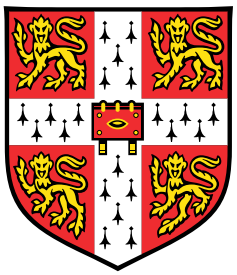
Approximate Inference: New Visions



Yingzhen Li

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Darwin College June 2018

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Yingzhen Li
June 2018

Acknowledgements

First and foremost, I would like to thank my supervisor, Richard E. Turner. Both of us took risks back in summer 2013: Rich just started his faculty career and was assembling his first team, and I was very new to the machine learning world. I will never forget the struggle in the first two years when I was desperate to make my algorithms work. Hence I want to express my deepest gratitude to Rich for his detailed guidance and patience until the "eureka" moment in spring 2015. Also I wouldn't have done many exciting projects later on if without his encouragement of exploration and generous allowance for independent research. So now looking back to the start of this journey, I realise that I was very fortunate to have decided to join Rich's group, and also the amazing Cambridge MLG/CBL.

I am indebted to my collaborators for their supports. José Miguel Hernández-Lobato co-worked with me on all the experimental details in Chapter 3 which largely serves as the basis of the first part. I learned a lot from Qiang Liu during the collaboration in my final year, which leads to plenty of inspiration in the second part of this thesis. Collaborating with Yarin Gal has quite substantial influence on my view of Bayesian deep learning. Special thanks also go to my co-authors during my PhD study: Thang Bui, Daniel Hernández-Lobato, Cuong Nguyen and Mark Rowland, all of them are knowledgable people to collaborate.

I also want to thank a few senior researchers in particular. I had very inspiring discussions with Zoubin Ghahramani and Sebastian Nowozin during my PhD viva, and I also appreciate Zoubin's efforts to constantly bringing in great speakers, and the precious schedule of 1-to-1 meeting with them. Martin Szummer and Ulrich Paquet, when they were both in Cambridge, provided advice when I was grinding message passing algorithms in the first two years. Stephan Mandt hosted my internship at Disney Research in Jun-Sept 2017, where I had a great summer there for both exciting research projects and enjoyable activities, including my first heavy metal concert.

I would like to thank all I have met in Cambridge for comfortable leisure time. I also thank the FFTF fellowship award from Schlumberger Foundation (2014, 2015, 2016).

I can never thank my family enough. My father kept reminding me to read books. My mother concerned a lot about my body health and leisure life. My sister Yinghong shared with me the joy and sorrow of tuning hyper-parameters of deep neural networks.

Abstract

Nowadays machine learning (especially deep learning) techniques are being incorporated to many intelligent systems affecting the quality of human life. The ultimate purpose of these systems is to perform automated decision making, and in order to achieve this, predictive systems need to return estimates of their confidence. Powered by the rules of probability, Bayesian inference is the gold standard method to perform coherent reasoning under uncertainty. It is generally believed that intelligent systems following the Bayesian approach can better incorporate uncertainty information for reliable decision making, and be less vulnerable to attacks such as data poisoning.

Critically, the success of Bayesian methods in practice, including the recent resurgence of Bayesian deep learning, relies on fast and accurate approximate Bayesian inference applied to probabilistic models. These approximate inference methods perform (approximate) Bayesian reasoning at a relatively low cost in terms of time and memory, thus allowing the principles of Bayesian modelling to be applied to many practical settings. However, more work needs to be done to scale approximate Bayesian inference methods to big systems such as deep neural networks and large-scale dataset such as ImageNet.

In this thesis we develop new algorithms towards addressing the open challenges in approximate inference. In the first part of the thesis we develop two new approximate inference algorithms, by drawing inspiration from the well known expectation propagation and message passing algorithms. Both approaches provide a unifying view of existing variational methods from different algorithmic perspectives. We also demonstrate that they lead to better calibrated inference results for complex models such as neural network classifiers and deep generative models, and scale to large datasets containing hundreds of thousands of data-points. In the second theme of the thesis we propose a new research direction for approximate inference: developing algorithms for fitting posterior approximations of arbitrary form, by rethinking the fundamental principles of Bayesian computation and the necessity of algorithmic constraints in traditional inference schemes. We specify four algorithmic options for the development of such new generation approximate inference methods, with one of them further investigated and applied to Bayesian deep learning tasks.

Publications

Some of the work in this thesis has been published in the following venues.

Section 2.3.3 adapts from:

- **Yingzhen Li** and Richard E. Turner.
  A Unifying Approximate Inference Framework from Variational Free Energy Relaxation.
  Advances in Approximate Inference workshop, NIPS, 2016.

Chapter 3 expands on:

- **Yingzhen Li**, José Miguel Hernández-Lobato and Richard E. Turner.
  Stochastic Expectation Propagation.
  Neural Information Processing Systems (NIPS), 2015.

Chapter 4 expands on:

- **Yingzhen Li** and Richard E. Turner.
  Rényi Divergence Variational Inference.
  Neural Information Processing Systems (NIPS), 2016.

Chapter 5 expands on:

- **Yingzhen Li** and Qiang Liu.
  Wild Variational Approximations.
  Advances in Approximate Inference workshop, NIPS, 2016.

Chapter 6 expands on:

- **Yingzhen Li** and Richard E. Turner.
  Gradient Estimators for Implicit Models.
  International Conference on Learning Representations (ICLR), 2018.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Can you tell whether a coin is fair or bent, given the following independent coin toss results:

head, tail, tail, head.

Maybe you would say "the coin is fair". But why, and how *confident* you are in your answer? Furthermore, if I offered you a bet on the next outcome of a coin flip, at what odds would you take that bet?

A Bayesian statistician can easily answer all of the above questions by following Bayesian decision theory [Berger, 2013; Bishop, 2006]. Before observing the coin flip results, he/she will first determine a *prior belief* on whether the coin is bent or not. Usually this belief is represented by two probability values $P(\text{fair})$ and $P(\text{bent})$ that sum to one. Then he/she builds a conditional probability – for example $P(\text{head}|\text{fair})$ is the probability of the coin flip "head" given that the coin is fair – to describe the *likelihood* of a fair/bent coin given an observation. After observing some simulation outcomes, he/she will adjust his/her *posterior belief* on the unknown properties of the coin using *Bayes' rule* [Bayes and Price, 1763; Laplace, 1820]:

$$P(\text{fair}|\text{coin flip results}) = \frac{P(\text{coin flip results}|\text{fair})P(\text{fair})}{P(\text{coin flip results})} \tag{1.1}$$

where in our example, the conditional and marginal probabilities are calculated by simply following the sum rule and product rule of probability:

$$P(\text{coin flip results}|A) = P(\text{head}|A)P(\text{tail}|A)P(\text{tail}|A)P(\text{head}|A), \quad A \in \{\text{fair}, \text{bent}\},$$

$$P(\text{coin flip results}) = P(\text{coin flip results}|\text{fair})P(\text{fair}) + P(\text{coin flip results}|\text{bent})P(\text{bent}).$$

The Bayesian statistician can finally answer the question, by picking the one with the largest *posterior probability*, i.e. the coin is fair if

$$P(\text{fair}|\text{coin flip results}) > P(\text{bent}|\text{coin flip results}),$$

or the coin is bent otherwise. Furthermore, he/she can tell us how confident he/she is, for example "the posterior probability of the coin being fair is 60%, so I am not very confident when saying the coin is fair". In general, by gathering more coin flip results (more than four simulation outcomes in this case), he/she continues adjusting the posterior belief for a fair/bent coin using Bayes' rule, and becomes more confident about the inferred answer. Having the inference result at hand, he/she can even predict the outcome of a future coin flip experiment, again with an uncertainty estimate, then decide whether he/she should accept the bet as well as the correct odds yielding a positive expected return.

The coin flip problem is just an elementary example of *Bayesian inference*, and in general these principles of inference and decision making apply to many real-world tasks as well. Powered by the rules of probability, Bayesian methods allow us to infer the unknown factors given the observed data, quantify the confidence of the inferred results, and make predictions with calibrated uncertainty estimates that support decision making. Critically, Cox [1946] argued that, reasoning under probability rules is the *only* way to perform coherent inference under uncertainty.[1] So ideally, Bayesian methods should be applied to any situation where well-calibrated inferences need to be made from data. These include prediction tasks such as classifying cat and dog images, and decision making tasks like determining the next action for a robot.

Despite having many desirable theoretical properties, Bayesian methods are less widely used in many exciting applications of artificial intelligence, especially those powered by deep learning [Goodfellow et al., 2016; LeCun et al., 2015; Schmidhuber, 2015] such as the AlphaGo system [Silver et al., 2016, 2017] that defeated human Go champion Ke Jie 3-0.[2] Although the Bayesian approach maintains a posterior distribution of all possible settings of the unknown factors which is desirable, it also requires computing the marginal probability of the observations (in our example $P(\text{coin flip results})$) that involves evaluating *all* possible settings of the neural network weights. Observing this, a deep learning practitioner might respond that "it takes *forever* to compute Bayes' rule for my task, so I'd better stick to a point estimate that minimises the training error." Two recent trends of deep learning applications

[1] Also see chapters 1 and 2 in Jaynes [2003].
[2] https://en.wikipedia.org/wiki/AlphaGo_versus_Ke_Jie

make the situation even worse for Bayesian inference: 1) the neural networks employed are getting deeper and wider [He et al., 2016; Huang et al., 2017, 2016], and 2) a typical dataset for deep learning tasks contains millions (if not billions) of instances [Abu-El-Haija et al., 2016; Deng et al., 2009].

But still, uncertainty quantification is crucial to achieving better deep learning. First, deep learning models are often over-parameterised, and using a point estimate can easily lead to over-fitting and poor predictive performance. Second, as nowadays deep learning techniques are being incorporated into many systems affecting the quality of human life (e.g. intelligent personal assistants, machine translation, even intelligence systems supporting autonomous driving and health care), it is crucial to make sure these systems know *how confident the model is when performing decision making*. Also, uncertainty information is essential to solving the exploration-exploitation trade-off, which helps learn a better agent faster in reinforcement learning and robotics applications [Deisenroth and Rasmussen, 2011; McAllister and Rasmussen, 2016]. Furthermore, Bayesian methods are the gold standard approaches for data efficiency, which is crucial to build a better prediction system for tasks that usually don't have *enough* labelled cases, and potentially present lots of missing values, e.g. medical applications.

Fortunately, if fast and accurate *approximation* schemes could be applied to the quantities that a Bayesian statistician would like to compute, then he/she can still perform (approximate) Bayesian inference and quantify the model uncertainty accordingly. Hence the primal challenge for Bayesian inference today is to design fast and accurate *approximate inference* algorithms for complex systems like neural networks and scale them to large datasets, which will be the main subject of the thesis. But before we delve into the development of such algorithms, in the rest of this introductory chapter I shall introduce the mathematical background for approximate inference, and identify fundamental research questions about this subject.

1.1 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.

1.1.1 Exact Bayesian inference as integration

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

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta})}{p(\mathcal{D})}, \tag{1.2}$$

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 and product 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 (1.2). Moreover, to predict the label $\mathbf{y}^*$ on unseen datum $\boldsymbol{x}^*$ a Bayesian statistician would compute the *predictive* distribution

$$p(\mathbf{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}, \tag{1.3}$$

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 statistics of the posterior, for example

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

[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 thesis.

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

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:

$$\int F(\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}, \tag{1.4}$$

and the goal of this thesis is to study how to perform this integration *pragmatically and efficiently*.

1.1.2 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 (1.4) 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 approximate the integral as

$$\int F(\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \approx \frac{1}{K} \sum_{k=1}^{K} F(\boldsymbol{\theta}_k). \tag{1.5}$$

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 (MCMC) [Gelman et al., 2014]. Unfortunately, in high dimensions these methods are likely to require a considerable number of samples (if the random variables are highly correlated), and the simulation time for MCMC can be prohibitively long (e.g. due to slow mixing).

[4] e.g. the prior is Gaussian and the neural network only has one hidden layer with ReLU activation. See Hernández-Lobato and Adams [2015] for details.

Now comes the brilliant idea of *approximate inference*: can we find another distribution $q(\boldsymbol{\theta})$ that makes the *computation* of 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 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

$$q^*(\boldsymbol{\theta}) = \arg\min_{q\in\mathcal{Q}} \mathrm{D}[q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})]. \tag{1.6}$$

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, 1959; Kullback and Leibler, 1951] which leads to the widely used *variational inference* algorithm [Beal, 2003; Ghahramani and Beal, 2000; Jordan et al., 1999]. In general an optimisation objective function $\mathcal{F}$ is designed to allow an accurate approximation to be obtained:

$$q^*(\boldsymbol{\theta}) = \arg\min_{q\in\mathcal{Q}} \mathcal{F}(q(\boldsymbol{\theta}); p(\boldsymbol{\theta}|\mathcal{D})), \tag{1.7}$$

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 constrained optimisation of 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 Chapter 2. Once q is obtained, at prediction time the Bayesian predictive distribution (1.3) is approximated by

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) \approx \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})q(\boldsymbol{\theta})d\boldsymbol{\theta}. \tag{1.8}$$

Other interesting quantities to be computed include the (approximated) *Bayesian averaged prediction* $\mathbf{y}^*_{\text{avg}} = \mathbb{E}_q[\mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{x}^*)]$ if $p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})$ is defined by a neural network $\mathbf{y}^*_{\text{pred}} = \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{x}^*)$ that is parametrised by $\boldsymbol{\theta}$.

[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 (other inference methods)**.** Not all statisticians and engineers agree with the Bayesian modelling paradigm. For example, in deep learning a dominating method for training neural networks is *loss function minimisation*, which first defines a loss function $\ell(\mathbf{y}, \hat{\mathbf{y}})$ between the true label $\mathbf{y}$ and the prediction $\hat{\mathbf{y}} = \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{x})$, then minimises the averaged loss computed on the dataset $\mathcal{D}$. Under fairly mild conditions we show in Section 2.5.2 that this corresponds to a *maximum likelihood estimation* (MLE) [Fisher, 1922] of the unknown parameters $\boldsymbol{\theta}$, which puts a uniform prior (which can be improper) on the weights $\boldsymbol{\theta}$. Also in many cases, adding regularisations such as the ℓ_2 or ℓ_1 regularisers corresponds to defining a prior distribution on $\boldsymbol{\theta}$, which turns the optimisation into a *maximum a posteriori* (MAP) problem. Both cases can be framed in the variational inference framework with the q distribution as a Dirac delta function.

Remark (direct approximations to the predictive distribution)**.** For Bayesian neural networks (introduced later) people are often more interested in the predictive distribution $p(\mathbf{y}^*|\boldsymbol{x}^*, \mathcal{D})$. Indeed, Snelson and Ghahramani [2005] and Korattikara et al. [2015] considered training a parametric model $\hat{p}(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\phi})$ to form a direct approximation to $p(\mathbf{y}^*|\boldsymbol{x}^*, \mathcal{D})$, where $p(\mathbf{y}^*|\boldsymbol{x}^*, \mathcal{D})$ is (approximately) computed with a carefully tuned MCMC sampler. The approximation is done by *distillation* [Hinton et al., 2015], which means the training data for the "student model" $\hat{p}(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\phi})$ is generated from the "teacher model" $p(\mathbf{y}^*|\boldsymbol{x}^*, \mathcal{D})$. Although this approximation is arguably more direct, with an explicit approximation $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$ one can perform many (approximate) integration tasks with different $F(\boldsymbol{\theta})$ functions at the same time.

Remark (a comparison to Bayesian quadrature)**.** Another important technique for approximating integrals is *Bayesian quadrature* [Ghahramani and Rasmussen, 2003; Kennedy and O'Hagan, 1996; O'Hagan, 1991], 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 thesis

we only study approximate inference methods and assume F is analytic and cheap to compute for a given configuration.

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

1.2 Questions to be answered for algorithmic design

We have just introduced the concept of approximate inference, but in a very vague way. Precisely what does an "accurate approximation" look like? In which sense can we claim the designed method is "computationally easier" than the exact integration problem? In general we need to answer the following questions when developing new variants of approximate inference algorithms:

Q1 What is the measure of "accuracy"?
It is crucial to define the accuracy measure for the specific tasks that approximate inference is applied to. For example Alice might care about the accuracy in terms of the (approximated) predictive likelihood (1.8), but instead Bob might want an accurate approximation to the model evidence $\log p(\mathcal{D})$ for model selection and/or hyper-parameter optimisation. Unfortunately, no approximate inference algorithm can provide satisfactory answers for any possible accuracy measure, hence users are encouraged to think about what they really want from an approximation procedure. In the rest of the thesis we answer this question by considering the (approximate) predictive likelihood, predictive error and the log marginal likelihood $\log p(\mathcal{D})$ ($\log p(\boldsymbol{x})$ in latent variable model settings) computed using the obtained approximate posterior.

Q2 Which optimisation procedure/objective function should be used?
Ideally we should directly minimise the error of approximation according to the choice of accuracy measure. For example, if the answer to Q1 is to obtain an approximate posterior with minimal error measure by a selected divergence, then minimising that divergence is arguably "the correct thing to do", and it would return the exact answer if the exact posterior is contained in the candidate distribution set $\mathbb{Q}$. However such a desired measure might be intractable to minimise, and most of the time we are forced to select an alternative, or a *surrogate* energy function due to tractability concerns. Hence Q2 can be rephrased as, given a set of "tractable" optimisation methods, how do we choose the best one of them such that an "accurate" (as defined in Q1) approximation could be obtained?

Q3 What does "computational efficiency" mean here? What constraints are present?
As discussed in Section 1.1.2, we resort to approximation because 1) analytical solutions are unavailable and 2) the computational resource is limited to perform numerical integration directly. The latter computational constraints include:

- *low time complexity* required by large-scale inference tasks or online algorithms that needs real-time inference;
- *low space complexity* that is crucial for the "big data, big model" settings (typically the case for deep learning);
- and *algorithmic tractability* that simply requires the objective function/gradients/-fixed point equations to be computable in a fast way, as these are sub-routines in the optimisation procedure.

Another practical concern might be, to what extent the designed algorithm could be incorporated into existing infrastructure, with minimal adjustment. For example, deep learning systems prefer gradient-based methods over fixed-point iterative updates, so that we should arguably prioritise gradient descent when designing approximate inference algorithms for them. These constraints are incorporated to the selection of both the inference algorithm and the approximate distribution family Q.

In principle, Q2 and Q3 should be addressed together. For example, new variational objectives are required when the approximate distribution contains mixture components [Jaakkola and Jordan, 1998; Maaløe et al., 2016; Ranganath et al., 2016b; Salimans et al., 2015; Tran et al., 2016]. In other words, the "tractability" in Q2 is mainly defined by the answers to Q3. But in many cases people simply select a widely used approximate inference method (e.g. variational inference) which eventually adds the *algorithmic* tractability constraints, and focus more on the design of approximate distributions. One of the main topics studied in this thesis is to remove as many as possible of these algorithmic restrictions, with the hope of enabling very flexible q distributions to be used for better inference results.

1.3 Thesis outline

The rest of the thesis is organised in two themes (or two parts):

I Understanding existing research: unifying variational methods.
This part of the thesis focuses on the optimisation procedures of existing approximate inference methods, and proposes generalised algorithms that provide unifying views.

Chapter 2 summarises existing literature on popular approximate inference algorithms and applications. Then in Chapters 3 and 4, two unifying views of variational methods will be presented from different perspectives, in order to both provide a comprehensive understanding and enable wider applications to Bayesian deep learning.

II Proposing a new research direction: wild approximate inference.
In this theme the focus turns to the discussion of approximate inference algorithms that suit complex approximations. In Chapter 5 I will revisit the principles of approximate inference again, and discuss the importance of developing new approximate inference methods that allow the use of *implicit* approximations or implicit sampling procedures. In Chapter 6 I will demonstrate with a concrete example how we can train this type of approximation, and demonstrate how this technique enables new applications of approximate inference such as to meta-learning.

Finally Chapter 7 concludes the thesis and discusses future directions of research.

To make the presentation concise I will move all derivation and proof details into Appendix A (except those are crucial to the presentation). Optional materials for further reading are also included in Appendix B. Additional comments and discussions are also presented as "remark" paragraphs just as readers might have seen in previous pages. All these materials can be safely skipped for first reading.

Part I

Unifying Variational Methods

Chapter 2

Divergences, Algorithms and Applications

To kickstart the discussion of variational methods unification, I provide a condensed introduction to two classes of well-known approximate inference techniques. I will first start by discussing statistical divergence measures since both of them are closely related to divergence minimisation. Then I will review the two methods in detail. Finally, I will touch on two main learning tasks in Bayesian deep learning as the primary applications of the approaches developed in this thesis.

2.1 Statistical divergences for probability distributions

Many approximate inference algorithms measure the approximation quality by considering the "closeness" between the target and the approximation. In this thesis, we will mainly focus on the case where both the target and the approximation are expressed by probability distributions. Then the concept of "closeness" is established as *divergence*. Before introducing the formal definition, we briefly discuss the definition of the *probability density function* (PDF)[1] as preparation.

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 a probability distribution P in interest which is defined on Σ, we can define its *probability density function* p by $dP = pd\mu$.[2] For

[1] For discrete case we refer PDF as probability *mass* functions (PMF).

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

simplicity in the rest of the thesis 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 $p \in \mathcal{P}$.

With all the preparation above we now provide a formal definition of divergence.

Definition 2.1. *(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 $\mathcal{D}[p||q] \geq 0$ for all $p, q \in \mathcal{P}$, and $\mathcal{D}[p||q] = 0$ iff. $p = q$.*

This definition 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. Hence there exist many available divergences to use, and in this section we review some of the popular choices. We start from the well-known Kullback-Leibler (KL) divergence and discuss its properties and applications. Then we move to a more general case and review α-divergences which will be the main divergence tools for the algorithms developed in the first part of the thesis. In Appendix B.1 we also briefly touch on two very general cases, f-divergence and Bregman divergence, and discuss their connections to α-divergences.

2.1.1 Kullback-Leibler (KL) divergence

Kullback-Leibler divergence [Kullback, 1959; Kullback and Leibler, 1951], 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 2.2. *(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*

$$\mathrm{KL}[p||q] = \int p(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}, \quad p, q \in \mathcal{P}, \tag{2.1}$$

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 (see Appendix B.5) we have (2.1) 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 (2.1) 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 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 (2.1) 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".

Later we will see how these two KL divergences have been applied to approximate inference algorithms such as the widely used variational inference [Beal, 2003; Jordan et al., 1999] and expectation propagation [Minka, 2001b]. But here we switch the topic to maximum likelihood estimation (MLE) [Fisher, 1922] for a moment, in which we will show that MLE is also 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:

$$\hat{\boldsymbol{\theta}}^{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}\in\Theta} \frac{1}{N}\sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}). \tag{2.2}$$

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 (Section 2.5.1).

2.1.2 Amari's α-divergences

Besides the KL divergence, do we have other choices of divergences for approximate inference? Certainly there are, and here we review a rich class of them called α-*divergences*. Interestingly there exist multiple (slightly) different definitions of α-divergences, and in the sequel we will formally introduce the version that will be the focus of this chapter. Before that I provide a short (and possibly incomplete) history for these developments in below.

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 an integral. That integral has later been referred as the *Chernoff* α-coefficient

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

which is used in all later variants of α-divergences.

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.[3] These two quantities are now referred to *Rényi entropy* and *Rényi divergence*, respectively, and the latter will be employed as the divergence tool in Chapter 4. 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 Amari [1985]; Efron [1975, 1978], 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 $\alpha \in \mathbb{R}$.

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

Definition 2.3. *(Amari's α-divergence) Amari's α-divergence* $\mathrm{D}_\alpha^A[\cdot||\cdot] : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$*, parameterised by* $\alpha \in \{\alpha : \mathcal{D}_\alpha^A[p||q] < +\infty\}$*, is defined as*

$$\mathrm{D}_\alpha^A[p||q] = \frac{4}{1-\alpha^2}\left(1 - \int p(\boldsymbol{\theta})^{\frac{1+\alpha}{2}} q(\boldsymbol{\theta})^{\frac{1-\alpha}{2}} d\boldsymbol{\theta}\right), \quad \alpha \neq \pm 1, \tag{2.3}$$

$$\mathrm{D}_1^A[p||q] := \lim_{\alpha \to 1} \mathrm{D}_\alpha^A[p||q] = \mathrm{KL}[p||q], \quad \alpha = 1, \tag{2.4}$$

$$\mathrm{D}_{-1}^A[p||q] := \lim_{\alpha \to -1} \mathrm{D}_\alpha^A[p||q] = \mathrm{KL}[q||p], \quad \alpha = -1. \tag{2.5}$$

In Amari's career he used α-divergence as a tool to study the geometry of distribution manifolds, and in particular, he claimed in Amari [2009] that his definition is the only divergence that belongs to both the f-divergences [Csiszár, 1963] (related to information theory) and Bregman divergences [Bregman, 1967] (related to geometry). These properties are not directly related to approximate inference, and we only discuss them in Appendix B.1.

Remark (Other α-divergence definitions)**.** There exist other definitions of α-divergence, and some of them are detailed here.

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

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

  By continuity in α we can show that $\lim_{\alpha \to 1} \mathrm{D}_\alpha^R[p||q] = \mathrm{KL}[p||q]$. We defer the detailed introduction of Rényi's definition to Section 4.1.

- 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 $\lim_{\alpha \to 1} \mathrm{D}_\alpha^T[p||q] = \mathrm{KL}[p||q]$.

2.2 Variational inference with KL-divergence

It seems from the introduction of the divergences above that divergence minimisation is an excellent idea to obtain an accurate approximation to the target distribution, where the measure of "accuracy" is also represented by the choice of divergence. However 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

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

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

2.2.1 Kullback-Leibler divergence and variational free-energy

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):

$$\arg\min_q \mathrm{KL}[q(\boldsymbol{\theta})||p(\boldsymbol{\theta})] = \arg\min_q \mathcal{F}_{\mathrm{VFE}}(q;p),$$

$$\mathcal{F}_{\mathrm{VFE}}(q;p) := \mathrm{KL}[q(\boldsymbol{\theta})||p(\boldsymbol{\theta})] - \log Z = \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p^*(\boldsymbol{\theta})} d\boldsymbol{\theta}. \tag{2.6}$$

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

$$\mathcal{L}_{\mathrm{VI}}(q;p) := -\mathcal{F}_{\mathrm{VFE}}(q;p) = \int q(\boldsymbol{\theta}) \log \frac{p^*(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}. \tag{2.7}$$

In posterior inference context (i.e. $p^*(\boldsymbol{\theta}) = p(\mathcal{D}, \boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta}) p_0(\boldsymbol{\theta})$) the following two formulations of the variational lower-bound have also been considered:

$$\mathcal{L}_{\mathrm{VI}}(q;p) = \mathbb{E}_q[\log p(\mathcal{D}|\boldsymbol{\theta})] - \mathrm{KL}[q(\boldsymbol{\theta})||p_0(\boldsymbol{\theta})], \tag{2.8}$$

$$\mathcal{L}_{\mathrm{VI}}(q;p) = \mathbb{E}_q[\log p(\mathcal{D}, \boldsymbol{\theta})] + \mathbb{H}[q(\boldsymbol{\theta})]. \tag{2.9}$$

where $\mathbb{H}[q(\boldsymbol{\theta})]$ is the Shannon entropy of the q distribution.

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:

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

Here Jensen's inequality (see appendix B.5) 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.

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) to Bayesian neural networks [Hinton and Van Camp, 1993; Peterson and Anderson, 1987] that will be surveyed in the application section. Especially since the development of Peterson and Anderson [1987], mean-field approximation started to be an attractive alternative to sampling methods for probabilistic inference in graphical models [Ghahramani, 1995; MacKay, 1997].

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; Beal, 2003; Ghahramani and Beal, 2000, 2001; Sato, 2001] that is used to perform posterior approximations of the model parameters and even model selection.

2.2.2 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 referred 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

$$q(\boldsymbol{\theta}) := \prod_{i=1}^{D} q_i(\theta_i). \tag{2.10}$$

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 (2.10). 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 (2.10) into (2.6) (and use $\boldsymbol{\theta}_{\neq j}$ to denote all the θ_i variables except θ_j):

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

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

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

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

$$q_j(\theta_j) = \frac{\exp\left[\int \prod_{i\neq j} q_i(\theta_i) \log p^*(\boldsymbol{\theta}) d\boldsymbol{\theta}_{\neq j}\right]}{\int \exp\left[\int \prod_{i\neq j} q_i(\theta_i) \log p^*(\boldsymbol{\theta}) d\boldsymbol{\theta}_{\neq j}\right] d\theta_j}. \tag{2.11}$$

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 (2.11):

$$\begin{aligned} \log q_1(\theta_1) &= \int q_2 \log p(\boldsymbol{\theta}, \mathcal{D}) d\theta_2 + \text{const} \\ &= \mathbb{E}_{q_2}\left[-\frac{1}{2}(\theta_1 - \mu_1)^2 \Lambda_{11} - (\theta_1 - \mu_1)\Lambda_{12}(\theta_2 - \mu_2)\right] + \text{const} \\ &= -\frac{1}{2}\theta_1^2 \Lambda_{11} + \theta_1 \mu_1 \Lambda_{11} - \theta_1 \Lambda_{12}(\mathbb{E}_{q_2}[\theta_2] - \mu_2) + \text{const} \\ &:= \log \mathcal{N}(\theta_1; m_1, \lambda^{-1}) + \text{const} \end{aligned} \tag{2.12}$$

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 again that we do not assume the approximation to be a Gaussian distribution in order to obtain the last equation in (2.12). Rather the Gaussian distribution solution came out from the derivation of the global optimum (2.11) 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 $q_1(\theta_1) = \mathcal{N}(\theta_1; \mu_1, \Lambda_{11}^{-1})$, and similarly $q_2(\theta_2) = \mathcal{N}(\theta_1; \mu_2, \Lambda_{22}^{-1})$ as the unique global optimum of variational mean-field approximation. A visualisation of the mean-field approximation is provided in Figure 2.1. 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

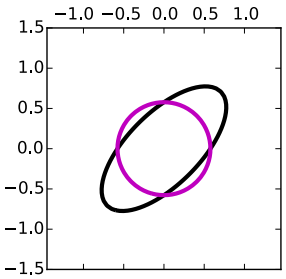


Fig. 2.1 Mean-field approximation to the exact posterior distribution in the Bayesian linear regression example (with one-sigma contours). The exact posterior contour is shown in black and the variational approximation is in purple.

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.

2.2.3 Monte Carlo variational inference

So far we have discussed variational inference algorithms for linear regression. But real world problems are much more complicated. Often, the term $\mathbb{E}_q[\log p^*(\boldsymbol{\theta})]$ 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 $p^*(\boldsymbol{\theta}) := p(\boldsymbol{\theta}, \mathcal{D})$ is proportional to the product of many likelihood functions, and evaluating $\mathbb{E}_q[\log p(\mathcal{D}|\boldsymbol{\theta})]$ 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 have attempted to do so, e.g. see Gershman et al. [2012]; 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" 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).

To see how the algorithm 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:

$$\mathcal{L}_{\text{VI}}(q;p) = \sum_{n=1}^{N} \mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})] + \mathbb{E}_q[\log p_0(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta})], \tag{2.13}$$

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

$$\mathbb{E}_q[\log p(\boldsymbol{x}_n|\boldsymbol{\theta})] \approx \frac{1}{K}\sum_{k=1}^{K} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}^k), \quad \boldsymbol{\theta}^k \sim q(\boldsymbol{\theta}). \tag{2.14}$$

This forms an *unbiased* estimation, and under mild assumptions (detailed in Chapter 4), the RHS term in (2.14) converges to the exact expectation value as $K \to +\infty$. The KL-divergence 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. In summary, with this "black-box" approach, one can approximate the variational lower-bound as

$$\mathcal{L}_{\text{VI}}^{\text{MC}}(q;p) = \frac{N}{|\mathcal{S}|}\sum_{n\in\mathcal{S}}\frac{1}{K}\sum_{k}\log p(\boldsymbol{x}_n|\boldsymbol{\theta}^k) + \frac{1}{K}\sum_{k}[\log p_0(\boldsymbol{\theta}^k) - \log q(\boldsymbol{\theta}^k)], \quad \boldsymbol{\theta}^k \sim q(\boldsymbol{\theta}), \tag{2.15}$$

and compute stochastic gradient descent on the MC approximation (2.15) with mini-batch $\mathcal{S} \sim \mathcal{D}^{|\mathcal{S}|}$.

Remark (MC samples for different observations)**.** In the MC approximation (2.15) 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 [Gal, 2016; Srivastava et al., 2014].

Remark (variance reduction for MC-VI)**.** MC-VI relies on stochastic gradient descent/ascent for optimisation, and the gradient of (2.15) wrt. the variational parameters $\boldsymbol{\phi}$ could have very high variance, especially when the latent variable is discrete. Thus recent work has been focused on variance reduction, often employing a pre-computed quantity correlated with the stochastic gradient as a *control variate* [Ross, 2002]. Discussions of

these methods are out of the scope of the thesis. Interested readers are pointed to e.g. Gu et al. [2016]; Mnih and Gregor [2014]; Paisley et al. [2012] for more details.

2.2.4 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, which is typically the case for *latent variable models* – probabilistic models that have unobserved variables attached to each data instance. 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 sub-routine. These issues had restricted the extensions of VI to many interesting cases, until the introduction of *amortised inference* that is detailed in below.

Let us start from the mean-field approximation example we had in Section 2.2.2 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; Roweis and Ghahramani, 1999]. 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 observed input-output pairs $(\boldsymbol{x}_n, y_n)$, 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 Section 2.2.2 we have the optimal solutions satisfying

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

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 [Kingma and Welling, 2014; Rezende et al., 2014; Salimans and Knowles, 2013], 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;
- generalisable to unseen observations, as the learned variational parameters only capture the global structure of the variational approximation;
- a good initialisation of q for unseen data, which will then be refined[4] by e.g. VI [Kim et al., 2018; Marino et al., 2018] or MCMC [Hoffman, 2017; Stuhlmüller et al., 2013].

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

[4] Usually amortised inference returns suboptimal approximation to each individual $p(\boldsymbol{z}_n|\boldsymbol{x}_n, y_n)$, – see Cremer et al. [2018] for a quantitative analysis – so refinements upon the amortised distribution are often useful.

designed using invertible transform [Kingma et al., 2016; Rezende and Mohamed, 2015]. Solutions for these problems are further discussed in the second part of the thesis.

Remark (a misconception of amortised inference)**.** Amortised inference is sometimes misunderstood as being equivalent to the variational auto-encoder [Kingma and Welling, 2014; Rezende et al., 2014] approach (discussed later) 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. $\boldsymbol{z},\boldsymbol{x} \sim p(\boldsymbol{z},\boldsymbol{x})$. They also refer $q(\boldsymbol{z}|\boldsymbol{x})$ as the "recognition model". Morris [2001] further applied the sleep step update to learn an approximation to the conditional distributions of a directed graphical model. We note that in both cases, there is no guarantee that q approximate the exact posterior well as it never observes real-world data in the sleep step. In Gaussian process (GP) literature, amortised inference has been applied to GP latent variable models (GPLVMs) to infer the latent variables, 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; Paige and Wood, 2016] and MCMC [Li et al., 2017].

2.3 Expectation propagation with α-divergences

This section reviews another important class of approximate inference algorithms: expectation propagation (EP)[Minka, 2001b; Opper and Winther, 2005]. EP can be viewed as a generalisation of the well-known sum-product algorithm and belief propagation [Pearl, 1982, 1988] for computing marginal distributions of a probabilistic (graphical) model. Also by generalising EP to minimising alpha-divergences, a broad class of approximate inference algorithms – including variational inference – is recovered. Many topics detailed below can also be founded in Koller and Friedman [2009]; Wainwright and Jordan [2008].

2.3.1 Factor graphs and exponential families

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 $P(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3)$

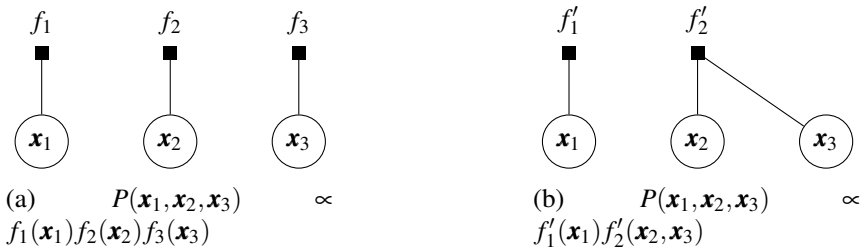


Fig. 2.2 Two different factor graph representations for the same probability distribution, if defining $f'_1 = f_1$, $f'_2 = f_2 f_3$.

which can be factorised into $P(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3) = f_1(\boldsymbol{x}_1)f_2(\boldsymbol{x}_2)f_3(\boldsymbol{x}_3)$, illustrated in Figure 2.2. To give a formal definition, a factor graph is a bipartite graph between variable nodes (circles) and factor nodes (squares), where a factor node f is connected to a variable node $\boldsymbol{x}_i$ iff. $\boldsymbol{x}_i$ 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 $P(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3) = f'_1(\boldsymbol{x}_1)f'_2(\boldsymbol{x}_2,\boldsymbol{x}_3)$, where $f'_1(\boldsymbol{x}_1) = f_1(\boldsymbol{x}_1)$ and $f'_2(\boldsymbol{x}_2,\boldsymbol{x}_3) = f_2(\boldsymbol{x}_2)f_3(\boldsymbol{x}_3)$. Factor graphs enable fast marginal computations using the sum-product algorithm, and in later sections we will see that the EP update procedure also depends on the factor graph structure directly. Therefore, factor graphs can also be viewed as "hyper-parameters" specified by the user, and selecting the appropriate factorisation is key to both the approximation accuracy and the computational efficiency. For simplicity in the rest of this chapter we assume the selected factorisation suits well for the learning method. To reduce notation clutter we use $\boldsymbol{x}_a$ to denote a subset of random variables $\{\boldsymbol{x}_i\}$ connected to a factor node f_a.

Exponential family distributions

Another important concept we introduce here is the *exponential family* distribution [Koopman, 1936]. As a motivating example, consider fitting a probabilistic model p to some empirical distribution $\hat{p}_{\mathcal{D}}$ by matching some statistical quantity $\boldsymbol{\Phi}(\boldsymbol{\theta})$. There exists infinitely many solutions for this problem, and here we would prefer the solution which has the maximum *entropy* $\mathbb{H}[p] = -\int p(\boldsymbol{\theta})\log p(\boldsymbol{\theta})d\boldsymbol{\theta}$. This means the solution produces samples that have the same statistics as the data distribution, but also retains maximal uncertainty (in terms of the entropy). Solving this constrained optimisation algorithm reveals that the optimal fit has an exponential family form as follows.

Definition 2.4. *(Exponential family) An exponential family distribution is defined as*

$$p(\boldsymbol{\theta}) = \frac{1}{Z}h(\boldsymbol{\theta})\exp\left[\boldsymbol{\lambda}^T\boldsymbol{\Phi}(\boldsymbol{\theta})\right], \tag{2.16}$$

where $h(\boldsymbol{\theta})$ is the base measure, $\boldsymbol{\lambda}$ is the natural parameter or canonical parameter, and $\boldsymbol{\Phi}(\boldsymbol{\theta})$ is the sufficient statistic.

The natural parameter $\boldsymbol{\lambda}$ of the maximum entropy solution is selected to satisfy the constraint $\mathbb{E}_p[\boldsymbol{\Phi}(\theta)] = \mathbb{E}_{\hat{p}_\mathcal{D}}[\boldsymbol{\Phi}(\theta)]$. If the exponential family is *regular*, i.e. the components of $\boldsymbol{\Phi}$ are linearly independent, then there exists a one-to-one mapping between the natural parameter and the expectation of the sufficient statistics $\boldsymbol{\mu} := \mathbb{E}_p[\boldsymbol{\Phi}(\boldsymbol{\theta})]$. Thus $\boldsymbol{\mu}$ is also named *moment parameter* or *mean parameter* of an exponential family distribution. For notational simplicity we also assume the base measure $h(\boldsymbol{\theta}) = 1$ and the above exponential family is regular. Hence the partition function Z is a function of the natural parameter $\boldsymbol{\lambda}$ (and equivalently a function of the moment parameter $\boldsymbol{\mu}$ as well) and we will also write $Z = Z(\boldsymbol{\lambda}) = Z(\boldsymbol{\mu})$ when necessary.

Here we include some important properties of exponential family distributions [Wainwright and Jordan, 2008], which can be proved by simple calculations.

Proposition 2.1. *A regular exponential family distribution (2.16) satisfies the following:*
(1) $\log Z(\boldsymbol{\lambda})$ *is convex in* $\boldsymbol{\lambda}$*;*
(2) $\boldsymbol{\mu} = \nabla_{\boldsymbol{\lambda}} \log Z(\boldsymbol{\lambda})$*, and* $Cov_p[\boldsymbol{\Phi}] = \nabla\nabla_{\boldsymbol{\lambda}} \log Z(\boldsymbol{\lambda})$*;*
(3) The Fenchel dual $(\log Z)^*(\boldsymbol{\mu}) = -\mathbb{H}[\boldsymbol{\mu}]$*, where* $\mathbb{H}[\boldsymbol{\mu}]$ *denotes the entropy of the exponential family distribution (2.16) with moment parameter* $\boldsymbol{\mu}$*.*

The last property is particularly interesting in variational inference context. Recall the Fenchel duality equation $\log Z(\boldsymbol{\lambda}) = \max_{\boldsymbol{\nu}} \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\nu} + \mathbb{H}[\boldsymbol{\nu}]$, where the maximum is obtained at $\boldsymbol{\nu}^* = \boldsymbol{\mu}$. This means if the target distribution belongs to some complicated exponential family[5] and the q distribution has moments $\boldsymbol{\nu}$, then the Fenchel duality equation is exactly the optimisation problem of VI (2.6). In particular, if q also belongs to the same exponential family, then the optimal approximation can be obtained by matching the moments $\boldsymbol{\nu} \leftarrow \boldsymbol{\mu}$, thus $q = p$.

2.3.2 Expectation propagation

Now we consider a distribution $p(\boldsymbol{\theta}) \propto \prod_a \tilde{f}_a(\boldsymbol{\theta}_a)$, where $\tilde{f}_a$ are the factors in the corresponding factor graph with associated variables $\boldsymbol{\theta}_a$. A prevalent example for such a distribution is the exact posterior, where now $\tilde{f}_a$ represents either the prior distribution or the likelihood function associated with a datum. The marginal probability of a subset of $\boldsymbol{\theta}$ can be complicated. The highly successful Expectation Propagation (EP) [Minka, 2001b] algorithm approximates these exact marginals by another factor graph with a collection of simpler

[5] Notice that we can also write the target distribution as $p(\boldsymbol{\theta}) = \frac{1}{Z}\exp\left[1 \cdot \log p^*(\boldsymbol{\theta})\right]$.

functions $q(\boldsymbol{\theta}) \propto \prod_a f_a(\boldsymbol{\theta}_a)$. It iteratively updates the approximating factors $\tilde{f}_a$ through the "exclusion-moment matching-inclusion" procedure, detailed in Algorithm 1.

To summarise the algorithm, we first define the "leave-one-out", or *cavity distribution*[6]

$$q_{-a} \propto \prod_{b \neq a} f_b(\boldsymbol{\theta}_b) \propto q(\boldsymbol{\theta}) / f_a(\boldsymbol{\theta}_a), \tag{2.17}$$

which is computed by multiplying all the other factors except the selected one. Also an ordering of this factor selection is called a *schedule* of the EP algorithm, and in the following we assume it is random. The second step is to compute the *tilted distribution* by inserting back the true factor $\tilde{f}_a$ that f_a approximates:

$$\tilde{p}_a(\boldsymbol{\theta}) \propto q_{-a}(\boldsymbol{\theta}) \tilde{f}_a(\boldsymbol{\theta}_a), \tag{2.18}$$

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 exponential family approximations, we solve this optimisation problem 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 $\boldsymbol{\Phi}(\boldsymbol{\theta}) = (\Phi_1(\boldsymbol{\theta}), ..., \Phi_d(\boldsymbol{\theta}))$, then we write the q distribution as

$$q(\boldsymbol{\theta}) \propto \exp(\boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{\Phi}(\boldsymbol{\theta})), \tag{2.19}$$

where $\boldsymbol{\lambda}$ denote the natural parameters. Zeroing the gradient of $\mathrm{KL}[\tilde{p}_a || q]$ wrt. $\boldsymbol{\lambda}$ (and viewing $\tilde{p}_a$ as constant) returns

$$\mathbb{E}_q [\boldsymbol{\Phi}(\boldsymbol{\theta})] = \mathbb{E}_{\tilde{p}_a} [\boldsymbol{\Phi}(\boldsymbol{\theta})], \tag{2.20}$$

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

$$\mathrm{proj}[p] = \arg\min_q \mathrm{KL}[p || q], \tag{2.21}$$

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

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

Algorithm 1 Expectation Propagation (without damping)

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

of $q^* = \text{proj}[\tilde{p}_a]$ we recover the update of f_a by

$$f_a(\boldsymbol{\theta}_a) \leftarrow q^*(\boldsymbol{\theta})/q_{-a}(\boldsymbol{\theta}). \tag{2.22}$$

To improve convergence damping updates can be applied to step 4 in Algorithm 1 as

$$f_a(\boldsymbol{\theta}_a) \leftarrow f_a(\boldsymbol{\theta}_a)^{1-\varepsilon} \left(\text{proj}[\tilde{p}_a]/q_{-a}(\boldsymbol{\theta})\right)^{\varepsilon}, \tag{2.23}$$

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:

$$q(\boldsymbol{\theta}) \leftarrow q_{-a}(\boldsymbol{\theta})f_a(\boldsymbol{\theta}_a). \tag{2.24}$$

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 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 until the development of Chapter 4. Otherwise, EP does *not* minimise the inclusive KL divergence to the target distribution. Thus 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.

Why EP works

In general, to obtain an accurate approximation for some complicated distribution is a difficult task, as estimating the approximate functions *jointly* is often intractable. Factor

graphs provide not only a factorised representation of the true distribution but also a guide for choosing the function families for approximation. Consider the minimisation task $q^* = \arg\min_q \mathrm{D}[p||q]$.[7] Below we discuss 3 difference methods for the optimisation problem, where we understand them as different schedules.[8]

The simplest way is independent factorised approximation, i.e. to approximate each factor independently. This returns a fast and parallelisable algorithm that a simple map-reduce scheme can be applied to. However, is it common that factors share random variables, in which the marginals of the selected factor's random variables can deviate from that factor's function value. From the nature of product we know that small errors of each factor approximation can easily accumulate to large errors of the final result q.

The statistics and control communities have noted the importance of including the dependence in approximation, resulting in the well-known assumed density filtering (ADF) algorithm [Maybeck, 1982]. It can be viewed as a restricted version of factorised approximation by greedily adding in more factors to the q distribution such that $q(\boldsymbol{\theta})f_a(\boldsymbol{\theta})$ matches $q(\boldsymbol{\theta})\tilde{f}_a(\boldsymbol{\theta})$ as close as possible. One simple example would be approximating $p(\boldsymbol{\theta}) = \tilde{f}_1(\boldsymbol{\theta})\tilde{f}_2(\boldsymbol{\theta})\tilde{f}_3(\boldsymbol{\theta})$ with $q(\boldsymbol{\theta}) = f_1(\boldsymbol{\theta})f_2(\boldsymbol{\theta})f_3(\boldsymbol{\theta})$. Assume we start from an accurate estimation of $f_1(\boldsymbol{\theta}) \approx \tilde{f}_1(\boldsymbol{\theta})$ (where now $q(\boldsymbol{\theta}) = f_1(\boldsymbol{\theta})$), ADF in current iteration incorporates $f_2(\boldsymbol{\theta})$ (or $f_3(\boldsymbol{\theta})$ depending on the schedule) to the q distribution, updates $f_2(\boldsymbol{\theta})$ such that it minimises $\mathrm{D}[q(\boldsymbol{\theta})\tilde{f}_2(\boldsymbol{\theta})||q(\boldsymbol{\theta})f_2(\boldsymbol{\theta})]$, and includes the new approximation $f_2(\boldsymbol{\theta})$ to the new $q(\boldsymbol{\theta}) \leftarrow f_1(\boldsymbol{\theta})f_2(\boldsymbol{\theta})$. The disadvantage of this approach is its sensitivity to the choice of the schedule, especially when a subset of variables is shared by most of the factors.

EP tries to address the problem of sensitivity to scheduling by looping through the factors repeatedly, giving a chance for inaccurate factors to "correct themselves". This puts further restrictions to the mean-field update by including all the factors in moment matching. The precision may oscillate as the approximation in the first few loops can be very poor, especially considering EP is reduced to ADF in the first iteration if initialising all the approximating factors as 1 (which is often the case). Also there is no guarantee for EP to converge. But EP (and in general message passing) is still an efficient algorithm, especially when applying to only a subset of factors (see the application to error-correcting codes [Peterson and Weldon, 1972] and TrueSkill algorithm [Herbrich et al., 2006]).

[7] Here the measurement can be any distance or divergence, each returns different benefits. In EP the divergence measure is the KL divergence.

[8] This is an explanatory discussion rewritten from my first year report.

Algorithm 2 Power EP with fraction α

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

Linking power EP and α-divergence

An extension of EP is power EP [Minka, 2004], which excludes a fraction of the approximation (i.e. $f_a(\boldsymbol{\theta})^\alpha$) 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 $f_a(\boldsymbol{\theta})^\alpha$ directly before recovering the update for $f_a(\boldsymbol{\theta})$.

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

$$\mathrm{D}_\alpha^M[p||q] = \frac{1}{\alpha(1-\alpha)}\left(1 - \int p(\boldsymbol{\theta})^\alpha q(\boldsymbol{\theta})^{1-\alpha} d\boldsymbol{\theta}\right) \tag{2.25}$$

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 $\mathrm{KL}[q||p] = \lim_{\alpha\to 0}\mathrm{D}_\alpha^M[p||q]$ 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 (2.19), and we fix $\tilde{p}_a(\boldsymbol{\theta})$ as the target. Then when $\alpha \neq 0, 1$,

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

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

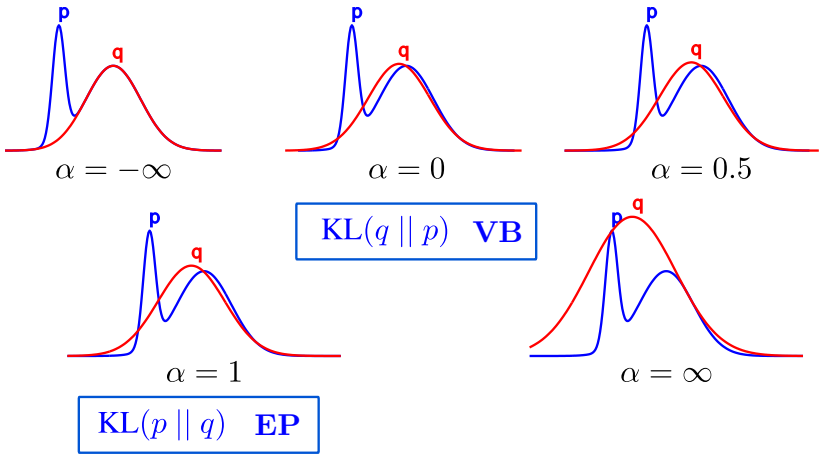


Fig. 2.3 An illustration of approximating distributions by α-divergence minimization. Here p and q shown in the graphs are unnormalized probability densities. Reproduced from Minka [2005]. Best viewed in color.

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.

Different α-divergences show different characteristics, and the family includes the two KL-divergences. Variational methods minimise the exclusive KL-divergence and provides a lower bound on model evidence. Furthermore, the exclusive KL-divergence has an unique advantage in that local optimisation can return optima of global approximation. However, Bayesian predictions often require just the marginals, and the inclusive KL-divergence is the only α-divergence which preserves them. Hence standard EP is preferred, which searches the optimisers with invariant sufficient statistics locally at $\boldsymbol{\theta}_a$.

To understand how the choice of other α values might affect the result of approximate inference, consider the problem of approximating a complicated distribution p with a tractable Gaussian distribution q by minimizing $\mathrm{D}_\alpha^M[p||q]$. The resulting (unnormalized) approximations obtained for different values of α are visualized in Figure 2.3. This shows that when α is a large positive number the approximation q tends to cover all the modes of p, while for $\alpha \to -\infty$ (assuming the divergence is finite) q is attracted to the mode with the largest probability mass. The optimal setting of α might reasonably be expected to depend on the learning task that is being considered.

Setting aside the analytic tractability of the computations, we note that the minimisation of a global α-divergence might not always be desirable. If the true posterior has many modes, then when a Gaussian approximation is deployed, a global approximation of this flavour that

is refined using $\alpha \geq 1$ will cover the modes, and can place substantial probability in the area where the true posterior has low probability (see the last plot in Figure 2.3).

2.3.3 EP energy: a primal-dual story

EP has been criticised for not having convergence guarantees and instead being a heuristic for posterior approximation. On the other hand, it has been shown to have superior performance compared to VI on a variety of tasks, e.g. Gaussian Process classification [Kuss and Rasmussen, 2005]. To understand these seemingly contradicting observations, here we provide an energy minimisation view of EP,[9] and discuss why EP could potentially lead to better approximations, and why it is difficult to prove the convergence of EP. Similar derivations are also available in Heskes and Zoeter [2002]; Minka [2005]; Opper and Winther [2005]; Wainwright and Jordan [2008]; Yedidia et al. [2001], and some extensions to handling latent variable models are provided in appendix B.2.

To streamline the notation, we consider approximating the intractable posterior $p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{Z} p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta})$ as a running example. In this case we denote $\tilde{f}_n(\boldsymbol{\theta}) = p(\boldsymbol{x}_n|\boldsymbol{\theta})$ and leave the prior $p_0(\boldsymbol{\theta})$ as it is. Using this notation we again write down the objective function of Variational inference (VI) called the *variational free energy* (VFE) [Beal, 2003; Jordan et al., 1999]:

$$\min_{q\in\mathcal{Q}} \mathrm{KL}[q||p] \Leftrightarrow \min_{q\in\mathcal{Q}} \mathcal{F}_{\mathrm{VFE}}(q) = \mathbb{E}_q \left[\log q(\boldsymbol{\theta}) - \log p_0(\boldsymbol{\theta}) - \sum_{n=1}^{N} \log \tilde{f}_n(\boldsymbol{\theta}) \right]. \tag{2.26}$$

From VFE to Bethe Free Energy

First we make use of the additivity of logarithm to rewrite an equivalent optimisation problem (recall that $\mathcal{P}$ is the space of all probability distributions):

$$\begin{aligned} \min_{q\in\mathcal{Q}} \mathcal{F}_{\mathrm{VFE}}(q) &= \min_{q\in\mathcal{Q}} \mathrm{KL}[q||p_0] - \sum_n \mathbb{E}_q \left[\log \frac{p_0(\boldsymbol{\theta}) \tilde{f}_n(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right] - N\mathrm{KL}[q||p_0] \\ &= \min_{q\in\mathcal{Q}, \{\tilde{p}_n \in \mathcal{P}\}} (1-N)\mathrm{KL}[q||p_0] - \sum_n \mathbb{E}_{\tilde{p}_n} \left[\log \frac{p_0(\boldsymbol{\theta}) \tilde{f}_n(\boldsymbol{\theta})}{\tilde{p}_n(\boldsymbol{\theta})} \right] \\ &\qquad \text{subject to } \tilde{p}_n = q, \forall n. \end{aligned} \tag{2.27}$$

Here in the first line we added N copies of $\mathrm{KL}[q||p_0]$ to the VFE and subtracted the same, and in the second line we *decoupled* the tilted distribution $\tilde{p}_n$ from q by introducing equality constraints. This means the above optimisation has the same fixed points as minimising VFE.

[9] material based on my NIPS 2016 approximate inference workshop abstract, see publication page.

The constraints can be *relaxed* to matching all the moments $\mathbb{E}_{\tilde{p}_n}[\boldsymbol{\theta}^k] = \mathbb{E}_q[\boldsymbol{\theta}^k]$ for $k \in \mathbb{N}$,[10] and a further crude relaxation suggests moment matching just for the first K moments[11] $\mathbb{E}_{\tilde{p}_n}[\boldsymbol{\theta}^k] = \mathbb{E}_q[\boldsymbol{\theta}^k], k = 1, 2, ..., K$. In the following we use a vectorial function $\boldsymbol{\Phi}(\boldsymbol{\theta})$ to summarise these constraints as $\mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}] = \mathbb{E}_q[\boldsymbol{\Phi}]$, where as an example for Gaussian EP: $\boldsymbol{\Phi}(\boldsymbol{\theta}) = [\boldsymbol{\theta}, \boldsymbol{\theta}\boldsymbol{\theta}^T]$. In general $\boldsymbol{\Phi}$ can contain any polynomial terms or other basis functions. This relaxation returns the following constrained optimisation problem:

$$\begin{aligned} &\min_{q \in \mathcal{Q}, \{\tilde{p}_n \in \mathcal{P}\}} \mathcal{F}_{\text{Bethe}}(\{\tilde{p}_n\}, q) \quad \text{subject to } \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}] = \mathbb{E}_q[\boldsymbol{\Phi}], \forall n, \\ &\mathcal{F}_{\text{Bethe}}(\{\tilde{p}_n\}, q) = (1-N)\text{KL}[q||p_0] - \sum_n \mathbb{E}_{\tilde{p}_n}\left[\log \frac{p_0(\boldsymbol{\theta})\tilde{f}_n(\boldsymbol{\theta})}{\tilde{p}_n(\boldsymbol{\theta})}\right]. \end{aligned} \tag{2.28}$$

$\mathcal{F}_{\text{Bethe}}(\{\tilde{p}_n\}, q)$ is the *Bethe free energy* [Bethe, 1935; Yedidia et al., 2001] that is usually presented in the context of probabilistic graphical models and belief propagation. Below we show how to derive its dual form that is usually discussed in EP literature [Minka, 2001a; Opper and Winther, 2005; Seeger, 2005].

> **Remark** (Tom Minka's original note)**.** Minka [2001a] formulated (2.28) as a minimax problem ($\min_{\{\tilde{p}_n\}} \max_q$) instead which seems questionable to me. First (2.28) relaxes the constraints in (2.27), meaning both should have the same optimisation direction. Then since (2.27) just decouples VFE (2.26) with equality constraints, it should be a pure minimisation and has the same stationary points. For graphical models the Bethe free energy optimisation problem is formulated as a pure minimisation problem like above (2.28), e.g. in Heskes [2002]; Wainwright and Jordan [2008] but also interestingly in pages 3-4 of Minka [2001a]. On the other hand, Minka's derivation of the dual energy differs from solving the Lagrangian and does not require the minimax assumption of the primal problem.

From Bethe to EP: a dual form representation

We provide a derivation in a similar way as Heskes [2002], starting from a note on the KL duality[12]

$$-\text{KL}[q||p_0] = \min_{\boldsymbol{\lambda}_q(\boldsymbol{\theta})} -\mathbb{E}_q[\lambda_q(\boldsymbol{\theta})] + \log \mathbb{E}_{p_0}\left[\exp[\lambda_q(\boldsymbol{\theta})]\right], \tag{2.29}$$

with $\lambda_q(\boldsymbol{\theta})$ a function to be specified later on. This duality is in the same spirit as deriving convex conjugate function for the log partition function of an exponential family distribution

[10] Having the same moments for p and q does not imply having the same *moment generating function*.
[11] The zeroth moment matching constraint is replaced by the constraint that $\tilde{p}_n$ integrates to 1.
[12] We include this step in order to connect to the EP energy with optimisation arguments all in the dual space.

