). Tree-based reparameterization for approximate inference on loopy graphs. In *Advances in neural information processing systems*, pages 1001–1008.

Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. (2005). A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1058–1066.

Wang, D., Liu, H., and Liu, Q. (2018a). Variational inference with tail-adaptive f-divergence. In *Advances in Neural Information Processing Systems*, pages 5737–5747.

Wang, D. and Liu, Q. (2016). Learning to draw samples: With application to amortized MLE for generative adversarial learning. *arXiv preprint arXiv:1611.01722*.

Wang, S. and Manning, C. (2013). Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning*, pages 118–126.

Wang, T., Wu, Y., Moore, D., and Russell, S. J. (2018b). Meta-learning mcmc proposals. In *Advances in neural information processing systems*, pages 4146–4156.

Webb, S. and Teh, Y. W. (2016). A tighter Monte Carlo objective with Renyi alpha-divergence measures. In *NIPS workshop on Bayesian deep learning*.

Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *International Conference on Learning Representations*.

Wiegerinck, W. and Heskes, T. (2003). Fractional belief propagation. In *Advances in Neural Information Processing Systems*, pages 438–445.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Wingate, D. and Weber, T. (2013). Automated variational inference in probabilistic programming. *arXiv preprint arXiv:1301.1299*.

Winn, J. M. and Bishop, C. M. (2005). Variational message passing. In *Journal of Machine Learning Research*, pages 661–694.

Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. (2017). On the quantitative analysis of decoder-based generative models. In *International Conference on Learning Representations*.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Bethe free energy, Kikuchi approximations, and belief propagation algorithms. *Advances in Neural Information Processing Systems*.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2018). Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.

Zhou, S. and Wolfe, D. A. (2000). On derivative estimation in spline regression. *Statistica Sinica*, pages 93–108.

Zhou, Y., Shi, J., and Zhu, J. (2020). Nonparametric score estimators. In *Proceedings of the 37th International Conference on Machine Learning*.

Zhu, H. and Rohwer, R. (1995). Information geometric measurements of generalisation. Technical report, Technical Report NCRG/4350. Aston University.