(see Proposition 2.1), if viewing $p_0(\boldsymbol{\theta})$ as the base measure. The equality is achieved by $q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \exp[\lambda_q(\boldsymbol{\theta})]$. Substitution into (2.28) then yields a transformed energy that we denoted as $\mathcal{F}_{\text{Bethe}}(\{\tilde{p}_n\}, q, \lambda_q(\boldsymbol{\theta}))$.

$$\begin{aligned}\mathcal{F}_{\text{Bethe}}(\{\tilde{p}_n\}, q, \lambda_q(\boldsymbol{\theta})) = (1-N)\mathbb{E}_q[\lambda_q(\boldsymbol{\theta})] + (N-1)\log \mathbb{E}_{p_0}\left[\exp[\lambda_q(\boldsymbol{\theta})]\right] \\ -\sum_n \mathbb{E}_{\tilde{p}_n}\left[\log \frac{p_0(\boldsymbol{\theta})\tilde{f}_n(\boldsymbol{\theta})}{\tilde{p}_n(\boldsymbol{\theta})}\right].\end{aligned} \tag{2.30}$$

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

$$\begin{aligned}\min_{q,\{\tilde{p}_n\},\lambda_q(\boldsymbol{\theta})} \max_{\{\boldsymbol{\lambda}_{-n},\nu_n,\nu\}} \mathcal{F}_{\text{Bethe}}(\{\tilde{p}_n\}, q, \lambda_q(\boldsymbol{\theta})) + \sum_n \boldsymbol{\lambda}_{-n}^T(\mathbb{E}_q[\boldsymbol{\phi}] - \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\phi}]) \\ + \sum_n \nu_n\left(\int \tilde{p}_n(\boldsymbol{\theta})d\boldsymbol{\theta} - 1\right) + \nu\left(\int q(\boldsymbol{\theta})d\boldsymbol{\theta} - 1\right).\end{aligned} \tag{2.31}$$

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

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

where the normalising constant is

$$Z_n = \int p_0(\boldsymbol{\theta})\tilde{f}_n(\boldsymbol{\theta}) \exp\left[\boldsymbol{\lambda}_{-n}^T \boldsymbol{\Phi}(\boldsymbol{\theta})\right] d\boldsymbol{\theta}.$$

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

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

We explicitly specify $\lambda_q(\boldsymbol{\theta}) = \boldsymbol{\lambda}_q^T \boldsymbol{\Phi}(\boldsymbol{\theta}) + \nu$ 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 (2.31) and enforcing the fixed point condition for q yields the *EP energy* [Minka, 2001a]:

$$\begin{aligned}\min_{\boldsymbol{\lambda}_q} \max_{\{\boldsymbol{\lambda}_{-n}\}} \mathcal{F}_{\text{EP}}(\boldsymbol{\lambda}_q, \{\boldsymbol{\lambda}_{-n}\}) = (N-1)\log \mathbb{E}_{p_0}\left[\exp[\boldsymbol{\lambda}_q^T \boldsymbol{\Phi}(\boldsymbol{\theta})]\right] - \sum_n \log Z_n, \\ \text{subject to } (N-1)\boldsymbol{\lambda}_q = \sum_n \boldsymbol{\lambda}_{-n}.\end{aligned} \tag{2.32}$$

Notice now the optimisation problem over q is dropped since (2.32) does not depend on it. To obtain the approximate posterior back, we make use of the tightness of the KL duality,

and define

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

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

EP as a fix point iteration method for solving the dual problem

EP [Minka, 2001b] proposes parametrising the (natural parameters of) local approximating factors $f_n \approx \tilde{f}_n$ instead of the approximate posterior q, with the goal of f_n capturing the effect of $\tilde{f}_n(\boldsymbol{\theta})$ on the exact posterior, by defining $f_n(\boldsymbol{\theta}) = \exp[\boldsymbol{\lambda}_n^T \boldsymbol{\Phi}(\boldsymbol{\theta})]$, $\boldsymbol{\lambda}_n = \boldsymbol{\lambda}_q - \boldsymbol{\lambda}_{-n}$. Thus by construction the constraint in (2.32) is automatically satisfied: $\boldsymbol{\lambda}_q = \sum_n \boldsymbol{\lambda}_n$ and $\boldsymbol{\lambda}_{-n} = \sum_{m \neq n} \boldsymbol{\lambda}_m$. Then EP runs a fixed point iteration algorithm to find a stationary point for $\{\boldsymbol{\lambda}_n\}_{n=1}^N$. More specifically the gradient of (2.32) w.r.t. the local parameter $\boldsymbol{\lambda}_n$ is

$$\nabla_{\boldsymbol{\lambda}_n} \mathcal{F}_{\text{EP}} = (N-1)\mathbb{E}_q[\boldsymbol{\Phi}(\boldsymbol{\theta})] - \sum_{m \neq n} \mathbb{E}_{\tilde{p}_m}[\boldsymbol{\Phi}(\boldsymbol{\theta})]. \tag{2.33}$$

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

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

which motivates the moment matching update (2.20) in EP.

A pictorial view on why EP can be better than VI

Folk wisdom suggests that EP, if it convergences, often provide more accurate approximations to the target distribution when compared with VI. This observation is explained pictorially in Figure 2.4. Recall that both algorithms can be viewed as minimising the Bethe free energy under constraints: for VI the constraints are equality constraints $q = \tilde{p}_n$, whereas EP instead uses moment matching constraints $\mathbb{E}_q[\boldsymbol{\Phi}] = \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}]$. In an augmented space $\mathcal{P}^{N+1}$, the search space of VI $\{(q, \tilde{p}_n) : q = \tilde{p}_n, q \in \mathcal{Q}\}$ is contained in the search space of EP $\{(q, \tilde{p}_1, ..., \tilde{p}_N) : q \in \mathcal{Q}, \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}] = \mathbb{E}_q[\boldsymbol{\Phi}]\}$, and can be of much smaller dimensions (e.g. the green line segment versus the blue region as shown in the Figure 2.4). Therefore if EP converges, then it is more likely to find a better minimum compared to VI. Empirical evidence also suggests that constrained Bethe free-energy evaluated at a fixed point is often a better approximation to the (negative log) marginal likelihood.

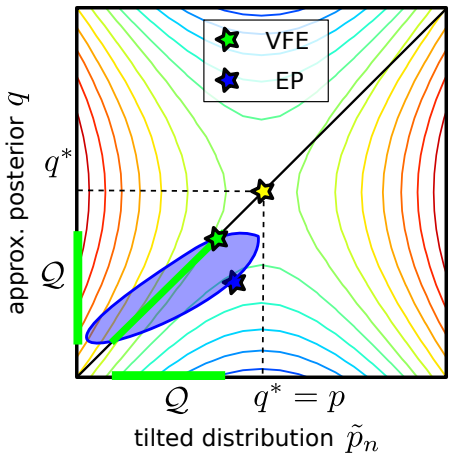


Fig. 2.4 EP versus VI as constrained energy optimisation problems, visualised by projecting the energy surface from the augmented space $\mathcal{P}^{N+1}$ to $\mathcal{P}^2$. Here the slash line across the space represents the subspace $\{(q, \tilde{p}_n) : q = \tilde{p}_n\}$, and the search space for VI (the green segment) is contained in the EP candidate set (the blue region). The stars indicate the optimal solutions returned by the exact (in yellow) and the approximate inference algorithms (green/blue). See main text for details.

Criticisms for EP's iterative update

The above fixed point iteration update has no convergence guarantee, which is one of the drawbacks of EP. The reason is that EP solves the dual problem of constrained Bethe free energy minimisation, and that dual problem turns out to be a minimax optimisation problem with constraints. Indeed, a double-loop algorithm [Heskes and Zoeter, 2002] should be applied to (2.32) if convergence is required. However in practice such a double-loop method has been shown to be much slower than EP. Also disappointedly, even when the exact posterior is contained in $\mathcal{Q}$, EP is not guaranteed to return it. Similar problems exist for belief propagation when the graph contains many loops, or when the relaxed polytope is not *tight* [Wainwright and Jordan, 2008].

From VFE to power EP

We now extend the above approach to power EP [Minka, 2004] which is a new contribution, although fairly straightforward. This procedure includes one modification to the Bethe free energy. Assume for each factor $\tilde{f}_n$ a power value $\alpha_n \neq 0$ is associated, with $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_N)$ and $\sum_n \frac{1}{\alpha_n} \neq 1$. Then the Bethe free energy with moment matching constraints is modified to:

$$\mathcal{F}_{\boldsymbol{\alpha}}(q, \{\tilde{p}_n\}) = \left(1 - \sum_n \frac{1}{\alpha_n}\right) \mathrm{KL}[q||p_0] - \sum_{n=1}^{N} \frac{1}{\alpha_n} \mathbb{E}_{\tilde{p}_n}\left[\log \frac{p_0(\boldsymbol{\theta}) \tilde{f}_n(\boldsymbol{\theta})^{\alpha_n}}{\tilde{p}_n(\boldsymbol{\theta})}\right]. \tag{2.34}$$

Similar to the derivation of (2.27), here we first added and subtracted $(\sum_n \frac{1}{\alpha_n})$ copies of $\mathrm{KL}[q||p_0]$, then decoupled $\tilde{p}_n$ from q, and relaxed the equality constraints to moment matching. Calculations following Section 2.3.3 also reveal the change of the fixed point condition for q to $(\sum_n \frac{1}{\alpha_n} - 1)\boldsymbol{\lambda}_q = \sum_n \frac{1}{\alpha_n}\boldsymbol{\lambda}_{-n}$. Define q as an exponential family distribution with natural parameter $\boldsymbol{\lambda}_q$ as before, and $\boldsymbol{\lambda}_n = (\boldsymbol{\lambda}_q - \boldsymbol{\lambda}_{-n})/\alpha_n$. We arrive at the power EP objective:

$$\left(\sum_n \frac{1}{\alpha_n} - 1\right) \log Z_q - \sum_n \frac{1}{\alpha_n} \log \int p_0(\boldsymbol{\theta}) \tilde{f}_n(\boldsymbol{\theta})^{\alpha_n} \exp\left[(\boldsymbol{\lambda}_q - \alpha_n \boldsymbol{\lambda}_n)^T \boldsymbol{\Phi}(\boldsymbol{\theta})\right] d\boldsymbol{\theta}. \quad (2.35)$$

The iterative process also enforces $\boldsymbol{\lambda}_q = \sum_n \boldsymbol{\lambda}_n$. Minka [2005] showed that (2.35) becomes an upper-bound of $\log Z$ when $\alpha_n > 0$ and $\sum_n \frac{1}{\alpha_n} < 1$. On the other hand, taking $\alpha_n \to 0, \forall n$ recovers $\mathcal{F}_{\mathrm{VFE}}$ but now the q distribution is restricted to have an exponential family form.

2.4 A battle between VI and EP: which I should prefer?

We have presented VI and EP as two main classes of approximate inference methods. Newcomers to this subject might be unsure about which algorithm they should use for their task. Inspired by the comparison table on the Infer.NET [Minka et al., 2014] user guide webpage,[13] here I put in Table 2.1 a comparison between VI and EP, with discussions in the following. We note here again that VI is a special case of power EP by setting the power $\alpha = 0$ (which is also referred to the variational message passing (VMP) algorithm [Minka, 2005; Winn and Bishop, 2005] that uses fixed-point iterative updates to optimise the variational lower-bound), and all the comparisons below are on VI versus power-EP with other α values.

- Global versus local approximations.
  As explained, VI constructs a *global* approximation $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$ while EP exploits the structure of the posterior distribution and proposes approximations to each of the factors instead (thus we call it *local*). Interesting observations apply for VMP: like local approximation methods, it also propagates messages between factors and store approximations locally; like global approximation methods, however, any stationary point of the VMP algorithm is also a stationary point of the variational lower-bound, no matter which factor graph structure is in use. Conversely, EP returns different approximation results when executed on different factor graph representations of the

[13] http://infernet.azurewebsites.net/docs/working%20with%20different%20inference%20algorithms.aspx

target distribution. Due to this reason we still call VMP a *global* approximation method, even when it also performs local computations.

- Zero-forcing versus mass-covering behaviour.
  VI exhibits zero-forcing behaviour exactly because it minimises the exclusive KL-divergence (see discussions in Section 2.1.1). For power EP the story is complicated: it can prefer mass-covering when having large $\alpha > 0$ values, but it can also be zero-forcing if, though not very often, negative α values are in use.

- Convergence guarantee.
  VI and VMP are guaranteed to converge, mainly because the variational lower-bound is upper-bounded. EP, like belief propagation on graphs, has no convergence guarantees in general, although practices suggest that it often converges to a local optimum.

- Running time/speed.
  VI with gradient descent often takes many iterations to converge, though each gradient step can be very cheap. EP, on the other hand, takes less iterations to run, but typically requires more time to compute the moment matching step. This is especially the case when there is no closed form solution for the moments, where an additional approximate inference procedure (e.g. MCMC) will be called as a sub-routine, which can be very expensive. VMP combines the advantages from both sides, often being the fastest method to find a local optimum.

- Accuracy.
  Assume both EP and VI use the same family of approximate distributions $\mathcal{Q}$. It is well known that variational free-energy approaches are biased and often severely so [Turner and Sahani, 2011]. Because of the zero-forcing behaviour, VI can be over-confident and miss important modes/correlation structure in the exact posterior. This often results in worse performance when calibrated uncertainty is required. EP is often more accurate when compared to VI, in some cases it can even be exact (e.g. with a tree-structure graph containing simple factors like Gaussians). However EP can diverge on certain cases and in this case one might prefer VI to at least obtain a reasonable approximation.

- Approximate posterior design.
  VI with Monte Carlo methods allows the deployment of more complex approximate posterior distributions (e.g. those parameterised by a neural network which will be introduced later). On the other hand, many applications of EP still use exponential family distributions (mainly Gaussians) until the development of Chapter 4.

Table 2.1 Comparing VI/VMP and power EP.

	VI/VMP	power EP ($\alpha \neq 0$)
global or local?	global (same for VMP)	local (depends on the factor graph)
behaviour	zero-forcing	zero-forcing/mass-covering (depends on α)
optimisation procedure	gradient descent (generic VI) fixed-point iterative updates (VMP)	fixed-point iterative updates
convergence	yes (theoretical guarantee)	yes (empirically) (with log-concave potentials)
accuracy	often less accurate (under-estimates uncertainty)	often more accurate (depends on the factor graph)

- Hyper-parameter optimisation and model selection.
  One might want to optimise the hyper-parameters of the model, or perform model selection to choose the best model. Both cases require a reliable approximation to $\log p(\mathcal{D})$. In this regard, VI provides a conservative (although possibly strongly biased) estimate of the marginal likelihood, which provides a safe option for approximate MLE. But EP's energy function can be hard to optimise: unless a fixed point is obtained, in theory the EP energy can be arbitrarily far from the exact marginal likelihood [Cunningham et al., 2011].[14] Even if EP converges, the corresponding energy function is not protected: it is neither an upper- nor a lower-bound to the model evidence.

Unfortunately I cannot provide a general case verdict for the two classes of algorithms, just like the folklore no-free lunch theorem [Wolpert and Macready, 1997] states "there exists no machine learning algorithm that dominates the others in all cases". To conclude the discussion, here comes two suggestions that have been validated by many results in the literature. Care should be taken for EP when the factor graph is densely connected and contains many loops. Also one should avoid using VI to approximate a non-smooth target density with a smooth q distribution, because the zero-forcing behaviour of VI may push the approximate posterior towards undesirable solutions like delta mass. This happens, in binary classification for example, when the likelihood function is a Heaviside function or a very sharp sigmoid function.

[14] Although some empirical results suggest that it is possible to perform hyper-parameter optimisation at the same time as EP runs [Hernández-Lobato and Hernández-Lobato, 2016].

2.5 Applications: Bayesian deep learning

Having discussed plenty of background material on divergences and approximate inference algorithms, in this section we turn to specify the probabilistic models that will be studied in the rest of the thesis. Historically, the Bayesian modelling community studied well-understood probabilistic models (conjugate models, generalised linear models, etc). Very recently the research theme of *Bayesian deep learning* has received much increasing interest, where *deep generative models* and *Bayesian neural networks* are mainly studied. Advances in approximate inference sit at the core of Bayesian deep learning research: we will briefly sketch how VI can be used to perform inference and learning in both models. In later chapters we will also develop advanced approximate inference methods, and demonstrate improved performances on Bayesian deep learning tasks.

2.5.1 Deep generative models

Generating realistic images, sound and text has always been one of the main themes 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 desired outputs. Similar to the latent variable model we discussed in Section 2.2.4, 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 $\boldsymbol{\theta}$.[15] 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.[16] As a concrete example, let us consider the following model:

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

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:

$$\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \tag{2.36}$$

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

[16] Another case where $p(\boldsymbol{x}|\boldsymbol{z})$ is implicitly defined by neural network transforms will be discussed in the second part of the thesis.

which involves integrating out all the latent variables $\boldsymbol{z}_n$ out and which is thus analytically intractable. Traditionally, the expectation maximisation (EM) algorithm [Dempster et al., 1977] is used here to train the parameters $\boldsymbol{\theta}$. But here we deploy another approach which uses the variational lower-bound as a surrogate loss:

$$\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \sum_{n=1}^{N} \mathcal{L}_{\text{VI}}(\boldsymbol{\phi},\boldsymbol{\theta};\boldsymbol{x}_n), \quad \mathcal{L}_{\text{VI}}(\boldsymbol{\phi},\boldsymbol{\theta};\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right]. \tag{2.37}$$

The amortised variational inference algorithm is deployed (see Section 2.2.4).

Here we introduce a neat trick called *reparameterisation* [Kingma and Welling, 2014; Rezende et al., 2014],[17] which, along with MC approximation, makes the variational lower-bound easy to handle. This trick comes from a very simple observation: given a distribution $p(\boldsymbol{z})$, if sampling $\boldsymbol{z} \sim p(\boldsymbol{z})$ is equivalent to first sampling a "noise" variable $\boldsymbol{\varepsilon} \sim p(\boldsymbol{\varepsilon})$ and then computing a mapping $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{\varepsilon})$, then the expectation of some function $F(\boldsymbol{z})$ under distribution $p(\boldsymbol{z})$ can be rewritten as

$$\mathbb{E}_{p(\boldsymbol{z})}[F(\boldsymbol{z})] = \mathbb{E}_{p(\boldsymbol{\varepsilon})}[F(\boldsymbol{f}(\boldsymbol{\varepsilon}))].$$

This convenient computation is also called the *law of the unconscious statistician* (LOTUS). To see how this trick works in the context of deep generative models, we consider a simple approximate posterior distribution, namely the factorised Gaussian:

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

Again $\boldsymbol{\mu}_{\boldsymbol{\phi}}$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}$ are mappings parameterised by deep neural networks with parameter $\boldsymbol{\phi}$. One can easily notice that drawing samples from this Gaussian is done by the following procedure

$$\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \Leftrightarrow \boldsymbol{\varepsilon} \sim q(\boldsymbol{\varepsilon}) = \mathcal{N}(\boldsymbol{\varepsilon};\boldsymbol{0},\boldsymbol{I}), \boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\varepsilon}, \tag{2.38}$$

where $\odot$ denotes element-wise product, and essentially (2.38) performs a change-of-variable operation. Usually the two mappings $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\cdot)$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\cdot)$ are represented by (deep) neural networks (with $\boldsymbol{\phi}$ the network weight matrices), and in this context the q distribution is also called the *recognition model* or the *inference network*. Following the LOTUS rule, the

[17] As we shall see this is *not* a new trick, although Kingma and Welling [2014] and Rezende et al. [2014] were the first to apply it in deep generative modelling context.

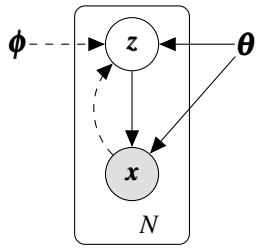


Fig. 2.5 The graphical model of VAE, showing the generative model and the inference network. Dash arrows imply dependencies in the q distribution. Reproduced from Kingma and Welling [2014].

variational lower-bound can be rewritten as

$$\mathcal{L}_{\text{VI}}(\boldsymbol{\phi},\boldsymbol{\theta};\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{\varepsilon})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})+\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})\odot\boldsymbol{\varepsilon})}{q_{\boldsymbol{\phi}}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})+\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})\odot\boldsymbol{\varepsilon}|\boldsymbol{x})}\right], \tag{2.39}$$

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

$$\mathcal{L}^{\text{MC}}_{\text{VI}}(\boldsymbol{\phi},\boldsymbol{\theta};\boldsymbol{x}) = \frac{1}{K}\sum_{k=1}^{K}\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})+\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})\odot\boldsymbol{\varepsilon}_k)}{q_{\boldsymbol{\phi}}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})+\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})\odot\boldsymbol{\varepsilon}_k|\boldsymbol{x})}, \quad \boldsymbol{\varepsilon}_k \sim q(\boldsymbol{\varepsilon}). \tag{2.40}$$

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 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 2.5.

2.5.2 Bayesian neural networks

Another exciting application of Bayesian inference is a class of methods called Bayesian neural networks (Bayesian NNs, BNNs), which maintains the uncertainty of the weight matrix assignments. Originated in the late 80s and early 90s in the last century [Hinton and Van Camp, 1993; MacKay, 1992; Neal, 1992, 1995; Peterson and Anderson, 1987],[18] it has recently attracted a lot of attention again, mainly due to the advance of modern approximate inference techniques. In this thesis we will also study BNNs as a test case for the developed approximate algorithms, and here we provide a concise review of the related key concepts.

We start by describing a normal deep neural network mapping $\mathbf{y} = \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ parameterised by a set of weights (and bias vectors which we omit here for simplicity) $\boldsymbol{\theta} = \{\mathbf{W}^l\}_{l=1}^{L}$. Then

[18] For a detailed history see [Gal, 2016, Section 2.2.1].

given a training dataset $\mathcal{D} = \{(\boldsymbol{x}_n, \mathbf{y}_n)\}_{n=1}^N$, one would first define a loss function $l(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x}))$ to measure the error between the supervision signal and the prediction, then train the neural network by minimising the empirical loss on the dataset

$$\min_{\boldsymbol{\theta}} \mathcal{F}(\boldsymbol{\theta}) = \sum_{n=1}^{N} l(\mathbf{y}_n, \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_n)).$$

Examples for such loss function include the mean squared error (or ℓ_2 error) $l(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x})) = ||\mathbf{y} - \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x})||_2^2$ for regression and cross entropy loss $l(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x})) = -\mathbf{y} \log f_{\boldsymbol{\theta}}(\boldsymbol{x}) - (1-\mathbf{y}) \log(1 - \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x}))$ for (binary) classification. To avoid overfitting, regularisers are often included in the optimisation objective, for example, adding an ℓ_2 regulariser would return the following optimisation problem

$$\min_{\boldsymbol{\theta}} \mathcal{F}_{\text{MAP}}(\boldsymbol{\theta}) = \sum_{n=1}^{N} l(\mathbf{y}_n, \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + \frac{\lambda}{2} ||\boldsymbol{\theta}||_2^2. \tag{2.41}$$

Let us consider the above training objective again but from a probabilistic modelling perspective. In this case the network weight matrices $\boldsymbol{\theta}$ are treated as random variables, and a Gaussian prior $p_0(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$ is attached.[19] For many loss functions (e.g. ℓ_2 loss), the likelihood function of $\boldsymbol{\theta}$ is then defined as[20]

$$p(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{Z} \exp(-l(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x})).$$

See LeCun et al. [2006] for an example justification. For most loss functions (e.g. mean square error and cross entropy) the normalising constant Z does not depend on the weights $\boldsymbol{\theta}$. Given the training dataset $\mathcal{D}$ the joint distribution is

$$p(\mathcal{D}, \boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\mathbf{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}),$$

thus one can easily show that the maximum a posteriori (MAP) training $\max_{\boldsymbol{\theta}} \log p(\mathcal{D}, \boldsymbol{\theta})$ is exactly equivalent to the problem of (2.41).

After framing the deep neural network as a probabilistic model, a Bayesian approach would find the posterior of the network weights $p(\boldsymbol{\theta}|\mathcal{D})$ and use the uncertainty information encoded in it for future predictions. Again the exact posterior is intractable, and approxi-

[19] Sometimes other prior distributions might be preferred, e.g. spike-and-slab prior [Louizos et al., 2017; Neal, 1995].

[20] This definition requires $\int \exp(-l(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{x}))d\mathbf{y}$ to be finite, which is true for many loss functions but not always the case.

mate inference would fit an approximate posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ parameterised by the variational parameters $\boldsymbol{\phi}$ to the exact posterior, and then use it to compute the (approximate) predictive distribution. As an example, here we consider a variational inference approach and write down the variational lower-bound accordingly [Barber and Bishop, 1998; Hinton and Van Camp, 1993]

$$\mathcal{L}_{\text{VI}}(\boldsymbol{\phi}) = \sum_{n=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p(\mathbf{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta})] - \text{KL}[q_{\boldsymbol{\phi}}||p_0]. \tag{2.42}$$

Solving problem (2.42) with a degenerate variational approximation $q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = \delta_{\boldsymbol{\theta}=\boldsymbol{\phi}}$ would result in a MAP solution (2.41), which is less desirable in terms of capturing uncertainty. However $q_{\boldsymbol{\phi}}$ should also not be too complicated in terms of the entailed computational complexity, as now $\boldsymbol{\theta}$ is typically a very long vector of thousands, or even millions of dimensions. Popular choices of such "efficient" variational approximations include the mean-field Gaussian approximation

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = \prod_{l=1}^{L} \prod_{i,j} \mathcal{N}(W_{ij}^l; \mu_{ij}^l, (\sigma_{ij}^l)^2), \quad \boldsymbol{\phi} = \{\mu_{ij}^l, \sigma_{ij}^l\}_{i,j,l},$$

which doubles the number of parameters to be optimised, and which is thus less memory demanding than a correlated Gaussian approximation for example. Also now the KL term in (2.42) has an analytical solution

$$\text{KL}[q_{\boldsymbol{\phi}}||p_0] = \sum_{i,j,l} \frac{1}{2} \left(\lambda (\mu_{ij}^l)^2 - 1 + \lambda (\sigma_{ij}^l)^2 - \log \lambda - 2 \log \sigma_{ij}^l \right).$$

But even so the error term $\mathbb{E}_q[\log p(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\theta})]$ is intractable due to the highly non-linear NN mapping $\mathbf{f}_{\boldsymbol{\theta}}$. Fortunately this issue is mitigated by employing the Monte Carlo (MC) approximation again:

$$\mathcal{L}_{\text{VI}}^{\text{MC}}(\boldsymbol{\phi}) = \sum_{n=1}^{N} \frac{1}{K} \sum_{k=1}^{K} [\log p(\mathbf{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}^k)] - \text{KL}[q_{\boldsymbol{\phi}}||p_0], \quad \boldsymbol{\theta}^k \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta}). \tag{2.43}$$

Usually to reduce the number of forward passes (i.e. computing $\log p(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\theta})$) K is often a small number, even as little as $K = 1$. Also for large datasets stochastic optimisation techniques are also applied, meaning that for mini-batch training with a subset

$\{(\boldsymbol{x}_m, \mathbf{y}_m)\}_{m=1}^M \sim \mathcal{D}^M$, the objective is (e.g. for $K = 1$)

$$\mathcal{L}_{\text{VI}}^{\text{MC}}(\boldsymbol{\phi}) = \frac{N}{M} \sum_{m=1}^{M} \log p(\mathbf{y}_m|\boldsymbol{x}_m, \boldsymbol{\theta}) - \text{KL}[q_{\boldsymbol{\phi}}||p_0], \quad \boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta}). \tag{2.44}$$

If the KL divergence is intractable it can also be approximated by Monte Carlo, i.e.

$$\text{KL}[q_{\boldsymbol{\phi}}||p_0] \approx \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) - \log p_0(\boldsymbol{\theta}), \ \boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}.$$

The reparameterisation trick discussed earlier is often applied to allow gradient back-propagation, and in the case of mean-field Gaussian approximations the weight matrices are sampled as $W_{ij}^l = \mu_{ij}^l + \sigma_{ij}^l \varepsilon$, $\varepsilon \sim \mathcal{N}(\varepsilon; 0, 1)$, and the log-likelihood terms in (2.44) is then computed by $\log p(\mathbf{y}_m|\boldsymbol{x}_m, \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon})$.

Remark (Other approximate inference methods for BNNs)**.** Although we presented BNNs using VI, I shall point out that other approximate inference methods do apply. Milestone papers of approximate inference techniques developed for BNNs include: Laplace approximation [MacKay, 1992], minimum description length [Hinton and Van Camp, 1993] (which is equivalent to VI), hybrid/Hamiltonian Monte Carlo [Neal, 1995], and variational inference [Barber and Bishop, 1998; Blundell et al., 2015; Gal and Ghahramani, 2016; Graves, 2011]. Recent applications of BNNs using non-VI Bayesian methods include stochastic gradient MCMC [Korattikara et al., 2015; Li et al., 2016a, for example], assumed density filtering/EP [Hernández-Lobato and Adams, 2015] and those that will be presented in the later Chapters.

2.6 Summary and outlook

In this chapter we have established the idea of divergence minimisation, and reviewed two classes of commonly used approximate inference techniques: variational inference (VI) and expectation propagation (EP). These two methods are closely related to each other but also have notable differences, both algorithmically and in terms of the solutions they return. In the rest of the first part (chapters 3 and 4), we will study in depth these similarities and differences, and as a motivation for the detailed analysis we provide a brief discussion as follows.

First, compared to VI, EP brings more serious computational burden, in that its memory cost scales linearly with the number of factors in the graph representing the target density.

Scalable approximations to EP are then discussed in Chapter 3, in which we demonstrate the scalability of our proposals to machine learning tasks in which EP is too expensive to handle.

Second, both methods can be motivated by divergence minimisation, in which VI minimises the exclusive KL divergence and EP minimises an inclusive KL divergence. The generalisation, i.e. power EP, minimises alpha-divergences which have the two KL divergences as special cases. However, VI performs *global* divergence minimisation, while EP minimises the selected divergence *locally*. The resulting differences in local optima and optimisation behaviour are further investigated in Chapter 4.

Chapter 3

Stochastic Expectation Propagation

Scalable approximate inference is key to the recent success of Bayesian deep learning [Blundell et al., 2015; Gal and Ghahramani, 2016; Hernández-Lobato and Adams, 2015]. Here scalability entails that the algorithm has low enough time and space complexity figures to be deployed on real-world datasets. By leveraging stochastic optimisation techniques, variational inference [Beal, 2003; Hoffman et al., 2013; Jordan et al., 1999] has been shown to be highly scalable even for modelling datasets comprising millions of documents [Broderick et al., 2013]. On the other hand, we have discussed at length in the last chapter why EP-like algorithms can be superior, which is indeed confirmed by a large set of *small* scale experiments [Barthelmé and Chopin, 2014; Cunningham et al., 2011; Kuss and Rasmussen, 2005]. As a reminder: EP constructs a posterior approximation by iterating simple local computations that refine factors which approximate the posterior contribution from *each* datapoint.

At first sight, EP might therefore appear well suited to large-data problems: the locality of computation makes the algorithm simple to be extended to stochastic optimisation scenarios. Indeed EP is certainly suited for fast approximate inference, and folklore suggests that it usually converges even faster than VI in practice, due to the exploitation of fixed-point iterative search. Despite the huge gain in time complexities, EP has garnered much less attention in the regard of large-scale learning tasks. This is because, the elegance of local computation has been bought at the price of a prohibitive memory overhead that grows with the number of data-points N, since each local approximating factor typically has the same complexity as the global approximation. In contrast, VI utilises global approximations that are refined directly, which prevents memory overheads from scaling with N. Thus VI is arguably better-suited for approximate Bayesian inference at large-scale, precisely because it is much more memory efficient (albeit if not utilising parallel computing).

Can we have the best of both worlds? That is, accurate global approximations that are derived from truly local computation. In this chapter we propose *stochastic expectation propagation* (SEP) to address this question, which are developed based upon the standard EP algorithm. Importantly, SEP only maintains a global approximation (like VI), thus reducing the memory footprint by a factor of N when compared to EP. However SEP updates the global approximation in a local way (like EP), with (damped) stochastic estimates on data sub-samples in an analogous fashion to stochastic variational inference (SVI) [Hoffman et al., 2013]. Indeed, the generalisation of the algorithm to the power-EP setting directly recovers SVI as a special case. We further extend the method to control the granularity of the approximation, and to treat models with latent variables without compromising on accuracy or entailing unnecessary memory demands. Finally, we demonstrate the scalability and accuracy of the method on a number of real world and synthetic datasets, also spanning a number of canonical machine learning tasks.

To the best of my knowledge, the development of SEP back in 2015 was the first successful attempt[1] to scale-up EP-like algorithms to large-scale data and stochastic optimisation settings, which also inspired many innovations in Chapter 4, the black-box alpha algorithm [Hernández-Lobato et al., 2016], and other applications.

Remark (Previous attempts to scale-up EP) **.** As EP appears to be the method of choice for some applications, researchers have attempted to push it to scale. A first approach is to swallow the large computational burden and simply use large data-structures to store the approximating factors (e.g. TrueSkill [Herbrich et al., 2006]). This approach can only be pushed so far. A second approach is to use the *assumed density filtering* (ADF) algorithm as a simple variant, which only requires a global approximation to be stored [Maybeck, 1982]. ADF, however, provides poorly calibrated uncertainty estimates [Minka, 2001b] which was one of the main motivating reasons for developing EP in the first place. A third idea, complementary to the one described here, is to use approximating factors that have simpler structure (e.g. low rank, [Qi et al., 2010]). This reduces memory consumption (e.g. for Gaussian factors from $\mathcal{O}(ND^2)$ to $\mathcal{O}(ND)$), but does not stop the scaling with N. A fourth idea uses EP to carve up the dataset [Gelman et al., 2014; Xu et al., 2014] using approximating factors for collections of data-points. This results in coarse-grained, rather than local, updates and other methods must be used to compute them. (Indeed, the spirit of Gelman et al. [2014]; Xu et al. [2014] is to extend sampling methods to large-datasets by exploiting EP's ability to split up inference problems into smaller parts, not EP itself.)

[1] except for those using parallel computing where the memory constraint is not in consideration.

3.1 Memory efficient factor tying

3.1.1 A quick comparison between EP and ADF

We begin by briefly setting-up the EP and assumed density filtering (ADF) algorithms for posterior approximation, and the readers are referred to Section 2.3.2 for a more detailed review. Recall that EP constructs the approximation $q(\boldsymbol{\theta})$ to the exact posterior as the following:[2]

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx q(\boldsymbol{\theta}), \quad p(\boldsymbol{\theta}|\mathcal{D}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta}), \quad q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{n=1}^{N} f_n(\boldsymbol{\theta}). \tag{3.1}$$

The goal of EP is to refine the approximate factors so that they capture the contribution of each of the likelihood terms to the posterior i.e. $f_n(\boldsymbol{\theta}) \approx p(\boldsymbol{x}_n|\boldsymbol{\theta})$. In this spirit, one approach would be to find each approximating factor $f_n(\boldsymbol{\theta})$ by minimising the Kullback Leibler (KL) divergence between the posterior and the distribution formed by replacing one of the likelihoods by its corresponding approximating factor, $\mathrm{KL}[p(\boldsymbol{\theta}|\mathcal{D})||p(\boldsymbol{\theta}|\mathcal{D})f_n(\boldsymbol{\theta})/p(\boldsymbol{x}_n|\boldsymbol{\theta})]$. Unfortunately, such an update is still intractable as it involves computing the full posterior. Instead, EP approximates this procedure by replacing the exact leave-one-out posterior $p_{-n}(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta}|\mathcal{D})/p(\boldsymbol{x}_n|\boldsymbol{\theta})$ on both sides of the KL by the approximate leave-one-out posterior (also called the *cavity* distribution) $q_{-n}(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})/f_n(\boldsymbol{\theta})$. Since this couples the updates for the approximating factors, the updates must now be iterated.

We summarise the update procedure for a single factor in Algorithm 3. Critically, the approximation step of EP involves local computations since one likelihood term is treated at a time. The assumption is that these local computations, although possibly requiring further approximation, are far simpler to handle compared to the full posterior $p(\boldsymbol{\theta}|\mathcal{D})$. In practice, EP often performs well when the updates are parallelised [Minka et al., 2014]. Moreover, by using approximating factors for groups of data-points, and then running additional approximate inference algorithms to perform the EP updates (which could include nested EP), EP carves up the data making it suitable for distributed approximate inference.

There is, however, one wrinkle that complicates deployment of EP at scale. Computation of the cavity distribution requires removal of the current approximating factor, which means any implementation of EP must store them explicitly necessitating an $\mathcal{O}(N)$ memory footprint. One option is to simply ignore the removal step and replace the cavity distribution with the full approximation, resulting in the assumed density filtering (ADF) algorithm (Algorithm 4) [Maybeck, 1982] that only maintains global approximation in memory. But as the moment

[2] Here we consider tractable prior distributions, e.g. Gaussians, otherwise further approximation can be applied and the presented results carry in that case.

Algorithm 3 EP

choose a factor f_n to refine:
compute cavity distribution $q_{-n}(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})/f_n(\boldsymbol{\theta})$
compute tilted distribution $\tilde{p}_n(\boldsymbol{\theta}) \propto p(\boldsymbol{x}_n|\boldsymbol{\theta})q_{-n}(\boldsymbol{\theta})$
moment matching: $f_n(\boldsymbol{\theta}) \leftarrow \texttt{proj}[\tilde{p}_n(\boldsymbol{\theta})]/q_{-n}(\boldsymbol{\theta})$
inclusion: $q(\boldsymbol{\theta}) \leftarrow q_{-n}(\boldsymbol{\theta})f_n(\boldsymbol{\theta})$

Algorithm 4 ADF

choose a datapoint $\boldsymbol{x}_n \sim \mathcal{D}$:
compute cavity distribution $q_{-n}(\boldsymbol{\theta}) = q(\boldsymbol{\theta})$
compute tilted distribution $\tilde{p}_n(\boldsymbol{\theta}) \propto p(\boldsymbol{x}_n|\boldsymbol{\theta})q_{-n}(\boldsymbol{\theta})$
moment matching: $f_n(\boldsymbol{\theta}) \leftarrow \texttt{proj}[\tilde{p}_n(\boldsymbol{\theta})]/q_{-n}(\boldsymbol{\theta})$
inclusion: $q(\boldsymbol{\theta}) \leftarrow q_{-n}(\boldsymbol{\theta})f_n(\boldsymbol{\theta})$

Algorithm 5 SEP

choose a datapoint $\boldsymbol{x}_n \sim \mathcal{D}$:
compute cavity distribution $q_{-1}(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})/f(\boldsymbol{\theta})$
compute tilted distribution $\tilde{p}_n(\boldsymbol{\theta}) \propto p(\boldsymbol{x}_n|\boldsymbol{\theta})q_{-1}(\boldsymbol{\theta})$
moment matching: $f_n(\boldsymbol{\theta}) \leftarrow \texttt{proj}[\tilde{p}_n(\boldsymbol{\theta})]/q_{-1}(\boldsymbol{\theta})$
inclusion: $q(\boldsymbol{\theta}) \leftarrow q_{-1}(\boldsymbol{\theta})f_n(\boldsymbol{\theta})$
implicit update: $f(\boldsymbol{\theta}) \leftarrow f(\boldsymbol{\theta})^{1-\frac{1}{N}} f_n(\boldsymbol{\theta})^{\frac{1}{N}}$

matching step now over-counts the underlying approximating factor (consider the new form of the objective $\mathrm{KL}[q(\boldsymbol{\theta})p(\boldsymbol{x}_n|\boldsymbol{\theta})||q(\boldsymbol{\theta})]$) the variance of the approximation shrinks to zero as multiple passes are made through the dataset. Early stopping, e.g. after a single pass through the data, is therefore required to prevent overfitting, and generally speaking ADF does not return uncertainties that are well-calibrated to the posterior. In the next section we introduce a new algorithm that sidesteps EP's large memory demands whilst avoiding the pathological behaviour of ADF.

3.1.2 Stochastic expectation propagation

In this section we introduce a new algorithm, inspired by EP, called stochastic expectation propagation (SEP) that combines the benefits of local approximation (tractability of updates, distributability, and parallelisability) with global approximation (reduced memory demands). The algorithm can be interpreted as a version of EP in which the approximating factors are tied, or alternatively as a corrected version of ADF that prevents overfitting. The key idea is that, at convergence, the approximating factors in EP can be interpreted as parameterising a global factor, $f(\boldsymbol{\theta})$, that captures the average effect of a likelihood on the posterior $f(\boldsymbol{\theta})^N = \prod_{n=1}^N f_n(\boldsymbol{\theta}) \approx \prod_{n=1}^N p(\boldsymbol{x}_n|\boldsymbol{\theta})$. In this spirit, the new algorithm employs direct iterative refinement of a global approximation comprising the prior and N copies of a single approximating factor, $f(\boldsymbol{\theta})$, that is $q(\boldsymbol{\theta}) \propto f(\boldsymbol{\theta})^N p_0(\boldsymbol{\theta})$.

SEP uses updates that are analogous to EP's in order to refine $f(\boldsymbol{\theta})$ in such a way that it captures the *average* effect a likelihood function has on the posterior. First the cavity distribution is formed by removing one of the copies of the factor, $q_{-1}(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})/f(\boldsymbol{\theta})$. Second, the corresponding likelihood is included to produce the tilted distribution $\tilde{p}_n(\boldsymbol{\theta}) \propto q_{-1}(\boldsymbol{\theta})p(\boldsymbol{x}_n|\boldsymbol{\theta})$ and, third, SEP finds an intermediate factor approximation by moment

matching, $f_n(\boldsymbol{\theta}) \leftarrow \texttt{proj}[\tilde{p}_n(\boldsymbol{\theta})]/q_{-1}(\boldsymbol{\theta})$. Finally, having updated the factor, it is included into the approximating distribution. It is important here *not* to make a full update since $f_n(\boldsymbol{\theta})$ captures the effect of just a single likelihood function $p(\boldsymbol{x}_n|\boldsymbol{\theta})$. Instead, damping should be employed to make a partial update $f(\boldsymbol{\theta}) \leftarrow f(\boldsymbol{\theta})^{1-\varepsilon} f_n(\boldsymbol{\theta})^{\varepsilon}$, and a natural choice uses $\varepsilon = 1/N$ which can be interpreted as minimising $\mathrm{KL}[\tilde{p}_n(\boldsymbol{\theta})||p_0(\boldsymbol{\theta})f(\boldsymbol{\theta})^N]$ in the moment update.

SEP is summarised in Algorithm 5. Unlike ADF, the cavity is formed by dividing out $f(\boldsymbol{\theta})$ which captures the average effect of the likelihood and prevents the posterior from collapsing. Like ADF, however, SEP only maintains the global approximation $q(\boldsymbol{\theta})$ since $f(\boldsymbol{\theta}) \propto (q(\boldsymbol{\theta})/p_0(\boldsymbol{\theta}))^{\frac{1}{N}}$ and $q_{-1}(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})^{1-\frac{1}{N}} p_0(\boldsymbol{\theta})^{\frac{1}{N}}$. When Gaussian approximating factors are used, for example, SEP reduces the storage requirement of EP from $\mathcal{O}(ND^2)$ to $\mathcal{O}(D^2)$ which is a substantial saving that enables models with many parameters to be applied to large datasets. We also provide a cartoon visualisation for motivating SEP in Figure 3.1.

3.2 Algorithmic extensions to SEP

SEP has been motivated from a practical perspective by the limitations inherent in EP and ADF. In this section we extend SEP in four orthogonal directions and through these extensions relate SEP to SVI. Many of the algorithms described in this section are summarised in Figure 3.2 and they are detailed in the following discussions.

3.2.1 Parallel SEP: relating the EP fixed points to SEP

The SEP algorithm outlined above approximates one likelihood at a time which can be computationally slow. However, it is simple to parallelise the SEP updates by following the same recipe by which EP is parallelised. Consider a minibatch comprising M datapoints (for a full parallel (batch) update use $M = N$). First we form the cavity distribution for each likelihood. Unlike EP these are all identical. Next, in parallel, compute M intermediate factors $f_m(\boldsymbol{\theta}) \leftarrow \texttt{proj}[\tilde{p}_m(\boldsymbol{\theta})]/q_{-1}(\boldsymbol{\theta})$. In EP these intermediate factors become the new likelihood approximations and the approximation is updated to $q(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{n \neq m} f_n(\boldsymbol{\theta}) \prod_m f_m(\boldsymbol{\theta})$. In SEP, the same update is used for the approximating distribution, which becomes $q(\boldsymbol{\theta}) \leftarrow p_0(\boldsymbol{\theta}) f_{old}(\boldsymbol{\theta})^{N-M} \prod_m f_m(\boldsymbol{\theta})$ and, by implication, the approximating factor is $f_{new}(\boldsymbol{\theta}) = f_{old}(\boldsymbol{\theta})^{1-M/N} \prod_{m=1}^M f_m(\boldsymbol{\theta})^{1/N}$. One way of understanding parallel SEP is as a double loop algorithm. The *inner loop* produces intermediate approximations $q_m(\boldsymbol{\theta}) \leftarrow \arg\min_q \mathrm{KL}[\tilde{p}_m(\boldsymbol{\theta})||q(\boldsymbol{\theta})]$; these are then combined in the *outer loop*: $q(\boldsymbol{\theta}) \leftarrow \arg\min_q \sum_{m=1}^M \mathrm{KL}[q(\boldsymbol{\theta})||q_m(\boldsymbol{\theta})] + (N-M)\mathrm{KL}[q(\boldsymbol{\theta})||q_{old}(\boldsymbol{\theta})]$.

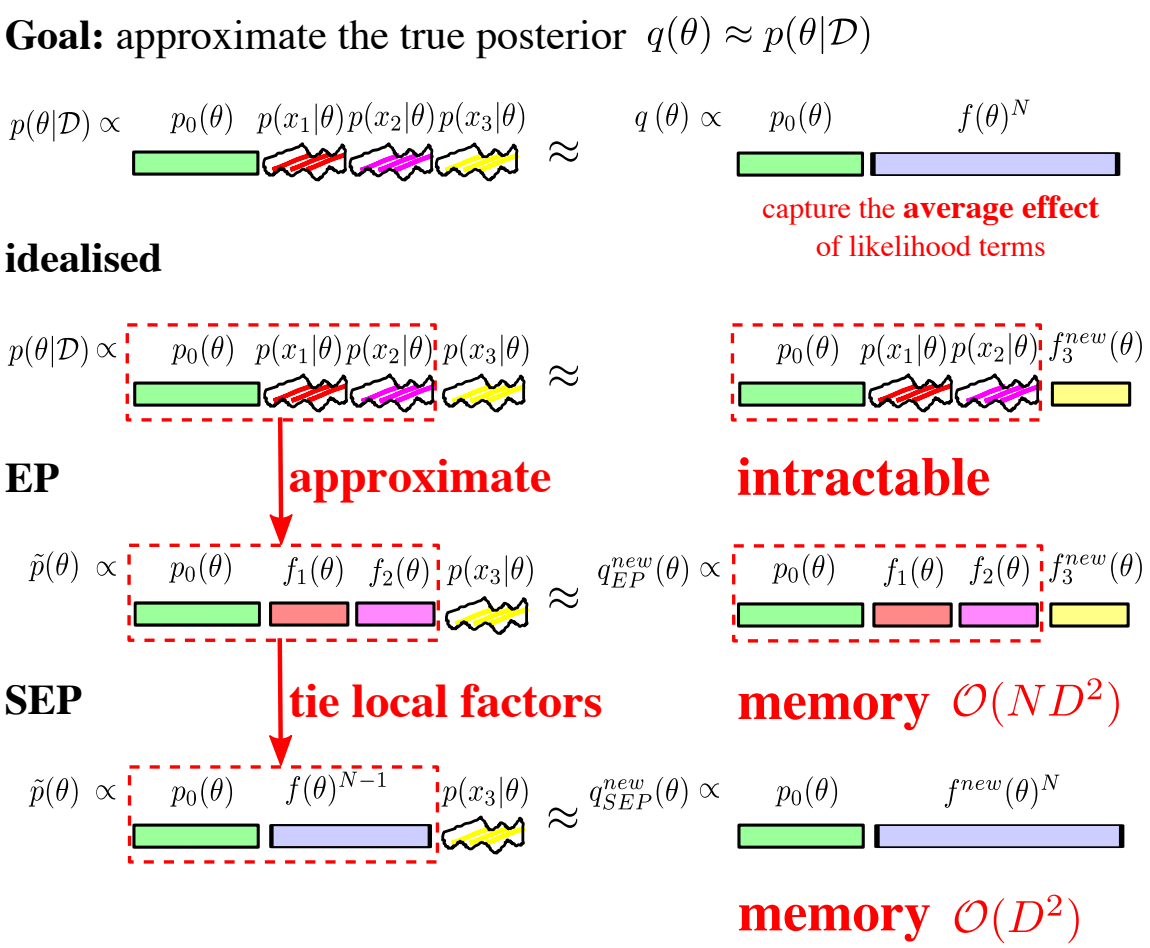


Fig. 3.1 A cartoon visualisation for the comparison of three algorithms. Here the boxes represent the prior and the approximating factors, and both of them are assumed to be tractable. The wiggled objects are the complicated likelihood terms to be approximated. An idealised algorithm would approximate each of the likelihood terms given the others fixed as contextual information, which is intractable. EP replaces the "contextual" likelihood terms with the approximating factors to form a cavity distribution, making the moment matching step tractable. Finally, SEP ties all the approximating factors, and updates the tied factor by including one randomly sampled likelihood term at each iteration. The space complexity figures are calculated with the assumption of Gaussian approximation.

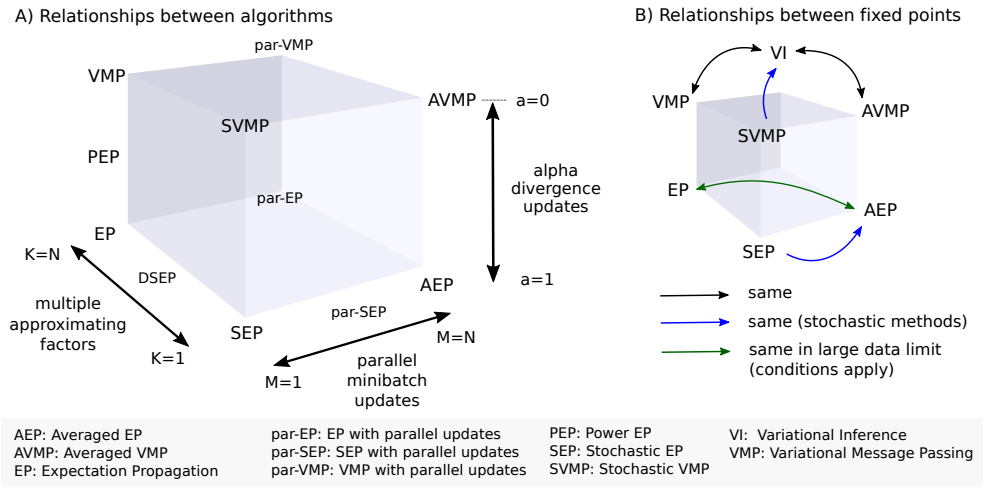


Fig. 3.2 Relationships between algorithms. Note that care needs to be taken when interpreting Minka's alpha-divergence as $a \to 0$. See the main text for further discussions.

For $M = 1$ parallel SEP reduces to the original SEP algorithm. For $M = N$ parallel SEP is equivalent to the so-called Averaged EP (AEP) algorithm proposed in a concurrent work [Dehaene and Barthelmé, 2015; Dehaene and Barthelmé, 2018] as a theoretical tool to study the convergence properties of normal EP. This work showed that when using Gaussian approximations, under fairly restrictive conditions,[3] AEP converges to the same fixed points as EP in the large data limit ($N \to \infty$).

There is another illuminating and arguably more direct connection between SEP and AEP. Since SEP's approximating factor $f(\boldsymbol{\theta})$ converges to the geometric average of the intermediate factors $\bar{f}(\boldsymbol{\theta}) \propto [\prod_{n=1}^{N} f_n(\boldsymbol{\theta})]^{\frac{1}{N}}$, SEP converges to the same fixed points as AEP, and therefore under certain conditions [Dehaene and Barthelmé, 2015; Dehaene and Barthelmé, 2018], to the same fixed points as EP. In practice decreasing learning rates, e.g. $\{\varepsilon_t\}$ satisfying the Robbins and Monro criterion [Robbins and Monro, 1951] $\sum_t \varepsilon_t = \infty$, $\sum_t \varepsilon_t^2 < \infty$, may be used to achieve convergence. It is possible that there are more direct relationships between EP and SEP's dynamics, but that is still an open question.

3.2.2 Stochastic power EP: relationships to variational methods

The SEP algorithm generalises to the power-EP case in a straight-forward manner. Instead of removing one copy of the tied factors in Algorithm 5, when power α is in use we remove α fraction of the tied factors:

$$q_{-\alpha} \propto q(\boldsymbol{\theta}) / f(\boldsymbol{\theta})^{\alpha}.$$

[3] e.g. log likelihood functions that are smooth with bounding conditions up to the 4th-order derivative, although this is not a necessary condition.

The moment matching step proceeds as in the power EP algorithm, precisely,

$$f_n(\boldsymbol{\theta})^{\alpha} \leftarrow \text{proj}[q_{-\alpha}(\boldsymbol{\theta})p(\boldsymbol{x}_n|\boldsymbol{\theta})^{\alpha}]/q_{-\alpha}(\boldsymbol{\theta}).$$

Similar to the power-EP case, the projection operator becomes I-projection when setting $\alpha \to 0$.

The relationship between variational inference and stochastic variational inference [Hoffman et al., 2013] mirrors the relationship between EP and SEP. Can these relationships be made more formal? In the following we show that, if the moment projection step in EP is replaced by a natural parameter matching step (i.e. I-projection) then the resulting algorithm is equivalent to the Variational Message Passing (VMP) algorithm [Minka, 2005; Winn and Bishop, 2005].

We first briefly sketch the VMP algorithm using the EP framework but replacing the moment matching step with natural parameter matching. We assume the approximate posterior $q(\boldsymbol{\theta})$ is in some exponential family:

$$q(\boldsymbol{\theta}) \propto \exp\left[\langle \boldsymbol{\lambda}_q, \boldsymbol{\Phi}(\boldsymbol{\theta})\rangle\right]. \tag{3.2}$$

At time t we have the current estimate of the natural parameter $\boldsymbol{\lambda}_q^t$, which is defined as the sum of local variational parameters: $\boldsymbol{\lambda}_q^t \triangleq \boldsymbol{\lambda}_0 + \sum_{n=1}^N \boldsymbol{\lambda}_n^t$.[4] Here $\boldsymbol{\lambda}_0$ represents the natural parameter of the prior distribution $p_0(\boldsymbol{\theta})$. VMP iteratively computes the update of each local estimate $\boldsymbol{\lambda}_n^{t+1}$ in the following procedure. First VMP computes the expected sufficient statistics $\hat{\boldsymbol{s}}_n$ about datapoint $\boldsymbol{x}_n$ using $\boldsymbol{\lambda}_q^t$, e.g. $\hat{\boldsymbol{s}}_n = E_q[t(z_n, x_n)]$ in the original SVI paper [Hoffman et al., 2013]. Then VMP forms the gradient by differentiating the variational lower-bound but with $q_{-1}(\boldsymbol{\theta})$ as the prior:

$$\nabla_{\boldsymbol{\lambda}_q^t} \mathcal{L}_{\text{VI}} \propto \boldsymbol{\lambda}_{-1}^t + \hat{\boldsymbol{s}}_n - \boldsymbol{\lambda}_q^t, \quad \boldsymbol{\lambda}_{-1}^t := \boldsymbol{\lambda}_q^t - \boldsymbol{\lambda}_n^t. \tag{3.3}$$

Next VMP zeros the gradient and recovers the current update $\boldsymbol{\lambda}_n^{t+1} = \hat{\boldsymbol{s}}_n$. The stochastic version of VMP, if extended in a way as SEP is developed from EP, defines the global variational parameters as $\boldsymbol{\lambda}_q^t \triangleq \boldsymbol{\lambda}_0 + N\boldsymbol{\lambda}^t$. It computes the expected sufficient statistics $\hat{\boldsymbol{s}}_n$ in the same way as VMP but changes the cavity to $\boldsymbol{\lambda}_{-1}^t = \boldsymbol{\lambda}_q^t - \boldsymbol{\lambda}^t$ in the variational lower-bound maximisation steps. Readers can verify that this returns the current update $\boldsymbol{\lambda}^{t+1} = \hat{\boldsymbol{s}}_n$ using the important fact that the local update (for example $\hat{\boldsymbol{s}}_n = E_q[t(z_n, x_n)]$) *only* depends on the global parameter $\boldsymbol{\lambda}_q^t$. Now since we tie all the local updates, the global parameter update

[4] This notation implicitly assumes that the prior also belongs to the approximate distribution family $\mathcal{Q}$. In general we can propose another factor to approximate $p_0(\boldsymbol{\theta})$, and our result still applies.

$\boldsymbol{\lambda}_q^{t+1} = \boldsymbol{\lambda}_0 + N\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}_0 + N\hat{\boldsymbol{s}}_n$. In practice we perform a damped update, where a typical choice of step size is $\varepsilon = 1/N$ like in SEP:

$$\boldsymbol{\lambda}_q^{t+1} \leftarrow (1 - \frac{1}{N})\boldsymbol{\lambda}_q^t + \frac{1}{N}(\boldsymbol{\lambda}_0 + N\boldsymbol{\lambda}^{t+1}) = \boldsymbol{\lambda}_0 + (N-1)\boldsymbol{\lambda}^t + \hat{\boldsymbol{s}}_n. \tag{3.4}$$

On the other hand, Mandt and Blei [2014] summarises SVI as to compute the current update by zeroing the gradient

$$\nabla_{\boldsymbol{\lambda}_q} \mathcal{L}_{\text{VI}} \propto \boldsymbol{\lambda}_0 + N\hat{\boldsymbol{s}}_n - \boldsymbol{\lambda}_q, \tag{3.5}$$

which returns $\boldsymbol{\lambda}_q^{t+1} = \boldsymbol{\lambda}_0 + N\hat{\boldsymbol{s}}_n$ as well. This implies that SVI, when using learning rate $\varepsilon = 1/N$, is equivalent to SVMP. More generally, the procedure can be applied any member of the PEP family of algorithms, but care has to be taken when taking the limiting cases $\alpha \to 0$. These results lend weight to the view that SEP is a natural stochastic generalisation of EP.

3.2.3 Distributed SEP: controlling granularity of the approximation

EP uses a fine-grained approximation comprising a single factor for each likelihood. SEP, on the other hand, uses a coarse-grained approximation comprising a signal global factor to approximate the average effect of all likelihood terms. One might worry that SEP's approximation is too severe if the dataset contains sets of data-points that have very different likelihood contributions (consider classifying handwritten digits into odd and even classes, for example). It might be more sensible in such cases to partition the dataset into J disjoint pieces $\{\mathcal{D}_j = \{\boldsymbol{x}_n\}_{n=N_{j-1}}^{N_j}\}_{j=1}^J$ with $N = \sum_{j=1}^J N_j$ and use an approximating factor for each partition. If normal EP updates are performed on the subsets, i.e. treating $p(\mathcal{D}_j|\boldsymbol{\theta})$ as a single true factor to be approximated, we arrive at the distributed EP (DEP) algorithm [Gelman et al., 2014; Xu et al., 2014], summarised in Algorithm 6 and detailed as the following.

After data partitioning, the posterior distribution and likelihood functions are denoted as

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p_0(\boldsymbol{\theta}) \prod_{j=1}^J p(\mathcal{D}_j|\boldsymbol{\theta}), \tag{3.6}$$

$$p(\mathcal{D}_j|\boldsymbol{\theta}) = \prod_{\boldsymbol{x}_n \in \mathcal{D}_j} p(\boldsymbol{x}_n|\boldsymbol{\theta}). \tag{3.7}$$

Next DEP assigns factors to each sub-dataset likelihood, i.e. $q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_{j=1}^J F_j(\boldsymbol{\theta})$ with each $F_j(\boldsymbol{\theta})$ approximating $p(\mathcal{D}_j|\boldsymbol{\theta})$. The projection step is no longer analytically tractable in general since the tilted distribution with multiple data-points often lacks a simple form.

Instead DEP handles moment matching by nesting additional approximations, e.g. MCMC, which might be undesirable in terms of time complexity figures.

How does the factor tying idea apply in this case? A naive implementation would simply tie the sub-dataset level factors, but one should notice that N_j might not be equal for different subsets. Instead we still construct datapoint level approximate factors similar to SEP, i.e. $q(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta}) f(\boldsymbol{\theta})^N$, but construct the update in DEP fashion. In other words, N_j factors are replaced by the likelihood terms in the jth subset in order to form the tilted distribution. Later these N_j copies are updated by exactly the same moment matching step as in DEP. We name this algorithm stochastic EP *on* data partitions and to distinguish from another algorithm presented later we abbreviate it as stochastic distributed EP (SDEP) (see Algorithm 7). It is trivial to parallelise this method just as one would do for DEP, and we abbreviate the corresponding version as ADEP.

To have a deterministic[5] counterpart of DEP, we consider SEP/AEP *inside* each partition. This strategy might be preferred when one wish to carve up the data and carry out deterministic inference routines distributed across machines. We name this approach as Distributed SEP/AEP (DSEP/DAEP). Different to both DEP and SDEP, the approximate posterior for DSEP is defined as $q(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) \prod_j f_j(\boldsymbol{\theta})^{N_j}$, with $f_j(\boldsymbol{\theta})^{N_j}$ approximating $p(\mathcal{D}_j|\boldsymbol{\theta})$. The computations are almost the same as SEP/AEP except that the updates only modifies the copies of the corresponded subset. This method is also detailed in Algorithm 8, and the differences between the two factor tying proposals are also visualised in Figure 3.3.

In summary, a flexible class of EP-like approximate inference algorithms can be derived, by carving up the dataset and designing the approximating factor structures and update procedures. Computational complexity figures are further presented in Section 3.3 to allow selections of algorithms confronting computation constraints.

3.2.4 SEP for latent variable models

Many applications of EP involve latent variable models. Although this is not the main focus of the development, we show that SEP is applicable in this case and again it prevents the memory footprint from scaling with N. Critically as we shall see, the local factors $g_n(\boldsymbol{z}_n)$ do not need to be maintained in memory, although retaining them in memory might provide better initialisations that speeds up convergence potentially. This means that all of the advantages of SEP carry over to more complex models involving latent variables.

We consider a model containing latent variables $\boldsymbol{z}_n$ associated with each observation $\boldsymbol{x}_n$, which are drawn i.i.d. from a prior $p_0(\boldsymbol{z}_n)$. SEP proposes approximations to the exact

[5] in the sense that the moment matching step does not employ Monte Carlo.

Algorithm 6 DEP

1: compute cavity distribution $q_{-j}(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})/F_j(\boldsymbol{\theta})$
2: compute tilted distribution $\tilde{p}_j(\boldsymbol{\theta}) \propto p(\mathcal{D}_j|\boldsymbol{\theta})q_{-j}(\boldsymbol{\theta})$
3: moment matching: $F_j(\boldsymbol{\theta}) \leftarrow \texttt{proj}[\tilde{p}_j(\boldsymbol{\theta})]/q_{-j}(\boldsymbol{\theta})$

Algorithm 7 SDEP

1: compute cavity distribution $q_{-j}(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})/f(\boldsymbol{\theta})^{N_j}$
2: compute tilted distribution $\tilde{p}_j(\boldsymbol{\theta}) \propto p(\mathcal{D}_j|\boldsymbol{\theta})q_{-j}(\boldsymbol{\theta})$
3: moment matching: $F_j(\boldsymbol{\theta}) \leftarrow \texttt{proj}[\tilde{p}_j(\boldsymbol{\theta})]/q_{-j}(\boldsymbol{\theta})$
4: inclusion: $f(\boldsymbol{\theta}) \leftarrow f(\boldsymbol{\theta})^{1-N_j/N}F_j(\boldsymbol{\theta})^{1/N}$

Algorithm 8 DSEP

1: compute cavity distribution $q_{-1}(\boldsymbol{\theta}) = q(\boldsymbol{\theta})/f_j(\boldsymbol{\theta})$
2: choose a datapoint $\boldsymbol{x}_n \sim \mathcal{D}_j$
3: compute tilted distribution $\tilde{p}_j^{(n)}(\boldsymbol{\theta}) \propto p(\boldsymbol{x}_n|\boldsymbol{\theta})q_{-1}(\boldsymbol{\theta})$
4: moment matching: $f_j^{(n)}(\boldsymbol{\theta}) \leftarrow \texttt{proj}[\tilde{p}_j^{(n)}(\boldsymbol{\theta})]/q_{-1}(\boldsymbol{\theta})$
5: inclusion: $f_j(\boldsymbol{\theta}) \leftarrow f_j(\boldsymbol{\theta})^{1-1/N_j}f_j^{(n)}(\boldsymbol{\theta})^{1/N_j}$

posterior over parameters and hidden variables

$$p(\boldsymbol{\theta}, \{\boldsymbol{z}_n\}|\mathcal{D}) \propto p_0(\boldsymbol{\theta}) \prod_n p_0(\boldsymbol{z}_n) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) \tag{3.8}$$

by tying the factors for the global parameter $\boldsymbol{\theta}$ but retaining the local factors for the hidden variables:

$$q(\boldsymbol{\theta}, \{\boldsymbol{z}_n\}) \propto p_0(\boldsymbol{\theta}) f(\boldsymbol{\theta})^N \prod_{n=1}^{N} g_n(\boldsymbol{z}_n). \tag{3.9}$$

In other words, SEP uses $f(\boldsymbol{\theta})g_n(\boldsymbol{z}_n)$ to approximate $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta})p_0(\boldsymbol{z}_n)$.

Next we show a critical advantage of SEP for approximating latent variable posterior: the local factors $g_n(\boldsymbol{z}_n)$ do not need to be maintained in memory (again it might help to do so for better initialisation). More formally, the cavity distribution is $q_{-n}(\boldsymbol{\theta}, \{\boldsymbol{z}_n\}) \propto q(\boldsymbol{\theta}, \{\boldsymbol{z}_n\})/(f(\boldsymbol{\theta})g_n(\boldsymbol{z}_n))$ and the tilted distribution is

$$\tilde{p}_n(\boldsymbol{\theta}, \{\boldsymbol{z}_n\}) \propto q_{-n}(\boldsymbol{\theta}, \{\boldsymbol{z}_n\}) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) p_0(\boldsymbol{z}_n).$$

This leads to a moment-update that minimises

$$\mathrm{KL}[p_0(\boldsymbol{\theta})f(\boldsymbol{\theta})^{N-1}p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta})p_0(\boldsymbol{z}_n) \prod_{m\neq n} g_m(\boldsymbol{z}_m) || p_0(\boldsymbol{\theta})f(\boldsymbol{\theta})^{N-1} f'(\boldsymbol{\theta}) g_n(\boldsymbol{z}_n) \prod_{m\neq n} g_m(\boldsymbol{z}_m)].$$

with respect to $f'(\boldsymbol{\theta})g_n(\boldsymbol{z}_n)$. Importantly, the terms involving $\prod_{m\neq n} g_m(\boldsymbol{z}_m)$ are cancelled, meaning that these factors do not contribute to the local approximation step. For simple models the moments of $\boldsymbol{z}_n$ can be computed analytically given $q_{-1}(\boldsymbol{\theta})$, thus $g_n(\boldsymbol{z}_n)$ is never stored in memory, resulting in a reduced memory footprint by a factor of N again. It is also possible to have latent variables globally shared or shared in a data piece $\mathcal{D}_j$. But we

Goal: approximate the true posterior $q(\theta) \approx p(\theta|\mathcal{D})$

$$p(\theta|\mathcal{D}) \propto p_0(\theta)\ p(\mathcal{D}_1|\theta)\ p(\mathcal{D}_2|\theta)$$

$$\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2,\ \mathcal{D}_1 = \{x_1\},\ \mathcal{D}_2 = \{x_2, x_3\}$$

$$N = 3, N_1 = 1, N_2 = 2$$

DEP

$$\tilde{p}(\theta) \propto p_0(\theta)\ F_1(\theta)\ p(\mathcal{D}_2|\theta) \approx q^{new}(\theta) \propto p_0(\theta)\ F_1(\theta)\ F_2^{new}(\theta)$$

SDEP

$$\tilde{p}(\theta) \propto p_0(\theta)\ f(\theta)^{N-N_2}\ p(\mathcal{D}_2|\theta) \approx q^{new}(\theta) \propto p_0(\theta)\ f^{new}(\theta)^N$$

DSEP

$$\tilde{p}(\theta) \propto p_0(\theta)\ f_1(\theta)^{N_1} f_2(\theta)^{N_2-1}\ p(x_3|\theta) \approx q^{new}(\theta) \propto p_0(\theta)\ f_1(\theta)^{N_1}\ f_2^{new}(\theta)^{N_2}$$

Fig. 3.3 A cartoon visualisation on comparing DEP, SDEP and DSEP. One should notice that the definitions of $F_j(\boldsymbol{\theta})$ in DEP is different from $f_j(\boldsymbol{\theta})$ in DSEP. Here SDEP's moment matching step would typically require another approximate inference procedure, e.g. simulating MCMC dynamics for a long time. On the other hand DSEP still use cheap updates computed on a single datum, which can be faster.

can also extend SEP to these latent variables accordingly, which still provides computation gains in space complexity. In mathematical form, assume $\boldsymbol{z}_j$ a latent variable shared in $\mathcal{D}_j$. A prevalent example for this is the latent Dirichlet allocation (LDA) model [Blei et al., 2003], where $\mathcal{D}_j$ represents the j^{th} document in the corpus and $\boldsymbol{z}_j$ represents the topic of the document. Then we construct $q(\boldsymbol{z}_j) \propto p_0(\boldsymbol{z}_j) g_k(\boldsymbol{z}_j)^{N_j}$ to approximate its posterior. This procedure still reduces memory by roughly a factor of N/J.

> **Remark** (amortised inference nested in SEP) **.** In practice people may prefer maintaining the g factors in memory, if the moment computation requires another optimisation inner-loop (which might be more expensive than the moment matching step itself). Examples of methods where this may be the case include latent Dirichlet allocation [Blei et al., 2003] that has a hierarchy of latent variables, where VI methods also store variational q distributions for some of the hidden variables. One potential recipe in this scenario is to

apply amortised inference techniques, where in this case we can optimise the variational parameter $g_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)$ for the factors attached to the latent variables. Another very closed related idea is to learn a model for the moments/messages passed in each SEP step with confidence estimates. A particular compelling feature of this approach is that one can control the use of the proposed message update by the model, and if rejected the algorithm can resort to the more expensive exact message computation using e.g. MCMC. Interested readers are directed to [Heess et al., 2013; Jitkrittum et al., 2015] for more details.

3.3 Computational complexity

Besides the approximation accuracy, time and space complexities are also crucial for an efficient approximate inference algorithm to be widely-applicable, especially in systems that handle very large-scale datasets and also require fast computations. To provide a direct comparison between the methods discussed so far, we present the space and time complexity factors by considering Gaussian approximations with full covariance matrices.

We assume an MCMC method is used for the DEP moment matching step, and assume it has time complexity $\mathcal{O}(H(n,D))$ if n likelihood terms are involved and the random variable $\boldsymbol{\theta}$ is of dimension D, since it handles multiple likelihood functions at the same time for a single factor update. We also denote the time complexity factor of a normal EP moment matching step as $\mathcal{O}(h(D))$. Also since EP-like methods handle the natural parameters, for Gaussian approximations it requires $\mathcal{O}(D^3)$ time for matrix inversion and $\mathcal{O}(D^2)$ for matrix-vector multiplication in the inclusion step. By assumption EP's update step for a single factor can be done in a fast way, which means for a full pass through of the dataset $H(N,D)+D^3+D^2 \geq N(h(D)+D^3+D^2)$. Furthermore in practice low-rank approximations can be applied to remove the $\mathcal{O}(D^3)$ factor if the random variable is very high dimensional, e.g. see Qi et al. [2010] in EP literature. For the concern of memory usage, every factor in use occupies $\mathcal{O}(D^2)$ storage, although sparse approximations can significantly improve space complexity figures as well.

With all the above set-ups, we present the complexity figures in Table 3.1, which apply to fully observed models, e.g. Bayesian neural networks for classification tasks. Like the comparison between batch and stochastic learning methods, AEP-type methods produce more robust updates, and can be significantly faster than SEP-type methods if executed in parallel. However DEP/AEP algorithms require storing the factors in local worker machines, so in this regard SEP provides significant advantage in memory consumption in the price of slower convergence.

Table 3.1 Complexity figures for the EP algorithms discussed (with Gaussian approximations, full covariance matrices). The time complexity numbers are counted on a full pass of dataset, and the global approximations are updated after each moment computation. We assume the dataset is evenly split into J disjoint subsets when applicable.

Algorithm	Time complexity	Space complexity
DEP (parallel)	$\mathcal{O}(D^3 + H(N/J, D) + D^2)$	$\mathcal{O}(JD^2)$
SDEP (sequel)	$\mathcal{O}(J(D^3 + H(N/J, D) + D^2))$	$\mathcal{O}(D^2)$
ADEP (parallel)	$\mathcal{O}(D^3 + H(N/J, D) + D^2)$	$\mathcal{O}(JD^2)$
DSEP (sequel)	$\mathcal{O}(N(D^3 + h(D) + D^2))$	$\mathcal{O}(D^2)$
DAEP (parallel)	$\mathcal{O}(J(D^3 + h(D) + D^2))$	$\mathcal{O}(JD^2)$
SEP	$\mathcal{O}(N(D^3 + h(D) + D^2))$	$\mathcal{O}(D^2)$
AEP (parallel)	$\mathcal{O}(D^3 + h(D) + D^2)$	$\mathcal{O}(ND^2)$
ADF (multi. pass)	$\mathcal{O}(N(h(D)))$	$\mathcal{O}(D^2)$
Normal EP	$\mathcal{O}(N(D^3 + h(D) + D^2))$	$\mathcal{O}(ND^2)$
Sampling	$\mathcal{O}(H(N, D))$	$\mathcal{O}(D^2)$
SVI	$\mathcal{O}(N(D^3 + \tilde{h}(D) + D^2))$	$\mathcal{O}(D^2)$

To conclude the discussion of computational efficiency we compare the complexity figures of SEP against those of SVI. Here we refer SVI to the general version which uses gradient descent, since the likelihood terms are very unlikely to be conjugate to the Gaussian approximation (thus the natural gradient descent algorithm [Hoffman et al., 2013] does not apply). It is straightforward to see that both algorithms require the same amount of storage, which is $\mathcal{O}(D^2)$ in the Gaussian case. For the gradient descent update, as the gradient of the entropy term $\mathbb{H}[q]$ in the variational lower-bound requires computing the covariance matrix, this means the precision matrix as one of the natural parameters needs to be inverted. Therefore the cost for each gradient descent step in SVI is $\mathcal{O}(\tilde{h}(D) + D^3 + D^2)$, with $\mathcal{O}(D^3)$ the matrix inversion time, $\mathcal{O}(D^2)$ the update time for the precision matrix, and $\mathcal{O}(\tilde{h}(D))$ the computation time for the gradients of other terms in the variational lower-bound. SVI might be more efficient in time complexity per iteration (i.e. the runtime for processing one incoming observation, therefore $\tilde{h}(D) \leq h(D)$). However, as SEP uses fixed-point iterative updates, in practice SEP often converges in less passes of data than SVI thus is faster in total runtime, which is something that the above complexity analysis cannot take into account. For example Hernández-Lobato and Hernández-Lobato [2016] showed that SEP, when applied to sparse GP classification, is significantly faster than SVI. SEP can be further accelerated using low-rank approximations when D is large, which typically reduces the matrix inversion time to $\mathcal{O}(Dd^2)$ if we use rank-d approximations.

3.4 Experiments

The purpose of the experiments was to evaluate SEP on a number of datasets (synthetic and real-world, small and large) and on a number of models (probit regression, mixture of Gaussians for clustering, and Bayesian neural networks).

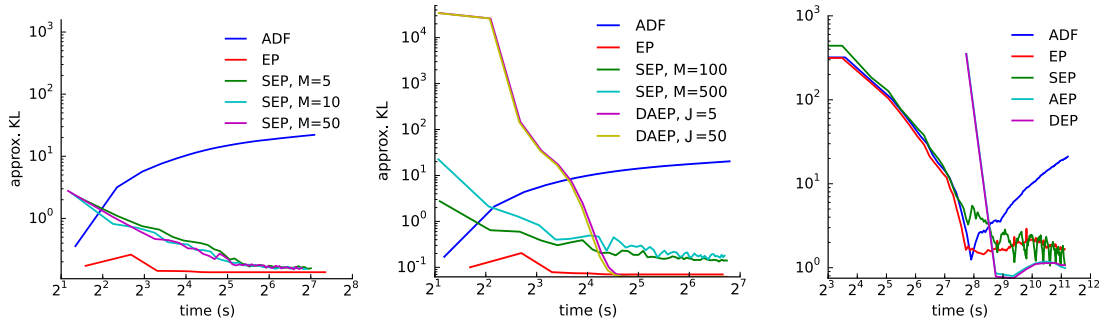
3.4.1 Bayesian probit regression

The first experiments considered a simple Bayesian classification problem and investigated the stability and quality of SEP in relation to EP and ADF as well as the effect of using mini-batches and varying the granularity of the approximation. The model comprised a probit likelihood function $P(\boldsymbol{y}_n = 1|\theta) = \Phi(\boldsymbol{\theta}^T \boldsymbol{x}_n)$ and a Gaussian prior over the hyper-plane parameter $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \gamma I)$. The synthetic data comprised $N = 5{,}000$ datapoints $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, where $\boldsymbol{x}_n$ were $D = 4$ dimensional and were either sampled from a single Gaussian distribution (Fig. 3.4(a)) or from a mixture of Gaussians (MoGs) with 5 components (Fig. 3.4(b)) to investigate the sensitivity of the methods to the homogeneity of the dataset. The labels were produced by sampling from the generative model. We followed Xu et al. [2014] to measure the performance by computing an approximation of $\mathrm{KL}[p(\boldsymbol{\theta}|\mathcal{D})||q(\boldsymbol{\theta})]$, where $p(\boldsymbol{\theta}|\mathcal{D})$ was replaced by a Gaussian that had the same mean and covariance as samples drawn from the posterior using the No-U-Turn sampler (NUTS) [Hoffman and Gelman, 2014]. This evaluation metric measures how close the approximated first and second moments are to those of the exact posterior, and emphasises well calibrated uncertainty estimations.

Results in Fig. 3.4(a) indicate that EP is the best performing method and that ADF collapses towards a delta function. SEP converges to a solution which appears to be of similar quality to that obtained by EP for the dataset containing Gaussian inputs, but slightly worse when the MoGs was used. Variants of SEP that used larger mini-batches fluctuated less, but typically took longer times to converge (although for the small mini-batches shown this effect is not clear). The utility of finer grained approximations depended on the homogeneity of the data. For the second dataset containing MoGs inputs (shown in Fig. 3.4(b)), finer grained approximations were found to be advantageous if the datapoints from each mixture component are assigned to the same approximating factor. Generally it was found that there is no advantage to retaining more approximating factors than there were clusters in the dataset.

Although not a main purpose, we further test the performance of SEP with sampling methods to compute moments (i.e. SDEP).[6] We re-use the settings of probit regression but change the probit unit to sigmoid function, making the moment projection analytically intractable. We randomly partition the dataset into $J = 20$ subsets $\{\mathcal{D}_j\}$, construct the

[6] code adjusted from `ep-stan`: https://github.com/gelman/ep-stan



(a) (b) (c)

Fig. 3.4 Bayesian logistic regression experiments. Panels (a) and (b) show synthetic data experiments. Panel (c) shows the performance of EP-like methods on Bayesian logistic regression with the moment matching step computed by NUTS. M denotes the mini-batch size for data sub-sampling (may be within a data piece for panel (c)). EP (in red curves) is the best performing method in terms of the approximate KL metric. SEP works slightly worse but with its performance approaching to EP as the running time grows. ADF over-counts the number of observations and thus returns worst results as expected.

approximate posterior with local factors over the subsets, and tie them in SEP/AEP as before. Note that we perform sequential computations for DEP and AEP although they are ideally suited for parallel computing. Again as presented in Figure 3.4(c), SEP performs almost as well as EP, which further justifies SEP even with sampling methods. Also AEP is indistinguishable from DEP, but it reduces memory by a factor of N/J.

To see if the trends carry to real-world datasets, we tested SEP's performance on six small binary classification datasets from the UCI machine learning repository.[7] We did not consider the effect of mini-batches or the granularity of the approximation, using $J = M = 1$. We ran the tests with damping and stopped learning after convergence (by monitoring the updates of approximating factors). The classification results are summarised in Table 3.2. ADF performs reasonably well on the mean classification error metric, presumably because it tends to learn a good approximation to the posterior mode. However, the posterior variance is poorly approximated and therefore ADF returns poor test log-likelihood scores. EP achieves significantly higher test log-likelihood than ADF indicating that a superior approximation to the posterior variance is attained. Crucially, SEP performs very similarly to EP, implying that SEP is an accurate alternative to EP even though it is refining a cheaper global posterior approximation.

[7] https://archive.ics.uci.edu/ml/index.html

Table 3.2 Average test results all methods on probit regression (mean and standard error reported). All methods capture a good posterior mode, however EP outperforms ADF in terms of test log-likelihood on almost all the datasets, with SEP performing similarly to EP.

	mean error			test log-likelihood		
Dataset	**ADF**	**SEP**	**EP**	**ADF**	**SEP**	**EP**
Australian	0.328±0.0127	**0.325±0.0135**	0.330±0.0133	-0.634±0.010	**-0.631±0.009**	**-0.631±0.009**
Breast	0.037±0.0045	**0.034±0.0034**	**0.034±0.0039**	-0.100±0.015	-0.094±0.011	**-0.093±0.011**
Crabs	0.056±0.0133	**0.033±0.0099**	0.036±0.0113	-0.242±0.012	-0.125±0.013	**-0.110±0.013**
Ionos	**0.126±0.0166**	0.130±0.0147	0.131±0.0149	-0.373±0.047	-0.336±0.029	**-0.324±0.028**
Pima	0.242±0.0093	0.244±0.0098	**0.241±0.0093**	-0.516±0.013	-0.514±0.012	**-0.513±0.012**
Sonar	**0.198±0.0208**	**0.198±0.0217**	**0.198±0.0243**	-0.461±0.053	-0.418±0.021	**-0.415±0.021**

3.4.2 DSEP experiments and grouping tests

The assumption we made in the main text to achieve SEP $\approx$ full EP is that the contributions of each likelihood term to the posterior are very similar. We show further results here on the approximation produced by different EP methods when we believe there exists heterogeneity in data. We generated synthetic XOR classification data by sampling from 4 unit Gaussians with means $(3,3)$, $(-3,-3)$, $(3,-3)$ and $(-3,3)$, and labelling the clusters centred at the former two as negative examples (and positive for the others). The model $p(y_n|\boldsymbol{x}_n,\boldsymbol{\theta})$ is kernel probit regression using RBF kernel with width $l = 1.0$, which is the same as the model presented in Section 3.4.1 except that the features are changed to kernel representations. This makes the feature vectors high dimensional, and the local nature of kernels also makes them very different if the datapoints belong to different clusters. We generated 50×4 test data and $\{10 \times 4, 20 \times 4, 50 \times 4\}$ training data and ran SEP/DSEP/full EP to approximate the posterior distribution of $\boldsymbol{\theta}$. For DSEP we partitioned the dataset into 4 subsets according to the associated centroid. Each experiment was repeated 10 times to collect average test data log-likelihood and classification error.

Table 3.3 shows the quantitative numbers of performances and Figure 3.5 visualises the contours of probability $p(y = 1|\boldsymbol{x},\mathcal{D})$ with true posterior approximated by $q(\boldsymbol{\theta})$. Interestingly SEP is slightly better then the others on the classification error metric. But importantly EP achieves the best test log-likelihood numbers and in general DSEP produces very similar results (shown by both the table and the figure), meaning that even for small datasets running full EP might be unnecessary. Also the three methods become indistinguishable when the size of the dataset N increases. We argue the main reason is that the posterior contributions are getting similar since more datapoints are observed in the circle of kernel width.

We further tested the robustness of all three methods to outliers. We reused the settings above and randomly flipped 10% labels of training data. Qualitative results in Figure 3.6 show that SEP is almost as robust as DSEP/EP in this example. We had tried different types

Table 3.3 Average test results of all methods on kernel probit regression.

	mean error			test log-likelihood		
N	**SEP**	**DSEP**	**EP**	**SEP**	**DSEP**	**EP**
10×4	**0.032±0.0058**	0.055±0.0127	**0.032±0.0097**	-0.405±0.011	-0.380±0.010	**-0.378±0.009**
20×4	**0.007±0.0014**	0.008±0.0024	0.012±0.0031	-0.326±0.007	-0.320±0.006	**-0.317±0.003**
50×4	**0.003±0.0010**	**0.003±0.0014**	0.006±0.0009	-0.243±0.004	**-0.233±0.007**	-0.238±0.003

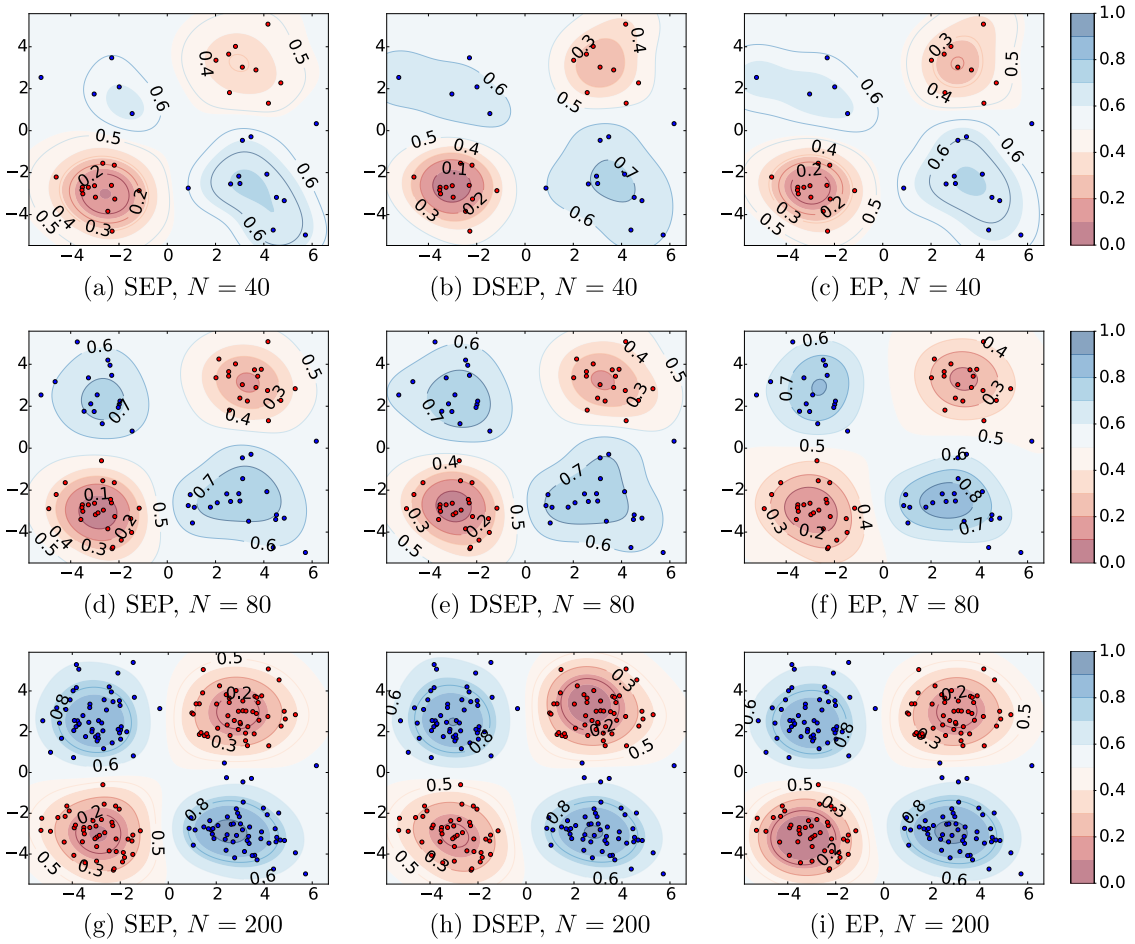
of outliers and failed to find the cases where EP/DSEP significantly outperforms SEP. Future work should answer the questions that when SEP gives bad approximations and whether it fails in the same way as EP fails.

To verify whether these conclusions about the granularity of the approximation hold in real datasets, we sampled $N = 1,000$ datapoints for each of the digits in MNIST and performed odd-vs-even classification. Each digit class was assigned its own global approximating factor, $J = 10$. We compare the log-likelihood of a test set using ADF, SEP ($J = 1$), full EP and DSEP ($J = 10$) in Figure 3.7. EP and DSEP significantly outperform ADF. DSEP is slightly worse than full EP initially, however it reduces the memory to 0.001% of full EP without losing substantial accuracy. SEP's accuracy was still increasing at the end of learning and was slightly better than ADF.

Mixture of Gaussians for clustering

The small scale experiments on probit regression indicate that SEP performs well for fully-observed probabilistic models. Although it is not the main focus of the section, we sought to test the flexibility of the method by applying it to a latent variable model, specifically a mixture of Gaussians (MoGs). A synthetic MoGs dataset containing $N = 200$ datapoints was constructed comprising 4 Gaussians. The means were sampled from a Gaussian distribution, $p(\boldsymbol{\mu}_j) = \mathcal{N}(\boldsymbol{\mu}; \boldsymbol{m}, \mathbf{I})$, the cluster identity variables $\boldsymbol{h}_n$ were sampled from a uniform categorical distribution, and each mixture component was isotropic $p(\boldsymbol{x}_n|\boldsymbol{h}_n) = \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_{\boldsymbol{h}_n}, 0.5^2\mathbf{I})$. EP, ADF and SEP were performed to approximate the joint posterior over the cluster means $\{\boldsymbol{\mu}_j\}$ and cluster identity variables $\{\boldsymbol{h}_n\}$ (the other parameters were assumed known).

Figure 3.8(a) visualises the approximate posteriors after 200 iterations. All methods return good estimates for the means, but ADF collapses towards a point estimate as expected. SEP, in contrast, captures the uncertainty and returns nearly identical approximations to EP. The accuracy of the methods is quantified in Fig. 3.8(b) by comparing the approximate posteriors to those obtained from NUTS. In this case the approximate KL-divergence measure is analytically intractable, instead we used the averaged Frobenius-norm (F-norm) of the difference of the Gaussian parameters fitted by NUTS and EP methods. These measures confirm that SEP approximates EP well.

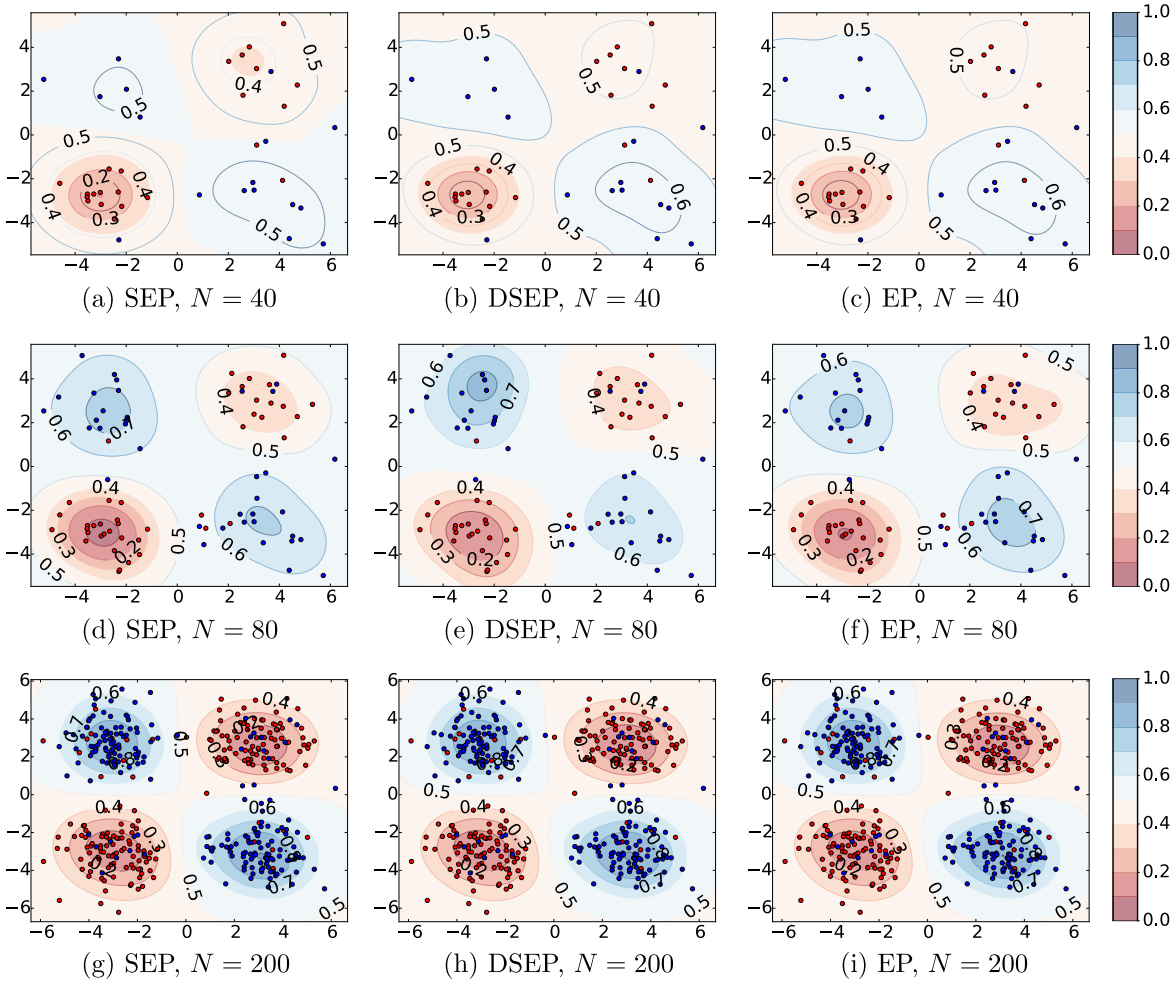


(a) SEP, $N = 40$ (b) DSEP, $N = 40$ (c) EP, $N = 40$

(d) SEP, $N = 80$ (e) DSEP, $N = 80$ (f) EP, $N = 80$

(g) SEP, $N = 200$ (h) DSEP, $N = 200$ (i) EP, $N = 200$

Fig. 3.5 Comparing predictions of kernel Probit regression trained by SEP/DSEP/EP, with increasing training data size N. Although not very significant, for $N = 40$ the decision boundary obtained by the DSEP method is more similar to that of the full-EP method. This difference vanishes as N increases.



(a) SEP, $N = 40$ (b) DSEP, $N = 40$ (c) EP, $N = 40$

(d) SEP, $N = 80$ (e) DSEP, $N = 80$ (f) EP, $N = 80$

(g) SEP, $N = 200$ (h) DSEP, $N = 200$ (i) EP, $N = 200$

Fig. 3.6 Comparing predictions of kernel probit regression trained by SEP/DSEP/EP, with 10% labels flipped. The same observations as to the results in 3.5 apply.

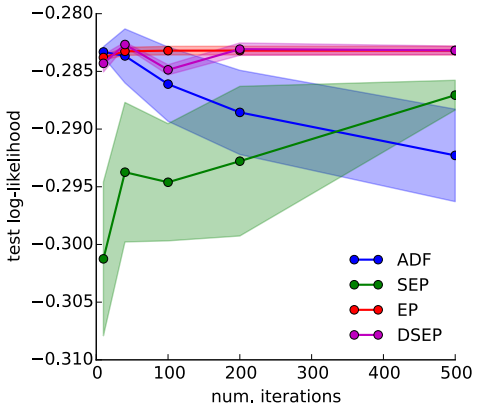
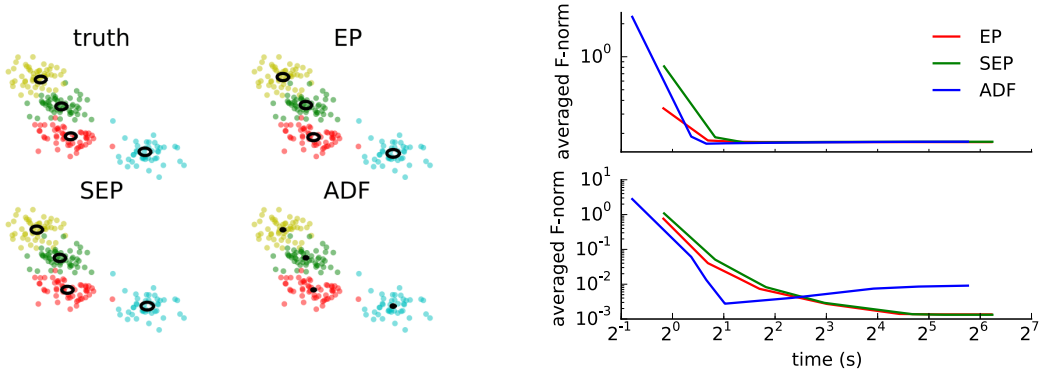


Fig. 3.7 DSEP experimental result on MNIST (mean and standard error reported, see text for full details).



(a) (b)

Fig. 3.8 Posterior approximation for the mean of the Gaussian components. (a) visualises posterior approximations over the cluster means (98% confidence level). The coloured dots indicate the true label (top-left) or the inferred cluster assignments (the rest). In (b) we show the error (in F-norm) of the approximate Gaussians' means (top) and covariances (bottom).

3.4.3 Probabilistic back-propagation for Bayesian neural nets

The final set of tests consider more complicated models and large datasets. Specifically we evaluate the methods for probabilistic back-propagation (PBP) [Hernández-Lobato and Adams, 2015], a recent state-of-the-art method for scalable Bayesian learning in neural network models.[8] Previous implementations of PBP perform several iterations of ADF over the training data. The moment-matching operations required by ADF are themselves intractable and they are approximated by first propagating the uncertainty on the synaptic weights forward through the network in a sequential way, and then computing the gradient of the marginal likelihood by back-propagation. Previous implementations of PBP are based on ADF to reduce the large memory cost that would be required by EP when the amount of available data is very large.

We performed neural network regression experiments with publicly available data sets and neural networks with one hidden layer. Table 3.5 lists the analysed data sets and shows summary statistics. We used neural networks with 50 hidden units in all cases except in the two largest ones, i.e., *Year Prediction MSD* and *Protein Structure*, where we used 100 hidden units. The different methods, SEP, EP and ADF were run by performing 40 passes over the available training data, updating the parameters of the posterior approximation after seeing each data point. The data sets were split into random training and test sets with 90% and 10% of the data, respectively. This splitting process was repeated 20 times for the small datasets, and the average test performances of each method were reported. In the two largest data sets,

[8] See Appendix A.1 for detailed update equations.

Table 3.4 Average test results for all methods on Bayesian neural networks (mean and standard error reported). Datasets are also from the UCI machine learning repository.

	RMSE			test log-likelihood		
Dataset	**ADF**	**SEP**	**EP**	**ADF**	**SEP**	**EP**
Kin8nm	0.098±0.0007	**0.088±0.0009**	0.089±0.0006	0.896±0.006	**1.013±0.011**	1.005±0.007
Naval	0.006±0.0000	**0.002±0.0000**	0.004±0.0000	3.731±0.006	**4.590±0.014**	4.207±0.011
Power	**4.124±0.0345**	4.165±0.0336	4.191±0.0349	**-2.837±0.009**	-2.846±0.008	-2.852±0.008
Protein	4.727±0.0112	**4.670±0.0109**	4.748±0.0137	-2.973±0.003	**-2.961±0.003**	-2.979±0.003
Wine	**0.635±0.0079**	0.650±0.0082	0.637±0.0076	-0.968±0.014	-0.976±0.013	**-0.958±0.011**
Year	**8.879± NA**	8.922±NA	8.914±NA	**-3.603± NA**	-3.924±NA	-3.929±NA

Year Prediction MSD and *Protein Structure*, we did the train-test splitting only one and five times respectively. The datasets were normalised so that the input features and the targets have zero mean and unit variance in the training set. The normalisation on the targets was removed for prediction.

Table 3.4 shows the average test RMSE and test log-likelihood for each method. Interestingly, SEP can outperform EP in this setting (possibly because the stochasticity enabled it to find better solutions), and typically it performed similarly. Surprisingly ADF often outperformed EP, although the results presented for ADF use a near-optimal number of sweeps and further iterations generally degraded performance. ADF's good performance is most likely due to an interaction with additional the moment-approximation that is required in PBP.

We also provide the memory consumption details for experiments using PBP in Table 3.5, where some of the results are also visualised in Figure 3.9. We observe substantial memory reductions by running SEP instead of EP, while still attaining similar accuracies. Especially for Year Prediction MSD dataset, which is a typical large-scale dataset both in the number of observations N and the dimensionality D, SEP achieves tens of gigabytes savings.[9] We performed the test for EP using a machine with more than 100GB RAM, while SEP only required 2.7GB memory, including the space of storing the dataset (roughly 1.9GB). These numbers reveal the huge memory requirement of full EP and further support SEP as a practical alternative in big data, big model settings.

To summarise, in all the experiments presented in this section, we observed that SEP performed almost equally well as full EP, and at the same time provided substantial memory complexity gains. By noticing that the posterior only cares about the product of the likelihood terms, we conjecture that, whilst SEP provides worse approximations to individual likelihood terms, it approximates the product of the product of the likelihood terms almost as well as the full EP case. This again justifies the motivation of SEP that it focuses only on

[9] In this case we used mean-field Gaussian approximations, meaning that a Gaussian factor costs $\mathcal{O}(HD)$ storage with $H = 50$ the number of hidden units.

Table 3.5 Datasets used in the experiments with neural networks. The memory figures reported include dataset storage and temporal maintenance of computation graphs in Theano [Bastien et al., 2012] ($\sim 100MB$ for small datasets and $\sim 1.9GB$ for Year Prediction MSD).

Dataset	N	D	MB (EP)	MB (SEP)	MB reduced
Kin8nm	8192	8	168.23	109.76	58.47
Naval Propulsion	11,934	16	261.75	113.92	147.83
Combined Cycle Power Plant	9568	4	148.70	110.99	37.71
Protein Structure	45,730	9	815.55	121.52	694.02
Wine Quality Red	1599	11	122.21	107.90	14.30
Year Prediction MSD	**515,345**	**90**	**67837.90**	**2730.55**	**65107.34**

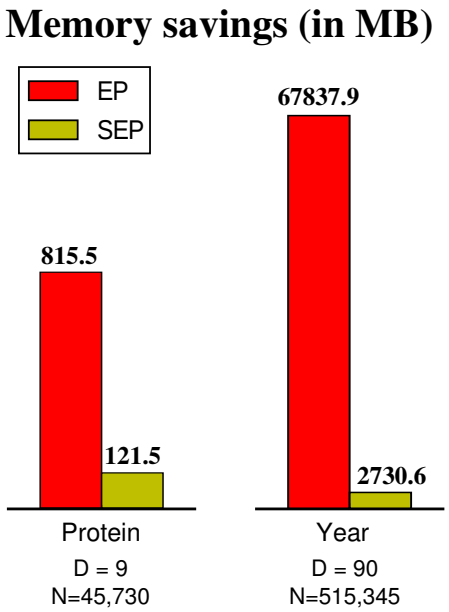


Fig. 3.9 A visualisation of storage savings by SEP.

the approximation to the "averaged likelihood", and full EP, which considering accurate approximations to each individuals, might seem like an over-kill in practice.

3.5 Summary

We have presented the stochastic expectation propagation (SEP) method for reducing EP's large memory consumption that is prohibitive for large datasets. We have connected the new algorithm to a number of existing methods including assumed density filtering, variational message passing, variational inference, stochastic variational inference and averaged EP. Experiments on Bayesian logistic regression (both synthetic and real world) and mixture of Gaussians clustering indicated that the new method had an accuracy that was competitive

with EP. Experiments using the probabilistic back-propagation approach to training Bayesian neural networks on large real world regression datasets again showed that SEP comparably to EP with a vastly reduced memory footprint.

One notable issue that is not addressed here is the theoretical properties of SEP. Does SEP have the same convergence properties as EP in expectation? What is the underlying energy function that SEP is minimising, or is there one at all? in the next chapter we will propose another unifying view of the existing variational methods, but from a very different angle: we will manipulate the energy functions to obtain both upper- and lower-bounds of the marginal likelihood, and discuss connections to VI and EP/SEP with further techniques. Our hope is that, by studying the energy functions, we can get more insights on how EP-like methods work, and better understand the principles of variational methods.

Remark (Further applications of SEP/DSEP)**.** The flexibility of the SEP algorithms allow further adjustments of the approximation procedure for different modelling scenarios. For example, we touched on the distributed version of SEP (DSEP) but did not push very far when this work was presented at NIPS 2015. Since then Zhe et al. [2016] adopted the factor tying idea to develop a highly scalable Bayesian algorithm for online click-through rate prediction on Yahoo! data. In more detail, their idea is DSEP essentially: they maintain the approximating factors for both positive and negative classes, and update them using cheap moment matching techniques. Their experimental results showed that DSEP returns better prediction accuracy compared to widely used algorithms like Vowpal Wabbit[a] and follow the regularised leader (FTRL)-proximal method [McMahan et al., 2013].

Another promising direction is the application of SEP algorithms to continual learning [Kirkpatrick et al., 2017; Ring, 1994, 1997], where information of previous data/task has to be maintained in some way in order to prevent catastrophic forgetting. Indeed the online elastic weight consolidation (EWC) method developed in Schwarz et al. [2018] can be viewed as a Laplace approximation version of SEP, which achieves state-of-the-art performance on multiple reinforcement learning tasks. Another interesting idea would combine the *coreset* algorithm with DSEP, where observed data-points in/out of the coreset are handled with different approximating factors.

[a] http://hunch.net/~vw/

Chapter 4

Rényi Divergence Variational Inference

We have discussed in the last chapter a class of EP-like algorithms, which unifies EP, SEP and VI from an algorithmic point of view. Approximate inference is also widely used as a sub-routine in approximate maximum likelihood algorithms and those used for model selections. Historically, VI has received most attentions in this regard. This is mainly because VI has elegant and useful theoretical properties, such as the fact that it proposes a lower-bound of the log-model evidence. On the other hand, as discussed in the previous chapter, the underlying objective function of SEP is unknown and might not even exist. Even the EP energy itself, although often providing more accurate approximations, has no bounding guarantees [Cunningham et al., 2011]. These undesirable issues make EP-like algorithms less appropriate for model selection and approximate MLE.

To (partially) address these issues, in this chapter we will present a new class of variational inference method using a variant of the α-divergences called Rényi divergence. We will develop both lower- and upper-bounds to the marginal likelihood, and draw connections to SEP and Black-box-α (introduced by us in Hernández-Lobato et al. [2016] but not included in the thesis). This framework is also computationally compelling for Bayesian deep learning, as it is compatible with gradient descent methods (unlike SEP methods which use moment matching). These favourable features are demonstrated with examples including variational auto-encoders and Bayesian neural networks. Throughout the development we will also discuss some theoretical properties of Monte Carlo approximations and data sub-sampling.

4.1 Rényi's α-divergence

We first review Rényi's α-divergence [Rényi, 1961; Van Erven and Harremoës, 2014]. Rényi's α-divergence, defined on $\{\alpha : \alpha > 0, \alpha \neq 1, |\mathrm{D}_\alpha^R| < +\infty\}$, measures the "closeness"

Table 4.1 Special cases in the Rényi divergence family.

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 ($\mathrm{KL}[q\|\|p]$) and EP ($\mathrm{KL}[p\|\|q]$)
$\alpha = 0.5$	$-2\log(1-\mathrm{Hel}^2[p\|\|q])$	function of the square *Hellinger distance*
$\alpha \to 0$	$-\log \int_{p(\boldsymbol{\theta})>0} q(\boldsymbol{\theta}) d\boldsymbol{\theta}$	zero when $\mathrm{supp}(q) \subseteq \mathrm{supp}(p)$ (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]

of two distributions p and q on a random variable $\boldsymbol{\theta} \in \Theta$:

$$\mathrm{D}_\alpha^R[p||q] = \frac{1}{\alpha-1} \log \int p(\boldsymbol{\theta})^\alpha q(\boldsymbol{\theta})^{1-\alpha} d\mu. \tag{4.1}$$

Here the constraint $|D_\alpha^R[p||q]| < +\infty$ is crucial for rewriting the divergence as the expectation under p or q (i.e. to change the measure from $d\mu$ to dP or dQ), i.e.

$$D_\alpha[p||q] = \frac{1}{\alpha-1} \log \mathbb{E}_p \left[\left(\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right)^{\alpha-1} \right] = \frac{1}{\alpha-1} \log \mathbb{E}_q \left[\left(\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right)^{\alpha} \right],$$

since it is possible that the definition (4.1) is infinity but one of the above expectations is finite. In the following we will use $d\mu = d\boldsymbol{\theta}$ w.l.o.g.

The definition is extended to $\alpha = 0, 1, +\infty$ by continuity. We note that when $\alpha \to 1$ the Kullback-Leibler (KL) divergence is recovered, which plays a crucial role in machine learning and information theory. Some other special cases are presented in Table 4.1. The method proposed in this work also considers $\alpha \leq 0$ (although (4.1) is no longer a divergence for these α values), and we include from Van Erven and Harremoës [2014] some useful properties for forthcoming derivations.

Proposition 4.1. *(Monotonicity) Rényi's α-divergence definition (4.1), extended to negative α, is **continuous** and **non-decreasing** on $\alpha \in \{\alpha : -\infty < \mathrm{D}_\alpha^R < +\infty\}$.*

Proposition 4.2. *(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 $\mathrm{D}_{-\infty}^R[p||q] = -\mathrm{D}_{+\infty}^R[q||p]$.*

A critical question that is still in active research is how to choose a divergence in this rich family to obtain optimal solution for a particular application, an issue which is discussed in Section 4.2.5.

4.2 Variational Rényi bound

Recall from previous section that the family of Rényi divergences includes the KL divergence. Perhaps variational free-energy approaches can be generalised to the Rényi case? 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 $\alpha > 0$. Now we consider the equivalent optimisation problem

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

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

$$\begin{aligned}\mathcal{L}_\alpha(q;\mathcal{D}) &:= \log p(\mathcal{D}) - \mathrm{D}_\alpha^R[q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})] \\ &= \log p(\mathcal{D}) - \frac{1}{\alpha-1}\log \mathbb{E}_q\left[\left(\frac{q(\boldsymbol{\theta})p(\mathcal{D})}{p(\boldsymbol{\theta},\mathcal{D})}\right)^{\alpha-1}\right] \\ &= \frac{1}{1-\alpha}\log \mathbb{E}_q\left[\left(\frac{p(\boldsymbol{\theta},\mathcal{D})}{q(\boldsymbol{\theta})}\right)^{1-\alpha}\right].\end{aligned} \tag{4.2}$$

We name this new objective the *variational Rényi (VR) bound*. Importantly the above definition can be extended to $\alpha \leq 0$, and the following theorem is a direct result of Proposition 4.1.

Theorem 4.1. *The objective* $\mathcal{L}_\alpha(q;\mathcal{D})$ *is* ***continuous*** *and* ***non-increasing*** *on* $\alpha \in \{\alpha : |\mathcal{L}_\alpha| < +\infty\}$. *Especially for all* $0 < \alpha_+ < 1$ *and* $\alpha_- < 0$,

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

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

Theorem 4.1 indicates that the VR bound can be useful for model selection by sandwiching the marginal likelihood with bounds computed using positive and negative α values, which we leave to future work. In particular $\mathcal{L}_0 = \log p(\mathcal{D})$ 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, and in the following we assume that this condition is satisfied.

Remark (on marginal likelihood estimation)**.** Though not fully discussed in this thesis, it is worth highlighting here the importance of robust marginal likelihood estimation, and thus the usefulness of sandwiching estimates (with both a lower- and an upper-bound). Dieng et al. [2017] applied the VR upper-bound to a number of real-world tasks. Grosse et al. [2016, 2015] proposed bi-directional Monte Carlo method that can be viewed as computing the VR bounds with a sequence of proposal distributions, aiming at reducing the mismatch between q and p thus improving the Monte Carlo estimates. Wu et al. [2017] applied this idea to perform the first attempt of robust predictive log-likelihood estimation for VAEs and generative adversarial networks (GANs) [Goodfellow et al., 2014].

4.2.1 Mean-field approximation revisited

We revisit in the following the mean-field approximation by optimising the VR bound, with Bayesian linear regression as an illustrating example. Recall the mean-field approximation factorises over the components of $\boldsymbol{\theta} = (\theta_1, ..., \theta_D)$: $q(\boldsymbol{\theta}) = \prod_i q_i(\theta_i)$. Re-writing the VR bound (4.2), we have

$$\begin{aligned}
\mathcal{L}_\alpha(q;\mathcal{D}) &= \frac{1}{1-\alpha}\log\int\prod_i q_i(\theta_i)\left(\frac{p(\boldsymbol{\theta},\mathcal{D})}{\prod_i q_i(\theta_i)}\right)^{1-\alpha} d\boldsymbol{\theta} \\
&= \frac{1}{1-\alpha}\log\int q_j(\theta_j)^\alpha\left(\int\prod_{i\neq j} q_i(\theta_i)\left(\frac{p(\boldsymbol{\theta},\mathcal{D})}{\prod_{i\neq j} q_i(\theta_i)}\right)^{1-\alpha} d\boldsymbol{\theta}_{\neq j}\right) d\theta_j \\
&:= \frac{1}{1-\alpha}\log\int q_j(\theta_j)^\alpha \tilde{p}_j(\theta_j)^{1-\alpha} d\theta_j + \text{const},
\end{aligned}$$

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

$$\log\tilde{p}_j(\theta_j) = \frac{1}{1-\alpha}\log\int\prod_{i\neq j} q_i(\theta_i)\left(\frac{p(\boldsymbol{\theta},\mathcal{D})}{\prod_{i\neq j} q_i(\theta_i)}\right)^{1-\alpha} d\boldsymbol{\theta}_{\neq j} + \text{const}.$$

Now maximising the VR bound (when $\alpha > 0$, and for $\alpha < 0$ we minimise the bound) is equivalent to minimising $\mathrm{D}_\alpha^R[q_j||\tilde{p}_j]$ (for $\alpha > 0$, and when $\alpha < 0$ we minimise $\mathrm{D}_{1-\alpha}^R[\tilde{p}_j||q_j]$), which means $\log q_j(\theta_j) = \log\tilde{p}_j(\theta_j) + \text{const}$ is the only global optimum of the mean-field approximation procedure. One can verify that when $\alpha \to 1$ it recovers the traditional variational mean-field approximation (see Section 2.2.2)

$$\lim_{\alpha\to 1}\log\tilde{p}_j(\theta_j) = \int\prod_{i\neq j} q_i(\theta_i)\log p(\boldsymbol{\theta},\mathcal{D})d\boldsymbol{\theta}_{\neq j} + \text{const},$$

and when $\alpha \to 0$ the fixed point equation returns the exact marginal of the posterior distribution:[1]

$$\lim_{\alpha \to 0} \tilde{p}_j(\theta_j) = p(\theta_j|\mathcal{D}).$$

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

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

Given the observations $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}$, 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

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

and define $q_i(\theta_i) = \mathcal{N}(\theta_i; m_i, \lambda_i^{-1})$ as a univariate Gaussian distribution. Then

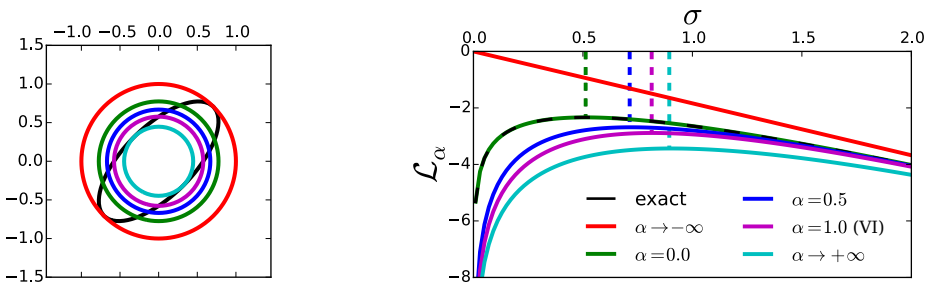
$$\begin{aligned}
\log q_1 &= \frac{1}{1-\alpha} \log \int q_2(\theta_2) \left(\frac{p(\boldsymbol{\theta}, \mathcal{D})}{q_2(\theta_2)} \right)^{1-\alpha} d\theta_2 + \text{const} \\
&= \frac{1}{1-\alpha} \log \int \exp\left[-\frac{1-\alpha}{2} (\boldsymbol{\theta} - \boldsymbol{\mu})^T \boldsymbol{\Lambda} (\boldsymbol{\theta} - \boldsymbol{\mu}) - \frac{\alpha}{2} \lambda_2 (\theta_2 - m_2)^2 \right] d\theta_2 + \text{const} \\
&= \frac{1}{1-\alpha} \log \int \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \tilde{\boldsymbol{\Sigma}}) d\theta_2 + \text{const} \\
&= \log \mathcal{N}(\theta_1; m_1, \lambda^{-1}) + \text{const}
\end{aligned}$$

where the new mean m_1 and the precision λ_1 satisfies

$$\begin{aligned}
m_1 = \mu_1 + C_1(\mu_2 - m_2), \quad C_1 = \frac{\alpha \lambda_2 \Lambda_{12}}{(1-\alpha)|\boldsymbol{\Lambda}| + \alpha \lambda_2 \Lambda_{11}}, \\
\lambda_1 = \Lambda_{11} - (1-\alpha)\Lambda_{12}((1-\alpha)\Lambda_{22} + \alpha\lambda_2)^{-1}\Lambda_{21}.
\end{aligned}$$

One can derive the terms m_2 and C_2 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 $q_1 = \mathcal{N}(\theta_1; \mu_1, \lambda_1^{-1})$, and similarly $q_2 = \mathcal{N}(\theta_1; \mu_2, \lambda_2^{-1})$ with $\lambda_2 = \Lambda_{22} - (1-\alpha)\Lambda_{21}((1-\alpha)\Lambda_{11} + \alpha\lambda_1)^{-1}\Lambda_{12}$. In this example λ_1, λ_2 are feasible for all α, and solving the fixed point equations, finally we have the stable

[1] This is *not* the unique fixed point of $\lim_{\alpha \to 0} \mathcal{L}_\alpha = \mathcal{L}_0$, since $\mathcal{L}_0 = \text{const}$ when q has full support.



(a) Approximated posterior. (b) Hyper-parameter optimisation.

Fig. 4.1 Mean-Field approximation for Bayesian linear regression (with one-sigma contours). C.f. Figure 2.1. In this case $\boldsymbol{\varphi} = \sigma$ the observation noise variance. As expected when $\alpha = 0$ the resulting bound coincides with the exact log marginal (see the green-black curve). The bound is tight as $\sigma \to +\infty$, biasing the VI solution to large σ values.

fixed point as

$$\lambda_1 = \rho_\alpha \Lambda_{11}, \quad \lambda_2 = \rho_\alpha \Lambda_{22}, \quad \rho_\alpha = \frac{1}{2\alpha}\left[(2\alpha - 1) + \sqrt{1 - \frac{4\alpha(1-\alpha)\Lambda_{12}^2}{\Lambda_{11}\Lambda_{22}}}\right].$$

The other solution for the quadratic formula is eliminated since it violates the assumptions that $\lambda_1 > 0$ (when $0 < \alpha < 1$) and $|\mathcal{L}_\alpha| < +\infty$ (when $\alpha < 0$ or $\alpha > 1$, since it requires $|\alpha \mathrm{diag}(\boldsymbol{\lambda}) + (1-\alpha)\boldsymbol{\Lambda}| > 0$). Thus the stable fixed point in this case is unique.

One can show that $\lim_{\alpha \to 1} \lambda_1 = \Lambda_{11}$ (the precision of the conditional distribution $p(\theta_1|\theta_2, \mathcal{D})$), $\lim_{\alpha \to 0} \lambda_1 = \Lambda_{11} - \Lambda_{12}\Lambda_{22}^{-1}\Lambda_{21}$ (the precision of the marginal distribution $p(\theta_1|\mathcal{D})$), and $\lim_{\alpha \to \pm\infty} \lambda_1 = \Lambda_{11} \pm |\Lambda_{12}|\sqrt{\Lambda_{11}\Lambda_{22}^{-1}}$ (similar results for λ_2). Also ρ_α is continuous and non-decreasing in α. This means one can interpolate between mass-covering ($\alpha \to -\infty$) and zero-forcing ($\alpha \to +\infty$, when using uni-modal approximations it is usually called mode-seeking) behaviour by increasing α values.

We visualise the analytical results for Bayesian linear regression in Figure 4.1(a) and 4.1(b). First as predicted, increasing α returns more confident estimate. Also notice that $\alpha \to +\infty$ (in cyan) returns non-zero uncertainty estimates (although it is more over-confident than VI) which is different from the maximum a posteriori (MAP) method that only returns a point estimate. Second, setting $\alpha = 0.0$ (in green) returns $q(\boldsymbol{\theta}) = \prod_i p(\theta_i|\mathcal{D})$ and the exact marginal likelihood $\log p(\mathcal{D})$ (Figure 4.1(b)). Also the approximate MLE is less biased for $\alpha = 0.5$ (in blue) since now the tightness of the bound is less hyper-parameter dependent.

4.2.2 Monte Carlo approximation of the VR bound

Although we have seen some nice properties of the VR bound optimisation through the mean-field approximation example, we also note that when $\alpha \neq 1$, the VR bound is usually just as intractable as the marginal likelihood for many other useful models. Also Theorem 4.1 suggests that the VR bound is to be minimised when $\alpha < 0$, which performs disastrously in MLE context. As we shall see, these issues are addressed by the MC approximation that we will be developing as follows, under certain conditions. Therefore, MC-VR can be applied to precisely the same set of models as MC-VI [Kucukelbir et al., 2015; Paisley et al., 2012; Ranganath et al., 2014; Salimans and Knowles, 2013].

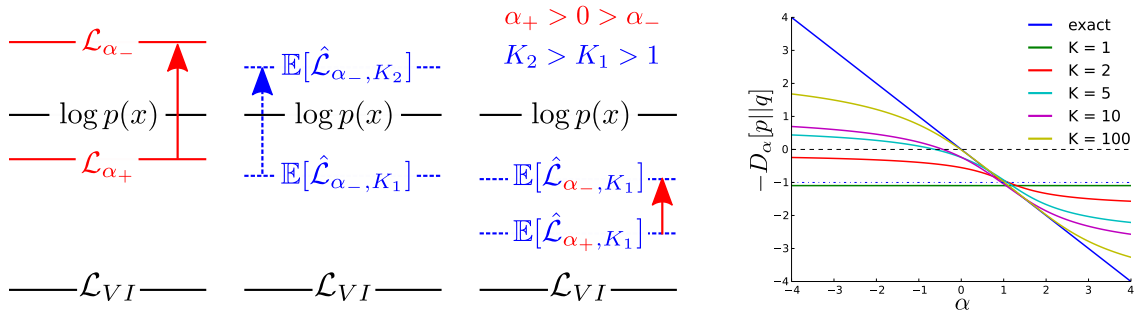
Consider learning a latent variable model with MLE as a running example, where the model is specified by a conditional distribution $p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{\varphi})$ and a prior $p(\boldsymbol{z}|\boldsymbol{\varphi})$ on the latent variable $\boldsymbol{z}$. Examples include latent variable models treated by the variational auto-encoder (VAE) approach [Kingma and Welling, 2014; Rezende et al., 2014] that parametrises the likelihood with a (deep) neural network. MLE requires $\log p(\boldsymbol{x})$ which is obtained by marginalising out $\boldsymbol{z}$ and is often intractable, so the VR bound is considered as an alternative optimisation objective. However instead of using exact bounds, a simple Monte Carlo (MC) method is deployed, which uses a finite number of samples $\boldsymbol{z}_k \sim q(\boldsymbol{z}|\boldsymbol{x}), k = 1,...,K$ to approximate the VR bound $\mathcal{L}_\alpha \approx \hat{\mathcal{L}}_{\alpha,K}$:

$$\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x}) = \frac{1}{1-\alpha}\log\frac{1}{K}\sum_{k=1}^{K}\left[\left(\frac{p(\boldsymbol{z}_k,\boldsymbol{x})}{q(\boldsymbol{z}_k|\boldsymbol{x})}\right)^{1-\alpha}\right]. \tag{4.3}$$

The importance weighted auto-encoder (IWAE) [Burda et al., 2016] is a special case of this framework with $\alpha = 0$ and $K < +\infty$. But unlike traditional VI, here the MC approximation is biased. Fortunately we can characterise the bias by the following theorems (proofs provided in Appendix A.2).

Theorem 4.2. *Assume* $\mathbb{E}_{\{\boldsymbol{z}_k\}_{k=1}^K}[|\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})|] < +\infty$ *and* $|\mathcal{L}_\alpha| < +\infty$. *Then* $\mathbb{E}_{\{\boldsymbol{z}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})]$ *as a function of* $\alpha \in \mathbb{R}$ *and* $K \geq 1$ *is:*
1) ***non-decreasing*** *in K for fixed* $\alpha \leq 1$, *and* ***non-increasing*** *in K for fixed* $\alpha \geq 1$;
2) $\mathbb{E}_{\{\boldsymbol{z}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})] \to \mathcal{L}_\alpha$ *as* $K \to +\infty$;
3) ***continuous*** *and* ***non-increasing*** *in* α *with fixed K.*

Corollary 4.1. *For finite K, either* $\mathbb{E}_{\{\boldsymbol{z}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})] < \log p(\boldsymbol{x})$ *for all* α, *or there exists* $\alpha_K \leq 0$ *such that* $\mathbb{E}_{\{\boldsymbol{z}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha_K,K}(q;\boldsymbol{x})] = \log p(\boldsymbol{x})$ *and* $\mathbb{E}_{\{\boldsymbol{z}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})] > \log p(\boldsymbol{x})$ *for all* $\alpha < \alpha_K$. *Also* α_K *is* ***non-decreasing*** *in K if exists, with* $\lim_{K\to 1}\alpha_K = -\infty$ *and* $\lim_{K\to+\infty}\alpha_K = 0$.



(a) MC approximated VR bounds. (b) Simulated MC approximations.

Fig. 4.2 (a) An illustration for the bounding properties of MC approximations to the VR bounds (non-increasing in α and non-decreasing in K when $\alpha \leq 1$). (b) The bias of the MC approximation, where the dash-dotted line on top of the green line ($K = 1$) is the analytical value of $-\mathrm{KL}[p||q]$. Best viewed in colour and see the main text for details.

The intuition behind the theorems is visualised in Figure 4.2(a). By definition, the exact VR bound is a lower-bound or upper-bound of $\log p(\boldsymbol{x})$ when $\alpha > 0$ or $\alpha < 0$, respectively. However the MC approximation $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$ biases the estimate towards $\mathcal{L}_{\mathrm{VI}}$, where the MC approximation of the bound can be improved using more samples. Thus for finite samples and under mild conditions, negative alpha values can potentially be used to improve the accuracy of the approximation, although the MC approximation for these alpha values is no longer guaranteed to be an upper bound. Figure 4.2(b) shows an empirical evaluation by computing the exact and the MC approximation of the Rényi divergence. In this example p, q are 2-D Gaussian distributions with $\boldsymbol{\mu}_p = [0,0]$, $\boldsymbol{\mu}_q = [1,1]$ and $\boldsymbol{\Sigma}_p = \boldsymbol{\Sigma}_q = \boldsymbol{I}$. The sampling procedure is repeated 200 times to estimate the expectation. Clearly for $K = 1$ it is equivalent to an unbiased estimate of the KL-divergence for all α (even though now the estimation is biased for D_α^R). For $K > 1$ and $\alpha < 1$, the MC method under-estimates the VR bound, and the bias decreases with increasing K. For $\alpha > 1$ the inequality is reversed also as predicted.

4.2.3 A unified implementation with the reparameterisation trick

Readers may have noticed that $\mathcal{L}_{\mathrm{VI}}$ has a different form compared to $\mathcal{L}_\alpha$ with $\alpha \neq 1$. In this section we show how to unify the implementation for all finite α settings using the *reparameterisation trick* [Kingma and Welling, 2014; Salimans and Knowles, 2013] as an example. This trick assumes the existence of the mapping $\boldsymbol{\theta} = g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon})$, where the distribution of the noise term $\boldsymbol{\varepsilon}$ satisfies $q(\boldsymbol{\theta})d\boldsymbol{\theta} = p(\boldsymbol{\varepsilon})d\boldsymbol{\varepsilon}$. Then the expectation of a function $F(\boldsymbol{\theta})$ over distribution $q(\boldsymbol{\theta})$ can be computed as $\mathbb{E}_{q(\boldsymbol{\theta})}[F(\boldsymbol{\theta})] = \mathbb{E}_{p(\boldsymbol{\varepsilon})}[F(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}))]$. One prevalent example is the Gaussian reparameterisation: $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \Rightarrow \boldsymbol{\theta} = \boldsymbol{\mu} + \Sigma^{\frac{1}{2}}\boldsymbol{\varepsilon}$, with $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, I)$.

Now we apply the reparameterisation trick to the VR bound

$$\mathcal{L}_\alpha(q_{\boldsymbol{\phi}};\boldsymbol{x}) = \frac{1}{1-\alpha}\log \mathbb{E}_{\boldsymbol{\varepsilon}}\left[\left(\frac{p(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}),\boldsymbol{x})}{q(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}))}\right)^{1-\alpha}\right]. \tag{4.4}$$

For notational ease we also write $g_{\boldsymbol{\phi}} = g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon})$. Then the gradient of the VR bound w.r.t. $\boldsymbol{\phi}$ (similar for $\boldsymbol{\varphi}$) is

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}}\mathcal{L}_\alpha(q_{\boldsymbol{\phi}};\boldsymbol{x}) &= \frac{1}{1-\alpha}\nabla_{\boldsymbol{\phi}}\log \mathbb{E}_{\boldsymbol{\varepsilon}}\left[\left(\frac{p(g_{\boldsymbol{\phi}},\boldsymbol{x})}{q(g_{\boldsymbol{\phi}})}\right)^{1-\alpha}\right] \\
&= \frac{1}{1-\alpha}\left(\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\left(\frac{p(g_{\boldsymbol{\phi}},\boldsymbol{x})}{q(g_{\boldsymbol{\phi}})}\right)^{1-\alpha}\right]\right)^{-1}\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\nabla_{\boldsymbol{\phi}}\left(\frac{p(g_{\boldsymbol{\phi}},\boldsymbol{x})}{q(g_{\boldsymbol{\phi}})}\right)^{1-\alpha}\right] \\
&= \frac{1}{1-\alpha}\left(\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\left(\frac{p(g_{\boldsymbol{\phi}},\boldsymbol{x})}{q(g_{\boldsymbol{\phi}})}\right)^{1-\alpha}\right]\right)^{-1}\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\left(\frac{p(g_{\boldsymbol{\phi}},\boldsymbol{x})}{q(g_{\boldsymbol{\phi}})}\right)^{1-\alpha}\nabla_{\boldsymbol{\phi}}(1-\alpha)\log\frac{p(g_{\boldsymbol{\phi}},\boldsymbol{x})}{q(g_{\boldsymbol{\phi}})}\right] \\
&= \mathbb{E}_{\boldsymbol{\varepsilon}}\left[w_\alpha(\boldsymbol{\varepsilon};\boldsymbol{\phi},\boldsymbol{x})\nabla_{\boldsymbol{\phi}}\log\frac{p(g_{\boldsymbol{\phi}},\boldsymbol{x})}{q(g_{\boldsymbol{\phi}})}\right].
\end{aligned} \tag{4.5}$$

where $w_\alpha(\boldsymbol{\varepsilon};\boldsymbol{\phi},\boldsymbol{x}) = \left(\frac{p(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}),\boldsymbol{x})}{q(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}))}\right)^{1-\alpha} \Big/ \mathbb{E}_{\boldsymbol{\varepsilon}}\left[\left(\frac{p(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}),\boldsymbol{x})}{q(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}))}\right)^{1-\alpha}\right]$ denotes the normalised importance weight. One can show that this recovers the the stochastic gradients of $\mathcal{L}_{\text{VI}}$ by setting $\alpha = 1$ in (4.5) since now $w_1(\boldsymbol{\varepsilon};\boldsymbol{\phi},\boldsymbol{x}) = 1$, which means the resulting algorithm unifies the computation for all finite α settings. For MC approximations, we use K samples to approximately compute the weight $\hat{w}_{\alpha,k}(\boldsymbol{\varepsilon}_k;\boldsymbol{\phi},\boldsymbol{x}) \propto \left(\frac{p(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}_k),\boldsymbol{x})}{q(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}_k))}\right)^{1-\alpha}$, $k = 1,\ldots,K$, and the stochastic gradient becomes

$$\nabla_{\boldsymbol{\phi}}\hat{\mathcal{L}}_{\alpha,K}(q_{\boldsymbol{\phi}};\boldsymbol{x}) = \sum_{k=1}^{K}\left[\hat{w}_{\alpha,k}(\boldsymbol{\varepsilon}_k;\boldsymbol{\phi},\boldsymbol{x})\nabla_{\boldsymbol{\phi}}\log\frac{p(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}_k),\boldsymbol{x})}{q(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}_k))}\right]. \tag{4.6}$$

When $\alpha = 1$, $\hat{w}_{1,k}(\boldsymbol{\varepsilon}_k;\boldsymbol{\phi},\boldsymbol{x}) = 1/K$, and it recovers the stochastic gradient VI method [Kingma and Welling, 2014].

To speed-up learning Burda et al. [2016] suggested back-propagating only one sample $\boldsymbol{\varepsilon}_j$ with $j \sim p_j = \hat{w}_{\alpha,j}$, which can be easily extended to our framework. Importantly, the use of different $\alpha < 1$ indicates the degree of emphasis placed upon locations where the approximation q under-estimates p, and in the extreme case $\alpha \to -\infty$, the algorithm chooses the sample that has the *maximum* unnormalised importance weight. We name this approach *VR-max* and summarise it and the general case in Algorithm 9. Note that VR-max (and VR-α with $\alpha < 0$ and MC approximations) does *not* minimise $\mathrm{D}^R_{1-\alpha}[p||q]$. It is true that

Algorithm 9 One gradient step for VR-α/VR-max with single backward pass. Here $\hat{w}(\boldsymbol{\varepsilon}_k;\boldsymbol{x})$ shorthands $\hat{w}_{0,k}(\boldsymbol{\varepsilon}_k;\boldsymbol{\phi},\boldsymbol{x})$ in the main text.

1: given the current datapoint $\boldsymbol{x}$, sample $\boldsymbol{\varepsilon}_1,...,\boldsymbol{\varepsilon}_K \sim p(\boldsymbol{\varepsilon})$
2: for $k = 1,...,K$, compute the unnormalised weight $\log \hat{w}(\boldsymbol{\varepsilon}_k;\boldsymbol{x}) = \log p(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}_k),\boldsymbol{x}) - \log q(g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}_k)|\boldsymbol{x})$
3: choose the sample $\boldsymbol{\varepsilon}_j$ to back-propagate:
if $|\alpha| < \infty$: $j \sim p_k$ where $p_k \propto \hat{w}(\boldsymbol{\varepsilon}_k;\boldsymbol{x})^{1-\alpha}$
if $\alpha = -\infty$: $j = \arg\max_k \log \hat{w}(\boldsymbol{\varepsilon}_k;\boldsymbol{x})$
4: return the gradients $\nabla_{\boldsymbol{\phi}} \log \hat{w}(\boldsymbol{\varepsilon}_j;\boldsymbol{x})$

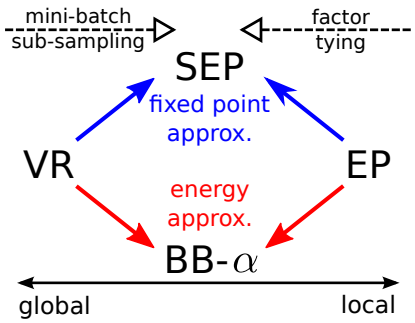


Fig. 4.3 Connecting local and global divergence minimisation.

$\mathcal{L}_\alpha \geq \log p(\boldsymbol{x})$ for negative α values. However Corollary 4.1 suggests that the tightest MC approximation for given K has non-positive α_K value, or might not even exist. Furthermore the optimal α_K value becomes more negative as the mismatch between q and p increases, e.g. the VAE uses a uni-modal q distribution to approximate the typically multi-modal exact posterior.

4.2.4 Stochastic approximation for large-scale learning

VR bounds can also be applied to full Bayesian inference with posterior approximation. However for large datasets full batch learning is very inefficient. Mini-batch training is non-trivial here since the VR bound cannot be represented by the expectation of a datapoint-wise loss, except when $\alpha = 1$ (VI). This section introduces two proposals for mini-batch training, and interestingly, this recovers two existing algorithms that were motivated from a different perspective. In the following we define the (geometric) "average likelihood" $\bar{f}_{\mathcal{D}}(\boldsymbol{\theta}) = [\prod_{n=1}^N p(\boldsymbol{x}_n|\boldsymbol{\theta})]^{\frac{1}{N}}$. Hence the joint distribution can be rewritten as $p(\boldsymbol{\theta},\mathcal{D}) = p_0(\boldsymbol{\theta})\bar{f}_{\mathcal{D}}(\boldsymbol{\theta})^N$. Also for a mini-batch of M datapoints $\mathcal{S} = \{\boldsymbol{x}_{n_1},...,\boldsymbol{x}_{n_M}\} \sim \mathcal{D}$, we define the "subset average likelihood" $\bar{f}_{\mathcal{S}}(\boldsymbol{\theta}) = [\prod_{m=1}^M p(\boldsymbol{x}_{n_m}|\boldsymbol{\theta})]^{\frac{1}{M}}$.

The first proposal considers *fixed point approximations* with mini-batch sub-sampling. It first derives the fixed point conditions for the variational parameters (e.g. the natural parameters of q) using the exact VR bound (4.2), then designs an iterative algorithm using those fixed point equations, but with $\bar{f}_{\mathcal{D}}(\boldsymbol{\theta})$ replaced by $\bar{f}_{\mathcal{S}}(\boldsymbol{\theta})$. The second proposal also applies this subset average likelihood approximation idea, but directly to the VR bound (4.2) (so this approach is named *energy approximation*):

$$\tilde{\mathcal{L}}_\alpha(q;\mathcal{S}) = \frac{1}{1-\alpha}\log \mathbb{E}_q\left[\left(\frac{p_0(\boldsymbol{\theta})\bar{f}_{\mathcal{S}}(\boldsymbol{\theta})^N}{q(\boldsymbol{\theta})}\right)^{1-\alpha}\right]. \tag{4.7}$$

Now we demonstrate with detailed derivations that fixed point approximation returns stochastic EP (SEP, see Chapter 3) , and black box alpha (BB-α) [Hernández-Lobato et al., 2016] corresponds to energy approximation. We derive the results in exponential family context, but in general these two principles of stochastic approximation also apply. Note that both methods use Minka's α-divergence, and for clear distinction here we use β instead to denote the corresponding α values for Minka's definition, and hence the two algorithms are returned respectively by taking $M = 1$ and $\alpha = 1 - \beta/N$.

Precisely, we assume the posterior approximation is defined as $q(\boldsymbol{\theta}) = \frac{1}{Z_q} p_0(\boldsymbol{\theta}) t(\boldsymbol{\theta})^N$. Often $t(\boldsymbol{\theta})$ is chosen to have an exponential family form $t(\boldsymbol{\theta}) \propto \exp[\langle \boldsymbol{\lambda}, \boldsymbol{\Phi}(\boldsymbol{\theta}) \rangle]$ with $\boldsymbol{\Phi}(\boldsymbol{\theta})$ denoting the sufficient statistic. Then picking $\alpha = 1 - \beta/N$, $\beta \neq 0$, we obtain the exact VR bound (by pulling out the normalising constant Z_q) as

$$\mathcal{L}_\alpha(q; \mathcal{D}) = \log Z_q + \frac{N}{\beta} \log \mathbb{E}_q \left[\left(\frac{\bar{f}_\mathcal{D}(\boldsymbol{\theta})}{t(\boldsymbol{\theta})} \right)^\beta \right]. \tag{4.8}$$

The first proposal considers deriving the exact fixed point conditions, then approximating them with mini-batch sub-sampling. In our example the exact fixed point condition for the variational parameters $\boldsymbol{\lambda}$ is

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}_\alpha(q; \mathcal{D}) = 0 \quad \Rightarrow \quad \mathbb{E}_q[\boldsymbol{\Phi}(\boldsymbol{\theta})] = \mathbb{E}_{\tilde{p}_\alpha}[\boldsymbol{\Phi}(\boldsymbol{\theta})], \tag{4.9}$$

with the tilted distribution defined as

$$\tilde{p}_\alpha(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})^\alpha p_0(\boldsymbol{\theta})^{1-\alpha} \bar{f}_\mathcal{D}(\boldsymbol{\theta})^{N(1-\alpha)} \propto p_0(\boldsymbol{\theta}) t(\boldsymbol{\theta})^{N-\beta} \bar{f}_\mathcal{D}(\boldsymbol{\theta})^\beta.$$

Now given a mini-batch of datapoints $\mathcal{S}$, the moment matching update can be approximated by replacing $\bar{f}_\mathcal{D}(\boldsymbol{\theta})$ with $\bar{f}_\mathcal{S}(\boldsymbol{\theta}) = [\prod_{m=1}^M p(\boldsymbol{x}_{n_m}|\boldsymbol{\theta})]^{\frac{1}{M}}$. More precisely, each iteration we sample a subset of data $\mathcal{S} = \{\boldsymbol{x}_{n_1}, ..., \boldsymbol{x}_{n_M}\} \sim \mathcal{D}$, and compute the new update for $\boldsymbol{\lambda}$ by first computing $\tilde{p}_{\alpha,\mathcal{S}}(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta}) t(\boldsymbol{\theta})^{N-\beta} \bar{f}_\mathcal{S}(\boldsymbol{\theta})^\beta$ then taking $\mathbb{E}_q[\boldsymbol{\Phi}(\boldsymbol{\theta})] \leftarrow \mathbb{E}_{\tilde{p}_{\alpha,\mathcal{S}}}[\boldsymbol{\Phi}(\boldsymbol{\theta})]$. This method returns SEP when $M = 1$, i.e. in each iteration only one datapoint is sampled to update the approximate posterior. Using larger mini-batch size $M > 1$ returns SDEP (see Section 3.2.3) and in this case further approximation might be required to compute the fixed point iterative updates.

The second proposal also applies this subset average likelihood approximation idea, but directly to the VR bound (4.8), with $\mathbb{E}_\mathcal{S}$ denoting the expectation over mini-batch sub-

sampling:

$$\mathbb{E}_{\mathcal{S}}\left[\tilde{\mathcal{L}}_\alpha(q;\mathcal{S})\right] = \log Z_q + \frac{N}{\beta}\mathbb{E}_{\mathcal{S}}\left[\log \mathbb{E}_q\left[\left(\frac{\bar{f}_{\mathcal{S}}(\boldsymbol{\theta})}{t(\boldsymbol{\theta})}\right)^\beta\right]\right]. \tag{4.10}$$

It recovers the energy function of BB-α when $M = 1$. Note that the original BB-α algorithm uses an adapted form of Amari's α-divergence, and the α value in the BB-α algorithm corresponds to β in our exposition. Now the gradient of this approximated energy function becomes

$$\nabla_{\boldsymbol{\lambda}}\mathbb{E}_{\mathcal{S}}\left[\tilde{\mathcal{L}}_\alpha(q;\mathcal{S})\right] = N(\mathbb{E}_q[\boldsymbol{\Phi}(\boldsymbol{\theta})] - \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\tilde{p}_{\alpha,\mathcal{S}}}[\boldsymbol{\Phi}(\boldsymbol{\theta})]). \tag{4.11}$$

We also provide a characterisation of the energy approximation (4.10) by the following theorem, with a proof presented in Appendix A.2.

Theorem 4.3. *If the approximate distribution $q(\boldsymbol{\theta})$ is Gaussian $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$, and the likelihood functions has an exponential family form $p(\boldsymbol{x}|\boldsymbol{\theta}) = \exp[\langle\boldsymbol{\theta},\boldsymbol{\Phi}(\boldsymbol{x})\rangle - A(\boldsymbol{\theta})]$, then for $\alpha \leq 1$ and $r > 1$ the stochastic approximation is bounded by*

$$\mathbb{E}_{\mathcal{S}}[\tilde{\mathcal{L}}_\alpha(q;\mathcal{S})] \leq \mathcal{L}_{1-(1-\alpha)r}(q;\mathcal{D}) + \frac{N^2(1-\alpha)r}{2(r-1)}\mathrm{tr}(\boldsymbol{\Sigma}\mathrm{Cov}_{\mathcal{S}\sim\mathcal{D}}(\bar{\boldsymbol{\Phi}}_{\mathcal{S}})).$$

The following corollary is a direct result of Theorem 4.3 applied to BB-α. Note here we follow the convention of BB-α to use $M = 1$ and overload the notation $\alpha = \beta$ and $\mathcal{L}_{BB-\alpha}(q;\mathcal{D}) = \mathbb{E}_{\{\boldsymbol{x}_n\}}\left[\tilde{\mathcal{L}}_{1-\alpha/N}(q;\{\boldsymbol{x}_n\})\right]$.

Corollary 4.2. *Assume the approximate posterior and the likelihood functions satisfy the assumptions in Theorem 4.3, then for $\alpha > 0$ and $r > 1$, the black-box alpha energy function is upper-bounded by*

$$\mathcal{L}_{BB-\alpha}(q;\mathcal{D}) \leq \mathcal{L}_{1-\frac{\alpha r}{N}}(q;\mathcal{D}) + \frac{N\alpha r}{2(r-1)}\mathrm{tr}(\boldsymbol{\Sigma}\mathrm{Cov}_{\mathcal{D}}(\boldsymbol{\Phi})).$$

It is interesting that both SEP and BB-α were originally proposed to approximate (power) EP [Minka, 2001b, 2004], which usually minimises α-divergences *locally*, and considers $M = 1$, $\alpha \in [1 - 1/N, 1)$ and exponential family distributions. These approximations were performed by factor tying, which significantly reduces the memory overhead of full EP and makes both SEP and BB-α scalable to large datasets just as is the case for SVI. The new derivation provides a theoretical justification from an energy perspective, and also sheds lights on the connections between *local* and *global* divergence minimisations as depicted in Figure 4.3. Note that all these methods recover SVI when $\alpha \to 1$, in which global and local divergence minimisation are equivalent. Also these results suggest again (but from a different perspective) that, recent attempts of distributed posterior approximation (by carving

up the dataset into pieces with $M > 1$ [Gelman et al., 2014; Xu et al., 2014]) can be extended to both SEP and BB-α.

SEP is arguably better justified since it returns the exact posterior if the approximation family $\mathcal{Q}$ is large enough to include the correct solution, just like VI and VR computed on the whole dataset. BB-α might still be biased even in this scenario. However, BB-α is much simpler to implement since the energy function can be optimised with stochastic gradient descent. Indeed BB-α considers the same black-box approach as used for VI, by computing a stochastic estimate of the energy function then using automatic differentiation tools to obtain the gradients.

4.2.5 Optimisation issues with α-divergences and MC approximations

It is in general an outstanding research question on how to select the divergence measure for a particular machine learning problem. In our case this corresponds to selecting the α value. Also an approximate inference algorithm can be evaluated with different performance measures, and it is generally impossible to find a single α value that returns the best performance on all evaluations. Thus we only present the evaluation in test error and test log-likelihood in the experiments, and use them to select the α values empirically.

We discuss two conjectures to explain the difficulty of selecting α in the Bayesian neural network experiments presented in later sections. The first conjecture is that zero-forcing algorithms ($\alpha \geq 1$) tend to favour minimising the test error, while mass-covering methods ($\alpha < 1$) tend to improve the test log-likelihood. However zero-forcing methods can fail as they might miss an important mode due to local optima. Similarly mass-covering methods can be pathological if the exact posterior includes modes that are very far away from each other. Furthermore, the form of the posterior will change with the number of observed datapoints N, so the "optimal" setting of α for a fixed task may change with N.

The second conjecture states that the MC approximation complicates the selection of α, since it favours zero-forcing (because of the bias introduced). For example, in order to maximise the quantity of the MC approximation the algorithm need to make $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$ finite first. However, as shown by Lemma A.1 in Appendix A.2, the MC approximation goes wrong if the support of q is strictly larger than the support of p. Hence to avoid this pathology the optimisation procedure will ensure $q = 0$ whenever p is zero. Also in order to avoid missing an important mode we already assumed that q is supported wherever p is supported. Combining with Theorem 4.2, we conjecture that the MC approximation makes the algorithm more "VI-like" compared to the exact case. In other words, when the MC approximation is deployed, the effective α value is closer to $\alpha = 1$ that is the value for VI (which is precisely the case if considering $K = 1$). This means, if there exists $\alpha_{\text{opt}} \neq 1$ for

a specific task, in practice one should use $\alpha \leq \alpha_{\text{opt}}$ (for $\alpha_{\text{opt}} < 1$, and should use $\alpha \geq \alpha_{\text{opt}}$ if $\alpha_{\text{opt}} > 1$) when running the MC algorithm. In general one should be very careful when estimating the ratio between distribution with Monte Carlo methods. Also the introduced MC approach usually has higher variance compared to the variational case (and the variance can be as high as importance sampling [Bamler et al., 2017]), so further methods like control variate techniques should be applied to reduce the sampling variance.

Still we want to emphasise again that for many problems, minimising an α-divergence other than the KL-divergence can be very useful, even when using MC approximations. Approximate EP has been applied to deep Gaussian process regression and has shown to achieve the state-of-the-art results for benchmark datasets [Bui et al., 2016b]. A recent paper [Depeweg et al., 2017] tested BB-α for model-based reinforcement learning with Bayesian neural networks. In their tests using $\alpha = 0.5$ successfully captured the bi-modality and heteroskedasticity in the predictive distribution, while VI failed disastrously.

4.3 Experiments

We evaluate the VR bound methods on Bayesian neural networks and variational auto-encoders. The implementation of all the experiments in Python is released at https://github.com/YingzhenLi/VRbound.

4.3.1 Bayesian neural networks

The first experiment considers Bayesian neural network regression. The datasets are collected from the UCI dataset repository.[2] We use a Gaussian prior $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \boldsymbol{I})$ for the network weights and Gaussian approximation to the true posterior $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_q, diag(\boldsymbol{\sigma}_q))$. We fit the parameters of q and the noise level σ by optimising the lower-bound. We follow the toy example in Section 4.2 and consider $\alpha \in \{-\infty, 0.0, 0.5, 1.0, +\infty\}$ in order to examine the effect of mass-covering/zero-forcing behaviour. Stochastic optimisation uses the energy approximation strategy proposed in Section 4.2.4.

For regression tests, we consider Protein and Year as the large datasets and the remainder as small datasets. For all small datasets we used single-layer neural networks with 50 hidden units (ReLUs), and for Protein and Year we used 100 units. The methods for comparison were run for 500 epochs on the small datasets and 100, 40 epochs for the large datasets, respectively. We used ADAM [Kingma and Ba, 2015] for optimisation with learning rate 0.001 and the standard setting for other parameters. For stochastic optimisation we used

[2] http://archive.ics.uci.edu/ml/datasets.html

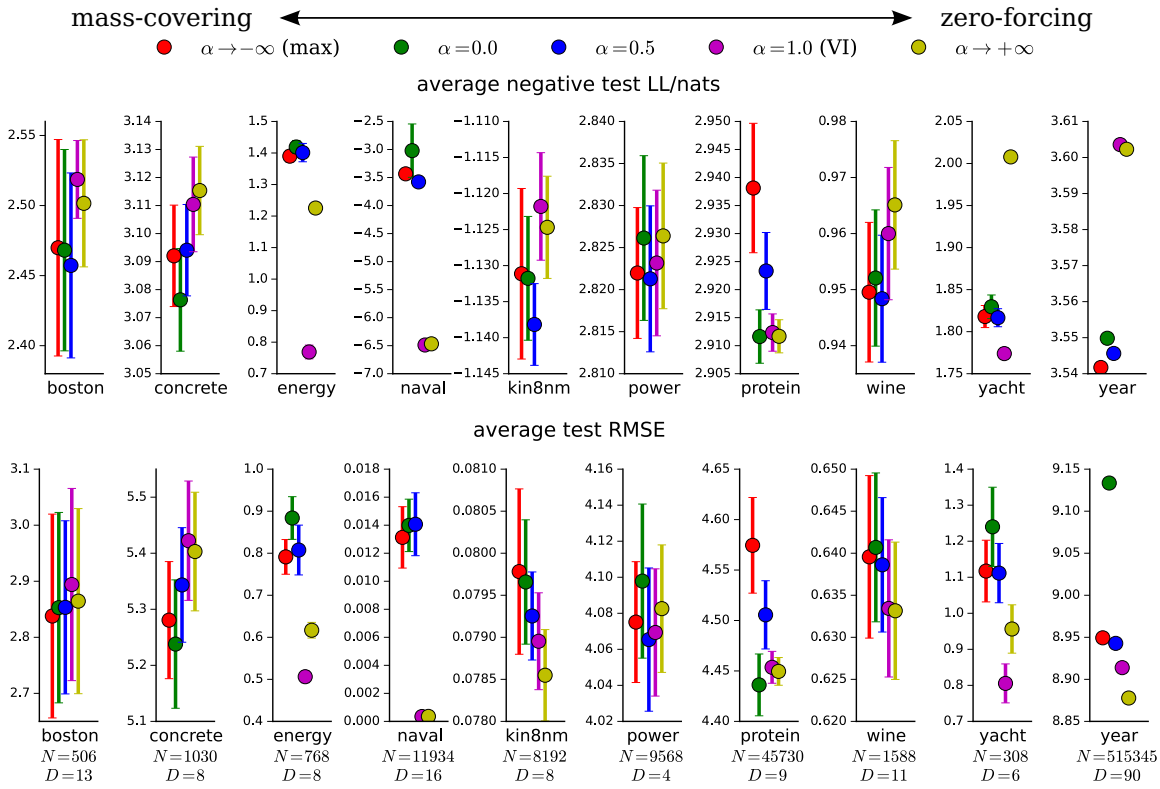


Fig. 4.4 Test LL and RMSE results for Bayesian neural network regression. The lower the better. The error bars show 1-standard deviation across 20 random splits of the data.

mini-batch size $M = 32$ and number of MC samples $K = 100$ and $K = 10$ for small and large datasets, respectively. The number of dataset random splits is 20 except for the large datasets, which is 5 and 1 for Protein and Year, respectively.

We summarise the test negative log-likelihood (LL) and RMSE with standard error (across different random splits except for Year) for selected datasets in Figure 4.4 and Table 4.2, 4.3. In the tables the best performing results are underlined, while the worse cases are also bold-faced. These results indicate that for posterior approximation problems, the optimal α may vary for different datasets (although for Boston and Power the performances are very similar).

It can be particularly tricky to approximate the posterior distribution of the weight matrices for a neural network. Since one can obtain the same output by swapping the positions of two hidden units and adjusting the corresponding in- and out-going weights, there exist symmetric modes in the exact posterior of the weight matrices. In this case mass-covering can be harmful, e.g. for Naval and Energy datasets, methods with $\alpha < 1.0$ values seem to be under-performing, not only for predictive error but also for test log-likelihood measure. But still, we observed two major trends. Zero-forcing/mode-seeking methods

Table 4.2 Regression experiment: Average negative test log likelihood/nats

Dataset	N	D	$\alpha \to -\infty$	$\alpha = 0.0$	$\alpha = 0.5$	$\alpha = 1.0$ **(VI)**	$\alpha \to +\infty$
boston	506	13	2.47±0.08	2.47±0.07	*2.46*±*0.07*	**2.52±0.03**	2.50±0.05
concrete	1030	8	3.09±0.02	*3.08*±*0.02*	3.09±0.02	3.11±0.02	**3.12±0.02**
energy	768	8	1.39±0.02	**1.42±0.02**	1.40±0.03	*0.77*±*0.02*	1.23±0.01
naval	11934	16	-3.43±0.08	**-3.02±0.48**	-3.58±0.08	*-6.49*±*0.04*	-6.47±0.09
kin8nm	8192	8	-1.13±0.01	-1.13±0.01	*-1.14*±*0.01*	**-1.12±0.01**	-1.12±0.01
power	9568	4	2.82±0.01	2.83±0.01	*2.82*±*0.01*	2.82±0.01	**2.83±0.01**
protein	45730	9	**2.94±0.01**	*2.91*±*0.00*	2.92±0.01	2.91±0.00	2.91±0.00
wine	1588	11	0.95±0.01	0.95±0.01	*0.95*±*0.01*	0.96±0.01	**0.97±0.01**
yacht	308	6	1.82±0.01	1.83±0.01	1.82±0.01	*1.77*±*0.01*	**2.01±0.00**
year	515345	90	*3.54*±NA	3.55±NA	3.55±NA	**3.60**±NA	3.60±NA
Average Rank			2.80±0.34	3.00±0.45	**2.20±0.37**	3.20±0.51	3.80±0.39

Table 4.3 Regression experiment: Average test RMSE

Dataset	N	D	$\alpha \to -\infty$	$\alpha = 0.0$	$\alpha = 0.5$	$\alpha = 1.0$ **(VI)**	$\alpha \to +\infty$
boston	506	13	*2.84*±*0.18*	2.85±0.17	2.85±0.15	**2.89±0.17**	2.86±0.17
concrete	1030	8	5.28±0.10	*5.24*±*0.11*	5.34±0.10	**5.42±0.11**	5.40±0.11
energy	768	8	0.79±0.04	**0.88±0.05**	0.81±0.06	*0.51*±*0.01*	0.62±0.02
naval	11934	16	0.01±0.00	0.01±0.00	**0.01±0.00**	*0.00*±*0.00*	0.00±0.00
kin8nm	8192	8	**0.08±0.00**	0.08±0.00	0.08±0.00	0.08±0.00	*0.08*±*0.00*
power	9568	4	4.08±0.03	**4.10±0.04**	*4.07*±*0.04*	4.07±0.04	4.08±0.04
protein	45730	9	**4.57±0.05**	*4.44*±*0.03*	4.51±0.03	4.45±0.02	4.45±0.01
wine	1588	11	0.64±0.01	**0.64±0.01**	0.64±0.01	0.63±0.01	*0.63*±*0.01*
yacht	308	6	1.12±0.09	**1.24±0.11**	1.11±0.08	*0.81*±*0.05*	0.96±0.07
year	515345	90	8.95±NA	**9.13**±NA	8.94±NA	8.91±NA	*8.88*±NA
Average Rank			3.40±0.38	3.70±0.51	3.20±0.31	2.40±0.45	**2.30±0.38**

tend to focus on improving the predictive error. Mass-covering methods, on the other hand, returns better test log-likelihood, which has been shown empirically to be correlated with the quality of the approximated uncertainty estimate [Depeweg et al., 2017; Gal, 2016; Hernández-Lobato and Adams, 2015]. In particular VI returns lower test log-likelihood for most of the datasets. Furthermore, $\alpha = 0.5$ produced overall good results for both test LL and RMSE, possibly because the skew symmetry is centred at $\alpha = 0.5$ and the corresponding divergence is the only symmetric distance measure in the family. Future work should develop algorithms to automatically select the best α values, although a naive approach could use validation sets.

4.3.2 Variational auto-encoders

The second experiments considers variational auto-encoders for unsupervised learning. We mainly compare three approaches: VAE ($\alpha = 1.0$), IWAE ($\alpha = 0$), and VR-max ($\alpha = -\infty$), which are implemented upon the publicly available code.[3] Four datasets are considered: Frey Face[4] (with 10-fold cross validation), Caltech 101 Silhouettes[5], MNIST[6] and OMNIGLOT[7]. The VAE model has $L = 1, 2$ stochastic layers with deterministic layers stacked between. The detailed numbers of stochastic layers L, number of hidden units, and the activation function are summarised in Table 4.4. The prefix of the number indicates whether this layer is **d**eterministic or **s**tochastic, e.g. d500-s200 stands for a neural network with one deterministic layer of 500 units followed by a stochastic layer of 200 units. For Frey Face data we train the models using learning rate 0.0005 and mini-batch size 100. For MNIST and OMNIGLOT we reuse the settings from Burda et al. [2016]: the training process runs for 3^i passes with learning rate $0.0001 \cdot 10^{-i/7}$ for $i = 0, \ldots, 7$, and the batch size is 20. For Caltech Silhouettes we use the same settings as MNIST and OMNIGLOT except that the training proceeded for $\sum_{i=0}^{7} 2^i = 255$ epochs. We reproduce the IWAE experiments to obtain a fair comparison, since the results in the original publication [Burda et al., 2016] mismatches those evaluated on the publicly available code.

We report test log-likelihood results in Table 4.5 by computing $\log p(\boldsymbol{x}) \approx \hat{\mathcal{L}}_{0,5000}(q; \boldsymbol{x})$ following Burda et al. [2016]. We also present some samples from the VR-max trained auto-encoders in Figure 4.7. Overall VR-max is almost indistinguishable from IWAE. Other positive alpha settings (e.g. $\alpha = 0.5$) return worse results, e.g. 1374.64 ± 5.62 for Frey Face and -85.50 for MNIST with $\alpha = 0.5$, $L = 1$ and $K = 5$. These worse results for $\alpha > 0$

[3] https://github.com/yburda/iwae
[4] http://www.cs.nyu.edu/~roweis/data.html
[5] https://people.cs.umass.edu/~marlin/data.shtml
[6] http://yann.lecun.com/exdb/mnist/
[7] https://github.com/brendenlake/omniglot

Table 4.4 Network architecture of tested VAE algorithms.

Dataset	L	architecture	activation	probability type (p/q)
Frey Face	1	d200-d200-s20	softplus	Gaussian/Gaussian
Caltech 101	1	d500-s200	tanh	Bernoulli/Gaussian
MNIST &	1	d200-d200-s50	tanh	Bernoulli/Gaussian
OMNIGLOT	2	d200-d200-s100-d100-d100-s50	tanh	Bernoulli/Gaussian

Table 4.5 Average Test log-likelihood. Results for VAE on MNIST and OMNIGLOT are collected from Burda et al. [2016].

Dataset	L	K	**VAE**	**IWAE**	**VR-max**
Frey Face	1	5	1322.96	**1380.30**	1377.40
($\pm$ std. err.)			$\pm$10.03	$\pm$**4.60**	$\pm$4.59
Caltech 101	1	5	-119.69	**-117.89**	-118.01
Silhouettes		50	-119.61	-117.21	**-117.10**
MNIST	1	5	-86.47	**-85.41**	-85.42
		50	-86.35	**-84.80**	-84.81
	2	5	-85.01	**-83.92**	-84.04
		50	-84.78	**-83.05**	-83.44
OMNIGLOT	1	5	-107.62	**-106.30**	-106.33
	1	50	-107.80	**-104.68**	-105.05
	2	5	-106.31	**-104.64**	-104.71
	2	50	-106.30	**-103.25**	-103.72

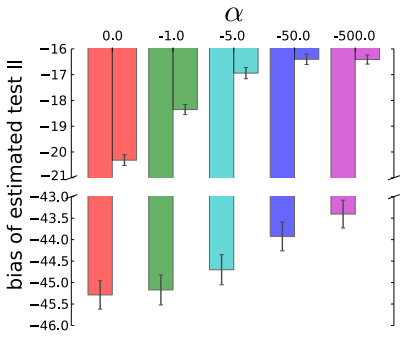
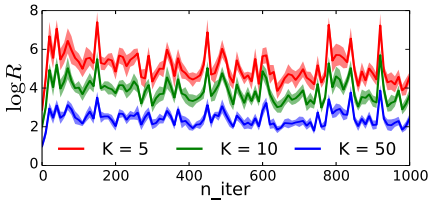
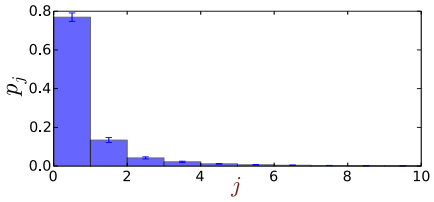


Fig. 4.5 Bias of sampling approximation to. Results for $K = 5, 50$ samples are shown on the left and right, respectively.



(a) Log of ratio $R = w_{max}/(1 - w_{max})$



(b) Weights of samples.

Fig. 4.6 Importance weights during training, see main text for details. Best viewed in colour.

indicate the preference of getting tighter approximations to the likelihood function for MLE problems. Small negative α values (e.g. $\alpha = -1.0, -2.0$) return better results on different splits of the Frey Face data, and overall the best α value is dataset-specific.[8]

VR-max's success might be explained by the tightness of the bound. To evaluate this, we compute the VR bounds on 100 test datapoints using the 1-layer VAE trained on Frey

[8] Since presentation of this work at NIPS 2016, Bui et al. [2016a] revisited this idea with slightly different architecture set-up, and showed that $\alpha \neq 0$ values are usually favoured over the IWAE approach.

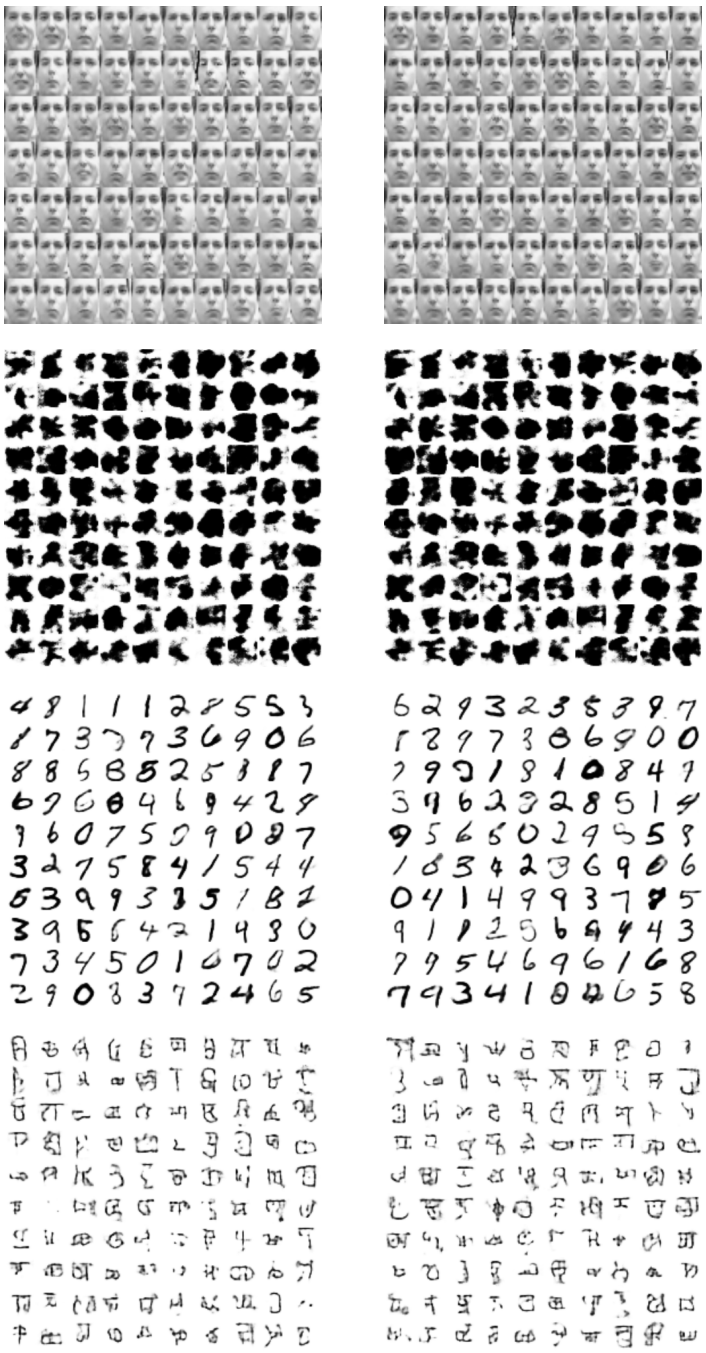


Fig. 4.7 Sampled images from the the best models trained with IWAE (left) and VR-max (right).

Face, with $K = \{5, 50\}$ and $\alpha \in \{0, -1, -5, -50, -500\}$. Figure 4.5 presents the estimated gap $\hat{\mathcal{L}}_{\alpha,K} - \hat{\mathcal{L}}_{0,5000}$. The results indicates that $\hat{\mathcal{L}}_{\alpha,K}$ provides a lower-bound, agreeing with the theoretical results presented in Section 4.2.2, and that gap is narrowed as $\alpha \to -\infty$. Also increasing K provides improvements. The standard error of estimation is almost constant for different α (with K fixed), and is negligible when compared to the MC approximation bias.

Another explanation for VR-max's success is that, the sample with the largest normalised importance weight w_{max} dominates the contributions of all the gradients. This is confirmed by tracking $R = \frac{w_{max}}{1-w_{max}}$ during training on Frey Face (Figure 4.6(a)). Also Figure 4.6(b) shows the 10 largest importance weights from $K = 50$ samples in descending order, which exhibit an exponential decay behaviour, with the largest weight occupying more than 75% of the probability mass. This means, the IWAE algorithm is not efficient in terms of sample efficiency, since the learning signal from back-propagation is dominated by the gradients computed on the particle with the largest weight. It also indicates that, VR-max can provide a fast approximation to IWAE when tested on CPUs or multiple GPUs with high communication costs. Indeed our numpy implementation of VR-max achieves up to a 3 times speed-up compared to IWAE (9.7s vs. 29.0s per epoch, tested on Frey Face data with $K = 50$ and batch size $M = 100$, CPU info: Intel Core i7-4930K CPU @ 3.40GHz). However this speed advantage is less significant when the gradients can be computed very efficiently on a single GPU.

4.4 Summary

We have introduced the variational Rényi bound and an associated optimisation framework. We have shown the richness of the new family, not only by connecting to existing approaches including VI/VB, SEP, BB-α, VAE and IWAE, but also by proposing the VR-max algorithm as a new special case. Empirical results on Bayesian neural networks and variational auto-encoders indicate that VR bound methods are widely applicable and can obtain state-of-the-art results. Future work will focus on both experimental and theoretical aspects. Theoretical work will study the interaction of the biases introduced by MC approximation and datapoint sub-sampling. Also a quantitative analysis of the MC approximation bias will be very useful, especially for model selection.

This ends the first part of the thesis, which has reviewed existing variational methods (Chapter 2), and proposed two unifying frameworks for both algorithmic (Chapter 3) and optimisation objective (Chapter 4) aspects. Experiments on Bayesian deep learning tasks also proved the successfulness of our efforts, on pushing variational algorithms towards wider applicability (far beyond conjugate models) and better scalability to "big data, big models". I do admit that, however, the lack of theoretically rigorous selection of approximate inference algorithms is one of the imperfections of the analysis presented here. Indeed, a guide on choosing the optimal inference method are needed for practitioners when applying the framework to their applications, which will be one of my research directions in the future.

So far we only studied different optimisation procedures assuming a given approximating distribution family (mean-field Gaussians in the empirical results), which is just one side of the story for approximate inference. The whole picture for this huge subject will never be comprehensive without discussing the other side, i.e. the construction of q distributions. In the second part of the thesis "wild approximate inference", I will revisit the fundamental questions in approximate inference, providing my point of view on the principles of approximate distribution design, and present one of the algorithms I developed during my PhD following these principles.

Part II

Wild Approximate Inference

Chapter 5

Wild Approximate Inference: Why and How

The design of approximating distributions is equally, if not more, crucial to the invention of optimisation algorithms such as the two algorithms presented in previous chapters. Mean-field approximations are often ineffective! Apparently if the $\mathcal{Q}$ family is expressive enough to contain the exact posterior, variational inference, either with KL divergence or Rényi divergence, would return it as the only global optimum. Hence if a more complex structure is included in the q distribution, the approximation might become much more accurate and potentially even exact. For this purpose, it would be useful to use universal functional approximators, e.g. neural networks, to expand the distribution family of approximations. However, it becomes very challenging to use these flexible approximations due to many restrictions, which cannot be addressed by simply investing more computational resources (say more running time and memory).

In recent years, one major research direction is to go beyond mean-field methods via development of flexible q distributions, which specifically addresses the intractability issues when applied to VI. For example, invertible transformations are utilised to construct q distributions which allow analytical evaluations of the approximate posterior density [Kingma et al., 2016; Louizos and Welling, 2017; Rezende and Mohamed, 2015]. Mixture distributions of more flexible forms are also introduced to capture the possible multi-modality of the exact posterior, with further approximations being proposed to account for extra computational challenges [Maaløe et al., 2016; Ranganath et al., 2016b; Salimans et al., 2015; Tran et al., 2016]. Though effective, these methods are complicated for practitioners to understand, let alone the design work itself that requires lots of care.

In the second theme of the thesis, an alternative research direction will be presented towards using flexible approximations in approximate inference. Concretely, instead of

designing approximate posterior distributions that fit into standard frameworks like VI, we would like to develop optimisation algorithms that enable approximations of *arbitrary* form. This goal is further justified by revisiting the fundamental problem that approximate inference is trying to solve – approximating an integral both accurately and quickly. Throughout the discussions, readers will realise that many computational constraints are irrelevant to inference itself: it is the chosen optimisation algorithm that introduces additional constraints on the q distribution design. Consequently, successful development of algorithms that present no more constraints would enable the usage of *arbitrary* approximate distributions, allowing practitioners to choose the q distribution that fits best to their particular tasks. In the following, we will discuss several research directions towards this goal, with one specific example detailed in the next chapter.

The rest of the chapter is organised as follows. In Section 5.1 we revisit the tractability issues in Bayesian inference and argue that *fast sampling* should be the only condition required by Monte Carlo based approximate inference algorithms. Based on this observation, we suggest using *implicit* distributions as approximate posterior distributions, with some examples provided in Section 5.2 to demonstrate the flexibility of this distribution class. As implicit distributions do not exhibit tractable densities, we propose a few research directions towards fitting these approximate posteriors in Section 5.3. Lastly we briefly discuss a research theme – meta-learning for approximate inference algorithms – in Section 5.4, which is enabled by wild approximate inference methods.

Remark (Related work)**.** The material in this chapter was originally presented in a NIPS 2016 approximate inference workshop abstract "Wild Variational Approximations" which is a joint work with Qiang Liu [Li and Liu, 2016] (see publication page). Before that the research scheme had not been established in a formal way, except for very few attempts [Ranganath et al., 2016a; Wang and Liu, 2016]. Later on, several groups explored an approach that blends VI and (adversarial) density ratio estimation [Huszár, 2017; Karaletsos, 2016; Mescheder et al., 2017; Shi et al., 2018; Tran et al., 2017], which corresponds to one of the algorithmic options described in this chapter. We argue that our work is more formal and general: we justify this line of research by examining the central topics in approximate inference, and point out four research directions for developing algorithms that can fit arbitrary q distributions that enable fast sampling.

5.1 Revisiting tractability issues in approximate inference

In Chapter 1 the definition of an approximate inference procedure is identified. However, here I invite the readers to consider the following question again:

What does tractability mean for an approximate inference algorithm?

This question touches the fundamental principles of approximate inference and it is very important answer it carefully. It helps us to understand the challenges that we face, and identify those that can be addressed by investing more *computational* resources, and those that require advanced *mathematical* tools. Think about the history of neural networks research as an analogy. The first challenge in the late 1960s came from *theoretical* limitations of perceptrons [Minsky and Papert, 1969], and the second issue in the last ten years of the 20th century was mainly the lack of *computational resources* (e.g. slow computers and small datasets) [LeCun et al., 2015]. It was only after identifying and addressing these two issues (back-propagation [Rumelhart et al., 1986], faster machine, GPU computing and big data) that the deep learning community started to revolutionise AI applications in the real world. But still, there are mathematical and computational challenges for modern deep learning which are waiting for discoveries and solutions, but these issues are out of the scope of our discussions.

To answer the above question, we first start by revisiting the definition of *approximate inference*, 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 $F(\boldsymbol{z})$ of interest under the exact posterior, which is $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})]$. Examples of such F functions include $F(\boldsymbol{z}) = \boldsymbol{z}$, $F(\boldsymbol{z}) = \boldsymbol{z}\boldsymbol{z}^{\mathrm{T}}$ (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 (i.e. computing a predictive distribution), and $F(\boldsymbol{z}) = \delta_A$ if one wishes to evaluate the distribution directly $p(\boldsymbol{z} \in A|\boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[\delta_A]$. 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.

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

$$\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})] \approx \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})]. \tag{5.1}$$

Critically, the primary tractability requirement here for the approximate posterior is the *fast computation* of the *approximate expectation* on the RHS 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 which 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 Section 2.2.3. 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 deep latent variable models. [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

$$\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})] \approx \frac{1}{K}\sum_{k=1}^{K} F(\boldsymbol{z}^k), \quad \boldsymbol{z}^k \sim q(\boldsymbol{z}|\boldsymbol{x}). \tag{5.2}$$

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* for a *given* function F. The latter option typically prefers traditional numerical integration methods since for different functions one would select different quadrature rules. On the other hand, fast sampling can also be a weaker condition: once we have obtained the samples from the approximate posterior, we can use them to compute an empirical estimate of the expectation for *any* integrable function. Hence methods that entail fast sampling might be preferred for tasks that require estimating expectations of a set of functions.

Unfortunately, except a few very recent attempts that will be detailed later, most approximate inference algorithms impose further constraints to the design of q. For example, MC-VI requires *fast density evaluation* and/or *fast density gradient evaluation* for $q(\boldsymbol{z}|\boldsymbol{x})$ given a configuration of $\boldsymbol{z}$. Importantly, this requirement is only presented in the VI optimisation procedure to seek for the best fit of q: once a (local) optimum is obtained, inference only requires evaluating the empirical expectation, thus there is no need to compute the density point-wise. Therefore the MC inference at test time enables the usage of *implicit distributions* [Diggle and Gratton, 1984; Mohamed and Lakshminarayanan, 2016] and *stochastic regularisation techniques (SRTs)* [e.g. dropout techniques, see Gal, 2016; Kingma et al., 2015; Krueger et al., 2017; Singh et al., 2016; Srivastava et al., 2014; Wan et al., 2013] that allow for the computation of the MC predictive inference but not for fast density evaluation.

But again at training time they are not applicable to traditional variational methods due to the fast density evaluation constraint.

These observations raise an outstanding research question:

Can we design efficient approximate inference algorithms to train arbitrary posterior approximations (including implicit distributions and SRTs)?

We will answer this in the next sections, by discussing proposals for training *wild approximate inference algorithms*.[1] We will also provide some examples of implicit posterior approximations that are not possible to be fitted using conventional approximate inference algorithms. Before that, I first address potential skepticism below, and argue why this is an outstanding but under-examined research direction.

Remark (a comparison between implicit distributions and SRTs)**.** As presented, it is desirable to remove the fast density evaluation constraint from the fitting procedure. However the reasons for doing so are different for implicit distributions and SRTs. First for implicit distributions, MC samples $\boldsymbol{z}^1,...,\boldsymbol{z}^K$ are generated (e.g. by warping noise variables with a neural network), in order to compute the empirical estimate $\frac{1}{K}\sum_{k=1}^K F(\boldsymbol{z}^k)$. The fast density evaluation constraint becomes an obstacle for fitting implicit distributions, as the underlying distribution of the samples does not exhibit a tractable form.
On the other hand, in Bayesian neural networks the function F is usually defined using the outputs of the neural network, i.e. $F(\boldsymbol{z}) = \frac{1}{N}\sum_{n=1}^N \tilde{F}(\mathbf{y}_n = \mathrm{NN}_{\boldsymbol{z}}(\boldsymbol{x}_n))$. Unlike implicit distributions, SRTs often apply random transformations to the hidden units, which leads to random outputs $\mathbf{y}_n^k, n = 1,...,N, k = 1,...,K$ and the empirical estimate $\frac{1}{KN}\sum_{k=1}^K\sum_{n=1}^N \tilde{F}(\mathbf{y}_n^k)$. More importantly, the random transformations applied to *different* inputs $\boldsymbol{x}_n$ are *different*, which is equivalent to sampling NK sets of weights $\boldsymbol{z}^{k,n}, n = 1,...,N, k = 1,...,K$ *implicitly*. Therefore even the underlying density of $\boldsymbol{z}^{k,n}$ might be tractable point-wise (consider Gaussian dropout [Kingma et al., 2015; Srivastava et al., 2014] as an example), evaluating them explicitly would cost $\mathcal{O}(NK)$ time or $\mathcal{O}(NK)$ memory for parallel computing. Therefore the fast density evaluation constraint becomes an obstacle again but for a different reason: intractability due to limited computational budget.

5.1.1 Is it necessary to evaluate the approximate posterior distribution?

One might argue that having an accessible q distribution allows the user to understand the properties of the exact posterior better. It might be true for low dimensional cases, as we

[1] Not to be confused with *black-box* variational inference [Ranganath et al., 2014].

can easily visualise the density function, and compare density values between samples to determine which is more probable. But I would disagree with this argument for the scenario of approximating multi-modal posterior distributions in high dimensions, which is often the application regime of interest. Some reasons are:

- First, enforcing the tractable density constraint means that in many cases, either we fit the posterior with a rather simple distribution (which has limited representational power), or a complex model such as a mixture density (which entails high computational costs).
- Second, even when setting aside the computational issues for density evaluation, visualising high dimensional distributions is itself still an open research problem. In this regard, many data visualisation techniques consider dimension reduction methods such as principal component analysis (PCA) [Pearson, 1901], self-organising map [Kohonen, 1998; Venna and Kaski, 2003] and t-SNE [van der Maaten and Hinton, 2008], which in fact only require samples from the distribution, not the density values.
- Finally as motivated above, many MC-based inference tasks do not require evaluating or comparing density values on samples. For those which do require density evaluation, one can then fit a density estimator on the samples from q. It can still be very convenient as in the MC estimation setting we typically assume fast sampling from the approximate posterior.

5.1.2 Comparisons to sampling-based 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 contributed 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 can one simulate the MCMC dynamics for an infinite amount of time. Furthermore, even some of these methods do construct *implicit* approximate posterior distributions in practice, they still add more constraints to the inference procedures, detailed in below.

- (Adaptive) importance sampling (IS) and sequential Monte Carlo (SMC).
  Importance sampling has a long history in statistics, e.g. see Geweke [1989]. Roughly speaking, it proposes samples from a rather simple distribution $\pi(\boldsymbol{z}|\boldsymbol{x})$, then "corrects" the sampling estimate by incorporating the importance weight

$$\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[F(\boldsymbol{z})] \approx \frac{1}{K}\sum_{k=1}^{K} w_k F(\boldsymbol{z}^k), \quad w_k = \frac{p(\boldsymbol{z}^k|\boldsymbol{x})}{\pi(\boldsymbol{z}^k|\boldsymbol{x})}, \quad \boldsymbol{z}^k \sim \pi(\boldsymbol{z}|\boldsymbol{x}). \tag{5.3}$$

  One can easily see the unbiasedness of the IS estimate, and under mild conditions one can also show it is consistent. Also self-normalised IS is sometimes used to obtain approximate posterior samples, which effectively constructs q as

$$q(\boldsymbol{z}|\boldsymbol{x}) = \sum_{k=1}^{K} \hat{w}_k \delta(\boldsymbol{z} = \boldsymbol{z}^k), \quad \hat{w}_k = \frac{w_k}{\sum_{k=1}^{K} w_j}, \quad \boldsymbol{z}^k \sim \pi(\boldsymbol{z}|\boldsymbol{x}). \tag{5.4}$$

  In this case the q distribution depends on the proposal π and the number of samples K, and again under mild conditions $q \to p$ when $K \to +\infty$. In this case the estimation is no longer unbiased, but practically the self-normalised IS estimate often enjoys the advantage of lower variance. Importantly, the q distribution is tractable and requires fast evaluation of the π density (up to a potentially unknown normalising constant). SMC can be viewed as importance sampling applied to time-series models (such as hidden Markov models), typically with extra techniques to improve sample efficiency.

  However, IS and SMC provide terrible approximations to the desired integral if the proposal π is very different from the target distribution p, mainly due to the high variance of the estimator. To address this issue, researchers have considered adapting the initial distribution to reduce the variance, therefore improving sample efficiency that is key to the success of IS in practice, Indeed the (unnormalised) optimal proposal distribution for IS is proportional to $|F(\boldsymbol{z})|p(\boldsymbol{z}|\boldsymbol{x})$, and in some cases the resulting estimator has zero variance, indicating that it requires only one (!) sample to compute the exact integral. Recently there is a plenty of research work on how to adapt the initial distribution and combine the approach with amortised inference [Burda et al., 2016; Cornebise, 2009; Gu et al., 2015; Le et al., 2017; Maddison et al., 2017a; Naesseth et al., 2017; Paige and Wood, 2016]. But still, the tractability constraint of $\pi(\boldsymbol{z}|\boldsymbol{x})$ largely restricts its analytic form to those have been used for VI.

- Markov Chain Monte Carlo (MCMC).
  An MCMC algorithm for posterior sampling is typically specified by a *transition distribution* (or transition kernel) $\mathcal{T}(\boldsymbol{z}'|\boldsymbol{z})$, with the following conditions often assumed:

(i) $\mathcal{T}$ has the target distribution $p(\boldsymbol{z}|\boldsymbol{x})$ as the *unique* stationary distribution:

$$p(\boldsymbol{z}'|\boldsymbol{x}) = \int \mathcal{T}(\boldsymbol{z}'|\boldsymbol{z}) p(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z}.$$

(ii) If defining

$$\mathcal{T}_T(\boldsymbol{z}_T|\boldsymbol{z}_0) = \int \prod_{t=0}^{T-1} \mathcal{T}(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t) d\boldsymbol{z}_{0:T-1},$$

then for any initial distribution $q_0(\boldsymbol{z}|\boldsymbol{x})$ the MCMC dynamics converges to the target distribution as $T \to +\infty$:

$$\lim_{T\to+\infty} q_T(\boldsymbol{z}|\boldsymbol{x}) = p(\boldsymbol{z}|\boldsymbol{x}), \quad q_T(\boldsymbol{z}|\boldsymbol{x}) := \int \mathcal{T}_T(\boldsymbol{z}|\boldsymbol{z}') q_0(\boldsymbol{z}'|\boldsymbol{x}) d\boldsymbol{z}'.$$

Conditions that such transition kernel $\mathcal{T}$ requires are described in e.g. chapter 11 of Gelman et al. [1995] and Brooks et al. [2011].

In practice one often specifies an initial distribution $q_0(\boldsymbol{z}|\boldsymbol{x})$ to draw starting particles, and stops simulating the transitions after T steps according to his/her computational budget. Consequently, this truncated Markov chain also induces an implicit q distribution

$$q(\boldsymbol{z}|\boldsymbol{x}) = \int \mathcal{T}_T(\boldsymbol{z}|\boldsymbol{z}') q_0(\boldsymbol{z}'|\boldsymbol{x}) d\boldsymbol{z}'. \tag{5.5}$$

In many applications the computational budget only allows simulations of a small number of transitions, e.g. when training big models with EM. Thus having a rapidly mixed chain would significantly reduce T, and to achieve this goal a lot of work has explored different designs of the transition kernel, to name a few see Ahn et al. [2012]; Ding et al. [2014]; Duane et al. [1987]; Girolami and Calderhead [2011]; Neal [2011]. However, these methods are often designed as a generic sampling algorithm, which might not be best suited for e.g. sampling from Bayesian neural network posterior distributions.

Observing the above, I would argue that recent advances of sampling-based inference method do not achieve the best *speed-accuracy trade-off* that is one of the most important topics in approximate inference. Indeed there are two potential directions to improve an MCMC procedure. First, as we will not be able to obtain the exact posterior from T-step MCMC simulations anyway, removing the asymptotic exactness requirement can potentially allow the best fit of the transition kernel, which makes $q(\boldsymbol{z}|\boldsymbol{x}) = q_T(\boldsymbol{z}|\boldsymbol{x})$ the best approximator to the exact posterior in such a $\mathcal{Q}$ class. Second, even when the transition kernel is constrained to leave the exact posterior invariant, learning the transition kernel would enable

fast convergence and bias reduction for the MCMC algorithm. These examples are closely related to meta-learning [Bengio et al., 1992; Naik and Mammone, 1992; Schmidhuber, 1987; Thrun and Pratt, 1998] for approximate inference, which is further discussed in Section 5.4. Similarly for IS, there exist estimators that use some "super-efficient" weights to enable significantly faster convergence rates than $\mathcal{O}(K^{-\frac{1}{2}})$ the usual convergence rate for the IS estimator (5.3) [Del Moral et al., 2006; Ghahramani and Rasmussen, 2003; Liu and Lee, 2017; Oates et al., 2017; O'Hagan, 1991]. More specifically, the recipe provided by Liu and Lee [2017] does not require a tractable initial distribution π at all. Although these estimators can be biased, in practice they often provide better speed-accuracy trade-off due to their better sample efficiency.

5.2 Examples of implicit distributions

In this section we provide further examples of implicit distributions that can be fitted to the exact posterior using the algorithmic proposals introduced later. As a reminder, we use $\boldsymbol{\phi}$ to denote the parameters for q, and will explicitly write $q(\boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ when necessary.

Ultimately for any uni-variate distribution, the sampling process can be described by deterministically transforming a uniform noise variable with the inverse *cumulative density function* (CDF) or quantile function:

$$z \sim p(z) \quad \Leftrightarrow \quad u \sim \text{Uniform}(0,1), \quad z = \text{CDF}_p^{-1}(u). \tag{5.6}$$

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. Nevertheless, the observation above inspires us to define the approximate distribution q by transforming a random noise variable $\boldsymbol{\varepsilon}$ through a deterministic mapping $\mathbf{f}$:

$$\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x}) \quad \Leftrightarrow \quad \boldsymbol{\varepsilon} \sim \pi(\boldsymbol{\varepsilon}), \quad \boldsymbol{z} = \mathbf{f}(\boldsymbol{\varepsilon}, \boldsymbol{x}). \tag{5.7}$$

This definition exhibits similar flavour as the reparameterisation trick [Kingma and Welling, 2014], and in the following we will also say q is *reparameterisable* if the sampling process of q follows the above procedure.

An important note here is that $\mathbf{f}$ might not be invertible, which differs from the invertible transform techniques discussed in [Kingma et al., 2016; Rezende and Mohamed, 2015]. Thus the q density cannot be evaluated by just calculating the Jacobian matrix as in the other case. It also implies that one cannot directly apply VI to find the best mapping $\mathbf{f}$ simply because

the entropy term $\mathbb{H}[q]$ is again intractable.Thus additional mathematical tools are required to handle this type of approximations, which will be detailed in later sections.

5.2.1 Neural network transform with noise inputs

The simplest way to define the transformation $\mathbf{f}$ is to construct a deterministic deep neural network $\mathrm{NN}_{\boldsymbol{\phi}}$ which takes both the observation $\boldsymbol{x}$ and the noise variable $\boldsymbol{\varepsilon}$ as input:

$$\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x}) \quad \Leftrightarrow \quad \boldsymbol{\varepsilon} \sim \pi(\boldsymbol{\varepsilon}), \quad \boldsymbol{z} = \mathrm{NN}_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x}). \tag{5.8}$$

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

$$\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x}) \quad \Leftrightarrow \quad \boldsymbol{\varepsilon} \sim \pi(\boldsymbol{\varepsilon}), \boldsymbol{W} \sim \pi(\boldsymbol{W}), \quad \boldsymbol{z} = \mathrm{NN}_{\boldsymbol{W}}(\boldsymbol{\varepsilon}, \boldsymbol{x}). \tag{5.9}$$

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 $\mathrm{CDF}_p^{-1} \circ \mathrm{CDF}_\pi$. 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].

5.2.2 Stochastic deep neural networks and recurrent neural networks

There has been a number of recent work that investigated latent variable models as q distributions, e.g. see Maaløe et al. [2016]; Ranganath et al. [2016b]; Salimans et al. [2015]; Tran et al. [2016]. In short, the q distribution is constructed by a series of conditional probabilities

$$q(\boldsymbol{z}|\boldsymbol{x}) = \int q(\boldsymbol{z}, \boldsymbol{z}_0, \boldsymbol{z}_1, ..., \boldsymbol{z}_{T-1}|\boldsymbol{x}) d\boldsymbol{z}_{0:T-1} = \int q(\boldsymbol{z}|\boldsymbol{z}_{T-1}, \boldsymbol{x}) \prod_{t=1}^{T-1} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}) d\boldsymbol{z}_{0:T-1}. \tag{5.10}$$

In the remainder of this section we will also write $\boldsymbol{z}_T = \boldsymbol{z}$. The most general form allows $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x})$ to have a different form at each time step t. Previous uses of this type of approximate distribution focused on the simple case where $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x})$ has tractable density (so that for small T the q distribution is explicit). We review some of the algorithms for fitting them in Appendix B.

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 $\boldsymbol{\varepsilon}_t$ as the input:

$$\boldsymbol{z}_t \sim q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1},\boldsymbol{x}) \quad \Leftrightarrow \quad \boldsymbol{z}_t = \mathbf{f}_t(\boldsymbol{z}_{t-1},\boldsymbol{\varepsilon}_t,\boldsymbol{x}). \quad \boldsymbol{\varepsilon}_t \sim \pi(\boldsymbol{\varepsilon}_t). \tag{5.11}$$

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 so that traditional approximate inference methods do no apply.

An interesting special case of the stochastic neural network approach considers tying the parameters of $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1},\boldsymbol{x})$ at every time step, which essentially makes the network a stochastic recurrent neural network (RNN). In fact this special case can be viewed as a truncated Markov chain, making the stochastic RNN approach closely related to MCMC. Although we will never be able to achieve the exactness guarantee for an MCMC algorithm, we can still use the intuition there to design an approximate q distribution that works well for our data. For example, Ma et al. [2015] claimed that a stochastic gradient MCMC (SG-MCMC) algorithm within the Itô diffusion framework can be framed into the following form (with discretisation)

$$\boldsymbol{z}_t = \boldsymbol{z}_{t+1} + \zeta_t[(\boldsymbol{D}(\boldsymbol{z}_t) + \boldsymbol{Q}(\boldsymbol{z}_t))\nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z} = \boldsymbol{z}_t|\boldsymbol{x}) + \Gamma(\boldsymbol{z}_t)] + \sqrt{2\zeta_t \boldsymbol{D}(\boldsymbol{z}_t)}\boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \tag{5.12}$$

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). But recall that asymptotic exactness is not required if one only cares about the approximation quality of $q_T(\boldsymbol{z}|\boldsymbol{x})$. This inspires us to derive the following stochastic RNN using NN-defined drift and diffusion: with ∇_t short-hands for $\nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z} = \boldsymbol{z}_t|\boldsymbol{x})$,

$$\boldsymbol{z}_t = \boldsymbol{z}_{t+1} + \zeta_t \mathbf{f}_1(\boldsymbol{z}_t,\boldsymbol{x},\nabla_t) + \zeta_t \mathbf{f}_2(\boldsymbol{z}_t,\boldsymbol{x},\nabla_t)\boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \tag{5.13}$$

with each of the $\mathbf{f}_i$ functions parameterised by a neural network, or even an RNN with memory modules [Graves et al., 2014; Hochreiter and Schmidhuber, 1997]. A promising direction would consider learning this stochastic RNN based approximate posterior distribution so that it generalises well to unseen target distributions. This meta-learning task can be addressed in a similar fashion as in Andrychowicz et al. [2016]; Li and Malik [2016] – see discussions in later sections and an initial experiment in Chapter 6.

Remark (stochastic gradient descent (SGD) as an approximate inference method)**.** SG-MCMC has a close relationship to SGD in that it can be viewed as adding a properly scaled noise term to a (pre-conditioned) SGD procedure. Therefore one can also draw inspiration from SGD for approximate posterior design. Indeed recently Duvenaud et al. [2016] has shown that the end points obtained by early stopping SGD (i.e. using finite T) results in a nonparametric variational posterior approximation. Later Mandt et al. [2016, 2017] also showed that under some constraints, the stationary distribution of SGD also forms an approximation to the posterior. The authors further proposed tuning the parameters (learning rate, pre-conditioning matrix, etc.) using variational inference. Similar results have also been discussed in Smith and Le [2018].

5.2.3 Learning to pass messages

An important research direction of graphical models is the fast computation of the marginal distributions of the random variables associated with the nodes in a graph. In formula, given a graphical model with graph $G = (V, E)$, marginal inference means the computation of $p(x_i) = \int p(x_i, \boldsymbol{x}_{\setminus i}) d\boldsymbol{x}_{\setminus i}$ for all $i \in V$. Message passing methods, such as belief propagation and EP as reviewed in the first theme, are popular approximate inference methods for such a purpose. Historically these graphs are constructed to have simple functions attached to each factor, making the computation of local messages tractable and fast. But the message computation is no longer analytic if non-linear mappings are adopted to describe the conditional distributions or potentials. Though not a primary focus of this chapter, below we discuss two recent approaches handling this challenging case with non-conventional methods, of which both do not require tractable densities of the messages or beliefs.

- Adversarial training for message approximation.
  In directed graphical model setting, we often specify the model distribution as $p(\boldsymbol{x}) = \prod_i p(x_i|\mathrm{pa}(x_i))$, in which $\mathrm{pa}(x_i)$ represent the variables attached to the parent nodes of x_i in the graph. Given observed variables $\boldsymbol{x}_o$, we utilise the graphical structure to design $q(\boldsymbol{x}) = q(\boldsymbol{x}_o) \prod_{i \notin o} q(x_i|\tilde{\mathrm{mb}}(x_i))$ where $\tilde{\mathrm{mb}}(x_i)$ denotes the variables in the Markov blanket of x_i needed for d-separation given $\boldsymbol{x}_o$, and the goal is to make q as close as possible to p. Assuming an implicit q density here, a naive idea defines a *global* Jensen-Shannon divergence [Lin, 1991] then using the generative adversarial network (GAN) [Goodfellow et al., 2014] idea to (approximately) minimise it. However computing this global divergence needs an "interpolated" distribution $m(\boldsymbol{x}) = \frac{1}{2}p(\boldsymbol{x}) + \frac{1}{2}q(\boldsymbol{x})$, which effectively destroys the graphical structure and thus requires the discriminator to take the entire set of variables $\boldsymbol{x}$ as the input. This can be very computationally

demanding for very big graphs. Instead, Karaletsos [2016] sketched an algorithm using *local* Jenson-Shanon divergences between $q(x_i, \tilde{\text{mb}}(x_i))$ and $p(x_i, \text{pa}(x_i))$,[2] where a GAN approximation for this divergence requires only inputs for a subset of variables $\{x_i, \text{pa}(x_i), \tilde{\text{mb}}(x_i)\}$ thus can be nested into a message passing procedure. This idea can also be extended to EP/SEP (see Chapters 2 and 3), in which we replace the M-projection step by minimising $\text{JS}[\tilde{p}_n(\boldsymbol{\theta})||q(\boldsymbol{\theta})]$ that is further approximated by a GAN procedure.

- Discriminative learning of the messages.
  Learning the parameters of an undirected graphical model often requires (loopy) belief propagation (LBP) repeatedly to infer the unobserved variables, which can be computationally challenging. However, notice that in many supervised learning tasks the latent variables are mostly served as a feature representation for later usage, in which for a test datum LBP is executed to obtain this embedding anyway. Therefore the prediction pipeline only requires the "approximate model" returned by message passing, and never touches the true distribution of the graphical model. Observing this, Zheng et al. [2015] and Dai et al. [2016b] proposed directly training the "approximate model" in an end-to-end fashion using supervision data. In particular, Song et al. [2011] took the intuition from kernel mean embeddings [Smola et al., 2007] that a distribution can be summarised with a point in a reproducing kernel Hilbert space (RKHS), and proposed a belief propagation algorithm to estimate the RKHS embeddings of marginal distributions. The graph neural network [Dai et al., 2016b; Gori et al., 2005; Li et al., 2016b; Scarselli et al., 2009] further extended this idea and parameterised the outgoing belief features as neural networks taking the incoming belief features as inputs.
  We note that this idea is *different* from directly using q as the underlying model. Unless the graph exhibits a tree structure (where q converges to p), the local beliefs obtained from LBP might not be *consistent*, i.e. these pseudo marginals *do not* always come from the same joint distribution. Rather, the modelling pipeline still uses an intractable but valid graphical model, where the discussed algorithms focus on improving the predictive inference performance directly.

[2] In general $q(x_i, \tilde{\text{mb}}(x_i))$ can be unnormalised (usually happens in message passing), and one can define divergences between unnormalised densities accordingly, see Minka [2005].

5.3 Algorithmic options for fitting arbitrary posterior approximations

The implicit distributions discussed in Section 5.2 cannot be fitted using many existing approximate inference methods such as MC-VI. In this section, we discuss four algorithmic options for training these approximations to the posterior.[3] One of the schemes is further developed in the next chapter. Other approaches that my colleagues and I proposed (following these ideas) include Li and Gal [2017]; Li et al. [2017] which are not presented in the thesis due to page limit.

5.3.1 Energy approximation

Assume q is reparameterisable,[4] i.e. $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \Leftrightarrow \boldsymbol{z} = \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{x})$. Then by the chain rule, the gradient of a given objective function $\mathcal{L}(\boldsymbol{\phi})$ w.r.t. $\boldsymbol{\phi}$ is $\nabla_{\boldsymbol{\phi}}\mathcal{L} = \nabla_{\boldsymbol{\phi}}\mathbf{f}\nabla_{\mathbf{f}}\mathcal{L}$. Therefore, if we have an approximation $\hat{\mathcal{L}}$ to the objective function, then we can approximate the gradient as $\nabla_{\boldsymbol{\phi}}\mathcal{L} \approx \nabla_{\boldsymbol{\phi}}\mathbf{f}\nabla_{\mathbf{f}}\hat{\mathcal{L}}$. We name this approach as *energy/objective approximation*.

Since the loss function $\mathcal{L}(\boldsymbol{\phi})$ is often defined as $\mathcal{L}(q_{\boldsymbol{\phi}})$, a naive method of energy approximation considers density estimation of $q_{\boldsymbol{\phi}}$ and a direct plug-in of the approximate density to the energy function. Typically a density estimator $\hat{q}$, e.g. kernel density estimators (KDE) [Fukunaga and Hostetler, 1975] or neural density estimators [Larochelle and Murray, 2011; Mackay and Gibbs, 1999], is fitted to the samples $\{\boldsymbol{z}^k = \mathbf{f}(\boldsymbol{\varepsilon}^k, \boldsymbol{x})\} \sim q$, and the gradient of $\log q$ is approximated as $\nabla_{\boldsymbol{\phi}} \log q(\boldsymbol{z}|\boldsymbol{x}) \approx \nabla_{\boldsymbol{z}} \log \hat{q}(\boldsymbol{z}|\boldsymbol{x})\nabla_{\boldsymbol{\phi}}\mathbf{f}$. One might even want to directly estimate $\log q$ if it turns out to be more accurate. However, practitioners should be careful with implementations using automatic differentiation tools, since the parameters of the density estimator $\hat{q}$ should not be differentiated through (even though they might depend on the samples $\boldsymbol{z}^k$).

The next idea considers analytical approximations to (part of) the energy function, e.g. the entropy term $\mathbb{H}[q]$ or the KL-divergence $\mathrm{KL}[q||p_0]$ in $\mathcal{L}_{\mathrm{VI}}$, and let the MC approach handle the rest. In MC-dropout [Gal and Ghahramani, 2016] $\mathrm{KL}[q||p_0]$ is approximated by an ℓ_2 regulariser of the neural network weights. In Li and Gal [2017] we extended this framework to α-divergence methods with further approximations. Due to the page limit, we skip the detailed discussions of this work.

[3] Again we note here that the discussed options are applicable to tractable q distributions as well. See discussions in the remark in Section 5.1.

[4] For non-reparameterisable distributions $\nabla_{\boldsymbol{\phi}}\mathbf{f}$ can be computed with further approximations such as the generalised reparameterisation trick [Ruiz et al., 2016].

A new direction for energy approximation applies density ratio estimation methods [Qin, 1998; Sugiyama et al., 2009, 2012]. This is done by introducing an auxiliary distribution $\tilde{q}$ and rewriting the variational lower-bound:

$$\mathcal{L}_{\text{VI}}(\boldsymbol{\theta}, q; \boldsymbol{x}) = \mathbb{E}_q \left[\log \frac{p_0(\boldsymbol{z}) p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})}{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})} + \log \frac{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z}|\boldsymbol{x})} \right]. \tag{5.14}$$

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

$$\nabla_{\boldsymbol{\phi}} \mathcal{L}_{\text{VI}} = \mathbb{E}_q \left[\nabla_{\boldsymbol{\phi}} \log \frac{p_0(\boldsymbol{z}) p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})}{\tilde{q}(\boldsymbol{z}|\boldsymbol{x})} + \nabla_{\boldsymbol{z}} \tilde{R}(\boldsymbol{z}) \nabla_{\boldsymbol{\phi}} \mathbf{f} \right]. \tag{5.15}$$

A simple example considers $\tilde{q} = p_0$ and the classification approach for ratio estimation. In short, we train a classifier

$$D(\boldsymbol{z} \text{ sampled from } p_0 \mid \boldsymbol{z}, \boldsymbol{x}) = (1 + \exp[-\tilde{R}(\boldsymbol{z})])^{-1}$$

to distinguish samples from p_0 and q. A related approach is the adversarial auto-encoder [Makhzani et al., 2015] which uses the prior distribution as an auxiliary. However, the objective function proposed in Makhzani et al. [2015] replaces the $\text{KL}[q||p_0]$ in the variational lower-bound with Jensen-Shannon divergence [Lin, 1991], which lacks a justification from a Bayesian point of view.[5] Also the density ratio estimation idea can be extended to a sequence of auxiliary distributions (in a similar spirit to annealed importance sampling [Neal, 2001]), which can also be adapted slowly during training to obtain a better approximation.

Remark (concurrent work on the density ratio estimation idea)**.** Since the presentation of the original material at the NIPS 2016 approximate inference workshop, this density ratio estimation proposal has also been independently considered in Huszár [2017]; Karaletsos [2016]; Mescheder et al. [2017]; Tran et al. [2017]. Specifically in the adversarial variational Bayes (AVB) paper [Mescheder et al., 2017], the authors first considered prior distribution as the auxiliary $\tilde{q}$, then discussed an advanced technique termed as *adaptive contrast*, which takes $\tilde{q}$ as a Gaussian approximation to 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 of randomness between different $q(\boldsymbol{z}_n|\boldsymbol{x}_n)$ distributions and thus reducing

[5] An optimal transport [Villani, 2008] perspective of adversarial auto-encoders is presented in Tolstikhin et al. [2018].

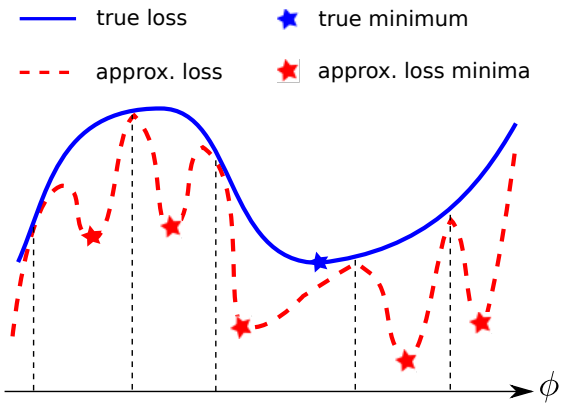


Fig. 5.1 A visualisation of the exact/approximate loss. See main text for further intuition.

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.

5.3.2 Direct gradient approximation

The recent development of machine learning algorithms, including VI and SG-MCMC, relies 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 visited by gradient descent, 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 5.1. Recall that practically, an optimiser only visits *finite* number of locations in the parameter space. As one would typically use a deep neural network to approximate the exact loss function, without careful control the deep net can potentially overfit to the observed evaluations, leading to strongly biased gradient updates and bad local optima.

Here are two concrete examples for further explanation.

- Variational EM as an approximation to MLE.
  The VAE algorithm can be viewed as an approximate MLE procedure, with the maximum likelihood objective $\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{D}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})]$ approximated by the variational lower-bound $\hat{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{VI}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}); \boldsymbol{x})]$. As shown in the mean-field example in

Section 4.2.1, this would bias the generative model $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ towards simple solutions, unless $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ perfectly approximates the exact posterior which is rarely the case. This issue is related to the "hidden unit over-pruning" problem [Burda et al., 2016; Sønderby et al., 2016]: even when $\boldsymbol{z}$ is of relatively high dimensions like a hundred, the VAE algorithm would turn off many of them, and learn a model which effectively uses much fewer units.

- GAN training.
  Recently generative adversarial networks (GANs) [Goodfellow et al., 2014] have attracted large attention from the deep learning community. In a nutshell, the original GAN algorithm proposes training the generator $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ by minimising the Jensen-Shannon divergence

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \mathrm{JS}[p_{\mathcal{D}}||p_{\boldsymbol{\theta}}] = \frac{1}{2}\mathrm{KL}\left[p_{\mathcal{D}}||\frac{p_{\mathcal{D}}+p_{\boldsymbol{\theta}}}{2}\right] + \frac{1}{2}\mathrm{KL}\left[p_{\boldsymbol{\theta}}||\frac{p_{\mathcal{D}}+p_{\boldsymbol{\theta}}}{2}\right]. \quad (5.16)$$

  However, the generative model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is implicitly defined in an analogous way as the deterministic transform discussed above. Thus point-wise evaluation of $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is intractable. Then the seminal GAN paper proposes approximating the Jenson-Shannon divergence [Lin, 1991] with a variational lower-bound, described by a *discriminator*:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \hat{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\mathcal{D}}[\log D_{\boldsymbol{\phi}}(\boldsymbol{x})] + \mathbb{E}_{p_{\boldsymbol{\theta}}}[\log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{x}))]. \quad (5.17)$$

  Arjovsky and Bottou [2017] pointed out a fundamental problem with this approach: since both $p_{\mathcal{D}}$ and $p_{\boldsymbol{\theta}}$ have low-dimensional support in a high-dimensional space, the discriminator, if powerful enough (which is typically the case when using neural networks), is very likely to overfit, thus it can perfectly separate the two support subspaces and provide meaningless gradients. Then in their follow-up work Arjovsky et al. [2017] proposed the Wasserstein GAN (WGAN), which uses Wasserstein distance [Villani, 2008] as the training objective, and in this case, the approximate loss function turns out to be

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}:||D_{\boldsymbol{\phi}}||_L \leq 1} \hat{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\mathcal{D}}[D_{\boldsymbol{\phi}}(\boldsymbol{x})] - \mathbb{E}_{p_{\boldsymbol{\theta}}}[D_{\boldsymbol{\phi}}(\boldsymbol{x})]. \quad (5.18)$$

  This modification, when adding more tricks to enforce the constraint $||D_{\boldsymbol{\phi}}||_L \leq 1$ such as a gradient penalty [Gulrajani et al., 2017], has largely solved the instability issue of GAN training.

Another interesting explanation for why WGAN works is that the power of the discriminator, or the test function $D_{\boldsymbol{\phi}}$, has been restrained. On the other hand, in the original GAN case, over-fitting frequently happens, particularly at the beginning of training, as neural network classifiers can easily fit almost any data, even that with random labels as claimed by Zhang et al. [2017]. Since smoothness is typically lost when over-fitting appears, it leads to poor approximations to the actual gradient and then a bad-performing model. Indeed Kodali et al. [2017] also showed that the original GAN training can be stabilised when the discriminator is also constrained to be 1-Lipschitz.

From the above two examples, 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. There exists a rich literature on (non-parametric) derivative estimation [De Brabanter et al., 2013; Fan and Gijbels, 1996; Ruppert and Wand, 1994; Stone, 1985; Zhou and Wolfe, 2000]; however, many of them require at least a noisy version of $\log q$ at the sampled locations, which is intractable in our case. Instead, Singh [1977] applied a kernel estimator directly to the first and higher order derivatives, and Sasaki et al. [2015] improved upon this idea by performing kernel ridge regression directly on the derivatives. Also Hyvärinen [2005] considered score matching methods for approximating $\nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})$, where follow-up papers [Sasaki et al., 2014; Strathmann et al., 2015] derived kernel-based solutions and applied them to tasks such as approximate Bayesian computation (ABC) [Beaumont et al., 2002]. The core idea of these methods is the use of integration by parts to avoid evaluations of the actual gradients, making them applicable in our context. In Chapter 6 we will further discuss gradient approximation techniques and propose new gradient estimators for implicit models and wild approximate inference.

Remark (denoising auto-encoder as a score function estimator)**.** It has been shown in Alain and Bengio [2014]; Särelä and Valpola [2005] 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 $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \sigma\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ by minimising the mean square error. Then the optimal DAE can be used to approximate the score function as $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) \approx \frac{1}{\sigma^2}(\text{DAE}^*(\boldsymbol{x}) - \boldsymbol{x})$. Sonderby et al. [2017] deployed this idea to train an implicit model for image super-resolution, providing some promising results in some metrics. However applying similar ideas to variational 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.

5.3.3 New optimisation objectives

In variational inference, the KL-divergence $\mathrm{KL}[q||p]$ is minimised to obtain the approximate posterior. In general, the KL-divergence minimisation can be replaced by other optimisation-based approximation methods, as long as with the guarantee of recovering the exact posterior if $\mathcal{Q}$ contains it. However simply replacing the objective with some other f-divergence [Ali and Silvey, 1966; Csiszár, 1963; Morimoto, 1963] does not simplify the problem as q has an intractable density. Variational and adversarial techniques for estimating f-divergences [Nguyen et al., 2007, 2010; Nowozin et al., 2016] do not apply either, as the exact posterior is difficult to sample.

One promising direction is to replace the KL divergence with Stein discrepancy [Barbour, 1988; Gorham and Mackey, 2015; Stein, 1972], which has a special form that does not require evaluating q nor sampling from p. Briefly speaking, Stein discrepancy involves a linear functional operator $\mathbf{O}_p$, called Stein operator, on a set of test functions $\mathcal{H} = \{h(\boldsymbol{z})\}$ such that $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[(\mathbf{O}_p h)(\boldsymbol{z})] = 0$ for $\forall h \in \mathcal{H}$. Then the associated *Stein discrepancy* is defined as $\mathbb{S}(q,p) = \sup_{h\in\mathcal{H}} \mathbb{E}_q[(\mathbf{O}_p h)(\boldsymbol{z})]$. For continuous density functions, a generic Stein operator derived from Stein's identity [Stein, 1972, 1981] is $(\mathbf{O}_p h)(\boldsymbol{z}) = \nabla_{\boldsymbol{z}} \log p(\boldsymbol{z},\boldsymbol{x}) h(\boldsymbol{z}) + \langle \nabla, h(\boldsymbol{z}) \rangle$, for which $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[(\mathbf{O}_p h)(\boldsymbol{z})] = 0$. Putting them together, we have the Stein discrepancy (equipped with norm $||\cdot||$) [Gorham and Mackey, 2015]

$$\mathbb{S}(q,p) = \sup_{h\in\mathcal{H}} ||\mathbb{E}_q[\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z},\boldsymbol{x}) h(\boldsymbol{z}) + \langle \nabla, h(\boldsymbol{z}) \rangle]||, \tag{5.19}$$

which only requires samples from q and the score function $\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z},\boldsymbol{x})$ and is thus indeed tractable.

Very recently Stein's method has been introduced to the approximate inference community. Ranganath et al. [2016a] defined $\mathcal{H}$ as parametric functions represented by neural networks, and obtained an approximate posterior by solving $\min_q \mathbb{S}(q,p)$. The authors approximated the minimax optimisation with gradient descent in an analogous way to GAN training [Goodfellow et al., 2014]. In contrast, analytic solution of the supremum in (5.19) exists if $\mathcal{H}$ 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) that will be further discussed in Chapter 6. Liu and Feng [2016] further developed an approximate inference algorithm by directly minimising the KSD between the exact and approximate posterior distributions.

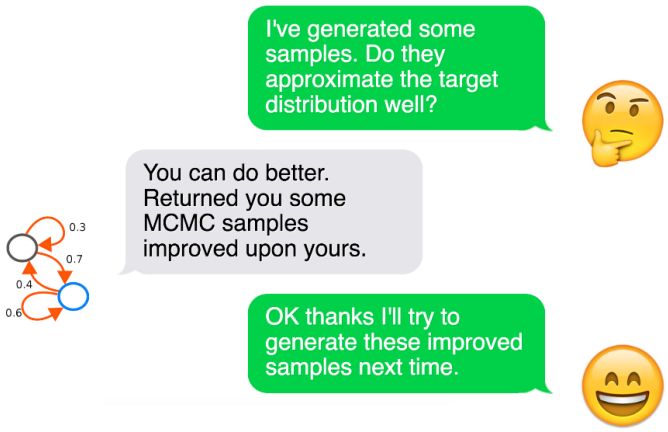


Fig. 5.2 A cartoon illustration of the amortised MCMC idea in Li et al. [2017].

5.3.4 Amortising stochastic 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, we sample $\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x})$, simulate T-step stochastic dynamics to obtain the updated particle $\boldsymbol{z}_T$, and update the q distribution to "catch-up" those updated particles. For example, Wang and Liu [2016] used this idea to amortise a deterministic dynamics called Stein variational gradient descent (SVGD) [Liu and Wang, 2016], where the "catch-up" step is defined by deliberately chaining the gradients $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \varepsilon \mathbb{E}_q[\nabla_{\boldsymbol{\phi}} \boldsymbol{z}(\boldsymbol{z}_T - \boldsymbol{z})]$. In Li et al. [2017] we extended this principle to MCMC methods and introduced different update rules for the (implicit) q distribution. The theoretical intuition behind this approach is illustrated in Figure 5.2. Since the MCMC "oracle" always improves the sample quality in terms of approximating the target distribution, 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 dynamics as long as they generate particles that are always approaching to the target distribution.

5.4 Application: meta-learning for approximate inference

Many inference algorithms discussed in this thesis, e.g. variational inference (with KL or Rényi divergences) and sampling (importance sampling, SMC, MCMC, etc.), are designed as

generic algorithms that can be directly applied to any integration problem. Can we automate the design of inference algorithms that are tailored to specific types of machine learning tasks (e.g. Bayesian neural network regression)? In this section we briefly discuss a meta-learning [Bengio et al., 1992; Naik and Mammone, 1992; Schmidhuber, 1987; Thrun and Pratt, 1998] approach towards this goal. The research scope of meta-learning is very broad, but in general the idea is to train a *learner* on one or multiple tasks, in order to acquire common knowledge on *how* to learn and generalise the learner to future tasks. Therefore meta-learning enables an inference algorithm to exploit commonly shared dependencies in a class of densities (say the posterior distribution of Bayesian neural network weights), and it is expected to produce a better approximation to the exact posterior.

In approximate inference context we define a learner as an algorithm $\mathcal{A} : \mathcal{P} \to \mathcal{Q}$ that maps the target distribution $p \in \mathcal{P}$ to an approximate distribution $q \in \mathcal{Q}$. For example, the classic variational inference algorithm can be rewritten as

$$\mathcal{A}_{\mathrm{VI}}(p(\boldsymbol{z}|\boldsymbol{x})) = \arg\max_{q \in \mathcal{Q}} \mathbb{E}_q \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} \right]. \tag{5.20}$$

Also the T-step simulation of an MCMC procedure with kernel $\mathcal{T}$ and initial distribution $q_0(\boldsymbol{z}|\boldsymbol{x})$ is

$$\mathcal{A}_{\mathcal{T},T,q_0}(p(\boldsymbol{z}|\boldsymbol{x})) = \int \mathcal{T}_T(\boldsymbol{z}|\boldsymbol{z}') q_0(\boldsymbol{z}'|\boldsymbol{x}) d\boldsymbol{z}'. \tag{5.21}$$

Often these algorithms are approximated executed, e.g. for variational inference the maximisation operator is approximated by T-step gradient ascent.

We consider meta-learning for approximate inference, which optimises $\mathcal{A}$ on a set of training densities $\{p_n(\boldsymbol{z})\}_{n=1}^N$, and generalises the learned algorithm $\mathcal{A}^*$ to unseen distributions. The training densities might come from a multi-task learning set-up, i.e. $p_n(\boldsymbol{z}) = p(\boldsymbol{z}|\mathcal{D}_n)$ where $\mathcal{D}_n$ is the dataset for task n. Then one can define $\mathcal{A}(p(\boldsymbol{z}|\mathcal{D})) = q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathcal{D})$ with e.g. a neural network, and then optimise the associated variational parameters $\boldsymbol{\phi}$ on the datasets $\mathcal{D}_1, ..., \mathcal{D}_N$. Further approximation techniques such as coresets [Huggins et al., 2016] and inducing points [Snelson and Ghahramani, 2006] can also be utilised If the dataset $\mathcal{D}_n$ is very big. In this case meta-learning for approximate inference algorithms can be viewed as amortised inference on *task* level, therefore many of the recently developed techniques can be deployed. But it might require the use of techniques discussed in previous sections, as one might prefer the learned algorithm $\mathcal{A}$ to produce implicit posterior approximations to the target densities.

In general the p_n densities might be derived from different probabilistic models with different observations and different latent variables, and the above neural network parameter-

isation is very likely to be sub-optimal. Instead of directly parameterising the q distributions, we propose learning an approximate inference algorithm $\mathcal{A}_{\boldsymbol{\phi}}$ based on optimisation and/or sampling. Consider learning a Markov process for posterior inference as an example. By parameterising $\mathcal{T}_n(\boldsymbol{z}'_n|\boldsymbol{z}_n) = \mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{z}'_n|\boldsymbol{z}_n; p_n)$ for all $n = 1, ..., N$, the sampler is learned by optimising some meta-objective $\mathcal{L}_{\text{meta}}$ on $\mathcal{A}_{\mathcal{T}_{\boldsymbol{\phi}},T,q_0}(p_n(\boldsymbol{z}_n))$ for all $n = 1, ..., N$. Thus the trained transition kernel $\mathcal{T}_{\boldsymbol{\phi}}$ will have its stationary distribution close to the target p_n, and/or it will have fast convergence and low bias properties if $\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{z}'_n|\boldsymbol{z}_n; p_n)$ is implicitly defined by a diffusion process with p_n as the stationary distribution [Ma et al., 2015]. The design of the meta-objective $\mathcal{L}_{\text{meta}}$ should avoid evaluating the density of the approximate posterior as an MCMC algorithm typically returns an implicit distribution. An initial experiment for learning samplers is presented in Chapter 6.

5.5 Summary

We presented *wild approximate inference* as a new research area within approximate inference. We established this area by investigating the fundamental question of what constitutes tractable approximate inference, and discussed potential restrictions introduced by both analytical approximate posteriors and conventional sampling methods. Then we provided examples of implicit approximations, and briefly discussed four algorithmic options for fitting them. Note that the recipes provided here are still mostly incomplete, and I must have missed many creative solutions developed by other researchers very recently. Nevertheless, it seems reasonable to believe that elucidating existing problems and pointing new research directions would help the community develop better approximate inference methods, thus leading to better Bayesian modelling.

In the next chapter, I will present one of our recent work that follows the gradient approximation proposal for wild approximate inference. There we will propose a new estimator of the score function $\nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})$, and evaluate its approximating accuracy by considering applications in Bayesian deep learning.

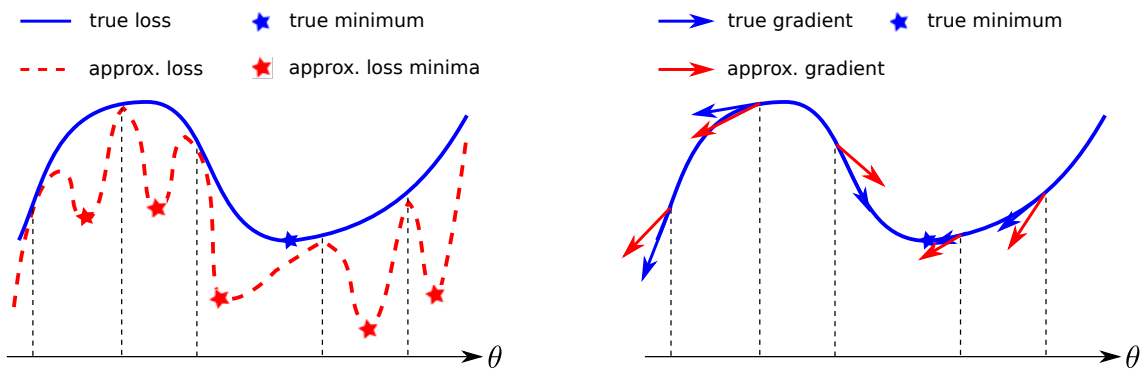
Chapter 6

Gradient Estimators for Implicit Models

Modelling is fundamental to the success of technological innovations for artificial intelligence. A powerful model learns a useful representation of the observations for a specified prediction task, and generalises to unknown instances that follow similar generative mechanics. A well established area of machine learning research focuses on developing *prescribed probabilistic models* [Diggle and Gratton, 1984], where learning is based on evaluating the probability of observations under the model. *Implicit probabilistic models*, on the other hand, are defined by a stochastic procedure that allows for direct generation of samples, but not for the evaluation of model probabilities. These are omnipresent in scientific and engineering research involving data analysis, for instance ecology, climate science and geography, where simulators are used to fit real-world observations to produce forecasting results.

Within the machine learning community, there is a recent interest in a specific type of implicit models, generative adversarial networks (GANs) [Goodfellow et al., 2014], which has been shown to be one of the most successful approaches to image generation [Arjovsky et al., 2017; Berthelot et al., 2017; Radford et al., 2016]. Very recently, implicit distributions have also been considered as approximate posterior distributions for Bayesian inference, e.g. see discussions in the last chapter and recent papers including Huszár [2017]; Karaletsos [2016]; Li et al. [2017]; Liu and Feng [2016]; Mescheder et al. [2017]; Shi et al. [2018]; Tran et al. [2017]; Wang and Liu [2016]. These examples demonstrate the superior flexibility of implicit models, which provide highly expressive means of modelling complex data structures.

Whilst prescribed probabilistic models can be learned by standard (approximate) maximum likelihood or Bayesian inference, implicit probabilistic models require substantially more severe approximations due to the intractability of the model distribution. Many existing approaches first approximate the model distribution or optimisation objective function and then use those approximations to learn the associated parameters. However, such approxima-



(a) approximate loss function (b) approximate gradients

Fig. 6.1 A comparison between the two approximation schemes. Since in practice the optimiser only visits finite number of locations in the parameter space, it can lead to overfitting if the neural network based functional approximator is not carefully regularised, and therefore the curvature information of the approximated loss can be very different from that of the original loss (shown in (a)). On the other hand, the gradient approximation scheme (b) can be more accurate since it only involves estimating the sensitivity of the loss function to the parameters in a local region.

tion can lead to unstable training and poor results, where a prevalent example is the original GAN framework [Goodfellow et al., 2014] that has been briefly sketched in Section 5.3.2. Recent ideas to address this issue for GANs suggest that restricting the representational power of the discriminator is effective in stabilising training [e.g. see Arjovsky et al., 2017; Kodali et al., 2017]. However, such restrictions, if not carefully crafted, often introduce undesirable biases, responsible for problems such as mode collapse in the context of GANs, and uncertainty underestimation in variational inference methods [Turner and Sahani, 2011].

In the previous chapter a number of proposals for wild approximate inference are presented. Critically, we believe these techniques are extendable to learning implicit models, and in this chapter we explore the *direct gradient approximation* idea as an alternative. A visualisation of the two approximation schemes is provided in Figure 6.1. More specifically we focus on approximating the score function, in which an accurate approximation of it then allows the application of many well-studied algorithms, such as maximum likelihood, maximum entropy estimation, variational inference and gradient-based MCMC, to implicit models. Concretely, our contributions include:

- the *Stein gradient estimator*, a novel generalisation of the score matching estimator [Hyvärinen, 2005], with both parametric and non-parametric versions;

- a comparison of the proposed estimator with the score matching and the KDE plug-in estimators on performing gradient-free MCMC, meta-learning of approximate posterior samplers for Bayesian neural networks, and entropy based regularisation of GANs.

6.1 Learning implicit probabilistic models

Given a dataset $\mathcal{D}$ containing i.i.d. samples we would like to learn a probabilistic model $p(\boldsymbol{x})$ for the underlying data distribution $p_{\mathcal{D}}(\boldsymbol{x})$. In the case of implicit models, $p(\boldsymbol{x})$ is defined by a generative process. For example, to generate images, one might define a generative model $p(\boldsymbol{x})$ that consists of sampling randomly a latent variable $\boldsymbol{z} \sim p_0(\boldsymbol{z})$ and then defining $\boldsymbol{x} = \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{z})$. Here $\mathbf{f}$ is a function parametrised by $\boldsymbol{\theta}$, usually a deep neural network or a simulator. We assume $\mathbf{f}$ to be differentiable w.r.t. $\boldsymbol{\theta}$. An extension to this scenario is presented by *conditional* implicit models, where the addition of a supervision signal $\mathbf{y}$, such as an image label, allows us to define a conditional distribution $p(\boldsymbol{x}|\mathbf{y})$ implicitly by the transformation $\boldsymbol{x} = \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{z}, \mathbf{y})$. A related methodology, *wild approximate inference* (Chapter 5) assumes a tractable joint density $p(\boldsymbol{x}, \boldsymbol{z})$, but uses implicit proposal distributions to approximate an intractable exact posterior $p(\boldsymbol{z}|\boldsymbol{x})$. Here the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ can likewise be represented by a deep neural network, but also by a truncated Markov chain, such as that given by Langevin dynamics with learnable step-size.

Whilst providing extreme flexibility and expressive power, the intractability of density evaluation also brings serious optimisation issues for implicit models. This is because many learning algorithms, e.g. maximum likelihood estimation (MLE), rely on minimising a distance/divergence/discrepancy measure $\mathrm{D}[p||p_{\mathcal{D}}]$, which often requires evaluating the model density [c.f. Liu and Feng, 2016; Ranganath et al., 2016a]. Thus good approximations to the optimisation procedure are the key to learning implicit models that can describe complex data structure. In the context of GANs, the Jensen-Shannon divergence is approximated by a variational lower-bound represented by a discriminator [Barber and Agakov, 2003; Goodfellow et al., 2014]. Related work for wild variational inference [Huszár, 2017; Li and Liu, 2016; Mescheder et al., 2017; Tran et al., 2017] uses a GAN-based technique to construct a density ratio estimator for q/p_0 [Mohamed and Lakshminarayanan, 2016; Sugiyama et al., 2009, 2012; Uehara et al., 2016] and then approximates the KL-divergence term in the variational lower-bound:

$$\mathcal{L}_{\mathrm{VI}}(q) = \mathbb{E}_q\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \mathrm{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_0(\boldsymbol{z})]. \tag{6.1}$$

In addition, Li and Liu [2016] and Mescheder et al. [2017] exploit the additive structure of the KL-divergence and suggest discriminating between q and an auxiliary distribution that is close to q, making the density ratio estimation more accurate. Nevertheless all these algorithms involve a minimax optimisation, and the current practice of gradient-based optimisation is notoriously unstable.

The stabilisation of GAN training is itself a recent trend of related research [e.g. see Arjovsky et al., 2017; Salimans et al., 2016]. However, as the gradient-based optimisation only interacts with gradients, there is no need to use a discriminator if an accurate approximation to the intractable gradients could be obtained. As an example, consider a variational inference task with the approximate posterior defined as $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \Leftrightarrow \boldsymbol{\varepsilon} \sim \pi(\boldsymbol{\varepsilon}), \boldsymbol{z} = \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x})$. Notice that the variational lower-bound can be rewritten as

$$\mathcal{L}_{\mathrm{VI}}(q) = \mathbb{E}_q\left[\log p(\boldsymbol{x}, \boldsymbol{z})\right] + \mathbb{H}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})], \tag{6.2}$$

the gradient of the variational parameters $\boldsymbol{\phi}$ can be computed by a sum of the path gradient of the first term (i.e. $\mathbb{E}_\pi\left[\nabla_{\mathbf{f}} \log p(\boldsymbol{x}, \mathbf{f}(\boldsymbol{\varepsilon}, \boldsymbol{x}))^{\mathrm{T}} \nabla_{\boldsymbol{\phi}} \mathbf{f}(\boldsymbol{\varepsilon}, \boldsymbol{x})\right]$) and the gradient of the entropy term $\nabla_{\boldsymbol{\phi}} \mathbb{H}[q(\boldsymbol{z}|\boldsymbol{x})]$. Expanding the latter, we have

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \mathbb{H}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})] &= -\nabla_{\boldsymbol{\phi}} \mathbb{E}_{\pi(\boldsymbol{\varepsilon})}[\log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x}))] \\
&= -\mathbb{E}_{\pi(\boldsymbol{\varepsilon})}[\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x}))] \\
&= -\mathbb{E}_{\pi(\boldsymbol{\varepsilon})}[\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})|_{\boldsymbol{z}=\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x})} + \nabla_{\boldsymbol{\phi}} \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x}) \nabla_{\mathbf{f}} \log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x})|\boldsymbol{x})] \\
&= -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})] - \mathbb{E}_{\pi(\boldsymbol{\varepsilon})}[\nabla_{\boldsymbol{\phi}} \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x}) \nabla_{\mathbf{f}} \log q_{\boldsymbol{\phi}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x})|\boldsymbol{x})],
\end{aligned} \tag{6.3}$$

in which the first term in the last line is zero [Roeder et al., 2017]. As we typically assume the tractability of $\nabla_{\boldsymbol{\phi}} \mathbf{f}$, an accurate approximation to $\nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})$ would remove the requirement of discriminators, speed-up the learning and obtain potentially a better model. Many gradient approximation techniques exist [De Brabanter et al., 2013; Fan and Gijbels, 1996; Stone, 1985; Zhou and Wolfe, 2000], and in particular, in the next section we will review kernel-based methods such as kernel density estimation [Singh, 1977] and score matching [Hyvärinen, 2005] in more detail, and motivate the main contribution of this chapter.

6.2 Gradient approximation with the Stein gradient estimator

We propose the *Stein gradient estimator* as a novel generalisation of the score matching gradient estimator. Before presenting it we first set-up the notation. Column vectors and matrices are boldfaced. The random variable under consideration is $\boldsymbol{x} \in \mathcal{X}$ with $\mathcal{X} = \mathbb{R}^{d\times 1}$ if not specifically mentioned. To avoid misleading notation we use the distribution $q(\boldsymbol{x})$ to derive the gradient approximations for general cases. As Monte Carlo methods are heavily used for implicit models, in the rest of the paper we mainly consider approximating the gradient $\boldsymbol{g}(\boldsymbol{x}^k) := \nabla_{\boldsymbol{x}^k} \log q(\boldsymbol{x}^k)$ for $\boldsymbol{x}^k \sim q(\boldsymbol{x}), k = 1, ..., K$. We use x^i_j to denote the jth element of the ith sample $\boldsymbol{x}^i$. We also denote the matrix form of the collected gradients as $\mathbf{G} := \left(\nabla_{\boldsymbol{x}^1} \log q(\boldsymbol{x}^1), \cdots, \nabla_{\boldsymbol{x}^K} \log q(\boldsymbol{x}^K)\right)^{\mathrm{T}} \in \mathbb{R}^{K\times d}$, and its approximation $\hat{\mathbf{G}} := \left(\hat{g}(\boldsymbol{x}^1), \cdots, \hat{g}(\boldsymbol{x}^K)\right)^{\mathrm{T}}$ with $\hat{g}(\boldsymbol{x}^k) = \nabla_{\boldsymbol{x}^k} \log \hat{q}(\boldsymbol{x}^k)$ for some $\hat{q}(\boldsymbol{x})$.

6.2.1 Stein gradient estimator: inverting Stein's identity

We start from introducing *Stein's identity* that was first developed for Gaussian random variables [Stein, 1972, 1981] then extended to general cases [Gorham and Mackey, 2015; Liu et al., 2016]. Let $\mathbf{h} : \mathbb{R}^{d\times 1} \to \mathbb{R}^{d'\times 1}$ be a differentiable multivariate test function which maps $\boldsymbol{x}$ to a column vector $\mathbf{h}(\boldsymbol{x}) = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), ..., h_{d'}(\boldsymbol{x})]^{\mathrm{T}}$. We further assume the *boundary condition* for $\mathbf{h}$:

$$q(\boldsymbol{x})\mathbf{h}(\boldsymbol{x})|_{\partial\mathcal{X}} = \mathbf{0}, \text{ or } \lim_{\boldsymbol{x}\to\boldsymbol{\infty}} q(\boldsymbol{x})\mathbf{h}(\boldsymbol{x}) = 0 \text{ if } \mathcal{X} = \mathbb{R}^d. \tag{6.4}$$

This condition holds for almost any test function if q has sufficiently fast-decaying tails (e.g. Gaussian tails). Now we introduce Stein's identity [Gorham and Mackey, 2015; Liu et al., 2016; Stein, 1981]

$$\mathbb{E}_q[\mathbf{h}(\boldsymbol{x})\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})^{\mathrm{T}} + \nabla_{\boldsymbol{x}}\mathbf{h}(\boldsymbol{x})] = \mathbf{0}, \tag{6.5}$$

in which the gradient matrix term $\nabla_{\boldsymbol{x}}\mathbf{h}(\boldsymbol{x}) = (\nabla_{\boldsymbol{x}} h_1(\boldsymbol{x}), \cdots, \nabla_{\boldsymbol{x}} h_{d'}(\boldsymbol{x}))^{\mathrm{T}} \in \mathbb{R}^{d'\times d}$. This identity can be proved using *integration by parts*: for the ith row of the matrix $\mathbf{h}(\boldsymbol{x})\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})^{\mathrm{T}}$, we have

$$\begin{aligned}
\mathbb{E}_q[h_i(\boldsymbol{x})\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})^{\mathrm{T}}] &= \int h_i(\boldsymbol{x})\nabla_{\boldsymbol{x}} q(\boldsymbol{x})^{\mathrm{T}} d\boldsymbol{x} \\
&= q(\boldsymbol{x})h_i(\boldsymbol{x})|_{\partial\mathcal{X}} - \int q(\boldsymbol{x})\nabla_{\boldsymbol{x}} h_i(\boldsymbol{x})^{\mathrm{T}} d\boldsymbol{x} \\
&= -\mathbb{E}_q[\nabla_{\boldsymbol{x}} h_i(\boldsymbol{x})^{\mathrm{T}}].
\end{aligned} \tag{6.6}$$

Observing that the gradient term $\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})$ of interest appears in Stein's identity (6.5), we propose the *Stein gradient estimator* by inverting Stein's identity. As the expectation in (6.5) is intractable, we further approximate the above with Monte Carlo (MC):

$$\frac{1}{K}\sum_{k=1}^{K} -\mathbf{h}(\boldsymbol{x}^k)\nabla_{\boldsymbol{x}^k} \log q(\boldsymbol{x}^k)^{\mathrm{T}} + \mathrm{err} = \frac{1}{K}\sum_{k=1}^{K} \nabla_{\boldsymbol{x}^k}\mathbf{h}(\boldsymbol{x}^k), \quad \boldsymbol{x}^k \sim q(\boldsymbol{x}^k), \tag{6.7}$$

with $\mathrm{err} \in \mathbb{R}^{d' \times d}$ the random error due to MC approximation, which has mean $\mathbf{0}$ and vanishes as $K \to +\infty$. Now by temporarily denoting

$$\mathbf{H} = \big(\mathbf{h}(\boldsymbol{x}^1), \cdots, \mathbf{h}(\boldsymbol{x}^K)\big) \in \mathbb{R}^{d' \times K}, \quad \overline{\nabla_{\boldsymbol{x}}\mathbf{h}} = \frac{1}{K}\sum_{k=1}^{K} \nabla_{\boldsymbol{x}^k}\mathbf{h}(\boldsymbol{x}^k) \in \mathbb{R}^{d' \times d},$$

equation (6.7) can be rewritten as

$$-\frac{1}{K}\mathbf{H}\mathbf{G} + \mathrm{err} = \overline{\nabla_{\boldsymbol{x}}\mathbf{h}}.$$

Thus we consider a ridge regression method (i.e. adding an ℓ_2 regulariser) to estimate $\mathbf{G}$:

$$\hat{\mathbf{G}}_V^{\mathrm{Stein}} := \operatorname*{arg\,min}_{\hat{\mathbf{G}} \in \mathbb{R}^{K \times d}} ||\overline{\nabla_{\boldsymbol{x}}\mathbf{h}} + \frac{1}{K}\mathbf{H}\hat{\mathbf{G}}||_F^2 + \frac{\eta}{K^2}||\hat{\mathbf{G}}||_F^2, \tag{6.8}$$

with $||\cdot||_F$ the Frobenius norm of a matrix and $\eta \geq 0$. Simple calculation shows that

$$\hat{\mathbf{G}}_V^{\mathrm{Stein}} = -(\mathbf{K} + \eta \boldsymbol{I})^{-1}\langle \nabla, \mathbf{K} \rangle, \tag{6.9}$$

where

$$\mathbf{K} := \mathbf{H}^{\mathrm{T}}\mathbf{H}, \quad \mathbf{K}_{ij} = \mathcal{K}(\boldsymbol{x}^i, \boldsymbol{x}^j) := \mathbf{h}(\boldsymbol{x}^i)^{\mathrm{T}}\mathbf{h}(\boldsymbol{x}^j),$$

$$\langle \nabla, \mathbf{K} \rangle := K\mathbf{H}^{\mathrm{T}}\overline{\nabla_{\boldsymbol{x}}\mathbf{h}}, \quad \langle \nabla, \mathbf{K} \rangle_{ij} = \sum_{k=1}^{K} \nabla_{x_j^k}\mathcal{K}(\boldsymbol{x}^i, \boldsymbol{x}^k).$$

One can show that the RBF kernel satisfies Stein's identity [Liu et al., 2016]. In this case $\mathbf{h}(\boldsymbol{x}) = \mathcal{K}(\boldsymbol{x}, \cdot), d' = +\infty$ and by the reproducing kernel property [Berlinet and Thomas-Agnan, 2011], $\mathbf{h}(\boldsymbol{x})^{\mathrm{T}}\mathbf{h}(\boldsymbol{x}') = \langle \mathcal{K}(\boldsymbol{x}, \cdot), \mathcal{K}(\boldsymbol{x}', \cdot) \rangle_{\mathcal{H}} = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$.

6.2.2 Stein gradient estimator minimises the kernelised Stein discrepancy

In this section we derive the Stein gradient estimator again, but from a divergence/discrepancy minimisation perspective. Stein's method also provides a tool for checking if two distributions $q(\boldsymbol{x})$ and $\hat{q}(\boldsymbol{x})$ are identical. If the test function set $\mathcal{H}$ (containing one-dimensional test functions $h \in \mathcal{H}$) is sufficiently rich, then one can define a Stein discrepancy measure (equipped with norm $||\cdot||$) by

$$\mathcal{S}(q,\hat{q}) := \sup_{\mathbf{h}\in\mathcal{H}} ||\mathbb{E}_q\left[\nabla_{\boldsymbol{x}} \log \hat{q}(\boldsymbol{x}) h(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} h(\boldsymbol{x})\right]||, \tag{6.10}$$

see Gorham and Mackey [2015] for an example derivation. The definition can be generalised to multi-dimension test functions $\mathbf{h}(\boldsymbol{x})$, and in particular, when $\mathcal{H}$ is defined as a unit ball in an RKHS induced by a kernel $\mathcal{K}(\boldsymbol{x},\cdot)$ and $||\cdot||$ is the ℓ_2 norm, Liu et al. [2016] and Chwialkowski et al. [2016] showed that the supremum in (6.10) can be analytically obtained as (recall $\boldsymbol{g}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log q(\boldsymbol{x}), \hat{\boldsymbol{g}}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log \hat{q}(\boldsymbol{x})$):

$$\mathcal{S}^2(q,\hat{q}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim q}\left[(\hat{\boldsymbol{g}}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x}))^{\mathrm{T}} \mathcal{K}(\boldsymbol{x},\boldsymbol{x}')(\hat{\boldsymbol{g}}(\boldsymbol{x}') - \boldsymbol{g}(\boldsymbol{x}'))\right], \tag{6.11}$$

which is also named the *kernelised Stein discrepancy* (KSD). Chwialkowski et al. [2016] showed that for C_0-universal kernels satisfying the boundary condition, KSD is indeed a discrepancy measure: $\mathcal{S}^2(q,\hat{q}) = 0 \Leftrightarrow q = \hat{q}$. Gorham and Mackey [2017] further characterised the power of KSD on detecting non-convergence cases. Furthermore, if the kernel is twice differentiable, then using the same technique as to derive (6.16) one can compute KSD by (with $\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'}$ shorthand for $\mathcal{K}(\boldsymbol{x},\boldsymbol{x}')$):

$$\mathcal{S}^2(q,\hat{q}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim q}\left[\hat{\boldsymbol{g}}(\boldsymbol{x})^{\mathrm{T}} \mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} \hat{\boldsymbol{g}}(\boldsymbol{x}') + \hat{\boldsymbol{g}}(\boldsymbol{x})^{\mathrm{T}} \nabla_{\boldsymbol{x}'} \mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} + \nabla_{\boldsymbol{x}} \mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'}^{\mathrm{T}} \hat{\boldsymbol{g}}(\boldsymbol{x}') + \mathrm{Tr}(\nabla_{\boldsymbol{x},\boldsymbol{x}'} \mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'})\right]. \tag{6.12}$$

In practice KSD is estimated with samples $\{\boldsymbol{x}^k\}_{k=1}^K \sim q$, and simple derivations show that the V-statistic of KSD can be reformulated as $\mathcal{S}_V^2(q,\hat{q}) = \frac{1}{K^2}\mathrm{Tr}(\hat{\mathbf{G}}^{\mathrm{T}}\mathbf{K}\hat{\mathbf{G}} + 2\hat{\mathbf{G}}^{\mathrm{T}}\langle\nabla,\mathbf{K}\rangle) + C$. Thus the l_2 error in (6.8) is equivalent to the V-statistic of KSD if $\mathbf{h}(\boldsymbol{x}) = \mathcal{K}(\boldsymbol{x},\cdot)$, and we have the following:

Theorem 6.1. $\hat{\mathbf{G}}_V^{\mathrm{Stein}}$ *is the solution of the following KSD V-statistic minimisation problem*

$$\hat{\mathbf{G}}_V^{\mathrm{Stein}} = \underset{\hat{\mathbf{G}}\in\mathbb{R}^{K\times d}}{\arg\min} \quad \mathcal{S}_V^2(q,\hat{q}) + \frac{\eta}{K^2}||\hat{\mathbf{G}}||_F^2. \tag{6.13}$$

One can also minimise the U-statistic of KSD to obtain gradient approximations, and a full derivation of which, including the optimal solution, can be found in Appendix A.3. In experiments we use V-statistic solutions and leave comparisons between these methods to future work.

6.2.3 Comparisons to existing kernel-based gradient estimators

There exist other gradient estimators that do not require explicit evaluations of $\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})$, e.g. the denoising auto-encoder (DAE) [Alain and Bengio, 2014; Vincent, 2011; Vincent et al., 2008] which, with infinitesimal noise, also provides an estimate of $\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})$ at convergence. However, applying such gradient estimators result in a double-loop optimisation procedure since the gradient approximation is repeatedly required for fitting implicit distributions, which can be significantly slower than the proposed approach. Therefore we focus on "quick and dirty" approximations and only include comparisons to kernel-based gradient estimators in the following.

KDE gradient estimator: plug-in estimator with density estimation

A naive approach for gradient approximation would first estimate the intractable density $\hat{q}(\boldsymbol{x}) \approx q(\boldsymbol{x})$ (up to a constant), then approximate the exact gradient by $\nabla_{\boldsymbol{x}} \log \hat{q}(\boldsymbol{x}) \approx \nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})$. Specifically, Singh [1977] considered kernel density estimation (KDE) $\hat{q}(\boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}^k) \times C.$, then differentiated through the KDE estimate to obtain the gradient estimator:

$$\hat{\mathbf{G}}_{ij}^{\mathrm{KDE}} = \sum_{k=1}^{K} \nabla_{x_j^i} \mathcal{K}(\boldsymbol{x}^i, \boldsymbol{x}^k) / \sum_{k=1}^{K} \mathcal{K}(\boldsymbol{x}^i, \boldsymbol{x}^k). \tag{6.14}$$

Interestingly for translation invariant kernels $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x} - \boldsymbol{x}')$ the *KDE gradient estimator* (6.14) can be rewritten as $\hat{\mathbf{G}}^{\mathrm{KDE}} = -\mathrm{diag}\,(\mathbf{K}\mathbf{1})^{-1} \langle \nabla, \mathbf{K} \rangle$. Inspecting and comparing it with the Stein gradient estimator (6.9), one might notice that the Stein method uses the full kernel matrix as the pre-conditioner, while the KDE method computes an averaged "kernel similarity" for the denominator. We conjecture that this difference is key to the superior performance of the Stein gradient estimator when compared to the KDE gradient estimator (see later experiments). The KDE method only collects the similarity information between $\boldsymbol{x}^k$ and other samples $\boldsymbol{x}^j$ to form an estimate of $\nabla_{\boldsymbol{x}^k} \log q(\boldsymbol{x}^k)$, whereas for the Stein gradient estimator, the kernel similarity between $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ for all $i, j \neq k$ are also incorporated. Thus it is reasonable to conjecture that the Stein method can be more sample efficient, which also implies higher accuracy when the same number of samples are collected.

Score matching gradient estimator: minimising MSE

The KDE gradient estimator performs indirect approximation of the gradient via density estimation, which can be inaccurate. An alternative approach directly approximates the gradient $\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})$ by minimising the expected ℓ_2 error w.r.t. the approximation $\hat{\boldsymbol{g}}(\boldsymbol{x}) = (\hat{g}_1(\boldsymbol{x}), \cdots, \hat{g}_d(\boldsymbol{x}))^{\mathrm{T}}$:

$$\mathcal{F}(\hat{\boldsymbol{g}}) := \mathbb{E}_q \left[||\hat{\boldsymbol{g}}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})||_2^2 \right]. \tag{6.15}$$

It has been shown in Hyvärinen [2005] that this objective can be reformulated as

$$\mathcal{F}(\hat{\boldsymbol{g}}) = \mathbb{E}_q \left[||\hat{\boldsymbol{g}}(\boldsymbol{x})||_2^2 + 2\langle \nabla, \hat{\boldsymbol{g}}(\boldsymbol{x}) \rangle \right] + C, \quad \langle \nabla, \hat{\boldsymbol{g}}(\boldsymbol{x}) \rangle = \sum_{j=1}^{d} \nabla_{x_j} \hat{g}_j(\boldsymbol{x}). \tag{6.16}$$

The key insight here is again the usage of integration by parts: after expanding the ℓ_2 loss objective, the cross term can be rewritten as $\mathbb{E}_q \left[\hat{\boldsymbol{g}}(\boldsymbol{x})^{\mathrm{T}} \nabla_{\boldsymbol{x}} \log q(\boldsymbol{x}) \right] = -\mathbb{E}_q \left[\langle \nabla, \hat{\boldsymbol{g}}(\boldsymbol{x}) \rangle \right]$, if assuming the boundary condition (6.4) for $\hat{\boldsymbol{g}}$ (see (6.6)). The optimum of (6.16) is referred as the *score matching gradient estimator*. The ℓ_2 objective (6.15) is also called *Fisher divergence* [Johnson, 2004] which is a special case of KSD (6.11) by selecting $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \delta_{\boldsymbol{x}=\boldsymbol{x}'}$. Thus the Stein gradient estimator can be viewed as a generalisation of the score matching estimator.

The comparison between the two estimators is more complicated. Certainly by the Cauchy-Schwarz inequality the Fisher divergence is stronger than KSD in terms of detecting convergence [Liu et al., 2016]. However it is difficult to perform direct gradient estimation by minimising the Fisher divergence, since (i) the Dirac kernel is non-differentiable so that it is impossible to rewrite the divergence in a similar form to (6.12), and (ii) the transformation to (6.16) involves computing $\nabla_{\boldsymbol{x}} \hat{\boldsymbol{g}}(\boldsymbol{x})$. So one needs to propose a *parametric* approximation to $\mathbf{G}$ and then optimise the associated parameters accordingly, and indeed Sasaki et al. [2014] and Strathmann et al. [2015] derived a parametric solution by first approximating the log density up to a constant as $\log \hat{q}(\boldsymbol{x}) := \sum_{k=1}^{K} a_k \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}^k) + C$, then minimising (6.16) to obtain the coefficients $\hat{a}_k^{\text{score}}$ and constructing the gradient estimator as

$$\hat{\mathbf{G}}_{i\cdot}^{\text{score}} = \sum_{k=1}^{K} \hat{a}_k^{\text{score}} \nabla_{\boldsymbol{x}^i} \mathcal{K}(\boldsymbol{x}^i, \boldsymbol{x}^k). \tag{6.17}$$

Therefore the usage of parametric estimation can potentially remove the advantage of using a stronger divergence. Conversely, the proposed Stein gradient estimator (6.9) is *non-parametric* in that it directly optimises over functions evaluated at locations $\{\boldsymbol{x}_k\}_{k=1}^K$. This brings in two key advantages over the score matching gradient estimator: (i) it removes the *approximation error* due to the use of restricted family of parametric approximations

and thus can be potentially more accurate; (ii) it has a much simpler and *ubiquitous* form that applies to *any kernel satisfying the boundary condition*, whereas the score matching estimator requires tedious derivations for different kernels repeatedly (see Appendix A.3).

In terms of computation speed, since in most of the cases the computation of the score matching gradient estimator also involves kernel matrix inversions, both estimators are of the same order of complexity, which is $\mathcal{O}(K^3 + K^2 d)$ (kernel matrix computation plus inversion). Low-rank approximations such as the Nyström method [Smola and Schökopf, 2000; Williams and Seeger, 2001] can enable speed-up, but this is not investigated in the paper. Again we note here that kernel-based gradient estimators can still be faster than e.g. the DAE estimator since no double-loop optimisation is required. Certainly it is possible to apply early-stopping for the inner-loop DAE fitting. However the resulting gradient approximation might be very poor, which leads to unstable training and poorly fitted implicit distributions.

Remark (score matching methods)**.** As a side note, score matching can also be used to learn the parameters of an unnormalised density. In this case the target distribution q would be the data distribution and $\hat{q}$ is often a Boltzmann distribution with intractable partition function. As a parameter estimation technique, score matching is also related to contrastive divergence [Hinton, 2002], pseudo likelihood estimation [Hyvärinen, 2006], and denoising auto-encoders [Vincent, 2011]. Generalisations of score matching methods are also presented in [Lyu, 2009; Marlin et al., 2010].

6.2.4 Adding predictive power

Though providing potentially more accurate approximations, the non-parametric estimator (6.9) has no predictive power as described so far. Crucially, many tasks in machine learning require predicting gradient functions at samples drawn from distributions other than q, for example, in MLE $q(\boldsymbol{x})$ corresponds to the model distribution which is learned using samples from the data distribution instead. To address this issue, we derive two *predictive* estimators, one generalised from the non-parametric estimator and the other minimises KSD using parametric approximations.

Predictions using the non-parametric estimator. Let us consider an unseen datum $\mathbf{y}$. If $\mathbf{y}$ is sampled from q, then one can also apply the non-parametric estimator (6.9) for gradient approximation, given the observed data $\mathbf{X} = \{\boldsymbol{x}^1, ..., \boldsymbol{x}^K\} \sim q$. Concretely, if writing $\hat{\boldsymbol{g}}(\mathbf{y}) \approx \nabla_{\mathbf{y}} \log q(\mathbf{y}) \in \mathbb{R}^{d \times 1}$ then the non-parametric Stein gradient estimator computed on

$\mathbf{X} \cup \{\mathbf{y}\}$ is

$$\begin{bmatrix} \hat{\boldsymbol{g}}(\mathbf{y})^{\mathrm{T}} \\ \hat{\mathbf{G}} \end{bmatrix} = -(\mathbf{K}^* + \eta \boldsymbol{I})^{-1} \begin{bmatrix} \nabla_{\mathbf{y}} \mathcal{K}(\mathbf{y},\mathbf{y}) + \sum_{k=1}^{K} \nabla_{\boldsymbol{x}^k} \mathcal{K}(\mathbf{y},\boldsymbol{x}^k) \\ \langle \nabla, \mathbf{K} \rangle + \nabla_{\mathbf{y}} \mathcal{K}(\cdot,\mathbf{y}) \end{bmatrix}, \quad \mathbf{K}^* = \begin{bmatrix} \mathbf{K}_{\mathbf{yy}} & \mathbf{K}_{\mathbf{yX}} \\ \mathbf{K}_{\mathbf{Xy}} & \mathbf{K} \end{bmatrix},$$

with $\nabla_{\mathbf{y}} \mathcal{K}(\cdot,\mathbf{y})$ denoting a $K \times d$ matrix with rows $\nabla_{\mathbf{y}} \mathcal{K}(\boldsymbol{x}^k,\mathbf{y})$, and $\nabla_{\mathbf{y}} \mathcal{K}(\mathbf{y},\mathbf{y})$ only differentiates through the second argument. Thus by simple matrix calculations, we have:

$$\begin{aligned} \nabla_{\mathbf{y}} \log q(\mathbf{y})^{\mathrm{T}} \approx & -\left(\mathbf{K}_{\mathbf{yy}} + \eta - \mathbf{K}_{\mathbf{yX}} (\mathbf{K} + \eta \mathbf{I})^{-1} \mathbf{K}_{\mathbf{Xy}}\right)^{-1} \\ & \left(\nabla_{\mathbf{y}} \mathcal{K}(\mathbf{y},\mathbf{y}) + \sum_{k=1}^{K} \nabla_{\boldsymbol{x}^k} \mathcal{K}(\mathbf{y},\boldsymbol{x}^k) + \mathbf{K}_{\mathbf{yX}} \hat{\mathbf{G}}_V^{\mathrm{Stein}} - \mathbf{K}_{\mathbf{yX}} (\mathbf{K} + \eta \mathbf{I})^{-1} \nabla_{\mathbf{y}} \mathcal{K}(\cdot,\mathbf{y}) \right). \end{aligned} \tag{6.18}$$

For translation invariant kernels, typically $\nabla_{\mathbf{y}} \mathcal{K}(\mathbf{y},\mathbf{y}) = \mathbf{0}$, and more conveniently,

$$\nabla_{\boldsymbol{x}^k} \mathcal{K}(\mathbf{y},\boldsymbol{x}^k) = \nabla_{\boldsymbol{x}^k} (\boldsymbol{x}^k - \mathbf{y}) \nabla_{(\boldsymbol{x}^k - \mathbf{y})} \mathcal{K}(\boldsymbol{x}^k - \mathbf{y}) = -\nabla_{\mathbf{y}} \mathcal{K}(\boldsymbol{x}^k,\mathbf{y}).$$

Thus equation (6.18) can be further simplified to (with column vector $\mathbf{1} \in \mathbb{R}^{K \times 1}$)

$$\begin{aligned} \nabla_{\mathbf{y}} \log q(\mathbf{y})^{\mathrm{T}} \approx & -\left(\mathbf{K}_{\mathbf{yy}} + \eta - \mathbf{K}_{\mathbf{yX}} (\mathbf{K} + \eta \mathbf{I})^{-1} \mathbf{K}_{\mathbf{Xy}}\right)^{-1} \\ & \left(\mathbf{K}_{\mathbf{yX}} \hat{\mathbf{G}}_V^{\mathrm{Stein}} - \left(\mathbf{K}_{\mathbf{yX}} (\mathbf{K} + \eta \mathbf{I})^{-1} + \mathbf{1}^{\mathrm{T}} \right) \nabla_{\mathbf{y}} \mathcal{K}(\cdot,\mathbf{y}) \right). \end{aligned} \tag{6.19}$$

In practice one would store the computed gradient $\hat{\mathbf{G}}_V^{\mathrm{Stein}}$, the kernel matrix inverse $(\mathbf{K} + \eta \mathbf{I})^{-1}$ and η as the "parameters" of the predictive estimator. For a new observation $\mathbf{y} \sim p$ in general, one can "pretend" $\mathbf{y}$ is a sample from q and apply the above estimator as well. The approximation quality depends on the similarity between q and p, and we conjecture here that this similarity measure, if can be described, is closely related to the KSD.

Fitting a parametric estimator using KSD. The non-parametric predictive estimator could be computationally demanding. Setting aside the cost of fitting the "parameters", in prediction the time complexity for the non-parametric estimator is $\mathcal{O}(K^2 + Kd)$. Also storing the "parameters" needs $\mathcal{O}(Kd)$ memory for $\hat{\mathbf{G}}_V^{\mathrm{Stein}}$ and $(\mathbf{K} + \eta \mathbf{I})^{-1}$. These costs make the non-parametric estimator undesirable for high-dimensional data, since in order to obtain accurate predictions it often requires K scaling with d as well. To address this, one can also minimise the KSD using parametric approximations, in a similar way as to derive the score matching estimator in Section 6.2.3. More precisely, we define a parametric approximation in a similar fashion as (6.17), and in Appendix A.3 we show that if the RBF kernel is used for both the KSD and the parametric approximation, then the linear coefficients $\boldsymbol{a} = (a_1, ..., a_K)^{\mathrm{T}}$

can be calculated analytically: $\hat{\boldsymbol{a}}_V^{\text{Stein}} = (\boldsymbol{\Lambda} + \eta \mathbf{I})^{-1}\boldsymbol{b}$, where

$$\begin{aligned}\boldsymbol{\Lambda} =& \mathbb{X} \odot (\mathbf{KKK}) + \mathbf{K}(\mathbf{K} \odot \mathbb{X})\mathbf{K} - ((\mathbf{KK}) \odot \mathbb{X})\mathbf{K} - \mathbf{K}((\mathbf{KK}) \odot \mathbb{X}),\\ \boldsymbol{b} =& (\mathbf{K}\text{diag}(\mathbb{X})\mathbf{K} + (\mathbf{KK}) \odot \mathbb{X} - \mathbf{K}(\mathbf{K} \odot \mathbb{X}) - (\mathbf{K} \odot \mathbb{X})\mathbf{K})\mathbf{1},\end{aligned} \tag{6.20}$$

with $\odot$ denoting element-wise product, and $\mathbb{X}$ denoting the "gram matrix" that has elements $\mathbb{X}_{ij} = (\boldsymbol{x}^i)^{\mathrm{T}}\boldsymbol{x}^j$. Then for an unseen observation $\mathbf{y} \sim p$ the gradient approximation returns $\nabla_{\mathbf{y}} \log q(\mathbf{y}) \approx (\hat{\boldsymbol{a}}_V^{\text{Stein}})^{\mathrm{T}} \nabla_{\mathbf{y}} \mathcal{K}(\cdot, \mathbf{y})$. In this case one only maintains the linear coefficients $\hat{\boldsymbol{a}}_V^{\text{Stein}}$ and computes a linear combination in prediction, which takes $\mathcal{O}(K)$ memory and $\mathcal{O}(Kd)$ time and therefore is computationally cheaper than the non-parametric prediction model (6.18).

6.3 Applications

We present some case studies that apply the gradient estimators to implicit models. Detailed settings (architecture, learning rate, etc.) are presented in Appendix A.3. Implementation is released at https://github.com/YingzhenLi/SteinGrad.

6.3.1 Synthetic example: Hamiltonian flow with approximate gradients

We first consider a simple synthetic example to demonstrate the accuracy of the proposed gradient estimator. More precisely we consider the kernel induced Hamiltonian flow (*not* an exact sampler) [Strathmann et al., 2015] on a 2-dimensional banana-shaped object: $\boldsymbol{x} \sim \mathcal{B}(\boldsymbol{x}; b = 0.03, v = 100) \Leftrightarrow x_1 \sim \mathcal{N}(x_1; 0, v), x_2 = \varepsilon + b(x_1^2 - v), \varepsilon \sim \mathcal{N}(\varepsilon; 0, 1)$. The approximate Hamiltonian flow is constructed using the same operator as in Hamiltonian Monte Carlo (HMC) [Duane et al., 1987; Neal, 2011], except that the exact score function $\nabla_{\boldsymbol{x}} \log \mathcal{B}(\boldsymbol{x})$ is replaced by the approximate gradients. We still use the exact target density to compute the rejection step as we mainly focus on testing the accuracy of the gradient estimators. We test both versions of the predictive Stein gradient estimator (see Section 6.2.4) since we require the particles of parallel chains to be independent with each other. We fit the gradient estimators on $K = 200$ training datapoints from the target density. The bandwidth of the RBF kernel is computed by the median heuristic and scaled up by a scalar between $[1, 5]$. All three methods are simulated for $T = 2,000$ iterations, they share the same initial locations that are constructed by target distribution samples plus Gaussian noises of standard deviation 2.0, and the results are averaged over 200 parallel chains.

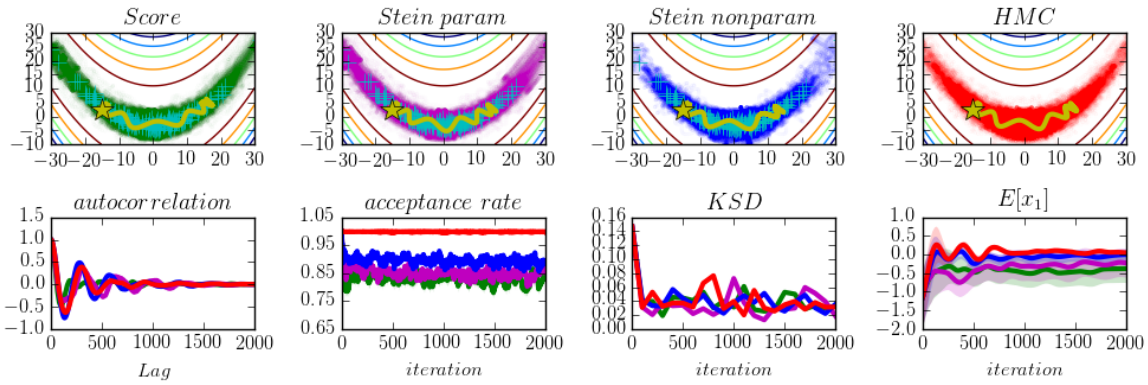


Fig. 6.2 Kernel induced Hamiltonian flow compared with HMC. Top: samples generated from the dynamics, training data (in cyan), an the trajectory of a particle for $T = 1$ to 200 starting at the star location (in yellow). Bottom: statistics computed during simulations. See main text for details.

We visualise the samples and some MCMC statistics in Figure 6.2. In general all the resulting Hamiltonian flows are HMC-like, which give us the confidence that the gradient estimators extrapolate reasonably well at unseen locations that are close to the training data. However all of these methods have trouble exploring the extremes, because at those locations there are very few or even no training data-points. Indeed we found it necessary to use large (but not too large) bandwidths, in order to both allow exploration of those extremes, and ensure that the corresponding test function is not too smooth. In terms of quantitative metrics, the acceptance rates are reasonably high for all the gradient estimators, and the KSD estimates (across chains) as a measure of sample quality are also close to that computed on HMC samples. The returned estimates of $\mathbb{E}[x_1]$ are close to zero which is the ground true value. We found that the non-parametric Stein gradient estimator is more sensitive to hyper-parameters of the dynamics, e.g. the stepsize of each HMC step. We believe a careful selection of the kernel (e.g. those with long tails) and a better search for the hyper-parameters (for both the kernel and the dynamics) can further improve the sample quality and the chain mixing time, but this is not investigated here.

6.3.2 Meta-learning of approximate posterior samplers for Bayesian NNs

One of the recent focuses on meta-learning has been on learning optimisers for training deep neural networks, e.g. see Andrychowicz et al. [2016]; Li and Malik [2016]. Could analogous goals be achieved for approximate inference? In this section we attempt to learn an approximate posterior sampler for Bayesian neural networks (Bayesian NNs, BNNs) that

generalises to *unseen* datasets and architectures, and we refer to Section 5.4 for a motivation of this approach.

In a nutshell, we consider a binary classification task: $p(y=1|\boldsymbol{x},\boldsymbol{\theta}) = \text{sigmoid}(\text{NN}_{\boldsymbol{\theta}}(\boldsymbol{x}))$, $p_0(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta};\mathbf{0},\mathbf{I})$. After observing the training data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, we first obtain the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D}) \propto p_0(\boldsymbol{\theta})\prod_{n=1}^N p(y_n|\boldsymbol{x}_n,\boldsymbol{\theta})$, then approximate the predictive distribution for a new observation as $p(y^*=1|\boldsymbol{x}^*,\mathcal{D}) \approx \frac{1}{K}\sum_{k=1}^K p(y^*=1|\boldsymbol{x}^*,\boldsymbol{\theta}^k), \boldsymbol{\theta}^k \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$. In this task we define an implicit approximate posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ as the following *stochastic* RNN $\boldsymbol{\theta}_{t+1} = \mathbf{f}(\boldsymbol{\theta}_t, \nabla_t, \boldsymbol{\varepsilon}_t)$: given the current location $\boldsymbol{\theta}_t$ and the mini-batch data $\{(\boldsymbol{x}_m, y_m)\}_{m=1}^M$, the update for the next step is

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \zeta\Delta_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t,\nabla_t) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t,\nabla_t)\odot\boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{\varepsilon};\mathbf{0},\mathbf{I}), \\ \nabla_t &= \nabla_{\boldsymbol{\theta}_t}\left[\frac{N}{M}\sum_{m=1}^M \log p(y_m|\boldsymbol{x}_m,\boldsymbol{\theta}_t) + \log p_0(\boldsymbol{\theta}_t)\right]. \end{aligned} \tag{6.21}$$

The coordinates of the noise standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t,\nabla_t)$ and the moving direction $\Delta_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t,\nabla_t)$ are parametrised by a *coordinate-wise* neural network, i.e.

$$\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t,\nabla_t) = [\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t(1),\nabla_t(1)),...,\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t(d),\nabla_t(d))]^{\text{T}}$$

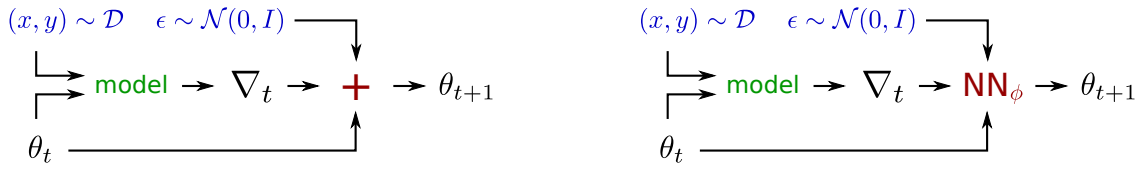
with $\boldsymbol{\theta}_t(i)$ denoting the i^{th} dimension of vector $\boldsymbol{\theta}_t$ (similarly for $\nabla_t(i)$ and $\Delta_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t,\nabla_t)$). If properly trained, this neural network will learn the best combination of the current location and gradient information, and produce approximate posterior samples efficiently on different probabilistic modelling tasks.

We propose using the variational inference objective (6.2) computed on the samples $\{\boldsymbol{\theta}_t^k\}$ to learn the variational parameters $\boldsymbol{\phi}$. More specifically, we simulate the approximate sampler for $T = 10$ transitions and sum over the variational lower-bounds computed on the samples of every step. This gives the maximisation objective

$$\mathcal{L}(\boldsymbol{\phi}) = \sum_{t=1}^T \mathcal{L}_{\text{VI}}(q_t),$$

with $q_t(\boldsymbol{\theta})$ as the marginal distribution of $\boldsymbol{\theta}_t$ (therefore depends on $\boldsymbol{\phi}$). In practice the variational lower-bound $\mathcal{L}_{\text{VI}}(q_t)$ is further approximated by Monte Carlo and data subsampling:

$$\mathcal{L}_{\text{VI}}(q_t) \approx \frac{N}{KM}\sum_{m=1}^M\sum_{k=1}^K \log p(y_m|\boldsymbol{x}_m,\boldsymbol{\theta}_t^k) + \log p_0(\boldsymbol{\theta}_t^k) - \log q_t(\boldsymbol{\theta}_t^k).$$



(a) SGLD

(b) NN-based approx. sampler

Fig. 6.3 Comparing the computation graphs of the two samplers. SGLD can be viewed as stochastic gradient descent plus properly scaled Gaussian noise. Instead of using the "plus" operation, the NN-based sampler combine the three inputs with a neural network, and the parameters $\boldsymbol{\phi}$ are then trained by the method described in the main text.

Since in this case the gradient of the log joint distribution can be computed analytically, we only approximate the gradient of the entropy term $\mathbb{H}[q]$ as in (6.3), with the exact score function replaced by the presented gradient estimators.

We briefly describe the test protocol. We take from the UCI repository [Lichman, 2013] six binary classification datasets (australian, breast, crabs, ionosphere, pima, sonar), train an approximate sampler on crabs with a small neural network that has one 20-unit hidden layer with *ReLU* activation, and generalise to the remaining datasets with a bigger network that has 50 hidden units and uses *sigmoid* activation. We use ionosphere as the validation set to tune ζ. The remaining 4 datasets are further split into 40% training subset for simulating samples from the approximate sampler, and 60% test subsets for evaluating the sampler's performance. Besides the gradient estimators we also compare with two baselines: an approximate posterior sampler trained by maximum a posteriori (MAP), and stochastic gradient Langevin dynamics (SGLD) [Welling and Teh, 2011] evaluated on the test datasets directly. A comparison between SGLD and the proposed neural network based approximate sampler is visualised in Figure 6.3.

For architecture details, we use a one hidden layer neural network with 20 hidden units to compute the noise standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t, \nabla_t)$ and the moving direction $\Delta_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t, \nabla_t)$ of the next update. Softplus non-linearity is used for the hidden layer and to compute the noise variance we apply ReLU activation to ensure non-negativity. The step-size ζ is selected as 10^{-5} which is tuned on the KDE approach. For SGLD step-size 10^{-5} also returns overall good results.[1] We report the results using the non-parametric Stein gradient estimator as we found it works better than the parametric version. The RBF kernel is applied for

[1] We note here that the results can be further improved with carefully tuned learning rate for both SGLD and the NN-based samplers, but here we are mainly interested in the same step-size set-up in order to compare the velocity of the particles defined by the underlying dynamics of the samplers.

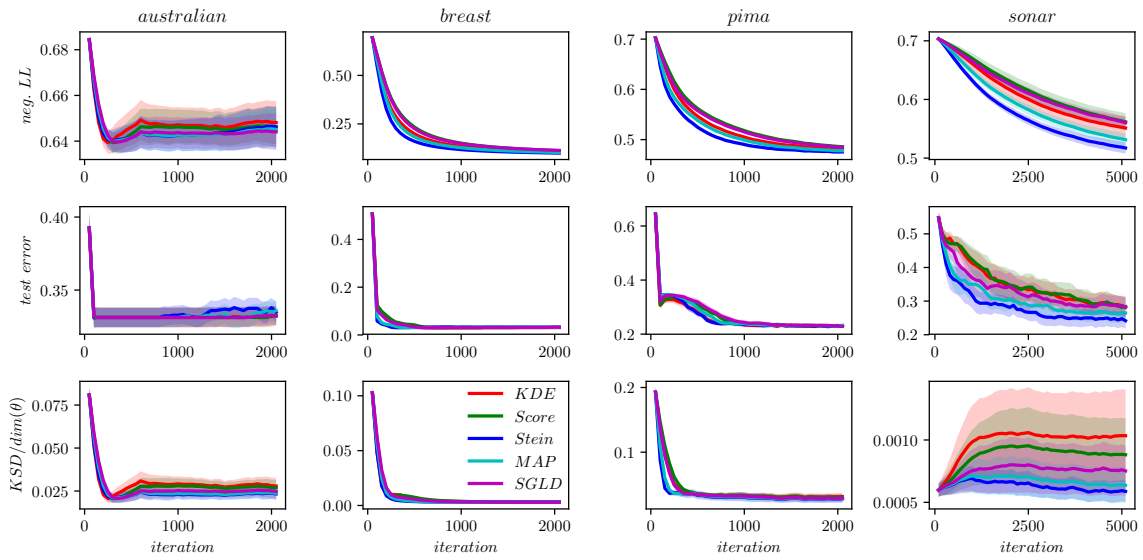


Fig. 6.4 Generalisation performances for trained approximate posterior samplers.

gradient estimation, with the hyper-parameters determined by a grid search on the bandwidth $\sigma^2 \in \{0.25, 1.0, 4.0, 10.0, \text{median trick}\}$ and $\eta \in \{0.1, 0.5, 1.0, 2.0\}$.

Figure 6.4 presents the (negative) test log-likelihood (LL), classification error, and an estimate of the KSD U-statistic $\mathcal{S}_U^2(p(\boldsymbol{\theta}|\mathcal{D}), q(\boldsymbol{\theta}))$ (with data sub-sampling) over 5 splits of each test dataset.[2] The Stein approach performs equally well or a little better than SGLD in terms of test-LL and test error. The KDE method is slightly worse and is close to MAP, indicating that the KDE estimator does not provide a very informative gradient for the entropy term. The score matching estimator method produces the worst results among the trained samplers even after carefully tuning the bandwidth and the regularisation parameter η, although the difference is not significant. Future work should investigate the usage of advanced recurrent neural networks such as an LSTM [Hochreiter and Schmidhuber, 1997], which is expected to return better performance.

6.3.3 Towards addressing mode collapse in GANs using entropy regularisation

GANs are notoriously difficult to train in practice. Besides the instability of gradient-based minimax optimisation which has been partially addressed by many recent proposals [Arjovsky et al., 2017; Berthelot et al., 2017; Salimans et al., 2016], they also suffer from mode collapse. We propose adding an entropy regulariser to the GAN generator loss. Concretely, assume the

[2] After conference publication (see publication page) I fixed a few issues of the previous implementation (mainly for the score matching estimator) and re-run the experiments, which is reported in this thesis.

generative model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is implicitly defined by $\boldsymbol{x} = \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{z} \sim p_0(\boldsymbol{z})$, then the generator's loss is defined by

$$\tilde{\mathcal{J}}_{\text{gen}}(\boldsymbol{\theta}) = \mathcal{J}_{\text{gen}}(\boldsymbol{\theta}) - \alpha \mathbb{H}[p_{\boldsymbol{\theta}}(\boldsymbol{x})], \tag{6.22}$$

where $\mathcal{J}_{\text{gen}}(\boldsymbol{\theta})$ is the original loss function for the generator from any GAN algorithm and α is a hyper-parameter. In practice the gradient of (6.22) is estimated using Monte Carlo.

As an illustrating example, in the following we consider the very recently proposed boundary equilibrium GAN (BEGAN) [Berthelot et al., 2017] approach. In BEGAN the discriminator is defined as an auto-encoder $D_{\boldsymbol{\varphi}}(\boldsymbol{x})$ that reconstructs the input $\boldsymbol{x}$. After selecting a ratio parameter $\gamma > 0$, a control rate β_0 initialised at 0, and a "learning rate" $\lambda > 0$ for the control rate, the loss functions for the generator $\boldsymbol{x} = \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{z} \sim p_0(\boldsymbol{z})$ and the discriminator are:

$$\begin{aligned}
&\mathcal{J}(\boldsymbol{x}) = ||D_{\boldsymbol{\varphi}}(\boldsymbol{x}) - \boldsymbol{x}||, \quad ||\cdot|| = ||\cdot||_2^2 \text{ or } ||\cdot||_1, \\
&\mathcal{J}_{\text{gen}}(\boldsymbol{\theta}; \boldsymbol{\varphi}) = \mathcal{J}(\mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{z})), \quad \boldsymbol{z} \sim p_0(\boldsymbol{z}) \\
&\mathcal{J}_{\text{dis}}(\boldsymbol{\varphi}; \boldsymbol{\theta}) = \mathcal{J}(\boldsymbol{x}) - \beta_t \mathcal{J}_{\text{gen}}(\boldsymbol{\theta}; \boldsymbol{\varphi}), \quad \boldsymbol{x} \sim \mathcal{D} \\
&\beta_{t+1} = \beta_t + \lambda(\gamma \mathcal{J}(\boldsymbol{x}) - \mathcal{J}(\mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{z}))).
\end{aligned} \tag{6.23}$$

The main idea behind BEGAN is that, as the reconstruction loss $\mathcal{J}(\cdot)$ is approximately Gaussian distributed, with $\gamma = 1$ the discriminator loss $\mathcal{J}_{\text{dis}}$ is (approximately) proportional to the Wasserstein distance between loss distributions induced by the data distribution $p_{\mathcal{D}}(\boldsymbol{x})$ and the generator $p(\boldsymbol{x})$. In practice it is beneficial to maintain the equilibrium $\gamma \mathbb{E}_{p_{\mathcal{D}}}[\mathcal{J}(\boldsymbol{x})] = \mathbb{E}_p[\mathcal{J}(\boldsymbol{x})]$ through the optimisation procedure described in (6.23) that is motivated by proportional control theory. This approach effectively stabilises training, however it suffers from catastrophic mode collapse problem (see the left most panel in Figure 6.5).

We empirically investigate the entropy regularisation idea on BEGAN using (continuous) MNIST. As described before, we simply subtract an entropy term from the generator's loss function, i.e. with K samples $\boldsymbol{z}^1, ..., \boldsymbol{z}^k \sim p_0(\boldsymbol{z})$,

$$\tilde{\mathcal{J}}_{\text{gen}}(\boldsymbol{\theta}; \boldsymbol{\varphi}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{J}(\mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{z}^k)) + \alpha \frac{1}{K} \sum_{k=1}^{K} \log p(\mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{z}^k)), \tag{6.24}$$

where the rest of the optimisation objectives remains as in (6.23). This procedure would maintain the equilibrium $\gamma \mathbb{E}_{p_{\mathcal{D}}}[\mathcal{J}(\boldsymbol{x})] = \mathbb{E}_p[\mathcal{J}(\boldsymbol{x})] - \alpha \mathbb{H}[p]$. We approximate the gradient $\nabla_{\boldsymbol{\theta}} \mathbb{H}[p]$ using the estimators presented above. For the purpose of updating the control rate β_t two strategies are considered to approximate the contribution of the entropy term. The first proposal considers a plug-in estimate of the entropy term with a KDE estimate of $p(\boldsymbol{x})$, which is consistent with the KDE estimator but not necessary with the other two (as they use kernels

when representing $\log p(\boldsymbol{x})$ or $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$). The second one uses a proxy of the entropy loss $-\tilde{\mathbb{H}}[p] = \frac{1}{K}\sum_{k=1}^{K} \nabla_{\boldsymbol{x}^k} \log \hat{p}(\boldsymbol{x}^k)^{\mathrm{T}} \boldsymbol{x}^k$ with $\nabla_{\boldsymbol{x}^k} \log \hat{p}(\boldsymbol{x}^k)$ is the approximate gradient obtained by the estimators. It is the surrogate loss used to (approximately) compute $\nabla_{\boldsymbol{\phi}} \mathbb{H}[p]$:

$$\nabla_{\boldsymbol{\phi}} \mathbb{H}[p] \approx \nabla_{\boldsymbol{\phi}} \tilde{\mathbb{H}}[p] = \frac{1}{K} \sum_{k=1}^{K} \nabla_{\boldsymbol{x}^k} \log \hat{p}(\boldsymbol{x}^k)^{\mathrm{T}} \nabla_{\boldsymbol{\phi}} \boldsymbol{x}^k.$$

We compare the non-parametric V-statistic Stein gradient estimator to both KDE and score matching estimators. We use a convolutional generative network and a convolutional auto-encoder and select the hyper-parameters of BEGAN $\gamma \in \{0.3, 0.5, 0.7\}$, $\alpha \in [0,1]$ and $\lambda = 0.001$. The Epanechnikov kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') := \frac{1}{d}\sum_{j=1}^{d}(1-(x_j - x'_j)^2)$ is used as the pixel values lie in a unit interval (see Appendix A.3 for the expression of the score matching estimator), and to ensure the boundary condition we clip the pixel values into range $[10^{-8}, 1-10^{-8}]$. Readers are referred to Appendix A.3 for a detailed experimental set-up.

The generated images are visualised in Figure 6.5. BEGAN without entropy regularisation fails to generate diverse samples even when trained with learning rate decay. The other three images clearly demonstrate the benefit of the entropy regularisation technique, with the Stein approach obtaining the highest diversity without compromising visual quality.

We further consider four metrics to assess the trained models quantitatively. First 500 samples are generated for each trained model, then we compute their nearest neighbours in the training set using ℓ_1 distance, and obtain a probability vector $\mathbf{p}$ by averaging over these neighbour images' label vectors. In Figure 6.6 we depict the entropy of $\mathbf{p}$ (top left), averaged ℓ_1 distances to the nearest neighbour (top right), and the difference between the largest and smallest elements in $\mathbf{p}$ (bottom right). The error bars are obtained by 5 independent runs. These results demonstrate that the Stein approach performs significantly better than the other two, in that it learns a better generative model not only faster but also in a more stable way. Interestingly the KDE approach achieves the lowest average ℓ_1 distance to nearest neighbours, possibly because it tends to memorise training examples. We next train a fully connected network $\pi(\mathbf{y}|\boldsymbol{x})$ on MNIST that achieves 98.16% text accuracy, and compute on the generated images an empirical estimate of the inception score [Salimans et al., 2016] $\mathbb{E}_{p(\boldsymbol{x})}[\mathrm{KL}[\pi(\mathbf{y}|\boldsymbol{x})||\pi(\mathbf{y})]]$ with $\pi(\mathbf{y}) = \mathbb{E}_{p(\boldsymbol{x})}[\pi(\mathbf{y}|\boldsymbol{x})]$ (bottom left panel). High inception score indicates that the generate images tend to be both realistic looking and diverse, and again the Stein approach out-performs the others on this metric by a large margin.

Concerning computation speed, all the three methods are of the same order: 10.20s/epoch for KDE, 10.85s/epoch for Score, and 10.30s/epoch for Stein.[3] This is because $K < d$ (in the experiments $K = 100$ and $d = 784$) so that the complexity terms are dominated by

[3] All the methods are timed on a machine with an NVIDIA GeForce GTX TITAN X GPU.

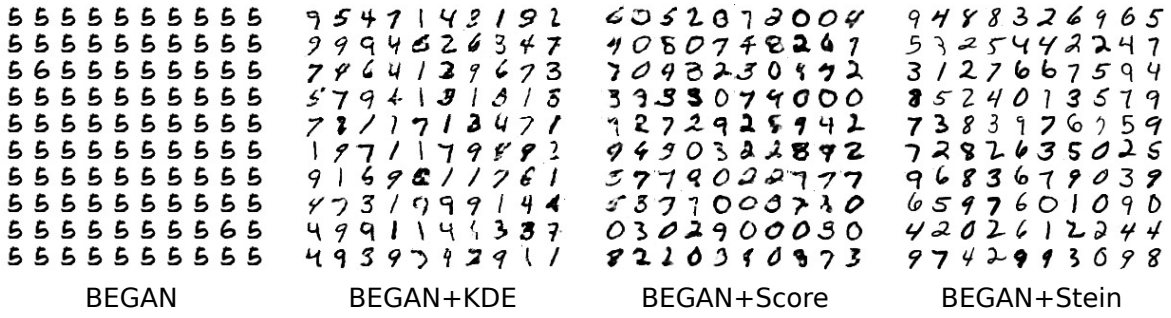


Fig. 6.5 Visualisation of generated images from trained BEGAN models.

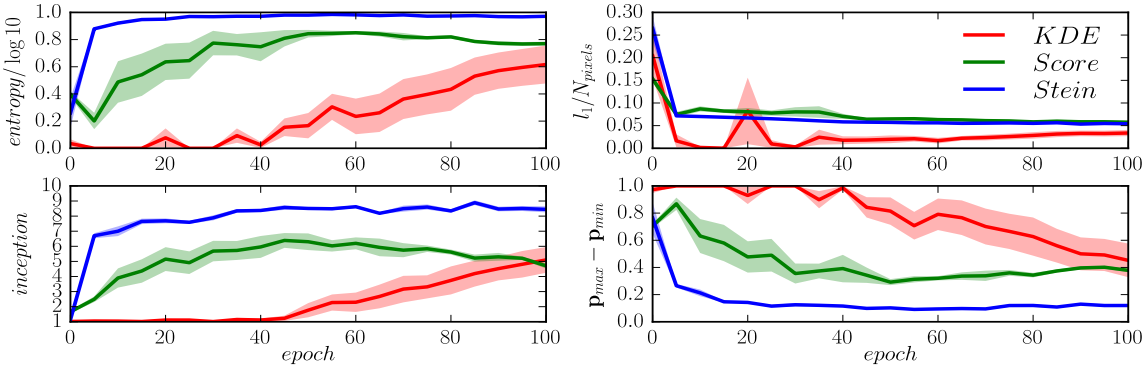


Fig. 6.6 Quantitative evaluation on entropy regularised BEGAN. The higher the better for the LHS panels and the other way around for the RHS ones. See main text for details.

kernel computations ($\mathcal{O}(K^2d)$) required by all the three methods. Also for a comparison, the original BEGAN method without entropy regularisation runs for 9.05s/epoch. Therefore the main computation cost is dominated by the optimisation of the discriminator/generator, and the proposed entropy regularisation can be applied to many GAN frameworks with little computational burden.

6.4 Summary

We have presented the Stein gradient estimator as a novel generalisation to the score matching gradient estimator. With a focus on learning implicit models, we have empirically demonstrated the efficacy of the proposed estimator by showing state-of-the-art results on three canonical learning tasks: approximating gradient-free MCMC, meta-learning for approximate posterior samplers, and unsupervised learning for image generation. Future work will expand the understanding of gradient estimators in both theoretical and practical sides. Theoretical development will compare both the V-statistic and U-statistic Stein gradient estimators. Practical work will improve the sample efficiency of kernel estimators in high dimensions and develop fast yet accurate approximations to the matrix inversion part. Finally

follow-up work will study the generalisation of the Stein gradient estimator to non-kernel settings and discrete distributions.

Here comes the end of the second theme of the thesis, where we have discussed principles of approximate inference algorithm design (Chapter 5) and presented one concrete example (Chapter 6) on learning wild approximations. Readers might have noticed that wild approximate inference is still a very new research direction: many related papers in citation are freshly baked within a year of this thesis submission, and the proposals in development are inspired by recent success in machine learning and deep learning. Thus both theoretical analyses and extensive comparisons to traditional approaches are much in need, which will then help identify the ideal application scenarios and possibly potential pitfalls of the method.

The presented two themes have substantial differences. In the next chapter, which concludes the thesis, I will summarise the contributions of the thesis to the approximate inference community, and discuss the connections and comparisons between the two themes. Expositions of important research questions will also be provided, and hopefully this will serve as a principled guide for future development of approximate inference algorithms.

Chapter 7

Conclusions and Future Work

Approximate inference is a huge subject, which studies both the *structure of the approximate distribution*, and the *optimisation algorithms* used to fit them. The thesis has mainly investigated the latter part, presenting a number of new approximate inference algorithms and their applications to Bayesian deep learning tasks. A summary of the contributions is provided in Section 7.1, where I will also discuss comparisons and connections between the two thesis themes. Much work is still required to enable generic q distributions to be fitted accurately and efficiently, and in Section 7.2 I will briefly provide my perspective on this matter, and raise several important questions to be answered in the future.

7.1 More discussions on the two themes

Although the thesis mainly discussed new optimisation algorithms for approximate inference, the material was organised into two themes.

I Unifying variational methods. (Chapters 2, 3, 4)
In this part of the thesis, we studied two main classes of variational algorithms, namely expectation propagation (EP) and variational inference (VI). The main contributions are the two novel frameworks: stochastic EP (Chapter 3) and Rényi divergence VI (Chapter 4). SEP significantly improves the memory efficiency of the EP-like algorithms, making them scalable to datasets comprising millions of instances. Extensions to the distributed SEP case further provide a highly flexible framework that unifies global and local approximation algorithms from an algorithmic perspective. Next we introduced the VR bound framework that enables the deployment of Monte Carlo methods for α-divergence methods, provides both upper- and lower-bounds to the marginal likelihood, and again connects global and local approaches, but from an energy point of view.

II Wild approximate inference. (Chapters 5, 6)

This theme presented a new research direction for approximate inference, with the goal of allowing *arbitrarily* complex approximate posteriors to be deployed. In Chapter 5, I argued that the density evaluation requirement should be removed when designing Monte Carlo based approximate inference algorithms. I showed that this allows the use of very flexible q distributions, and presented several examples of them. Later I wrote a generic guide for these designs, containing four different algorithmic options. One of the schemes (direct gradient approximation) was further developed in Chapter 6, in which an initial experiment on meta-learning for approximate inference was presented.

The two themes were motivated in very different ways. The study of the unified frameworks aimed at developing new algorithms that both advance the understanding of existing variational methods, and adapt well to the application areas for Bayesian deep learning. All the discussions and comparisons in that part were based on one assumption: the candidate approximate distribution family $\mathcal{Q}$ is pre-defined, such as mean-field Gaussians. Indeed in the empirical studies we used implementations of almost the same set-up, except that we tuned a few of the algorithmic settings such as the α values and the number of MC samples K.

Wild approximate inference, on the other hand, encourages the use of very complex approximate distributions. There is no reason to expect an accurate approximation with a distribution that factorises in general, even when it is fitted using algorithms with elegant theoretical properties (like VI/EP studied in the first theme). However, instead of providing recipes for complex approximate distribution designs (see some recent examples in Appendix B) which fit the existing algorithms, we made a bold move to developing algorithms that enable approximations of arbitrary form. By doing this, we allow the users to focus on the *statistical properties* that they want to plant into the approximations, rather than the *algorithmic tractability* aspects that restrict the flexibility of the q distribution.

Considering hyper-parameter optimisation, there is another notable difference between the two themes. For the first theme, we designed energy functions to allow training both the hyper-parameters and the q distribution, where as for the second theme, we decoupled the algorithms for inference and learning.[1] In this way, wild approximate inference is more in line with numerical integration methods like quadrature and Monte Carlo: all of them treat (approximate) Bayesian inference as a *computational* task, and leave the learning procedure to hyper-parameter optimisation and potentially model selection. However, when compared to the VAE approach, it becomes harder to understand the interaction between the approximate inference results and the hyper-parameter updates. In this regard, the VR

[1] Although for example the approximate maximum likelihood procedure does not require the inference optimisation to converge.

bound framework might be preferred, as both procedures are coupled, and can be analysed by examining the single loss function in use. We will continue the discussion on this matter in the next section.

Nevertheless, there is one important technique that has repeatedly been discussed in both themes: the Monte Carlo approximation method. It has been crucial to the recent success of Bayesian deep learning [Blundell et al., 2015; Gal, 2016; Graves, 2011; Kingma and Welling, 2014; Li et al., 2017; Rezende et al., 2014], as for deep neural networks most of the quantities that a Bayesian would like to compute are analytically intractable, even when using approximate posterior distributions with simple forms. In this thesis, except for SEP, the MC approximation is applied to all the other algorithms when performing empirical evaluations. Even SEP is compatible with Monte Carlo methods: when the moment matching step has no analytical solution, Monte Carlo methods can be deployed to estimate the sufficient statistics of the tilted distribution that are required for the update [Barthelmé and Chopin, 2011; Gelman et al., 2014].

7.2 Open problems

Arguably, approximate inference is the key subject for research in Bayesian deep learning and large-scale Bayesian modelling in general. Therefore it is important to identify the challenges and open questions for future investigation. The final section of the thesis is organised in three parts. In the first two subsections, I discuss open research problems for the two themes. Then in the end, I ask the final question on the relationship between approximate inference and Bayesian modelling.

7.2.1 Research challenges for EP-like methods

Is SEP guaranteed to converge?

One can study the energy functions and prove that Rényi divergence VI and BB-α methods are guaranteed to converge under mild conditions. However, since the development in 2015, little progress has been made to find the energy (or Lyapunov) function of SEP. One important research direction is to identify the existence of this energy function. On the other hand, Dehaene and Barthelmé [2015]; Dehaene and Barthelmé [2018] used the AEP (the batch version of SEP) to study the convergence of EP without looking at an energy function. Although having assumed some restrictive conditions for simplicity, the proof ideas in Dehaene and Barthelmé [2015]; Dehaene and Barthelmé [2018] can still be useful for future research.

More theoretical supports on selecting the α-divergence?

We have conjectured the behaviour of the α-divergence methods using both toy examples (see Section 4.2.1) and empirical results. However, there exists no formal theory so far which support (or disprove) these hypotheses. Furthermore, most of these intuitions apply to uni-modal approximations, and it is still largely unknown how the approximation methods perform when multi-modal distributions are fitted using α-divergence methods. Perhaps proving theoretical results for the general set-up might be hard, so a sensible plan would first propose theorems on toy problems, and also design (counter) examples that (dis)agree with the intuitions provided in this thesis. Also algorithmic design for automatic divergence selection would be much welcomed, in this regard, one can use Bayesian optimisation methods [Močkus, 1975; Snoek et al., 2012], or design an objective function to learn the α values [Dikmen et al., 2015].

Generalisation results for future observations?

In many applications of Bayesian deep learning, the predictive performances are mainly the focus of the model evaluation. Variational inference might be preferred in this regard, as under some assumptions, the PAC-Bayes framework provides generalisation bounds on future observations (coming from the same underlying distribution as the training data) that are directly related to the variational free-energy [Germain et al., 2016; McAllester, 1998]. Preliminary results showed that some generalisation bounds of other forms could also be proved using Rényi divergences [Bégin et al., 2016], however it is unclear how this is related to the α-divergence framework studied in this thesis. Future research on this challenge would also advance the understanding of α-divergence methods, therefore helping address the above question as well.

7.2.2 Open challenges for wild approximate inference

How do we design the approximate posterior?

A good design for the approximate posterior is still very much in need, even when the wild approximate inference framework allows more flexible q distributions to be fitted by relaxing the algorithmic tractability constraints. For example, the density ratio estimation method discussed in Section 5.3 requires a rather significant number of MC samples, which can be very expensive when applied to mini-batch optimisations for latent variable models. One very interesting solution to this issue is by sharing the randomness across the inference procedure associated with different latent variables [Mescheder et al., 2017]. Another important concern considers "better compression of posterior correlation", i.e. how we can design the approximation to capture the correlation between variables with the least amount

of computation and memory. This can be especially useful for Bayesian neural networks, and having the structural information of the posterior can significantly improve both predictive performance and uncertainty calibration.

Variance reduction/control variate methods for wild approximate inference?

With Monte Carlo approximations, the stochastic gradients used to update the q distributions can be very noisy. Efforts have been made recently to reduce the variance of the gradient signal for the MC-VI method, including control variate approaches, see Gu et al. [2016]; Mnih and Gregor [2014]; Paisley et al. [2012]. We note that variance reduction has also been an important research direction for both reinforcement learning [Greensmith et al., 2004; Gu et al., 2017] and stochastic optimisation [Defazio et al., 2014; Johnson and Zhang, 2013; Le Roux et al., 2012].[2] It might be helpful to borrow ideas from those fields to develop control variate methods for wild approximate inference.

Wild approximate inference methods for discrete latent variables?

One of the imperfections of this thesis is that we did not discuss approximating posterior distributions for discrete variables. In this case the path derivative (i.e. differentiation through an MC sample) is not available, and traditional VI has resorted to the REINFORCE gradient [Williams, 1992], potentially with control variate methods [Gu et al., 2016; Mnih and Gregor, 2014; Mnih and Rezende, 2016; Titsias and Lázaro-Gredilla, 2015]. However, many of these methods still rely on a tractable q density (or at least the tractability of the score function), which is not applicable in the wild approximate inference set-up. Some ad-hoc solutions include 1) a continuous relaxation of the discrete density, such as the Concrete distribution [Jang et al., 2017; Maddison et al., 2017b] which has some successful applications [Gal et al., 2017; Kusner and Hernández-Lobato, 2016]; 2) weak derivatives or Stein discrepancies defined on discrete distributions, e.g. see Ranganath et al. [2016a].

Principled approaches towards meta-learning for approximate inference?

We briefly discussed the research direction of meta-learning for approximate inference algorithms in Section 5.4, with an initial attempt also presented in Section 6.3.2. Another interesting approach would consider learning *variational objectives* for fitting the posterior distributions, i.e. the algorithm is defined by

$$\mathcal{A}_{\boldsymbol{\phi}}(p_n(\boldsymbol{z}_n)) = \arg\max_{q \in Q} \mathcal{L}_{\boldsymbol{\phi}}(p_n(\boldsymbol{z}_n), q(\boldsymbol{z}_n)), \tag{7.1}$$

[2] Three NIPS 2016 tutorials – on VI, deep RL and stochastic optimisation, respectively – had spent a considerate amount of time discussing variance reduction.

and the parameters $\boldsymbol{\phi}$ are learned by optimising a meta-objective $\mathcal{L}_{\text{meta}}$ on $\mathcal{A}_{f_{\boldsymbol{\phi}}}(p_n(\boldsymbol{z}_n))$ for all $n = 1,...,N$. However there exists no principled guide for the construction of such meta-objective in order to learn a superior algorithm. It might be useful to consider some desirable properties that one would like to incorporate. For example we might want: 1) reduced bias when $\mathcal{L}_{\boldsymbol{\phi}}$ is used as a surrogate for maximum likelihood, see Turner and Sahani [2011]; 2) reduced variance when Monte Carlo is deployed in the algorithm $\mathcal{A}_{\boldsymbol{\phi}}$. The former property can be achieved by learning a lower-bound $\mathcal{L}_{\boldsymbol{\phi}}$ to the log marginal likelihood that is (approximately) equally tight everywhere (c.f. Section 4.2.1), and for the latter, one might consider learning transferable control variates across different densities.

7.2.3 Approximate inference as computation, or modelling?

Conceptually, Bayesian inference is easy: one just needs to specify a model (including a prior distribution and a likelihood function), and later use a posterior distribution to study the unknown factors and quantify uncertainty given the observations. In particular, as a natural result of Bayes' rule, the posterior distribution is completely determined by the model and the observations. Therefore traditionally, exact posterior evaluation has been regarded as a *computational* task: no more assumptions are required, and the only thing left is to proceed the computation of Bayes' rule. Model selection can also be performed using the Bayes factor for example, or even frequentist style hypothesis tests on the "predictive model" $p(\boldsymbol{x}^*|\mathcal{D}) = \int p(\boldsymbol{x}^*|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$. Both of them can be computed easily if assuming exact inference is available.

Practically, however, Bayesian inference is hard: for most useful models in the real world, it is impractical to compute the exact posterior unless we have an unlimited amount of computational resources. Fortunately, we now have many efficient approximate inference algorithms, some of them approximate the computation of the posterior, and some of them directly approximate the predictive distribution. All of them are motivated as providing an accurate approximation to the Bayesian inference procedure, therefore at first glance, they should follow the philosophy of exact inference and be treated as a computation task as well. From this perspective, researchers are encouraged to develop new algorithms that focus on improving the approximation accuracy and running speed, and discovering better designs of the q distribution to reduce the biases further.

On the other hand, the Bayesian decision process relies on the inference results which can only be performed approximately in practice. Critically, as long as we are proposing approximations, we are making further assumptions about the inference procedure and thus to the whole (approximate) decision making process! In this sense, we should also add the inference procedure to the decision, or *predictive modelling* procedure. Here is a simple

example in favour of this claim: consider a model whose posterior distribution is a truncated Gaussian. Then VI with a Gaussian approximation returns terrible approximations (because the zero-forcing property of VI would force the q distribution to be a Dirac delta function), and based on this approximate inference result, the later model checking process can reject this under-performing "approximate Bayesian prediction model" even when the original model with exact inference is well behaved. We can potentially identify this false rejection of the model (in the usual Bayesian sense) by changing the approximation algorithm to EP, or the q distribution to allow a non-smooth density. Indeed this view has also been discussed in e.g. Dawid [1984]; Gelman et al. [1996]; Gelman and Shalizi [2013]; Meng [1994] which adds a bit of frequentist flavour to the predictive model selection procedure.

What is the take home message from this interpretation? Besides the encouragement of developing better approaches to improve the inference procedure (which is the same call as from the perspective of computation), an interesting direction is to understand the bias of existing inference methods. With a better understanding of how these biases might interact with downstream decision making tasks, we can then carefully select the best inference method to achieves a desirable "predictive model". For example, VI is known for penalising complicated models (if the prior prefers the simple ones) [Hinton and Van Camp, 1993]. Therefore, we can design specific types of the approximate posterior (e.g. Gaussian dropout [Kingma et al., 2015; Srivastava et al., 2014]) to further sparsify Bayesian neural networks without compromising the predictive accuracy [Louizos et al., 2017; Molchanov et al., 2017].

Finally, just to be clear: both views of approximate inference are useful for discovering new research directions that are generally beneficial to enrich the literature. For example, they can be used to propose criteria for evaluating an approximate inference procedure, and derive new principles for algorithmic design. I hope that, by investigating this problem, the community can draw inspiration from both views, and obtain a better understanding of the roles of approximate inference for modern Bayesian analysis.

References

Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. (2016). YouTube-8M: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*.

Ahn, S., Korattikara, A., and Welling, M. (2012). Bayesian posterior sampling via stochastic gradient fisher scoring. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1771–1778.

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.

Altosaar, J., Ranganath, R., and Blei, D. M. (2017). Proximity variational inference. *arXiv preprint arXiv:1705.08931*.

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. and Nagaoka, H. (2000). *Methods of information geometry*. Oxford University Press.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989.

Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*.

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.

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.

Bamler, R., Zhang, C., Opper, M., and Mandt, S. (2017). Perturbative black box variational inference. In *Advances in Neural Information Processing Systems*, pages 5086–5094.

Barber, D. and Agakov, F. (2003). The IM algorithm: a variational approach to information maximization. In *Advances in Neural Information Processing Systems*, pages 201–208.

Barber, D. and Bishop, C. M. (1998). Ensemble learning in Bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238.

Barbour, A. D. (1988). Stein's method and Poisson process convergence. *Journal of Applied Probability*, pages 175–184.

Barthelmé, S. and Chopin, N. (2011). ABC-EP: Expectation propagation for likelihood-free Bayesian computation. In *Proceedings of the 28th International Conference on Machine Learning*, pages 289–296.

Barthelmé, S. and Chopin, N. (2014). Expectation propagation for likelihood-free inference. *Journal of the American Statistical Association*, 109(505):315–333.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

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.

Beaumont, M. A., Zhang, W., and Balding, D. J. (2002). Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035.

Beck, A. and Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.

Bégin, L., Germain, P., Laviolette, F., and Roy, J.-F. (2016). PAC-Bayesian bounds based on the Rényi divergence. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics*, pages 435–444.

Bengio, S., Bengio, Y., Cloutier, J., and Gecsei, J. (1992). On the optimization of a synaptic learning rule. In *Conference on Optimality in Biological and Artificial Networks*.

Berger, J. O. (2013). *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media.

Berlinet, A. and Thomas-Agnan, C. (2011). *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media.

Berthelot, D., Schumm, T., and Metz, L. (2017). BEGAN: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*.

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.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1613–1622.

Bregman, L. M. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3).

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan, M. I. (2013). Streaming variational Bayes. In *Advances in Neural Information Processing Systems*, pages 1727–1735.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov chain Monte Carlo*. CRC press.

Bui, T. D., Hernández-Lobato, D., Hernández-Lobato, J. M., Li, Y., and Turner, R. E. (2016a). Black-box alpha divergence for generative models. In *NIPS workshop on Advances in Approximate Bayesian Inference*.

Bui, T. D., Hernández-Lobato, D., Hernández-Lobato, J. M., Li, Y., and Turner, R. E. (2016b). Deep Gaussian processes for regression using approximate expectation propagation. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1472–1481.

Burda, Y., Grosse, R., and Salakhutdinov, R. (2016). Importance weighted autoencoders. In *International Conference on Learning Representations*.

Chen, L. H. (1975). Poisson approximation for dependent trials. *The Annals of Probability*, pages 534–545.

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.

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.

Cornebise, J. (2009). *Adaptive Sequential Monte Carlo Methods*. PhD thesis, Université Pierre et Marie Curie.

Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. John Wiley & Sons.

Cox, R. T. (1946). Probability, frequency and reasonable expectation. *American journal of physics*, 14(1):1–13.

Cremer, C., Li, X., and Duvenaud, D. (2018). Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*.

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.

Cunningham, J. P., Hennig, P., and Lacoste-Julien, S. (2011). Gaussian probabilities and expectation propagation. *arXiv preprint arXiv:1111.6832*.

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.

Dawid, A. P. (1984). Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, pages 278–292.

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.

Deisenroth, M. and Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, pages 465–472.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436.

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.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 248–255.

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. (2017). Learning and policy search in stochastic dynamical systems with Bayesian neural networks. *International Conference on Learning Representations*.

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.

Dikmen, O., Yang, Z., and Oja, E. (2015). Learning the information divergence. *IEEE transactions on pattern analysis and machine intelligence*, 37(7):1442–1454.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. In *Advances in Neural Information Processing Systems*, pages 3203–3211.

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*.

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.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216 – 222.

Duvenaud, D., Maclaurin, D., and Adams, R. (2016). Early stopping as nonparametric variational inference. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics*, pages 1070–1077.

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.

Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A*, 222:309–368.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40.

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.

Gal, Y., Hron, J., and Kendall, A. (2017). Concrete dropout. In *Advances in Neural Information Processing Systems*, pages 3584–3593.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian data analysis*. Chapman and Hall.

Gelman, A., Meng, X.-L., and Stern, H. (1996). Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, pages 733–760.

Gelman, A. and Shalizi, C. R. (2013). Philosophy and the practice of Bayesian statistics. *British Journal of Mathematical and Statistical Psychology*, 66(1):8–38.

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*.

Germain, P., Bach, F., Lacoste, A., and Lacoste-Julien, S. (2016). PAC-Bayesian theory meets bayesian inference. In *Advances in Neural Information Processing Systems*, pages 1884–1892.

Gershman, S. J., Hoffman, M. D., and Blei, D. M. (2012). Nonparametric variational inference. In *Proceedings of the 29th International Conference on Machine Learning*, pages 235–242.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica: Journal of the Econometric Society*, pages 1317–1339.

Ghahramani, Z. (1995). Factorial learning and the EM algorithm. In *Advances in Neural Information Processing Systems*, pages 617–624.

Ghahramani, Z. and Beal, M. J. (2000). Variational inference for Bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems*, pages 449–455.

Ghahramani, Z. and Beal, M. J. (2001). Propagation algorithms for variational Bayesian learning. In *Advances in Neural Information Processing Systems*, pages 507–513.

Ghahramani, Z. and Rasmussen, C. E. (2003). Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 505–512.

Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

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

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.

Gorham, J. and Mackey, L. (2017). Measuring sample quality with kernels. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1292–1301.

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 729–734. IEEE.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing machines. *arXiv preprint arXiv:1410.5401*.

Greensmith, E., Bartlett, P. L., and Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530.

Grosse, R. B., Ancha, S., and Roy, D. M. (2016). Measuring the reliability of MCMC inference with bidirectional Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 2451–2459.

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., Ghahramani, Z., and Turner, R. E. (2015). Neural adaptive sequential Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 2629–2637.

Gu, S., Levine, S., Sutskever, I., and Mnih, A. (2016). MuProp: Unbiased backpropagation for stochastic neural networks. In *International Conference on Learning Representations*.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. (2017). Q-prop: Sample-efficient policy gradient with an off-policy critic. 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.

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. *Journal of Machine Learning Research*, 18(106):1–37.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778.

Heess, N., Tarlow, D., and Winn, J. (2013). Learning to pass expectation propagation messages. In *Advances in Neural Information Processing Systems*, pages 3219–3227.

Herbrich, R., Minka, T., and Graepel, T. (2006). Trueskill™: A Bayesian skill rating system. In *Advances in Neural Information Processing Systems*, pages 569–576.

Hernández-Lobato, D. and Hernández-Lobato, J. M. (2016). Scalable Gaussian process classification via expectation propagation. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics*, pages 168–176.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1861–1869.

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.

Heskes, T. and Zoeter, O. (2002). Expectation propagation for approximate inference in dynamic Bayesian networks. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 216–223. Morgan Kaufmann Publishers Inc.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

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.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

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.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. W. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347.

Hoffman, M. D. and Gelman, A. (2014). The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *The Journal of Machine Learning Research*, 15(1):1593–1623.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. (2017). Densely connected convolutional networks. In *Conference Vision and Pattern Recognition*.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer.

Huggins, J., Campbell, T., and Broderick, T. (2016). Coresets for scalable Bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088.

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.

Hyvärinen, A. (2006). Consistency of pseudolikelihood estimation of fully visible Boltzmann machines. *Neural Computation*, 18(10):2283–2292.

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*.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge University Press.

Jitkrittum, W., Gretton, A., Heess, N., Eslami, S., Lakshminarayanan, B., Sejdinovic, D., and Szabó, Z. (2015). Kernel-based just-in-time learning for passing expectation propagation messages. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 405–414. AUAI Press.

Johnson, O. T. (2004). *Information theory and the central limit theorem*. World Scientific.

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.

Khan, M. E., Baqué, P., Fleuret, F., and Fua, P. (2015). Kullback-Leibler proximal variational inference. In *Advances in Neural Information Processing Systems*, pages 3402–3410.

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., 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*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. (2017). How to train your DRAGAN. *arXiv preprint arXiv:1705.07215*.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, 21(1):1–6.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Koopman, B. O. (1936). On distributions admitting a sufficient statistic. *Transactions of the American Mathematical society*, 39(3):399–409.

Korattikara, A., Rathod, V., Murphy, K. P., and Welling, M. (2015). Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446.

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.

Kusner, M. J. and Hernández-Lobato, J. M. (2016). GANs for sequences of discrete elements with the Gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*.

Kuss, M. and Rasmussen, C. E. (2005). Assessing approximate inference for binary Gaussian process classification. *The Journal of Machine Learning Research*, 6:1679–1704.

Laplace, P. S. (1820). *Théorie analytique des probabilités*. Courcier.

Larochelle, H. and Murray, I. (2011). The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37.

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 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.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, 1:0.

Li, C., Chen, C., Carlson, D., and Carin, L. (2016a). Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1788–1794. AAAI Press.

Li, K. and Malik, J. (2016). Learning to optimize. *arXiv preprint arXiv:1606.01885*.

Li, Y. and Gal, Y. (2017). Dropout inference in Bayesian neural networks with alpha-divergences. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2052–2061.

Li, Y. and Liu, Q. (2016). Wild variational approximations. *NIPS 2016 approximate inference workshop*.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2016b). Gated graph sequence neural networks. 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*.

Lichman, M. (2013). UCI machine learning repository.

Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Liu, Q. and Feng, Y. (2016). Two methods for wild variational inference. *arXiv preprint arXiv:1612.00081*.

Liu, Q. and Lee, J. (2017). Black-box importance sampling. In *Proceedings of the Twentieth International Conference on Artificial Intelligence and Statistics*, pages 952–961.

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., Ullrich, K., and Welling, M. (2017). Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3290–3300.

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.

Lyu, S. (2009). Interpretation and generalization of score matching. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 359–366. AUAI Press.

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. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

MacKay, D. J. (1997). Ensemble learning for hidden Markov models. Technical report, University of Cambridge.

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*.

Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Mandt, S. and Blei, D. (2014). Smoothed gradients for stochastic variational inference. In *Advances in Neural Information Processing Systems*, pages 2438–2446.

Mandt, S., Hoffman, M., and Blei, D. (2016). A variational analysis of stochastic gradient algorithms. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 354–363.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35.

Marino, J., Yue, Y., and Mandt, S. (2018). Learning to infer.

Marlin, B., Swersky, K., Chen, B., and Freitas, N. (2010). Inductive principles for restricted Boltzmann machine learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 509–516.

Maybeck, P. S. (1982). *Stochastic models, estimation and control*. Academic Press.

McAllester, D. A. (1998). Some PAC-Bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 230–234. ACM.

McAllister, R. and Rasmussen, C. E. (2016). Data-efficient reinforcement learning in continuous-state POMDPs. *arXiv preprint arXiv:1602.02523*.

McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., Chikkerur, S., Liu, D., Wattenberg, M., Hrafnkelsson, A. M., Boulos, T., and Kubica, J. (2013). Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM.

Meng, X.-L. (1994). Posterior predictive p-values. *The Annals of Statistics*, pages 1142–1160.

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., Winn, J., Guiver, J., Webster, S., Zaykov, Y., Yangel, B., Spengler, A., and Bronskill, J. (2014). Infer.NET 2.6. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

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.

Minsky, M. and Papert, S. (1969). *Perceptrons*. MIT press.

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.

Mnih, A. and Rezende, D. (2016). Variational inference for Monte Carlo objectives. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2188–2196.

Močkus, J. (1975). On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer.

Mohamed, S. and Lakshminarayanan, B. (2016). Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*.

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*.

Naik, D. K. and Mammone, R. (1992). Meta-neural networks that learn by learning. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 1, pages 437–442. IEEE.

Neal, R. M. (1992). Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Technical report, University of Toronto.

Neal, R. M. (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

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.

Nemirovski, A. and Yudin, D. B. (1983). *Problem Complexity and Method Efficiency in Optimization*. J. Wiley @ Sons, New York.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2007). Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization. In *Advances in Neural Information Processing Systems*, pages 1089–1096.

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., 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.

Oates, C. J., Girolami, M., and Chopin, N. (2017). Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):695–718.

O'Hagan, A. (1991). Bayes-Hermite quadrature. *Journal of statistical planning and inference*, 29(3):245–260.

Opper, M. and Winther, O. (2005). Expectation consistent approximate inference. *The Journal of Machine Learning Research*, 6:2177–2204.

Paige, B. and Wood, F. (2016). Inference networks for sequential Monte Carlo in graphical models. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 3040–3049.

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.

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)*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Peterson, C. and Anderson, J. R. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019.

Peterson, W. W. and Weldon, E. J. (1972). *Error-correcting codes*. MIT press.

Qi, Y., Abdel-Gawad, A. H., and Minka, T. P. (2010). Sparse-posterior Gaussian processes for general likelihoods. In *Uncertainty and Artificial Intelligence*.

Qin, J. (1998). Inferences for case-control and semiparametric two-sample density ratio models. *Biometrika*, 85(3):619–630.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*.

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.

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.

Ring, M. B. (1994). *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin.

Ring, M. B. (1997). CHILD: A first step towards continual learning. *Machine Learning*, 28(1):77–104.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

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.

Ross, S. M. (2002). *Simulation*. Academic Press, Inc., Orlando, FL, USA, 3rd edition.

Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear gaussian models. *Neural Computation*, 11(2):305–345.

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.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:9.

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.

Sasaki, H., Noh, Y.-K., and Sugiyama, M. (2015). Direct density-derivative estimation and its application in KL-divergence approximation. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 809–818.

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.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*.

Seeger, M. (2005). Expectation propagation for exponential families. Technical report, EPFL-REPORT-161464.

Shi, J., Sun, S., and Zhu, J. (2018). Kernel implicit variational inference. In *International Conference on Learning Representations*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

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.

Smith, S. L. and Le, Q. V. (2018). A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*.

Smola, A., Gretton, A., Song, L., and Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer.

Smola, A. J. and Schökopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 911–918.

Snelson, E. and Ghahramani, Z. (2005). Compact approximations to bayesian predictive distributions. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 840–847.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

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*.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 3738–3746.

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.

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.

Thrun, S. and Pratt, L. (1998). *Learning to learn*. Springer Science & Business Media.

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.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2018). Wasserstein auto-encoders. In *International Conference on Learning Representations*.

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*.

Tsallis, C. (1988). Possible generalization of Boltzmann-Gibbs statistics. *Journal of statistical physics*, 52(1-2):479–487.

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.

Uehara, M., Sato, I., Suzuki, M., Nakayama, K., and Matsuo, Y. (2016). Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920*.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Van Erven, T. and Harremoës, P. (2014). Rényi divergence and Kullback-Leibler divergence. *Information Theory, IEEE Transactions on*, 60(7):3797–3820.

Venna, J. and Kaski, S. (2003). Visualizing high-dimensional posterior distributions in Bayesian modeling. In *Artificial Neural Networks and Neural Information Processing-Supplementary proceedings ICANN/ICONIP 2003*.

Villani, C. (2008). *Optimal transport: old and new*. Springer Science & Business Media.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674.

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. 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. and Liu, Q. (2016). Learning to draw samples: With application to amortized MLE for generative adversarial learning. *arXiv preprint arXiv:1611.01722*.

Webb, S. and Teh, Y. W. (2016). A tighter Monte Carlo objective with Renyi alpha-divergence measures. In *NIPS workshop on Bayesian deep learning*.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pages 681–688.

Williams, C. K. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Winn, J. M. and Bishop, C. M. (2005). Variational message passing. In *Journal of Machine Learning Research*, pages 661–694.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

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*.

Xu, M., Lakshminarayanan, B., Teh, Y. W., Zhu, J., and Zhang, B. (2014). Distributed Bayesian posterior sampling via moment sharing. In *Advances in Neural Information Processing Systems*, pages 3356–3364.

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*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*.

Zhe, S., Lee, K.-c., Zhang, K., and Neville, J. (2016). Online spike-and-slab inference with stochastic expectation propagation. *NIPS 2016 approximate inference workshop*.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537.

Zhou, S. and Wolfe, D. A. (2000). On derivative estimation in spline regression. *Statistica Sinica*, pages 93–108.

Zhu, H. and Rohwer, R. (1995). Information geometric measurements of generalisation. Technical report, Technical Report NCRG/4350. Aston University.

Appendix A

Proofs and Derivations

A.1 Derivations of PBP in Chapter 3

We briefly describe the approximation techniques used in probabilistic back-propagation (PBP) for training Bayesian neural networks [Hernández-Lobato and Adams, 2015]. In this case the model is the following:

$$\begin{aligned} p(\mathbf{y}_n|\boldsymbol{x}_n, W, \gamma) = \mathcal{N}(\mathbf{y}_n; \mathrm{NN}_W(\boldsymbol{x}_b), \gamma^{-1}), \quad p(W|\lambda) = \prod_{i,j,l} \mathcal{N}(W_{ij}^l; 0, \lambda^{-1}), \\ p(\lambda) = \mathrm{Gamma}(\lambda; \alpha_0^\lambda, \beta_0^\lambda), \quad p(\gamma) = \mathrm{Gamma}(\gamma; \alpha_0^\gamma, \beta_0^\gamma), \end{aligned} \tag{A.1}$$

where $\alpha_0^\lambda, \beta_0^\lambda, \alpha_0^\gamma, \beta_0^\gamma$ are prior parameters that are all set to 6. We are interested in computing the approximate posterior of the following form:

$$q(W, \gamma, \lambda) = \prod_{i,j,l} \mathcal{N}(W_{ij}^l; \mu_{ij}^l, v_{ij}^l) \mathrm{Gamma}(\lambda; \alpha^\lambda, \beta^\lambda) \mathrm{Gamma}(\gamma; \alpha^\gamma, \beta^\gamma), \tag{A.2}$$

which is iteratively solved by EP. Since the moments of the random variables can also be obtained by differentiating through the (log) normalising constant of the tilted distribution $\log Z$, PBP first approximately computes the normalising constant by

$$\begin{aligned} Z &= \int p(\mathbf{y}|\boldsymbol{x}, W, \gamma) q(W, \gamma, \lambda) dW d\gamma d\lambda \\ &\approx \int \mathcal{N}(\mathbf{y}; \boldsymbol{z}_L, \gamma^{-1}) \mathcal{N}(\boldsymbol{z}_L|\boldsymbol{m}_L, \boldsymbol{v}_L) \mathrm{Gamma}(\gamma; \alpha^\gamma, \beta^\gamma) d\boldsymbol{z}_L d\gamma \quad \text{// PBP forward pass} \\ &= \int \mathcal{T}(\mathbf{y}; \boldsymbol{z}_L, \beta^\gamma/\alpha^\gamma, 2\alpha^\gamma) \mathcal{N}(\boldsymbol{z}_L|\boldsymbol{m}_L, \boldsymbol{v}_L) d\boldsymbol{z}_L \quad \text{// student-t distribution} \\ &\approx \mathcal{N}(\mathbf{y}; \boldsymbol{m}_L, \beta^\gamma/(\alpha^\gamma - 1) + \boldsymbol{v}_L). \quad \text{// Gaussian approx.} \end{aligned} \tag{A.3}$$

It remains to specify the forward pass of PBP which approximate $\mathrm{NN}_W(\boldsymbol{x}), W \sim q(W)$ by $\boldsymbol{z}_L \sim \mathcal{N}(\boldsymbol{z}_L|\boldsymbol{m}_L, \boldsymbol{v}_L)$. A short summary of the idea is the following. Assume the inputs $\boldsymbol{z}_{l-1}$ to the l^{th} layer are Gaussian distributed: $\boldsymbol{z}_{l-1} \sim \mathcal{N}(\boldsymbol{z}_{l-1}|\boldsymbol{m}_{l-1}, \boldsymbol{v}_{l-1})$, then when the network is wide, due to the central limit theorem the pre-activation $\boldsymbol{a}_l = W\boldsymbol{z}_{l-1}/\sqrt{\dim(\boldsymbol{z}_{l-1})}$ is approximately Gaussian distributed with mean and variance determined by the means and variances of $\boldsymbol{z}_{l-1}$ and W^l. If the activation function is ReLU, then the distribution of $\boldsymbol{z}_l$ is a mixture of delta mass $\delta(\boldsymbol{z}_l = \boldsymbol{0})$ and truncated Gaussian at zero. The last step approximates this mixture distribution with a Gaussian distribution that matches the mean and variance, which leads to $\mathcal{N}(\boldsymbol{z}_l|\boldsymbol{m}_l, \boldsymbol{v}_l)$ that is used in later forward pass. We refer to Hernández-Lobato and Adams [2015] for the detailed derivation of $\boldsymbol{m}_l$ and $\boldsymbol{v}_l$.

In the SEP experiments for PBP the update equations are almost the same, except that Z is computed using the cavity distribution $q_{-1}(W, \gamma, \lambda) \propto p(W|\lambda)p(\lambda)p(\gamma)f(W)^{N-1}f(\gamma)^{N-1}f(\lambda)$.

A.2 Proofs of theorems in Chapter 4

A.2.1 Proof of Theorem 4.2

Proof. 1) First we prove for $\alpha \leq 1$, $\mathbb{E}_{\{\boldsymbol{h}_k\}}[\hat{\mathcal{L}}_{\alpha,K}]$ is non-decreasing in K. It is straight forward to show the results holds for $\alpha = 1$. We follow the proof in Burda et al. [2016] for fixed $\alpha < 1$. Let $K > 1$ and the subset of indices $I = \{i_1, ..., i_{K'}\} \subset \{1, ..., K\}, K' < K$ randomly sampled from integers 1 to K. Then for any $\alpha < 1$:

$$
\begin{aligned}
\mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}] &= \frac{1}{1-\alpha}\mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\log \frac{1}{K}\sum_{k=1}^{K}\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha}\right] \\
&= \frac{1}{1-\alpha}\mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\log \mathbb{E}_{I\subset\{1,\ldots,K\}}\left[\frac{1}{K'}\sum_{k=1}^{K'}\left(\frac{p(\boldsymbol{h}_{i_k},\boldsymbol{x})}{q(\boldsymbol{h}_{i_k})}\right)^{1-\alpha}\right]\right] \\
&\geq \frac{1}{1-\alpha}\mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\mathbb{E}_{I\subset\{1,\ldots,K\}}\left[\log \frac{1}{K'}\sum_{k=1}^{K'}\left(\frac{p(\boldsymbol{h}_{i_k},\boldsymbol{x})}{q(\boldsymbol{h}_{i_k})}\right)^{1-\alpha}\right]\right] \qquad (\log x \text{ is concave}) \\
&= \frac{1}{1-\alpha}\mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\log \frac{1}{K'}\sum_{k=1}^{K'}\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha}\right] = \mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^{K'}}[\hat{\mathcal{L}}_{\alpha,K'}]
\end{aligned}
$$

We used Jensen's inequality of logarithm for the lower-bounding result here. When $\alpha > 1$ we can proof similar result but with inequality reversed, simply because now $1 - \alpha < 0$.

2) Next we prove that, when $K \to \infty$ and $|\mathcal{L}_\alpha| < +\infty$, we have $\mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}] \to \mathcal{L}_\alpha$ if $\hat{\mathcal{L}}_{\alpha,K}$ is absolutely integrable wrt. $qd\mu = dQ$ for all $K \geq 1$ (in other words $\mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[|\hat{\mathcal{L}}_{\alpha,K}|] < +\infty$). We only prove it for $\alpha \leq 1$, and for $\alpha > 1$ it can be proved in a similar way. First we use Jensen's inequality again for all finite K:

$$\begin{aligned}\mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}] &= \frac{1}{1-\alpha}\mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\log \frac{1}{K}\sum_{k=1}^K \left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha}\right] \\ &\leq \frac{1}{1-\alpha}\log \mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\frac{1}{K}\sum_{k=1}^K \left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha}\right] = \mathcal{L}_\alpha.\end{aligned}$$

This implies $\limsup_{K\to+\infty} \mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}] \leq \mathcal{L}_\alpha$.

Then as an intermediate result we prove $\hat{\mathcal{L}}_{\alpha,K} \to \mathcal{L}_\alpha$ almost surely when $K \to \infty$. For $\alpha \neq 1$, since function log is continuous we again swap the limit and logarithm:

$$\lim_{K\to+\infty} \frac{1}{1-\alpha}\log \frac{1}{K}\sum_{k=1}^K \left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha} = \frac{1}{1-\alpha}\log \lim_{K\to+\infty} \frac{1}{K}\sum_{k=1}^K \left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha}.$$

Now since we assume $|\mathcal{L}_\alpha| < +\infty$, this implies $\mathbb{E}_q\left[\left(\frac{p(\boldsymbol{h},\boldsymbol{x})}{q(\boldsymbol{h}|\boldsymbol{x})}\right)^{1-\alpha}\right]$ is finite. Also notice for all α values the ratio p/q is non-negative. Thus by the strong law of large numbers we have

$$\lim_{K\to+\infty} \frac{1}{K}\sum_{k=1}^K \left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha} = \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{x})}\left[\left(\frac{p(\boldsymbol{h},\boldsymbol{x})}{q(\boldsymbol{h}|\boldsymbol{x})}\right)^{1-\alpha}\right] \text{ a. s.},$$

then $\hat{\mathcal{L}}_{\alpha,K} \to \mathcal{L}_\alpha$ almost surely as $K \to +\infty$. When $\alpha = 1$ we can use similar method to prove $\lim_{K\to+\infty} \hat{\mathcal{L}}_{1,K} = \mathcal{L}_{\text{VI}}$ almost surely.

Finally, using the non-increasing in α result we will prove later we have $\hat{\mathcal{L}}_{\alpha,K} \geq \hat{\mathcal{L}}_{1,K}$. Thus we can apply Fatou's Lemma and obtain the following almost surely (notice $\mathbb{E}[\hat{\mathcal{L}}_{1,K}] = \mathcal{L}_{\text{VI}}$ for all K):

$$\begin{aligned}\mathcal{L}_\alpha - \mathcal{L}_{\text{VI}} &= \mathbb{E}[\lim_{K\to+\infty} \hat{\mathcal{L}}_{\alpha,K} - \hat{\mathcal{L}}_{1,K}] \\ &\leq \liminf_{K\to+\infty} \mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K} - \hat{\mathcal{L}}_{1,K}] \\ &= \liminf_{K\to+\infty} \mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}] - \mathcal{L}_{\text{VI}}.\end{aligned}$$

Combining with the supremum bound, we have $\mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}] \to \mathcal{L}_\alpha$ when K goes to infinity. For $\alpha > 1$ we use Jensen's inequality to bound the limit infimum and the non-increasing property in α to bound the limit supremum. Thus the convergence result holds for all

$\alpha \in \{\alpha : |\mathcal{L}_\alpha| < +\infty\}$.

3) $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$ is non-increasing in α: since expectation preserves monotonicity, it is sufficient to prove the result for $\hat{\mathcal{L}}_{\alpha,K}$. This can be proved in similar way as Theorem 3 and 39 in Van Erven and Harremoës [2014], and we include the prove here for completeness. Notice that for $\alpha < \beta$ function $x^{\frac{1-\alpha}{1-\beta}}$ defined on $x > 0$ is convex when $\alpha < 1$ and concave when $\alpha > 1$. So applying Jensen's inequality:

$$\begin{aligned}\hat{\mathcal{L}}_{\alpha,K} = \frac{1}{1-\alpha}\log\frac{1}{K}\sum_{k=1}^{K}\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha} &= \frac{1}{1-\alpha}\log\frac{1}{K}\sum_{k=1}^{K}\left(\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\beta}\right)^{\frac{1-\alpha}{1-\beta}} \\ &\geq \frac{1}{1-\alpha}\log\left(\frac{1}{K}\sum_{k=1}^{K}\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\beta}\right)^{\frac{1-\alpha}{1-\beta}} = \hat{\mathcal{L}}_{\beta,K}.\end{aligned}$$

Continuity in α: First we show $\hat{\mathcal{L}}_{\alpha,K}$ is continuous in α when $p(\boldsymbol{h}_k,\boldsymbol{x}) \neq 0$ for $\boldsymbol{h}_k \sim q$. For $\alpha \neq 0, 1, \infty$ and for any sequence $\{\alpha_n\} \to \alpha$ it is sufficient to show that

$$\begin{aligned}&\lim_{n\to\infty}\log\frac{1}{K}\sum_k q(\boldsymbol{h}_k|\boldsymbol{x})^{\alpha_n}p(\boldsymbol{h}_k,\boldsymbol{x})^{1-\alpha_n} \\ =&\log\lim_{n\to\infty}\frac{1}{K}\sum_k q(\boldsymbol{h}_k|\boldsymbol{x})^{\alpha_n}p(\boldsymbol{h}_k,\boldsymbol{x})^{1-\alpha_n} \quad (\log x \text{ is a continuous function}) \\ =&\log\frac{1}{K}\sum_k \lim_{n\to\infty} q(\boldsymbol{h}_k|\boldsymbol{x})^{\alpha_n}p(\boldsymbol{h}_k,\boldsymbol{x})^{1-\alpha_n} \quad (\text{finite sum}) \\ =&\log\frac{1}{K}\sum_k q(\boldsymbol{h}_k|\boldsymbol{x})\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\lim_{n\to\infty}\alpha_n} \quad (a^x \text{ is continuous in } x \text{ for all } a > 0) \\ =&\log\frac{1}{K}\sum_k q(\boldsymbol{h}_k|\boldsymbol{x})^{\alpha}p(\boldsymbol{h}_k,\boldsymbol{x})^{1-\alpha}.\end{aligned}$$

We note that since we assume $\hat{\mathcal{L}}_{\alpha,K}$ is absolutely integrable, we have $p/q > 0$ almost everywhere on the support of q. Hence $\{\hat{\mathcal{L}}_{\alpha_n,K}\}$ has point-wise limit $\hat{\mathcal{L}}_{\alpha,K}$ almost everywhere as $n \to +\infty$.

For $\alpha = 0, 1, \infty$ the Rényi divergence is defined by continuity so one can use the same technique to show the continuity of $\hat{\mathcal{L}}_{\alpha,K}$ on those α values for fixed K. Then since $\alpha_n \to \alpha$, for any $\varepsilon > 0$, there exists n that is large enough such that $\alpha_m \in (\alpha-\varepsilon, \alpha+\varepsilon)$ for all $m > n$. Using the monotonicity result, we have for $\forall m > n$, $\hat{\mathcal{L}}_{\alpha_m,K}$ is bounded in the interval $(\hat{\mathcal{L}}_{\alpha+\varepsilon,K}, \hat{\mathcal{L}}_{\alpha-\varepsilon,K})$ and by assumption we have $\mathbb{E}[|\hat{\mathcal{L}}_{\alpha-\varepsilon,K}|] < +\infty$ and $\mathbb{E}[|\hat{\mathcal{L}}_{\alpha+\varepsilon,K}|] < +\infty$. This allows us to apply the dominated convergence theorem to prove $\lim_{n\to+\infty}\mathbb{E}[\hat{\mathcal{L}}_{\alpha_n,K}] =$

$\mathbb{E}[\lim_{n\to+\infty}\hat{\mathcal{L}}_{\alpha_n,K}] = \mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$. Thus we have proved that $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$ is continuous on $\alpha \in \{|\mathcal{L}_\alpha| < +\infty\}$ if $\hat{\mathcal{L}}_{\alpha,K}$ is absolutely integrable.

□

A.2.2 Proof of Corollary 4.1

The next question we are interested is that, given a fixed number of samples K, can we tune the α parameter to achieve the best approximation to the marginal likelihood? This is an important question as in practice only finite amount of computation resource is allowed. In the following we will discuss a corollary result based on Theorem 4.2, but to prove it we first introduce the following lemmas. As we assume $\text{supp}(p) \subseteq \text{supp}(q)$, there might exist some regions that $q > 0$ but $p = 0$. We define $\rho = \int_{\text{supp}(q)\setminus\text{supp}(p)} dQ$ with $dQ = qd\mu$, and rewrite the computation of $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$. The following lemma shows the importance of the absolute integrable assumption in Theorem 4.2.

Lemma A.1. *Assume $\rho > 0$. Then for all finite K and $\alpha < 0$, $\mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})] = -\infty$ and thus $\hat{\mathcal{L}}_{\alpha,K}$ is not integrable wrt. $qd\mu = dQ$.*

Proof. We define $\tilde{q}$ as the q distribution restricted on the support of p, i.e. $\tilde{q} = q/(1-\rho)$ defined on $\text{supp}(p)$. Then for any fixed $K < +\infty$ and $\alpha < 0$, we have

$$\begin{aligned}\mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K\sim q}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})] =& \rho^K \log 0 + \sum_{k=1}^{K}\binom{K}{k}\rho^{K-k}(1-\rho)^k\left(\mathbb{E}_{\{\boldsymbol{h}_j\}_{j=1}^k\sim\tilde{q}}[\hat{\mathcal{L}}_{\alpha,k}(\tilde{q};\boldsymbol{x})] + \frac{\log k}{1-\alpha}\right)\\ &-(1-\rho^K)\left(\log(1-\rho)+\frac{\log K}{1-\alpha}\right).\end{aligned}$$

Thus $\mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})] = -\infty$ for all finite K and $\alpha < 0$. □

The above example shows the pathology of MC approximation which is further discussed in Section 4.2.5. From now on we assume $\hat{\mathcal{L}}_{\alpha,K}$ is absolutely integrable w.r.t. dQ in order to apply Theorem 4.2.

Lemma A.2. *Assume $\alpha < 0$, $\hat{\mathcal{L}}_{\alpha,K}$ absolutely integrable wrt. $qd\mu = dQ$ for all K, $\mathcal{L}_\alpha > \mathcal{L}_{VI}$, and $|\mathcal{L}_\alpha| < +\infty$. Then there exists $1 \le K_\alpha < +\infty$ such that for all $K \le K_\alpha < K'$, $\mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^K}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})] \le \log p(\boldsymbol{x}) < \mathbb{E}_{\{\boldsymbol{h}_k\}_{k=1}^{K'}}[\hat{\mathcal{L}}_{\alpha,K'}(q;\boldsymbol{x})]$. Also K_α is **non-decreasing** in α with $\lim_{\alpha\to 0} K_\alpha = +\infty$ and $\lim_{\alpha\to-\infty} K_\alpha \ge 1$.*

Proof. 1) Existence of K_α: first from Theorem 4.2 we have $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$ is non-decreasing in K when $\alpha < 0$. Then since for all α, $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,1}] = \mathcal{L}_{\text{VI}} \le \log p(\boldsymbol{x})$, we have $K_\alpha \ge 1$ if K_α exists. Also from Theorem 4.2 we have $\lim_{K\to+\infty}\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}] = \mathcal{L}_\alpha > \log p(\boldsymbol{x})$ for all $\alpha < 0$. Hence for

$\varepsilon = \mathcal{L}_\alpha - \log p(\boldsymbol{x})$ there exist K that is finite but large enough such that $\mathcal{L}_\alpha - \mathbb{E}[\hat{\mathcal{L}}_{\alpha,K'}] < \varepsilon$ for all $K' > K$. Now we can define $\varepsilon = \mathcal{L}_\alpha - \log p(\boldsymbol{x})$ and take K_α as the minimum of such K, and it is straight-forward to show that $1 \leq K_\alpha < +\infty$.

2) K_α is non-decreasing in α: suppose there exist $\alpha > \beta$ such that $K_\alpha < K_\beta$. Then there exist $K_\alpha < K \leq K_\beta$ such that $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}] > \log p(\boldsymbol{x}) \geq \mathbb{E}[\hat{\mathcal{L}}_{\beta,K}]$. But Theorem 4.2 says $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$ is non-increasing in α, a contradiction.

3) Since $\lim_{K\to+\infty} \mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}] = \mathcal{L}_\alpha$ and $\mathcal{L}_\alpha \downarrow \log p(\boldsymbol{x})$ when $\alpha \uparrow 0$, we have $\lim_{\alpha\to 0} K_\alpha = +\infty$. Also since K_α is non-decreasing in α and is lower-bounded by 1, we have the limit exists and $\lim_{\alpha\to-\infty} K_\alpha \geq 1$. □

Now we prove Corollary 4.1 which answers the question of approximating the marginal likelihood with finite sample MC bound. It is sufficient to prove the corollary with the conditions assumed in Lemma A.2 since $K_\alpha = +\infty$ for the other cases, and if so for all $\alpha < 0$, then $\alpha_K = -\infty$ for all finite K.

Proof. 1) Existence of α_K for $\lim_{\alpha\to-\infty} K_\alpha < K < +\infty$: from Lemma A.2 we can find $\alpha > \beta$ such that $K_\alpha \geq K \geq K_\beta$. This means $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}] \leq \log p(\boldsymbol{x}) \leq \mathbb{E}[\hat{\mathcal{L}}_{\beta,K}]$. Since $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$ is continuous in α for any fixed K, there exits $\alpha \leq \gamma \leq \beta$ to have $\mathbb{E}[\hat{\mathcal{L}}_{\gamma,K}] = \log p(\boldsymbol{x})$. Note that γ might not be unique, so we define α_K as the minimum of such γ, which also gives $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}] > \log p(\boldsymbol{x})$ for all $\alpha < \alpha_K$.

2) α_K is non-decreasing in K: suppose there exist $K < K'$ with $\alpha_K > \alpha_{K'}$. Then we can find $\alpha_K > \alpha > \alpha_{K'}$ such that $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}] > \log p(\boldsymbol{x}) = \mathbb{E}[\hat{\mathcal{L}}_{\alpha_{K'},K'}] \geq \mathbb{E}[\hat{\mathcal{L}}_{\alpha,K'}]$. But from Theorem 4.2 $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}]$ is non-decreasing in K, a contradiction.

3) Since $\lim_{K\to+\infty} \mathbb{E}[\hat{\mathcal{L}}_{\alpha,K}] = \mathcal{L}_\alpha$ and $\mathcal{L}_\alpha \downarrow \log p(\boldsymbol{x})$ when $\alpha \uparrow 0$, we have $\lim_{K\to+\infty} \alpha_K = 0$. Also for all α, $\mathbb{E}[\hat{\mathcal{L}}_{\alpha,1}] = \mathcal{L}_{VI} \leq \log p(\boldsymbol{x})$, so $\lim_{K\to 1} \alpha_K = -\infty$.

□

A.2.3 Proof of Theorem 4.3

Proof. We substitute the exponential family likelihood term into the stochastic approximation of the VR bound with $\alpha < 1$, and use Hölder's inequality for any $1/r + 1/s = 1$, $r > 1$ (define $\tilde{\alpha} = 1 - (1-\alpha)r$):

$$\begin{aligned}
\mathbb{E}_{\mathbb{S}}[\tilde{\mathcal{L}}_\alpha(q;\mathbb{S})] &= \frac{1}{1-\alpha}\log\mathbb{E}_q[\left(\frac{p_0(\boldsymbol{\theta})\bar{f}_{\mathcal{D}}(\boldsymbol{\theta})^N}{q(\boldsymbol{\theta})}\frac{\bar{f}_{\mathbb{S}}(\boldsymbol{\theta})^N}{\bar{f}_{\mathcal{D}}(\boldsymbol{\theta})^N}\right)^{1-\alpha}] \\
&\leq \mathcal{L}_{\tilde{\alpha}}(q;\mathcal{D}) + \frac{1}{(1-\alpha)s}\mathbb{E}_{\mathbb{S}}\left\{\log\mathbb{E}_q[\exp[N(1-\alpha)s\langle\bar{\boldsymbol{\Phi}}_{\mathbb{S}} - \bar{\boldsymbol{\Phi}}_{\mathcal{D}}, \boldsymbol{\theta}\rangle]]\right\} \\
&= \mathcal{L}_{\tilde{\alpha}}(q;\mathcal{D}) + \frac{1}{(1-\alpha)s}\mathbb{E}_{\mathbb{S}}[K_{\boldsymbol{\theta}}(N(1-\alpha)s(\bar{\boldsymbol{\Phi}}_{\mathbb{S}} - \bar{\boldsymbol{\Phi}}_{\mathcal{D}}))],
\end{aligned}$$

where $\bar{\boldsymbol{\Phi}}_{\mathcal{S}}$ and $\bar{\boldsymbol{\Phi}}_{\mathcal{D}}$ denote the mean of the sufficient statistic $\boldsymbol{\Phi}(\boldsymbol{x})$ on the mini-batch $\mathcal{S}$ and the whole dataset $\mathcal{D}$, respectively. For Gaussian distribution $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the cumulant generating function $K_{\boldsymbol{\theta}}(\boldsymbol{t})$ has a closed form

$$K_{\boldsymbol{\theta}}(\boldsymbol{t}) = \boldsymbol{\mu}^T \boldsymbol{t} + \frac{1}{2} \boldsymbol{t}^T \boldsymbol{\Sigma} \boldsymbol{t}.$$

Define $\boldsymbol{t}_{\mathcal{S}} = N(1-\alpha)s\Delta_{\mathcal{S}}$ with $\Delta_{\mathcal{S}} = \bar{\boldsymbol{\Phi}}_{\mathcal{S}} - \bar{\boldsymbol{\Phi}}_{\mathcal{D}}$, then $\mathbb{E}_{\mathcal{S}}[\boldsymbol{t}_{\mathcal{S}}] = \mathbf{0}$ and the upper-bound becomes

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}[\tilde{\mathcal{L}}_{\alpha}(q;\mathcal{S})] &\leq \mathcal{L}_{\tilde{\alpha}}(q;\mathcal{D}) + \frac{1}{(1-\alpha)s} \mathbb{E}_{\mathcal{S}}[K_{\boldsymbol{\theta}}(\boldsymbol{t}_{\mathcal{S}})] \\
&= \mathcal{L}_{\tilde{\alpha}}(q;\mathcal{D}) + \frac{1}{(1-\alpha)s} \mathbb{E}_{\mathcal{S}}[\boldsymbol{\mu}^T \boldsymbol{t}_{\mathcal{S}} + \frac{1}{2} \boldsymbol{t}_{\mathcal{S}}^T \boldsymbol{\Sigma} \boldsymbol{t}_{\mathcal{S}}] \\
&= \mathcal{L}_{\tilde{\alpha}}(q;\mathcal{D}) + \frac{N^2(1-\alpha)s}{2} \mathbb{E}_{\mathcal{S}}[\Delta_{\mathcal{S}}^T \boldsymbol{\Sigma} \Delta_{\mathcal{S}}] \\
&= \mathcal{L}_{\tilde{\alpha}}(q;\mathcal{D}) + \frac{N^2(1-\alpha)s}{2} \mathrm{tr}(\boldsymbol{\Sigma} \mathrm{Cov}_{\mathcal{S}\sim\mathcal{D}}(\bar{\boldsymbol{\Phi}}_{\mathcal{S}})).
\end{aligned}$$

Applying the condition of Hölder's inequality $1/r + 1/s = 1$ proves the result. □

A.3 Derivations and experimental details of Chapter 6

A.3.1 Parametric Stein gradient estimator: the RBF kernel case

Direct minimisation of KSD V-statistic and U-statistic

The V-statistic of KSD is the following: given samples $\boldsymbol{x}^k \sim q, k = 1, ..., K$ and recall $\mathbf{K}_{jl} = \mathcal{K}(\boldsymbol{x}^j, \boldsymbol{x}^l)$

$$\mathcal{S}_V^2(q, \hat{q}) = \frac{1}{K^2} \sum_{j=1}^{K} \sum_{l=1}^{K} \left[\hat{\boldsymbol{g}}(\boldsymbol{x}^j)^{\mathrm{T}} \mathbf{K}_{jl} \hat{\boldsymbol{g}}(\boldsymbol{x}^l) + \hat{\boldsymbol{g}}(\boldsymbol{x}^j)^{\mathrm{T}} \nabla_{\boldsymbol{x}^l} \mathbf{K}_{jl} + \nabla_{\boldsymbol{x}^j} \mathbf{K}_{jl}^{\mathrm{T}} \hat{\boldsymbol{g}}(\boldsymbol{x}^l) + \mathrm{Tr}(\nabla_{\boldsymbol{x}^j, \boldsymbol{x}^l} \mathbf{K}_{jl}) \right]. \tag{A.4}$$

The last term $\nabla_{\boldsymbol{x}^j, \boldsymbol{x}^l} \mathbf{K}_{jl}$ will be ignored as it does not depend on the approximation $\hat{\boldsymbol{g}}$. Using matrix notations defined in the main text, readers can verify that the V-statistic can be computed as

$$\mathcal{S}_V^2(q, \hat{q}) = \frac{1}{K^2} \mathrm{Tr}(\mathbf{K} \hat{\mathbf{G}} \hat{\mathbf{G}}^{\mathrm{T}} + 2\langle \nabla, \mathbf{K} \rangle \hat{\mathbf{G}}^{\mathrm{T}}) + C. \tag{A.5}$$

Using the cyclic invariance of matrix trace leads to the desired result in the main text. The U-statistic of KSD removes terms indexed by $j = l$ in (A.4), in which the matrix form is

$$\mathbb{S}_U^2(q,\hat{q}) = \frac{1}{K(K-1)}\mathrm{Tr}((\mathbf{K}-\mathrm{diag}(\mathbf{K}))\hat{\mathbf{G}}\hat{\mathbf{G}}^{\mathrm{T}} + 2(\langle\nabla,\mathbf{K}\rangle - \nabla\mathrm{diag}(\mathbf{K}))\hat{\mathbf{G}}^{\mathrm{T}}) + C. \tag{A.6}$$

with the jth row of $\nabla\mathrm{diag}(\mathbf{K})$ defined as $\nabla_{\boldsymbol{x}^j}\mathcal{K}(\boldsymbol{x}^j,\boldsymbol{x}^j)$. For most translation invariant kernels this extra term $\nabla\mathrm{diag}(\mathbf{K}) = \mathbf{0}$, thus the optimal solution of $\hat{\mathbf{G}}$ by minimising KSD U-statistic is

$$\hat{\mathbf{G}}_U^{\mathrm{Stein}} = -(\mathbf{K}-\mathrm{diag}(\mathbf{K}) + \eta\boldsymbol{I})^{-1}\langle\nabla,\mathbf{K}\rangle. \tag{A.7}$$

Parametric Stein estimator with RBF kernel

We define a parametric approximation in a similar way as for the score matching estimator:

$$\log\hat{q}(\boldsymbol{x}) := \sum_{k=1}^{K} a_k\mathcal{K}(\boldsymbol{x},\boldsymbol{x}^k) + C, \quad \mathcal{K}(\boldsymbol{x},\boldsymbol{x}') = \exp\left[-\frac{1}{2\sigma^2}||\boldsymbol{x}-\boldsymbol{x}'||_2^2\right]. \tag{A.8}$$

Now we show the optimal solution of $\boldsymbol{a} = (a_1,...,a_K)^{\mathrm{T}}$ by minimising (A.4). To simplify derivations we assume the approximation and KSD use the same kernel. First note that the gradient of the RBF kernel is

$$\nabla_{\boldsymbol{x}}\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') = \frac{1}{\sigma^2}\mathcal{K}(\boldsymbol{x},\boldsymbol{x}')(\boldsymbol{x}'-\boldsymbol{x}). \tag{A.9}$$

Substituting (A.9) into (A.4):

$$\mathbb{S}_V^2(q,\hat{q}) = C + \clubsuit + 2\spadesuit,$$

$$\clubsuit = \frac{1}{K^2}\sum_{k=1}^{K}\sum_{k'=1}^{K}\sum_{j=1}^{K}\sum_{l=1}^{K} a_k a_{k'}\mathbf{K}_{kj}\mathbf{K}_{jl}\mathbf{K}_{lk'}\frac{1}{\sigma^4}(\boldsymbol{x}^k-\boldsymbol{x}^j)^{\mathrm{T}}(\boldsymbol{x}^{k'}-\boldsymbol{x}^l),$$

$$\spadesuit = \frac{1}{K^2}\sum_{k=1}^{K}\sum_{j=1}^{K}\sum_{l=1}^{K} a_k\mathbf{K}_{kj}\mathbf{K}_{jl}\frac{1}{\sigma^4}(\boldsymbol{x}^k-\boldsymbol{x}^j)^{\mathrm{T}}(\boldsymbol{x}^j-\boldsymbol{x}^l).$$

We first consider summing the j, l indices in $\clubsuit$. Recall the "gram matrix" $\mathbb{X}_{ij} = (\boldsymbol{x}^i)^{\mathrm{T}}\boldsymbol{x}^j$, the inner product term in $\clubsuit$ can be expressed as $\mathbb{X}_{kk'} + \mathbb{X}_{jl} - \mathbb{X}_{kl} - \mathbb{X}_{jk'}$. Thus the summation

over j, l can be re-written as

$$\begin{aligned}\boldsymbol{\Lambda} &:= \sum_{j=1}^{K}\sum_{l=1}^{K}\mathbf{K}_{kj}\mathbf{K}_{jl}\mathbf{K}_{lk'}(\mathbb{X}_{kk'}+\mathbb{X}_{jl}-\mathbb{X}_{kl}-\mathbb{X}_{jk'}) \\ &= \mathbb{X}\odot(\mathbf{KKK})+\mathbf{K}(\mathbf{K}\odot\mathbb{X})\mathbf{K}-((\mathbf{KK})\odot\mathbb{X})\mathbf{K}-\mathbf{K}((\mathbf{KK})\odot\mathbb{X}).\end{aligned}$$

And thus $\clubsuit = \frac{1}{\sigma^4}\boldsymbol{a}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{a}$. Similarly the summation over j, l in $\spadesuit$ can be simplified into

$$\begin{aligned}-\boldsymbol{b} &:= \sum_{j=1}^{K}\sum_{l=1}^{K}\mathbf{K}_{kj}\mathbf{K}_{jl}(\mathbb{X}_{kj}+\mathbb{X}_{jl}-\mathbb{X}_{kl}-\mathbb{X}_{jj}) \\ &= -(\mathbf{K}\mathrm{diag}(\mathbb{X})\mathbf{K}+(\mathbf{KK})\odot\mathbb{X}-\mathbf{K}(\mathbf{K}\odot\mathbb{X})-(\mathbf{K}\odot\mathbb{X})\mathbf{K})\mathbf{1},\end{aligned}$$

which leads to $\spadesuit = -\frac{1}{\sigma^4}\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}$. Thus minimising $\mathcal{S}_V^2(q,\hat{q})$ plus an l_2 regulariser returns the Stein estimator $\boldsymbol{a}_V^{\mathrm{Stein}}$ in the main text.

Similarly we can derive the solution for KSD U-statistic minimisation. The U statistic can also be represented in quadratic form $\mathcal{S}_U^2(q,\hat{q}) = C + \tilde{\clubsuit} + 2\tilde{\spadesuit}$, with $\tilde{\spadesuit} = \spadesuit$ and

$$\tilde{\clubsuit} = \clubsuit - \frac{1}{K^2}\sum_{k=1}^{K}\sum_{k'=1}^{K}\sum_{j=1}^{K}a_k a_{k'}\mathbf{K}_{kj}\mathbf{K}_{jj}\mathbf{K}_{jk'}\frac{1}{\sigma^4}(\mathbb{X}_{kk'}+\mathbb{X}_{jj}-\mathbb{X}_{kj}-\mathbb{X}_{jk'}).$$

Summing over the j indices for the second term, we have

$$\begin{aligned}&\sum_{j=1}^{K}\mathbf{K}_{kj}\mathbf{K}_{jj}\mathbf{K}_{jk'}(\mathbb{X}_{kk'}+\mathbb{X}_{jj}-\mathbb{X}_{kj}-\mathbb{X}_{jk'}) \\ &= \mathbb{X}\odot(\mathbf{K}\mathrm{diag}(\mathbf{K})\mathbf{K})+\mathbf{K}\mathrm{diag}(\mathbf{K}\odot\mathbb{X})\mathbf{K}-((\mathbf{K}\mathrm{diag}(\mathbf{K}))\odot\mathbb{X})\mathbf{K}-\mathbf{K}((\mathrm{diag}(\mathbf{K})\mathbf{K})\odot\mathbb{X}).\end{aligned}$$

Working through the analogous derivations reveals that $\hat{\boldsymbol{a}}_U^{\mathrm{Stein}} = (\tilde{\boldsymbol{\Lambda}}+\eta\mathbf{I})^{-1}\boldsymbol{b}$, with

$$\begin{aligned}\tilde{\boldsymbol{\Lambda}} =& \mathbb{X}\odot(\mathbf{K}(\mathbf{K}-\mathrm{diag}(\mathbf{K}))\mathbf{K})+\mathbf{K}((\mathbf{K}\odot\mathbb{X})-\mathrm{diag}(\mathbf{K}\odot\mathbb{X}))\mathbf{K} \\ &-((\mathbf{K}(\mathbf{K}-\mathrm{diag}(\mathbf{K})))\odot\mathbb{X})\mathbf{K}-\mathbf{K}(((\mathbf{K}-\mathrm{diag}(\mathbf{K}))\mathbf{K})\odot\mathbb{X}).\end{aligned}$$

A.3.2 Score matching estimator: the Epanechnikov kernel case

In this section we provide analytical solutions for the score matching estimator (more specifically the linear coefficient $\boldsymbol{a} = (a_1, ..., a_K)$) for the case of the Epanechnikov kernel. The solution for the RBF kernel case is referred to Strathmann et al. [2015].

The Epanechnikov kernel is defined as $\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') = \frac{1}{d}\sum_{i=1}^{d}(1-(x_i-x_i')^2)$, where the first and second order gradients w.r.t. x_i is

$$\nabla_{x_i}\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') = \frac{2}{d}(x_i'-x_i), \quad \nabla_{x_i}\nabla_{x_i}\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') = -\frac{2}{d}.$$

Thus the score matching objective with $\log\hat{q}(\boldsymbol{x}) = \sum_{k=1}^{K} a_k\mathcal{K}(\boldsymbol{x},\boldsymbol{x}^k) + C$ is reduced to

$$\begin{aligned}\mathcal{F}(\boldsymbol{a}) &= \frac{1}{K}\sum_{j=1}^{K}\left[||\sum_{k=1}^{K}a_k\frac{2}{d}(\boldsymbol{x}^k-\boldsymbol{x}^j)||_2^2 - 2\sum_{k=1}^{K}a_k\frac{2}{d}d\right]\\ &= \frac{4}{K}\sum_{j=1}^{K}\left[\frac{1}{d^2}\sum_{k=1}^{K}\sum_{k'=1}^{K}a_ka_{k'}(\boldsymbol{x}^k-\boldsymbol{x}^j)^{\mathrm{T}}(\boldsymbol{x}^{k'}-\boldsymbol{x}^j) - \boldsymbol{a}^{\mathrm{T}}\mathbf{1}\right]\\ &:= 4(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{a} - \boldsymbol{a}^{\mathrm{T}}\mathbf{1}),\end{aligned}$$

with the matrix elements

$$\boldsymbol{\Sigma}_{kk'} = \frac{1}{d^2}\left[(\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{x}^{k'} + \frac{1}{K}\sum_{j=1}^{K}\left(||\boldsymbol{x}^j||_2^2 - (\boldsymbol{x}^k+\boldsymbol{x}^{k'})^{\mathrm{T}}\boldsymbol{x}^j\right)\right].$$

Define the "gram matrix" $\mathbb{X}_{ij} = (\boldsymbol{x}^i)^{\mathrm{T}}\boldsymbol{x}^j$, we write the matrix form of $\boldsymbol{\Sigma}$ as

$$\boldsymbol{\Sigma} = \frac{1}{d^2}\left[\mathbb{X} + \frac{1}{K}\left(\mathrm{Tr}(\mathbb{X}) - 2\mathbb{X}\mathbf{1}\mathbf{1}^{\mathrm{T}}\right)\right].$$

Thus with an l_2 regulariser, the fitted coefficients are

$$\hat{\boldsymbol{a}}^{\text{score}} = \frac{d^2}{2}\left[\mathbb{X} + \frac{1}{K}\left(\mathrm{Tr}(\mathbb{X}) - 2\mathbb{X}\mathbf{1}\mathbf{1}^{\mathrm{T}}\right) + \eta\mathbf{I}\right]^{-1}\mathbf{1}.$$

A.3.3 Score matching estimator: the Cauchy kernel case

The Cauchy kernel is defined as $\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') = \left(1+\frac{||\boldsymbol{x}-\boldsymbol{x}'||_2^2}{2\sigma^2}\right)^{-1}$, where the first and second order gradients w.r.t. x_i is

$$\begin{aligned}\nabla_{x_i}\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') &= \frac{1}{\sigma^2}\mathcal{K}(\boldsymbol{x},\boldsymbol{x}')^2(x_i'-x_i),\\ \nabla_{x_i}\nabla_{x_i}\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') &= -\frac{1}{\sigma^2}\mathcal{K}(\boldsymbol{x},\boldsymbol{x}')^2 + \frac{2}{\sigma^4}\mathcal{K}(\boldsymbol{x},\boldsymbol{x}')^3(x_i'-x_i)^2.\end{aligned}$$

We separate the score matching objective in two parts and handle each of them in the following. Writing $\mathbf{K}^{\mathbb{2}} := \mathbf{K}\odot\mathbf{K}$, we have

$$\begin{aligned}
\sum_{j=1}^{K} ||\boldsymbol{g}(\boldsymbol{x}^j)||_2^2 &= \frac{1}{\sigma^4}\sum_{j=1}^{K}\left(\sum_{k=1}^{K} a_k \mathcal{K}(\boldsymbol{x}^k,\boldsymbol{x}^j)^2(\boldsymbol{x}^k-\boldsymbol{x}^j)\right)^{\mathrm{T}}\left(\sum_{k'=1}^{K} a_{k'} \mathcal{K}(\boldsymbol{x}^{k'},\boldsymbol{x}^j)^2(\boldsymbol{x}^{k'}-\boldsymbol{x}^j)\right) \\
&= \frac{1}{\sigma^4}\sum_{k,k'} a_k a_{k'} \sum_{j=1}^{K} \mathbf{K}_{jk}^2 \mathbf{K}_{jk'}^2 (\boldsymbol{x}^k-\boldsymbol{x}^j)^{\mathrm{T}}(\boldsymbol{x}^{k'}-\boldsymbol{x}^j) \\
&= \frac{1}{\sigma^4}\sum_{k,k'} a_k a_{k'} \sum_{j=1}^{K} \mathbf{K}_{jk}^2 \mathbf{K}_{jk'}^2 (\mathbb{X}_{kk'}+\mathbb{X}_{jj}-\mathbb{X}_{jk}-\mathbb{X}_{jk'}) \\
&= \frac{1}{\sigma^4}\boldsymbol{a}^{\mathrm{T}}\left[\mathbf{K}^2\mathbf{K}^2\odot\mathbb{X}+\mathbf{K}^2\mathrm{diag}(\mathbb{X})\mathbf{K}^2-(\mathbf{K}^2\odot\mathbb{X})\mathbf{K}^2-\mathbf{K}^2(\mathbf{K}^2\odot\mathbb{X})\right]\boldsymbol{a}
\end{aligned}$$

$$\begin{aligned}
2\sum_{j=1}^{K}\langle\nabla,\boldsymbol{g}(\boldsymbol{x}^j)\rangle &= \frac{2}{\sigma^2}\sum_{j=1}^{K}\sum_{k=1}^{K} a_k\left(-d\mathcal{K}(\boldsymbol{x}^k,\boldsymbol{x}^j)^2+2\mathcal{K}(\boldsymbol{x}^k,\boldsymbol{x}^j)^3\frac{||\boldsymbol{x}^k-\boldsymbol{x}^j||_2^2}{\sigma^2}\right) \\
&= \frac{2}{\sigma^2}\sum_{j=1}^{K}\sum_{k=1}^{K} a_k\left(-d\mathbf{K}_{jk}^2+4\mathbf{K}_{jk}^3(\mathbf{K}_{jk}^{-1}-1)\right) \\
&= \frac{2}{\sigma^2}\sum_{k=1}^{K} a_k\sum_{j=1}^{K}\mathbf{K}_{jk}^2(4-d-4\mathbf{K}_{jk}) \\
&= -\frac{2}{\sigma^2}\boldsymbol{a}^{\mathrm{T}}\left[\mathbf{K}^2\odot(4\mathbf{K}+(d-4)\mathbf{I})\right]\mathbf{1}
\end{aligned}$$

Therefore the optimal coefficient for the Cauchy kernel case is $\hat{\boldsymbol{a}}^{\text{score}} = (\boldsymbol{\Lambda}+\eta\mathbf{I})^{-1}\boldsymbol{b}$, with

$$\begin{aligned}
\boldsymbol{\Lambda} &= \mathbf{K}^2\mathbf{K}^2\odot\mathbb{X}+\mathbf{K}^2\mathrm{diag}(\mathbb{X})\mathbf{K}^2-(\mathbf{K}^2\odot\mathbb{X})\mathbf{K}^2-\mathbf{K}^2(\mathbf{K}^2\odot\mathbb{X}), \\
\boldsymbol{b} &= \sigma^2\mathbf{K}^2\odot(4\mathbf{K}+(d-4)\mathbf{I}).
\end{aligned}$$

A.3.4 Parametric Stein gradient estimator with the Cauchy kernel

The derivation for the parametric Stein gradient estimator with the Cauchy kernel is very similar with that for the RBF kernel:

$$\mathcal{S}_V^2(q,\hat{q}) = C+\diamondsuit+2\heartsuit,$$

$$\diamondsuit = \frac{1}{K^2}\sum_{k=1}^{K}\sum_{k'=1}^{K}\sum_{j=1}^{K}\sum_{l=1}^{K} a_k a_{k'}\mathbf{K}_{kj}^2\mathbf{K}_{jl}\mathbf{K}_{lk'}^2\frac{1}{\sigma^4}(\boldsymbol{x}^k-\boldsymbol{x}^j)^{\mathrm{T}}(\boldsymbol{x}^{k'}-\boldsymbol{x}^l),$$

$$\heartsuit = \frac{1}{K^2}\sum_{k=1}^{K}\sum_{j=1}^{K}\sum_{l=1}^{K} a_k\mathbf{K}_{kj}^2\mathbf{K}_{jl}^2\frac{1}{\sigma^4}(\boldsymbol{x}^k-\boldsymbol{x}^j)^{\mathrm{T}}(\boldsymbol{x}^j-\boldsymbol{x}^l).$$

Similar rearrangement as in the RBF kernel case returns $\diamondsuit = \frac{1}{\sigma^4}\boldsymbol{a}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{a}$ and $\heartsuit = -\frac{1}{\sigma^4}\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}$, with

$$\boldsymbol{\Lambda} = \mathbb{X} \odot (\mathbf{K}^{\mathbb{2}}\mathbf{K}\mathbf{K}^{\mathbb{2}}) + \mathbf{K}^{\mathbb{2}}(\mathbf{K} \odot \mathbb{X})\mathbf{K}^{\mathbb{2}} - ((\mathbf{K}^{\mathbb{2}}\mathbf{K}) \odot \mathbb{X})\mathbf{K}^{\mathbb{2}} - \mathbf{K}^{\mathbb{2}}((\mathbf{K}\mathbf{K}^{\mathbb{2}}) \odot \mathbb{X}),$$
$$\boldsymbol{b} = (\mathbf{K}^{\mathbb{2}}\mathrm{diag}(\mathbb{X})\mathbf{K}^{\mathbb{2}} + (\mathbf{K}^{\mathbb{2}}\mathbf{K}^{\mathbb{2}}) \odot \mathbb{X} - \mathbf{K}^{\mathbb{2}}(\mathbf{K}^{\mathbb{2}} \odot \mathbb{X}) - (\mathbf{K}^{\mathbb{2}} \odot \mathbb{X})\mathbf{K}^{\mathbb{2}})\mathbf{1},$$

which leads to the solution $\hat{\boldsymbol{a}}_V^{\mathrm{Stein}} = (\boldsymbol{\Lambda} + \eta\mathbf{I})^{-1}\boldsymbol{b}$.

A.3.5 Experimental detals

Neural network based sampler experiment

For the training task, we use a one hidden layer neural network with 20 hidden units to compute the noise variance and the moving direction of the next update. In a nutshell it takes the ith coordinate of the current position and the gradient $\boldsymbol{\theta}_t(i), \nabla_t(i)$ as the inputs, and output the corresponding coordinate of the moving direction $\Delta_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t, \nabla_t)(i)$ and the noise variance $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t, \nabla_t)(i)$. Softplus non-linearity is used for the hidden layer and to compute the noise variance we apply ReLU activation to ensure non-negativity. The step-size ζ is selected as 10^{-5} which is tuned on the KDE approach. For SGLD step-size 10^{-5} also returns overall good results.

The training process is the following. For each iteration, we simulate the approximate sampler for $T = 10$ transitions and sum over the variational lower-bounds computed on the samples of every step. Concretely, the maximisation objective is an MC estimate of

$$\mathcal{L}(\boldsymbol{\phi}) = \sum_{t=1}^{T} \mathcal{L}_{\mathrm{VI}}(q_t),$$

where $q_t(\boldsymbol{\theta})$ is implicitly defined by the marginal distribution of $\boldsymbol{\theta}_t$ that is dependent on $\boldsymbol{\phi}$ and $q_0(\boldsymbol{\theta}_0)$. The simulated samples at time T are stored to initialise the Markov chain for the next iteration, i.e. $q_0(\boldsymbol{\theta}_0) = \frac{1}{K}\sum_{k=1}^{K}\delta(\boldsymbol{\theta}_0 = \boldsymbol{\theta}_T^k)$ with $\boldsymbol{\theta}_T^k$ obtained from the last iteration. For every 50 iterations we restart the simulation by randomly sampling the locations from the prior, meaning that $q_0(\boldsymbol{\theta}_0) = p(\boldsymbol{\theta}_0)$ at this iteration. The MAP baseline considers an alternative objective function by removing the $\log q_t(\boldsymbol{\theta}_t)$ term from the above MC-VI objective. Early stopping is applied using the validation dataset, and the learning rate is set to 0.001, the number of epochs is set to 500.

We perform hyper-parameter search for the kernel, i.e. a grid search on the bandwidth $\sigma^2 \in \{0.25, 1.0, 4.0, 10.0, \text{median trick}\}$ and $\eta \in \{0.01, 0.1, 0.5, 1.0, 2.0\}$. We found the median heuristic is sufficient for the KDE and Stein approaches. However, we failed to

obtain desirable results using the score matching estimator with median heuristics, and for other settings the score matching approach underperforms when compared to KDE and Stein methods.

Entropy-regularised BEGAN experiments

In the experiment, we construct a deconvolutional net for the generator and a convolutional auto-encoder for the discriminator. The convolutional encoder consists of 3 convolutional layers with filter width 3, stride 2, and number of feature maps [32, 64, 64]. These convolutional layers are followed by two fully connected layers with [512, 64] units. The decoder and the generative net have a symmetric architecture but with stride convolutions replaced by deconvolutions. ReLU activation function is used for all layers except the last layer of the generator, which uses sigmoid non-linearity. The reconstruction loss in use is the squared ℓ_2 norm $||\cdot||_2^2$. The randomness $p_0(\boldsymbol{z})$ is selected as uniform distribution in [-1, 1] as suggested in the original paper [Berthelot et al., 2017]. The mini-batch size is set to $K = 100$. Learning rate is initialised at 0.0002 and decayed by 0.9 every 10 epochs, which is tuned on the KDE model. The selected γ and α values are: for KDE estimator approach $\gamma = 0.3, \alpha\gamma = 0.05$, for score matching estimator approach $\gamma = 0.3, \alpha\gamma = 0.1$, and for Stein approach $\gamma = 0.5$ and $\alpha\gamma = 0.3$. The presented results use the KDE plug-in estimator for the entropy estimates (used to tune β) for the KDE and score matching approaches. Initial experiments found that for the Stein approach, using the KDE entropy estimator works slightly worse than the proxy loss, thus we report results using the proxy loss. An advantage of using the proxy loss is that it directly relates to the approximate gradient. Furthermore we empirically observe that the performance of the Stein approach is much more robust to the selection of γ and α when compared to the other two methods.

Appendix B

Optional Materials

B.1 Other divergences

Although not usually used in approximate inference context, in this section we briefly review another two families of divergences, namely f-divergences and Bregman divergences. Both of them are very general divergence families: they contain many useful cases, and more interestingly, they have some nice connections to the two KL divergences.

f-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 B.1. *(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*

$$\mathrm{D}_f[p||q] = \int q(\boldsymbol{\theta}) f\left(\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right) d\boldsymbol{\theta}, \quad p, q \in \mathcal{P}. \tag{B.1}$$

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 $f(x) = \frac{4}{1-\alpha^2}(1 - x^{\frac{1+\alpha}{2}}) - \frac{2}{1-\alpha}(x-1)$. 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. This means many approximate inference algorithms using α-divergence can be adapted to the general f-divergence case without major modification, which also justifies the focus of α-divergences in this thesis.

Bregman divergences

Bregman divergences were introduced by Bregman [1967] as an important tool for geometry studies. It also use convex functions defined on a convex set Ω.

Definition B.2. *(Bregman divergence) Given a strictly convex, and differentiable function $f : \Omega \to \mathbb{R}$, the corresponding Bregman divergence on $\mathcal{P}$ is defined as a function $\mathrm{B}_f[\cdot||\cdot] : \Omega \times \Omega \to \mathbb{R}$ with the following form*

$$\mathrm{B}_f[r||s] = f(r) - f(s) - \langle \nabla f(s), r - s \rangle_\Omega, \quad r, s \in \Omega, \tag{B.2}$$

where $\langle \cdot, \cdot \rangle_\Omega$ denotes the inner product defined on Ω.

Bregman divergences have also been extensively applied to design and analyse algorithms for optimisation. A widely used algorithm called *mirror gradient descent* [Beck and Teboulle, 2003; Nemirovski and Yudin, 1983] can be viewed as solving an optimisation problem using proximal methods with a "regulariser" as the Bregman divergence between the current and the proposed locations

$$\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}} \quad \{\langle \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(\boldsymbol{\theta}_t) \rangle + \frac{1}{\gamma_t} \mathrm{B}_f[\boldsymbol{\theta}||\boldsymbol{\theta}_t]\}. \tag{B.3}$$

This and related optimisation techniques have recently been incorporated to approximate inference algorithms in order to improve convergence, e.g. see Altosaar et al. [2017]; Dai et al. [2016a]; Khan et al. [2015].

To conclude the review of divergences, we include a theoretical result proved by Amari [2009], which shows the deep connections between α-divergence and the above two families of divergences.

Proposition B.1. *([Amari, 2009]) Amari's α-divergence is the unique class of divergence sitting at the intersection of f-divergences and Bregman divergences classes.*

Seeing why α-divergence is a special case of Bregman divergence is slightly involved. In this case Ω is defined as the set $\Omega = \{k_\alpha(p) : p \in \mathcal{P}\}$ with

$$k_\alpha(p(\boldsymbol{\theta})) = \frac{2}{1-\alpha}\left(p(\boldsymbol{\theta})^{\frac{1-\alpha}{2}} - 1\right), \quad \alpha \neq 1$$
$$k_\alpha(p(\boldsymbol{\theta})) = \log p(\boldsymbol{\theta}), \quad \alpha = 1,$$

and the inner product is defined as $\langle g,h\rangle_\Omega = \int g(\boldsymbol{\theta})h(\boldsymbol{\theta})d\boldsymbol{\theta}$. Then simple calculations show that $D^A_\alpha[p||q] = B_{f_\alpha}[k_\alpha(p)||k_\alpha(q)]$ with $f_\alpha(g(\boldsymbol{\theta})) = \frac{2}{1+\alpha}\int k_\alpha^{-1}(g(\boldsymbol{\theta}))d\boldsymbol{\theta}$. The statement of uniqueness is proved in Amari [2009] which is skipped here.

Perhaps one of the more well-known results compared to the above proposition is the relationship to the KL divergence in exponential family setup. Consider two exponential family distributions $p(\boldsymbol{\theta})$, $q(\boldsymbol{\theta})$ with sufficient statistic $\boldsymbol{\Phi}(\boldsymbol{\theta})$ and natural parameters $\boldsymbol{\lambda}_p$, $\boldsymbol{\lambda}_q$, respectively. Recall the log partition function

$$A(\boldsymbol{\lambda}) := \log \int \exp[\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{\theta})]d\boldsymbol{\theta}$$

is convex in $\boldsymbol{\lambda}$, and here the inner product is defined as $\langle \boldsymbol{x},\mathbf{y}\rangle := \boldsymbol{x}^{\mathrm{T}}\mathbf{y}$. Then simple calculation reveals that

$$\begin{aligned}\mathrm{KL}[p||q] &= A(\boldsymbol{\lambda}_q) - A(\boldsymbol{\lambda}_p) - \langle \boldsymbol{\lambda}_q - \boldsymbol{\lambda}_p, \mathbb{E}_p[\boldsymbol{\Phi}(\boldsymbol{\theta})]\rangle \\ &= A(\boldsymbol{\lambda}_q) - A(\boldsymbol{\lambda}_p) - \langle \boldsymbol{\lambda}_q - \boldsymbol{\lambda}_p, \nabla_{\boldsymbol{\lambda}_p} A(\boldsymbol{\lambda}_p)\rangle \quad \text{// Proposition 2.1} \\ &= \mathrm{B}_A[\boldsymbol{\lambda}_q||\boldsymbol{\lambda}_p].\end{aligned} \tag{B.4}$$

B.2 Sketching variational methods by constraint relaxations

This is an optional section continuing the discussion of Section 2.3.3 in the main text. The material is adapted from my NIPS 2016 approximate inference workshop abstract (see publication page).

B.2.1 Further constraint relaxations by weighted averaging

In EP we need N Lagrange multipliers $\{\boldsymbol{\lambda}_{-n}\}$ because of the individual moment matching constraint $\mathbb{E}_q[\boldsymbol{\Phi}] = \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}]$. This can bring a large amount of memory burden for large datasets and "big" models. To solve this issue, we start from the re-formulated energy function (2.34), and then replace the N number of equality constraints $q = \tilde{p}_n$ by a single one that we call *weighted averaged moment matching*: $\mathbb{E}_q[\boldsymbol{\Phi}] = \sum_n w_n \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}]$, with $\boldsymbol{w} = (w_1,...,w_n)$ denote the weighing vector that sum to 1. The motivation here is to reduce the number of Lagrange multipliers (thus saving memory), but still to ensure q is close to the tilted distributions in some averaged measure. Empirical evaluations have shown state-of-the-art performances for this relaxation method [Hernández-Lobato et al., 2016] even though it is a very bad approximation to the N equality constraints in (2.27).

We proceed to solve the corresponding constrained optimisation problem. We also apply the KL duality (2.29) to (2.34) (but for simplicity now we directly use $\boldsymbol{\lambda}_q^T\boldsymbol{\Phi}(\theta)$), and denote

the transformed energy (in a similar way as in (2.30)) as $\mathcal{F}_{\boldsymbol{\alpha}}(\{\tilde{p}_n\}, q, \boldsymbol{\lambda}_q)$. Now we have the following Lagrangian

$$\min_{q,\{\tilde{p}_n\},\boldsymbol{\lambda}_q} \max_{\lambda,\{\nu_n\},\nu} \mathcal{F}_{\boldsymbol{\alpha}}(\{\tilde{p}_n\}, q, \boldsymbol{\lambda}_q) + \boldsymbol{\lambda}^T(\mathbb{E}_q[\boldsymbol{\Phi}] - \sum_n w_n \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}]) + \ldots \tag{B.5}$$

where we have omitted the multipliers for the normalisation constraints. This returns $\tilde{p}_n(\boldsymbol{\theta}) = \frac{1}{Z_n} p_0(\boldsymbol{\theta}) f_n(\boldsymbol{\theta})^{\alpha_n} \exp\left[\alpha_n w_n \boldsymbol{\lambda}^T \boldsymbol{\Phi}(\boldsymbol{\theta})\right]$, but the fixed point condition for q becomes $\left(\sum_n \frac{1}{\alpha_n} - 1\right) \boldsymbol{\lambda}_q = \boldsymbol{\lambda}$, indicating $\boldsymbol{\lambda}$ as a function of $\boldsymbol{\lambda}_q$. Thus we can directly obtain a single-loop algorithm for the following minimisation

$$\min_{\boldsymbol{\lambda}_q} \left(\sum_n \frac{1}{\alpha_n} - 1\right) \log Z_q - \sum_n \frac{1}{\alpha_n} \log \int p_0(\boldsymbol{\theta}) f_n(\boldsymbol{\theta})^{\alpha_n} \exp\left[\beta_n \boldsymbol{\lambda}_q \boldsymbol{\Phi}(\boldsymbol{\theta})\right] d\boldsymbol{\theta} \tag{B.6}$$

with $\beta_n = \left(\sum_m \frac{1}{\alpha_m} - 1\right) \alpha_n w_n$. Black-box alpha (BB-$\alpha$) [Hernández-Lobato et al., 2016] is a special case of the above by defining $\alpha_n = \alpha, \forall n$ and $w_n = (\frac{1}{\alpha_n} - \frac{1}{N}) / (\sum_m \frac{1}{\alpha_m} - 1) = \frac{1}{N}$, which leads to $\beta_n = 1 - \frac{\alpha}{N}$. In this case the weighing vector $\boldsymbol{w}$ is implicitly defined by the choice of α.

> **Remark.** The original derivation of BB-α in Hernández-Lobato et al. [2016] directly started from the power EP energy (2.35) and tied the local parameters $\boldsymbol{\lambda}_n$ (or in other words it added new constraints in the dual space). Our derivation here provides a proper justification from a constrained primal energy optimisation perspective.

B.2.2 Distributed black-box alpha

Variational methods have been shown to be very efficient for distributed computing, e.g. see Gelman et al. [2014]; Hasenclever et al. [2017]; Xu et al. [2014] for EP and Broderick et al. [2013] for VI. In our set-up this is equivalent to a different decoupling strategy. It first groups the factors into K subset-level factors $F_k = \prod f_{n_k}$ with the corresponding index set $N_k = \{n_k\}$, where usually $N_i \cap N_j = \emptyset$, $i \neq j$ and $\cup_k N_k = \{1, \ldots, N\}$, then it performs EP at the group level. Now we rewrite the variational free energy and also assume for simplicity $\alpha_k \neq 0$ and $\alpha_{n_k} = \alpha_k$ for $n_k \in N_k$:

$$\mathcal{F}_{\mathrm{VFE}}(q) = \left(1 - \sum_k \frac{1}{\alpha_k}\right) \mathrm{KL}[q||p_0] - \sum_k \frac{1}{\alpha_k} \mathbb{E}_q \left[\log \frac{p_0(\boldsymbol{\theta}) F_k(\boldsymbol{\theta})^{\alpha_k}}{q(\boldsymbol{\theta})}\right]. \tag{B.7}$$

One can easily show the distributed EP objective by repeating the decoupling and Lagrangian computation procedures as in Section 2.3.3, with the moment-matching constraints applied

to $\tilde{p}_k$ for subset-level factor F_k rather than for individual factor f_n. An equivalent derivation starts from power EP in Section 2.3.3 but with an extra set of constraints restricting $\tilde{p}_i = \tilde{p}_j, \forall i, j \in N_k$. Simple calculations reveal that in this case $1/\alpha_k$ in (B.7) equals to the sum of $1/\alpha_{n_k}$ for all $n_k \in N_k$.

Monte Carlo (MC) methods are deployed to compute these moments since now we incorporate multiple factors in the second integral. The EP/MC mixed approach combines the advantages from both worlds: it provides more accurate approximations than full EP since it's less "local", while it remains faster than full MC methods (as the tilted distribution contains less difficult factors) and is straight-forward to parallelise. In an extreme case where $K = 1$ and $\alpha \neq 0, 1$, the above derivation recovers the VR bound (see Chapter 4), with the q distribution restricted to exponential families.

We further present a mixed approach that nests BB-α in distributed EP, and again for simplicity we assume $\alpha_{n_k} = \alpha_k$ for $n_k \in N_k$. We still decouple all the q distributions associated with factor $\tilde{f}_n$ to $\tilde{p}_n$ as in (2.34), but then relax the equality constraint to $\frac{1}{|N_k|}\sum_{n_k \in N_k} \mathbb{E}_{\tilde{p}_{n_k}}[\boldsymbol{\Phi}] = \mathbb{E}_q[\boldsymbol{\Phi}]$, $\forall k$. Solving the Lagrangian and defining $\boldsymbol{\lambda}_q = \sum_k \boldsymbol{\lambda}_k$ returns the following energy:

$$(\sum_k \frac{|N_k|}{\alpha_k} - 1)\log Z_q - \sum_k \frac{1}{\alpha_k} \sum_{n_k \in N_k} \log \int p_0(\boldsymbol{\theta}) f_{n_k}(\boldsymbol{\theta})^{\alpha_k} \exp\left[(\boldsymbol{\lambda}_q - \frac{\alpha_k}{|N_k|}\boldsymbol{\lambda}_k)^T \boldsymbol{\Phi}(\boldsymbol{\theta})\right] d\boldsymbol{\theta}, \tag{B.8}$$

This means the local parameters $\boldsymbol{\lambda}_k$ are updated in a BB-α fashion, while the final approximation q is constructed in an EP way. Potentially this approach can both reduce the approximation bias of BB-α (as we use more than one set of local parameters) and be computationally faster than distributed EP (as often now the moments for the tilted distribution become tractable).

B.2.3 Nested variational methods

We further provide several EP-like recipes for latent variable models. As we shall see again, different decoupling and constraint relaxation strategies return algorithms that have different global and local behaviour.

Assume now the exact posterior becomes $p(\boldsymbol{\theta}, \{\boldsymbol{z}_n\}|\{\boldsymbol{x}_n\}) \propto p_0(\boldsymbol{\theta}) \prod_n p_0(\boldsymbol{z}_n) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta})$. We note that the algorithms discussed below also applies to models that has *intermediate-level variables*, i.e. those latent variables that are attached to a subset of data. The goal is to approximate the exact posterior of both $\boldsymbol{\theta}$ and $\boldsymbol{z}_n$, and we assume a factorised approximation $q(\boldsymbol{\theta}, \{\boldsymbol{z}_n\}) = q(\boldsymbol{\theta}) \prod_n q(\boldsymbol{z}_n|\boldsymbol{x}_n)$. Note that in VI/VB literature the local variational approximation is often denoted as $q(\boldsymbol{z}_n)$. However as at optimum $q(\boldsymbol{z}_n)$ depends on $\boldsymbol{x}_n$, here we explicitly write down this dependence as $q(\boldsymbol{z}_n|\boldsymbol{x}_n)$. This notation is also convenient for

the application of amortised inference (see Section 2.2.4) which uses a recognition model (i.e. sharing parameters between $q(\boldsymbol{z}_n)$) to parameterise the local variational distribution.

Full power EP/BB-α treatment

We repeat the term rearranging and argument decoupling procedure as in (2.34). This returns:

$$\mathcal{F}_{\boldsymbol{\alpha}}(q,\{\tilde{p}_n\}) = \left(1-\sum_n \frac{1}{\alpha_n}\right)\mathrm{KL}[q(\boldsymbol{\theta})||p_0(\boldsymbol{\theta})] + \sum_n \left(1-\frac{1}{\alpha_n}\right)\mathrm{KL}[q(\boldsymbol{z}_n|\boldsymbol{x}_n)||p_0(\boldsymbol{z}_n)] \\ -\sum_n \frac{1}{\alpha_n}\mathbb{E}_{\tilde{p}_n}\left[\log \frac{p_0(\boldsymbol{\theta})p_0(\boldsymbol{z}_n)p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})^{\alpha_n}}{\tilde{p}_n(\boldsymbol{\theta},\boldsymbol{z}_n)}\right], \tag{B.9}$$

subject to $\tilde{p}_n(\boldsymbol{\theta},\boldsymbol{z}_n) = q(\boldsymbol{\theta})q(\boldsymbol{z}_n|\boldsymbol{x}_n), \forall n$. The next step is to relax the equality constraint to moment-matching constraints denoted as $\mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}(\boldsymbol{\theta},\boldsymbol{z}_n)] = \mathbb{E}_q[\boldsymbol{\Phi}(\boldsymbol{\theta},\boldsymbol{z}_n)]$. The choice of the sufficient statistic $\boldsymbol{\Phi}$ also plays an important role here, and for simplicity we omit the dependence to the observations $\boldsymbol{x}_n$, and assume a *factorised sufficient statistic*, i.e. $\boldsymbol{\Phi}(\boldsymbol{\theta},\boldsymbol{z}_n) = [\boldsymbol{\Phi}(\boldsymbol{\theta}),\boldsymbol{\psi}(\boldsymbol{z}_n)]$. We leave the general case to future work.

Now we proceed to solve the fixed points using similar methods presented in the main text. We still use the KL duality (2.29) for $q(\boldsymbol{\theta})$, and write that for the latent variables as

$$-\mathrm{KL}[q(\boldsymbol{z}_n|\boldsymbol{x}_n)||p_0(\boldsymbol{z}_n)] = \min_{\eta(\boldsymbol{z}_n,\boldsymbol{x}_n)} -\mathbb{E}_q[\eta(\boldsymbol{z}_n,\boldsymbol{x}_n)] + \log \mathbb{E}_{p_0}\left[\exp[\eta(\boldsymbol{z}_n,\boldsymbol{x}_n)]\right]. \tag{B.10}$$

To simplify computations we assume $\eta(\boldsymbol{z}_n,\boldsymbol{x}_n) = \boldsymbol{\eta}(\boldsymbol{x}_n)^T\boldsymbol{\psi}(\boldsymbol{z}_n)$ w.l.o.g. Substitution into (B.9) returns the modified form $\mathcal{F}_{\boldsymbol{\alpha}}(q,\{\tilde{p}_n\},\boldsymbol{\lambda}_q,\{\boldsymbol{\eta}(\boldsymbol{x}_n)\})$ which is defined in a similar way as (2.30). Associating Lagrange multiplier $\boldsymbol{\lambda}_{-n}$ and $\boldsymbol{\eta}_n$ for the moment matching constraints of $\boldsymbol{\Phi}$ and $\boldsymbol{\psi}$ respectively, we have the following Lagrangian

$$\mathcal{F}_{\boldsymbol{\alpha}}(q,\{\tilde{p}_n\},\boldsymbol{\lambda}_q,\{\boldsymbol{\eta}(\boldsymbol{x}_n)\}) + \sum_n \frac{1}{\alpha_n}\left[\boldsymbol{\lambda}_{-n}^T(\mathbb{E}_q[\boldsymbol{\Phi}] - \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}]) + \boldsymbol{\eta}_n^T(\mathbb{E}_q[\boldsymbol{\psi}] - \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\psi}])\right] + \ldots \tag{B.11}$$

where we have omitted the multipliers for the normalisation constraints. Finding the fixed point wrt. $\tilde{p}_n$ returns the following tilted distribution:

$$\tilde{p}_n(\boldsymbol{\theta},\boldsymbol{z}_n) = \frac{1}{Z_n}p_0(\boldsymbol{\theta})p_0(\boldsymbol{z}_n)p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})^{\alpha_n}\exp\left[\boldsymbol{\lambda}_{-n}^T\boldsymbol{\Phi}(\boldsymbol{\theta}) + \boldsymbol{\eta}_n^T\boldsymbol{\psi}(\boldsymbol{z}_n)\right]. \tag{B.12}$$

Also zeroing the gradient wrt. q yields the fixed point conditions (up to a constant)

$$\left(\sum_n \frac{1}{\alpha_n} - 1\right)\boldsymbol{\lambda}_q = \sum_n \frac{1}{\alpha_n}\boldsymbol{\lambda}_{-n},$$

just like in power EP, and $(1-\alpha_n)\boldsymbol{\eta}(\boldsymbol{x}_n) = \boldsymbol{\eta}_n$. The second one is similar to the BB-α case so we directly assume it holds and drop the optimisation problem of $\boldsymbol{\eta}_n$. Furthermore, substituting $\tilde{p}_n$ back into (B.11) and zeroing the gradient wrt. q, we arrive at the following power EP energy

$$\begin{aligned}(\sum_n \frac{1}{\alpha_n} - 1)\log \int p_0(\boldsymbol{\theta}) \exp\left[\boldsymbol{\lambda}_q^T \boldsymbol{\Phi}(\boldsymbol{\theta})\right] d\boldsymbol{\theta} + \sum_n (\frac{1}{\alpha_n} - 1)\log \int p_0(\boldsymbol{z}_n) \exp\left[\boldsymbol{\eta}(\boldsymbol{x}_n)^T \boldsymbol{\psi}(\boldsymbol{z}_n)\right] d\boldsymbol{z}_n \\ - \sum_n \frac{1}{\alpha_n} \log \int p_0(\boldsymbol{\theta}) p_0(\boldsymbol{z}_n) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta})^{\alpha_n} \exp\left[\boldsymbol{\lambda}_{-n}^T \boldsymbol{\Phi}(\boldsymbol{\theta}) + (1-\alpha_n)\boldsymbol{\eta}(\boldsymbol{x}_n)^T \boldsymbol{\psi}(\boldsymbol{z}_n)\right] d\boldsymbol{\theta} d\boldsymbol{z}_n \\ \text{subject to } (\sum_n \frac{1}{\alpha_n} - 1)\boldsymbol{\lambda}_q = \sum_n \frac{1}{\alpha_n} \boldsymbol{\lambda}_{-n}.\end{aligned} \tag{B.13}$$

To make the KL duality tight we define the approximation q obtained from the dual energy optimisation as

$$q(\boldsymbol{\theta}) = \frac{1}{Z_q} p_0(\boldsymbol{\theta}) \exp\left[\boldsymbol{\lambda}_q^T \boldsymbol{\Phi}(\boldsymbol{\theta})\right], \quad q(\boldsymbol{z}_n|\boldsymbol{x}_n) = \frac{1}{Z_q(\boldsymbol{x}_n)} p_0(\boldsymbol{z}_n) \exp\left[\boldsymbol{\eta}(\boldsymbol{x}_n)^T \boldsymbol{\psi}(\boldsymbol{z}_n)\right].$$

We also present a much cleaner version of the energy function by substituting the q distributions into the power EP energy (with a further definition $\boldsymbol{\lambda}_n = \frac{1}{\alpha_n}(\boldsymbol{\lambda}_q - \boldsymbol{\lambda}_{-n})$) and rearranging terms:

$$\begin{aligned}-\log \int p_0(\boldsymbol{\theta}) \exp\left[\boldsymbol{\lambda}_q^T \boldsymbol{\Phi}(\boldsymbol{\theta})\right] d\boldsymbol{\theta} - \sum_n \frac{1}{\alpha_n} \log \mathbb{E}_{q(\boldsymbol{\theta})q(\boldsymbol{z}_n|\boldsymbol{x}_n)} \left[\left(\frac{p_0(\boldsymbol{z}_n) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta})}{q(\boldsymbol{z}_n|\boldsymbol{x}_n) \exp\left[\boldsymbol{\lambda}_n^T \boldsymbol{\Phi}(\boldsymbol{\theta})\right]}\right)^{\alpha_n}\right] \\ \text{subject to } \boldsymbol{\lambda}_q = \sum_n \boldsymbol{\lambda}_n.\end{aligned} \tag{B.14}$$

The BB-α variant can be derived similarly by further constraint relaxations. The detailed derivation is omitted here, but in summary we keep the constraint for $\boldsymbol{\psi}$ but modify the other constraint by $\mathbb{E}_{q(\boldsymbol{\theta})}[\boldsymbol{\Phi}] = \frac{1}{N}\sum_n \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}]$. One can show that the dual energy becomes (again after rearranging terms)

$$-\sum_n \frac{1}{\alpha} \log \mathbb{E}_{q(\boldsymbol{\theta})q(\boldsymbol{z}_n|\boldsymbol{x}_n)} \left[\left(\frac{p_0(\boldsymbol{\theta})^{\frac{1}{N}} p_0(\boldsymbol{z}_n) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta})}{q(\boldsymbol{\theta})^{\frac{1}{N}} q(\boldsymbol{z}_n|\boldsymbol{x}_n)}\right)^{\alpha}\right], \tag{B.15}$$

with $q(\boldsymbol{\theta})$ and $q(\boldsymbol{z}_n|\boldsymbol{x}_n)$ defined as exponential family distributions. Extensions to general q distributions can be justified using Rényi divergences minimisation methods presented in Chapter 4.

Nesting power-EP/BB-α in VI

This section discusses how to use power EP/BB-α as an inner loop computation for variational inference. It starts from (B.9), but only applies constraint relaxations to the moments for the latent variables. In other words, now the constraints are $\mathbb{E}_q[\boldsymbol{\psi}(\boldsymbol{z}_n)] = \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\psi}(\boldsymbol{z}_n)]$ and $q(\boldsymbol{\theta}) = \tilde{p}_n(\boldsymbol{\theta})$. For computational convenience we assume $\tilde{p}_n(\boldsymbol{\theta}, \boldsymbol{z}_n) = q(\boldsymbol{\theta})\tilde{p}_n(\boldsymbol{z}_n|\boldsymbol{\theta})$ w.l.o.g. so that there is only one set of constraints $\mathbb{E}_{q(\boldsymbol{z}_n|\boldsymbol{x}_n)}[\boldsymbol{\psi}(\boldsymbol{z}_n)] = \mathbb{E}_{q(\boldsymbol{\theta})\tilde{p}_n(\boldsymbol{z}_n|\boldsymbol{\theta})}[\boldsymbol{\psi}(\boldsymbol{z}_n)]$ (besides the normalisation ones). Denote $\boldsymbol{\eta}_n$ as the associated Lagrange multipliers for the moment maching constraints, then solving the corresponding Lagrangian yields $(1-\alpha_n)\boldsymbol{\eta}(\boldsymbol{x}_n) = \boldsymbol{\eta}_n$ again, and

$$\tilde{p}_n(\boldsymbol{z}_n|\boldsymbol{\theta}) = \frac{1}{Z_n(\boldsymbol{x}_n, \boldsymbol{\theta})} p_0(\boldsymbol{z}_n) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta})^{\alpha_n} \exp\left[(1-\alpha_n)\boldsymbol{\eta}(\boldsymbol{x}_n)^T \boldsymbol{\psi}(\boldsymbol{z}_n)\right]. \tag{B.16}$$

Note that now the normalising constant $Z_n(\boldsymbol{x}_n, \boldsymbol{\theta})$ is also a function of $\boldsymbol{\theta}$. We still keep the free-form optimisation for $q(\boldsymbol{\theta})$. Substitution of $\tilde{p}_n(\boldsymbol{z}_n|\boldsymbol{\theta})$ into the Lagrangian returns a "mixed form" of energy (again after rearranging terms)

$$\min_q \mathrm{KL}[q(\boldsymbol{\theta})||p_0(\boldsymbol{\theta})] - \sum_n \mathbb{E}_{q(\boldsymbol{\theta})}\left[\frac{1}{\alpha_n}\log \mathbb{E}_{q(\boldsymbol{z}_n|\boldsymbol{x}_n)}\left[\left(\frac{p_0(\boldsymbol{z}_n)p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})}{q(\boldsymbol{z}_n|\boldsymbol{x}_n)}\right)^{\alpha_n}\right]\right], \tag{B.17}$$

where $q(\boldsymbol{z}_n|\boldsymbol{x}_n)$ is restricted to have an exponential family form:

$$q(\boldsymbol{z}_n|\boldsymbol{x}_n) \propto p_0(\boldsymbol{z}_n)\exp\left[\boldsymbol{\eta}(\boldsymbol{x}_n)^T\boldsymbol{\psi}(\boldsymbol{z}_n)\right].$$

Remark. A naive modification of the constraints to $\mathbb{E}_q[\boldsymbol{\Phi}(\boldsymbol{\theta})] = \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}(\boldsymbol{\theta})]$ and $q(\boldsymbol{z}_n|\boldsymbol{x}_n) = \tilde{p}_n(\boldsymbol{z}_n)$ does *not* lead to a nested approach of VI in power-EP/BB-α. We omit the details here, but a try-out for the BB-α variant returns the following energy

$$\min_q \sum_n \mathrm{KL}[q(\boldsymbol{z}_n|\boldsymbol{x}_n)||p_0(\boldsymbol{z}_n)] - \sum_n \frac{1}{\alpha}\mathbb{E}_{q(\boldsymbol{z}_n|\boldsymbol{x}_n)}\log \mathbb{E}_{q(\boldsymbol{\theta})}\left[\left(\frac{p_0(\boldsymbol{\theta})^{1/N}p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})}{q(\boldsymbol{\theta})^{1/N}}\right)^{\alpha}\right], \tag{B.18}$$

with $q(\boldsymbol{\theta})$ also restricted as an exponential family distribution.

Nesting VI in power-EP/BB-α

Recall in the beginning we mentioned that the decoupling process plays a crucial role on the dual form of the energy. Now we decouple the variational distributions in a different way:

$$\mathcal{F}_{\boldsymbol{\alpha}}(q,\{\tilde{p}_n\}) = \left(1-\sum_n \frac{1}{\alpha_n}\right)\mathrm{KL}[q(\boldsymbol{\theta})||p_0(\boldsymbol{\theta})] - \sum_n \frac{1}{\alpha_n}\mathbb{E}_{\tilde{p}_n(\boldsymbol{\theta})q(\boldsymbol{z}_n|\boldsymbol{x}_n)}\left[\log \frac{p_0(\boldsymbol{\theta})p_0(\boldsymbol{z}_n)^{\alpha_n}p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})^{\alpha_n}}{\tilde{p}_n(\boldsymbol{\theta})q(\boldsymbol{z}_n|\boldsymbol{x}_n)^{\alpha_n}}\right], \tag{B.19}$$

In this case we only relax the constraints to $\tilde{p}_n(\boldsymbol{\theta})$ to moment matching: $\mathbb{E}_q[\boldsymbol{\Phi}] = \mathbb{E}_{\tilde{p}_n}[\boldsymbol{\Phi}]$. We also reuse the derivations in Section 2.3.3 of the main text by noticing now

$$\log f_n(\boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{z}_n|\boldsymbol{x}_n)}\left[\log \frac{p_0(\boldsymbol{z}_n)p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})}{q(\boldsymbol{z}_n|\boldsymbol{x}_n)}\right], \tag{B.20}$$

and thus omit the detailed expression of the energy functions. Readers are referred to (2.35) in the main text. This algorithm (with the power EP variant) returns the same stationary points as VI if the model is conjugate, i.e. $\log p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})$ is linear in $\boldsymbol{\Phi}(\boldsymbol{\theta})$.

Variational message passing between tilted distributions

Applying similar derivations as in B.2.3 to the decoupling B.9 returns a slightly different algorithm that applies variational message passing (VMP) [Winn and Bishop, 2005] to the tilted distributions. Here we also directly enforce the factorisation assumption, i.e. $\tilde{p}(\boldsymbol{\theta},\boldsymbol{z}_n) = \tilde{p}_n(\boldsymbol{\theta})\tilde{p}_n(\boldsymbol{z}_n)$. This returns the same fixed point as with the joint tilted distribution version if we optimise the free energy under equality constraints. However the stationary points differs from those derived above when relaxing the constraint to moment matching. This is because, as $\tilde{p}$ factorises, the fixed point solution of the Lagrangian should contain "messages" sent between $\tilde{p}_n(\boldsymbol{\theta})$ and $\tilde{p}(\boldsymbol{z}_n)$. To be precise, we solve the fixed point of the Lagrangian (B.11) (but with factorised $\tilde{p}$). The fixed point conditions for q remain the same, but those for $\tilde{p}$ become

$$\tilde{p}_n(\boldsymbol{\theta}) = \frac{1}{Z_n}p_0(\boldsymbol{\theta})\exp\left[\boldsymbol{\lambda}_{-n}^T\boldsymbol{\Phi}(\boldsymbol{\theta}) + \alpha_n\mathbb{E}_{\tilde{p}(\boldsymbol{z}_n)}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})]\right], \tag{B.21}$$

$$\tilde{p}_n(\boldsymbol{z}_n) = \frac{1}{Z_n(\boldsymbol{x}_n)}p_0(\boldsymbol{z}_n)\exp\left[\boldsymbol{\eta}_n^T\boldsymbol{\psi}(\boldsymbol{z}_n) + \alpha_n\mathbb{E}_{\tilde{p}_n(\boldsymbol{\theta})}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})]\right]. \tag{B.22}$$

Substituting these new fixed point equations back to the Lagrangian returns a different dual energy function

$$\begin{aligned}(\sum_n \frac{1}{\alpha_n} - 1)\log Z_q + \sum_n (\frac{1}{\alpha_n} - 1)\log Z_q(\boldsymbol{x}_n) - \sum_n \frac{1}{\alpha_n}(\log Z_n + \log Z_n(\boldsymbol{x}_n)) \\ + \sum_n \mathbb{E}_{\tilde{p}_n(\boldsymbol{\theta})\tilde{p}_n(\boldsymbol{z}_n)}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta})] \\ \text{subject to } (\sum_n \frac{1}{\alpha_n} - 1)\boldsymbol{\lambda}_q = \sum_n \frac{1}{\alpha_n}\boldsymbol{\lambda}_{-n}.\end{aligned} \tag{B.23}$$

The BB-α version can also be derived similarly which we omit here.

In general the above algorithm behaves differently when compared to VMP, since 1) the local messages are computed using the tilted distributions and 2) for non-conjugate models the tilted distributions contain more complex structure compared to q. In below we provide a fixed point iteration procedure to find the stationary points of the above constrained optimisation problem.

1 Select a datapoint $\boldsymbol{x}_n$;

2 Compute cavity distribution $q_{-n}(\boldsymbol{\theta})$ (either using power EP or BB-α);

3 Compute cavity distribution $q_{-1}(\boldsymbol{z}_n)$ when $\alpha_n \neq 1$, otherwise $q_{-1}(\boldsymbol{z}_n) = p_0(\boldsymbol{z}_n)$;

4 Run VMP/VI on this single datapoint $\boldsymbol{x}_n$ with prior terms changed to $q_{-n}(\boldsymbol{\theta})$ and $q_{-1}(\boldsymbol{z}_n)$. This procedure calculates $\tilde{p}_n(\boldsymbol{\theta})$ and $\tilde{p}(\boldsymbol{z}_n)$;

5 Compute the moment matching update $q_{new}(\boldsymbol{\theta}) \leftarrow \text{proj}[\tilde{p}_n(\boldsymbol{\theta})]$ (similarly for $q(\boldsymbol{z}_n)$);

6 Compute the final update for q with q_{new} (either using power EP or BB-α).

B.3 VR-bound optimisation for discrete distributions

Although in Chapter 4 we mainly focused on continuous distributions, here we also briefly show that a recent variance reduction technique for discrete distributions, called variational inference with Monte Carlo objectives (VIMCO) [Mnih and Rezende, 2016], can also be applied to the MC approximated VR bounds. This proposal has also been considered in Webb and Teh [2016] and their initial evaluation showed that using Rényi divergences provides advantages over the original VIMCO approach.

Consider the gradient of the MC approximated bound w.r.t. $\boldsymbol{\phi}$, the parameter of the recognition model:

$$\nabla_{\boldsymbol{\phi}}\mathbb{E}_{\{\boldsymbol{h}_k\}}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})] = \sum_{j=1}^{K}\mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\frac{1}{1-\alpha}\log\frac{1}{K}\sum_{k=1}^{K}\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha}\nabla_{\boldsymbol{\phi}}\log q(\boldsymbol{h}_j|\boldsymbol{x})\right] - \mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\hat{w}_{\alpha,k}\nabla_{\boldsymbol{\phi}}\log q(\boldsymbol{h}_k|\boldsymbol{x})\right],$$

with $\hat{w}_{\alpha,k} = \left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha} / \sum_{k=1}^{K}\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha}$. Notice that for any function $f(\boldsymbol{h}_{k\neq j})$ we have

$$\mathbb{E}_{\{\boldsymbol{h}_k\}}[f(\boldsymbol{h}_{k\neq j})\nabla_{\boldsymbol{\phi}}\log q(\boldsymbol{h}_j|\boldsymbol{x})] = \mathbb{E}_{\{\boldsymbol{h}_{j\neq k}\}}[f(\boldsymbol{h}_{k\neq j})\mathbb{E}_{q(\boldsymbol{h}_k|\boldsymbol{x})}[\nabla_{\boldsymbol{\phi}}\log q(\boldsymbol{h}_j|\boldsymbol{x})]] = 0.$$

This means we can choose some function $f(\boldsymbol{h}_{k\neq j})$ that is correlated with the MC estimate to reduce the variance of the stochastic gradient. We follow Mnih and Rezende [2016] to define

$$\Delta_{-j} := \frac{1}{1-\alpha}\log\frac{1}{K}\sum_{k=1}^{K}\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha} - \frac{1}{1-\alpha}\log\frac{1}{K-1}\sum_{k\neq j}\left(\frac{p(\boldsymbol{h}_k,\boldsymbol{x})}{q(\boldsymbol{h}_k|\boldsymbol{x})}\right)^{1-\alpha},$$

so the gradient w.r.t. $\boldsymbol{\phi}$ can be re-written as

$$\nabla_{\boldsymbol{\phi}}\mathbb{E}_{\{\boldsymbol{h}_k\}}[\hat{\mathcal{L}}_{\alpha,K}(q;\boldsymbol{x})] = \sum_{j=1}^{K}\mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\Delta_{-j}\log q(\boldsymbol{h}_j|\boldsymbol{x})\right] - \mathbb{E}_{\{\boldsymbol{h}_k\}}\left[\hat{w}_{\alpha,k}\nabla_{\boldsymbol{\phi}}\log q(\boldsymbol{h}_k|\boldsymbol{x})\right]. \quad \text{(B.24)}$$

B.4 Going beyond mean-field: further examples

B.4.1 Invertible transformations and normalising flows

In Section 5.2 we have discussed distributions constructed by warping a random noise variable through a deep neural network. However the non-linear mapping $\mathbf{f}$ might not be invertible, resulting in an intractable density. Instead, here we introduce the idea of *invertible transformations* which allow tractable evaluation point-wise, and discuss extensions such as normalising flows.

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

$$q(\mathbf{y}) = q(\boldsymbol{z})|\det(\nabla_{\boldsymbol{z}}\mathbf{f}(\boldsymbol{z}))|^{-1}. \tag{B.25}$$

Then when density evaluation is required on a given $\mathbf{y}$, one would 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 $\det(\nabla_{\boldsymbol{z}}\mathbf{f}(\boldsymbol{z}))$. 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 $\boldsymbol{z}_T$ given the distribution of $\boldsymbol{z}_0$:

$$q(\boldsymbol{z}_T) = q(\boldsymbol{z}_0)\prod_{t=1}^{T}|\det(\nabla_{\boldsymbol{z}_{t-1}}\mathbf{f}_t(\boldsymbol{z}_{t-1}))|^{-1}. \tag{B.26}$$

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 *introduce by the 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:

  $$\boldsymbol{\theta}^{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \quad \sum_{n=1}^{N}\{\log p(\boldsymbol{z}_0^n) - \sum_{t=1}^{T}\log|\det(\nabla_{\boldsymbol{z}_{t-1}^n}\mathbf{f}_t(\boldsymbol{z}_{t-1}^n))|\}|_{\boldsymbol{z}_T^n=\boldsymbol{x}_n}. \tag{B.27}$$

  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:

$$\mathcal{L}_{VI}(q;\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}_T)}[\log p(\boldsymbol{x},\boldsymbol{z}_T)] - \mathbb{E}_{q(\boldsymbol{z}_T)}[\log q(\boldsymbol{z}_T)]. \tag{B.28}$$

  The reconstruction term can be easily calculated following the LOTUS rule (see Section 2.5.1)

$$\mathbb{E}_{q(\boldsymbol{z}_T)}[\log p(\boldsymbol{x},\boldsymbol{z}_T)] = \mathbb{E}_{q(\boldsymbol{z}_0)}[\log p(\boldsymbol{x},\mathbf{f}_T \circ \mathbf{f}_{T-1} \circ \cdots \circ \mathbf{f}_1(\boldsymbol{z}_0))],$$

  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. Some examples include:

- Individual/block mapping:
  denoting $z_t[i]$ as the i^{th} element of the vector $\boldsymbol{z}_t$

$$q(\boldsymbol{z}_0) = \prod_{i=1}^{d} q(z_0[i]), \quad z_T[i] = f_T \circ f_{T-1} \circ \cdots \circ f_1(z_0[i]),$$

  and obviously the Jacobian matrix is diagonal, and the determinant can be computed in linear time. This idea can also be generalised to block 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.

- 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 $\mathbf{y}^2$. 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

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

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

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

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 [Neal, 2011], which essentially deploys a *deterministic* dynamics in the augmented space of $\{\boldsymbol{z}, \boldsymbol{p}\}$. Importantly this flow (with Leapfrog discretisation) is *volume preserving* in the joint space $(\boldsymbol{z}, \boldsymbol{p})$, i.e. the determinant of the Jacobian is 1 [Neal, 2011]. 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.

B.4.2 Mixture and hierarchical densities for posterior approximations

Besides invertible transformations which "go deeper" and construct highly non-linear warppings, another direction for improving posterior approximations is to "go wider", i.e. using mixture densities with many simple components. For example, Jaakkola and Jordan [1998] had applied a Gaussian mixture density to approximate the intractable exact posterior, and indeed with sufficiently many components, the mixture density can approximate *any* distribution *arbitrarily well*. However, their method relies on a further approximation to the variational lower-bound and applies to only a small set of probabilistic models. In the following we present the recent retake on fitting this type of approximate densities, which introduces auxiliary variables/bounds and is compatible with the MC estimation framework.

As a simple example, consider the following mixture density:

$$q(\boldsymbol{z}|\boldsymbol{x}) = \int q(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{x})q(\boldsymbol{a}|\boldsymbol{x})d\boldsymbol{a}.$$

In the Gaussian mixture density example, $q(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{x})$ is a Gaussian distribution with its mean and variance determined by the *auxiliary variable* $\boldsymbol{a}$, and $q(\boldsymbol{a}|\boldsymbol{x})$ is typically a categorical distribution representing the probability of selecting a component. Substituting this mixture

density into the variational lower-bound, we have

$$\mathcal{L}_{\text{VI}}(q;\boldsymbol{x}) = \mathbb{E}_q[\log p(\boldsymbol{x},\boldsymbol{z})] - \mathbb{E}_q[\log \int q(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{x})q(\boldsymbol{a}|\boldsymbol{x})d\boldsymbol{a}], \tag{B.29}$$

and computing the entropy term can be very expensive if q contains many, or even infinite number of component. To address this issue, a first attempt by Gershman et al. [2012] introduced a further lower-bound to the entropy term, again using the Jensen's inequality:

$$\begin{aligned}
\mathbb{H}[q] &= -\mathbb{E}_q[\log q(\boldsymbol{z}|\boldsymbol{x})] \\
&= -\int q(\boldsymbol{a}|\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{x}) \log q(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z} d\boldsymbol{a} \\
&\geq -\int q(\boldsymbol{a}|\boldsymbol{x}) \log \left(\int q(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{x}) d\boldsymbol{z} \right) d\boldsymbol{a} \\
&= -\int q(\boldsymbol{a}|\boldsymbol{x}) \log \left(\int q(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{a}',\boldsymbol{x})q(\boldsymbol{a}'|\boldsymbol{x}) d\boldsymbol{a}' d\boldsymbol{z} \right) d\boldsymbol{a}.
\end{aligned} \tag{B.30}$$

According to the fact that the integral of the product of Gaussians is still Gaussian, the integral inside the logarithm can also be expressed as a mixture of Gaussian. However, this does not remove the requirement of evaluating all components thus no computational gains.

An alternative, and more appealing approach, considers adding an auxiliary density $r(\boldsymbol{a}|\boldsymbol{z},\boldsymbol{x})$ for the sake of obtaining a better lower-bound. More precisely, we subtract from the variational lower-bound (B.29) the KL divergence from $q(\boldsymbol{a}|\boldsymbol{z},\boldsymbol{x})$ to $r(\boldsymbol{a}|\boldsymbol{z},\boldsymbol{x})$:

$$\mathcal{L}_{\text{VI}}(q;\boldsymbol{x}) - \text{KL}[q(\boldsymbol{a}|\boldsymbol{z},\boldsymbol{x})||r(\boldsymbol{a}|\boldsymbol{z},\boldsymbol{x})] = \mathbb{E}_q \left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})r(\boldsymbol{a}|\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{x})q(\boldsymbol{a}|\boldsymbol{x})} \right], \tag{B.31}$$

and this auxiliary lower-bound is tight (to the variational lower-bound) iff. $r(\boldsymbol{a}|\boldsymbol{z},\boldsymbol{x}) = q(\boldsymbol{a}|\boldsymbol{z},\boldsymbol{x})$. Therefore we can optimise the auxiliary distribution $r(\boldsymbol{a}|\boldsymbol{z},\boldsymbol{x})$ as well to reduce the bias of the auxiliary bound, which usually leads to improved approximation quality of the q distribution. Furthermore, MC estimation methods is applicable to the auxiliary lower-bound that also returns an unbiased estimator.

The auxiliary lower-bound idea was extensively discussed in Maaløe et al. [2016]; Ranganath et al. [2016b]; Salimans et al. [2015]; Tran et al. [2016], and specifically, Salimans et al. [2015] considered a hierarchy of auxiliary variables, i.e. $\boldsymbol{a} = \{\boldsymbol{a}_1, ..., \boldsymbol{a}_L\}$, and

$$q(\boldsymbol{z},\boldsymbol{a}|\boldsymbol{x}) = q(\boldsymbol{z}|\boldsymbol{a}^L,\boldsymbol{x})q(\boldsymbol{a}^1|\boldsymbol{x}) \prod_{l=1}^{L-1} q(\boldsymbol{a}^{l+1}|\boldsymbol{a}^l,\boldsymbol{x}).$$

The corresponding auxiliary distribution is then defined as

$$r(\boldsymbol{a}|\boldsymbol{z},\boldsymbol{x}) = r(\boldsymbol{a}^L|\boldsymbol{z},\boldsymbol{x}) \prod_{l=1}^{L-1} r(\boldsymbol{a}^l|\boldsymbol{a}^{l+1},\boldsymbol{x}).$$

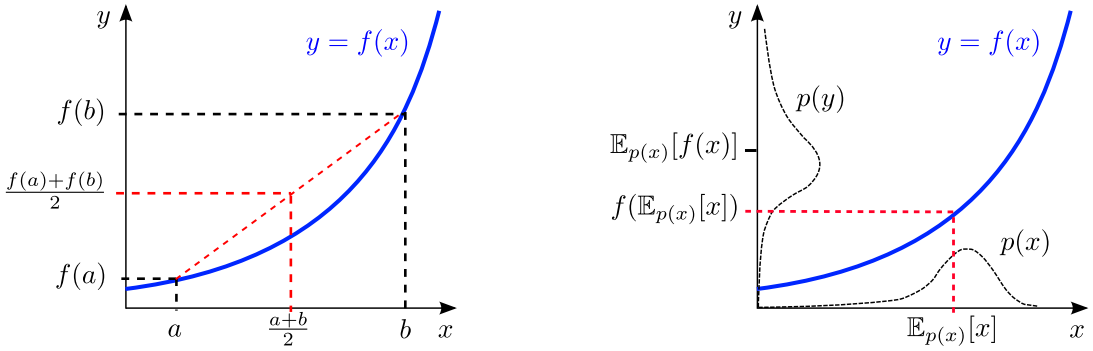
B.5 Inequalities and identities

B.5.1 Jensen's inequality

Proposition B.2. *(Jensen's inequality) If $f(x)$ is a convex function, then for any distribution $p(x)$,*

$$\mathbb{E}_{p(x)}[f(x)] \geq f(\mathbb{E}_{p(x)}[x]).$$

A visual proof is provided in Figure B.1.



(a) for discrete distributions (b) for continuous distributions

Fig. B.1 A visual proof of the Jensen's inequality.

B.5.2 Hölder's inequality

Proposition B.3. *(Hölder's Inequality) Let $(\mathcal{X},\Sigma,\mu)$ be a measure space and let $p,q \in [1,+\infty]$ satisfying $\frac{1}{p}+\frac{1}{q}=1$, then for all measurable functions f, g on $\mathcal{X}$,*

$$\int |f(x)g(x)|d\mu \leq \left(\int |f(x)|^p d\mu\right)^{\frac{1}{p}} \left(\int |g(x)|^q d\mu\right)^{\frac{1}{q}}.$$

Setting $p=q=2$ recovers the Cauchy-Schwarz inequality.

Proof. We assume $\exists x \in \mathcal{X}$ s.t. $f(x) \neq 0$ w.l.o.g., then

$$\begin{aligned}
\int |f(x)g(x)|d\mu &= \left(\int |f(x)|^p d\mu\right) \int [(|g(x)||f(x)|^{1-p})^q]^{\frac{1}{q}} \frac{|f(x)|^p}{\int |f(x)|^p d\mu} d\mu \\
&\leq \left(\int |f(x)|^p d\mu\right) \left[\int (|g(x)||f(x)|^{1-p})^q \frac{|f(x)|^p}{\int |f(x)|^p d\mu} d\mu\right]^{\frac{1}{q}} \\
&= \left(\int |f(x)|^p d\mu\right)^{1-\frac{1}{q}} \left[\int |g(x)|^q d\mu\right]^{\frac{1}{q}} \\
&= \left(\int |f(x)|^p d\mu\right)^{\frac{1}{p}} \left[\int |g(x)|^q d\mu\right]^{\frac{1}{q}}.
\end{aligned}$$

For the second line we used the Jensen's inequality, with the function $h(x) = x^{\frac{1}{q}}$ which is convex when $q >= 1$, and the probability measure $p(x) = \frac{|f(x)|^p}{\int |f(x)|^p d\mu}$. For the third and the last lines we used the assumption that $\frac{1}{p} + \frac{1}{q} = 1$. □

B.5.3 Stein's identity

We provide a proof of Stein's identity in 1D case. The general case can be proved accordingly.

Proposition B.4. *(Stein's identity, one dimensional case) Let $p(x)$ be a differentiable probability density function of $x \in \mathbb{R}$. Let $h(x)$ be a function such that $\lim_{x\to\pm\infty} p(x)h(x) = 0$. Then*

$$\mathbb{E}_p[h(x)\nabla_x \log p(x) + \nabla_x h(x)] = 0.$$

Proof. Using integration by parts, we have

$$\begin{aligned}
\mathbb{E}_p[h(x)\nabla_x \log p(x) + \nabla_x h(x)] &= \int p(x)h(x)\nabla_x \log p(x)dx + \int p(x)\nabla_x h(x)dx \\
&= \int h(x)\nabla_x p(x)dx + \int p(x)\nabla_x h(x)dx \\
&= \int \nabla_x [h(x)p(x)]dx \\
&= h(x)p(x)|_{-\infty}^{+\infty} = 0. \qquad \text{// boundary condition}
\end{aligned}$$

□

Stein [1972] proposed the following lemma for Gaussian distribution.

Lemma B.1. *([Stein, 1972]) Let $p(x) = \mathcal{N}(x; \mu, \sigma^2)$, and the function $h(x)$ satisfies $|\mathbb{E}_p[g(x)(x-\mu)]| < +\infty$ and $|\mathbb{E}_p[\nabla_x g(x)]| < +\infty$. Then*

$$\mathbb{E}_p[g(x)(x-\mu)] = \sigma^2 \mathbb{E}_p[\nabla_x g(x)].$$

Later, Chen [1975] extended the characterisation to Poisson distribution.

Lemma B.2. *([Chen, 1975]) A random variable x taking values in $\mathbb{N}$ is Poisson distributed with rate λ, if and only if $P(x)$ satisfies*

$$\mathbb{E}_P[\lambda h(x+1) - xh(x)] = 0$$

for all bounded functions $h : \mathbb{N} \to \mathbb{R}$.

Proof. Assume x is Poisson distributed with rate λ, then

$$\begin{aligned}\mathbb{E}_P[\lambda h(x+1) - xh(x)] &= \sum_{n=0}^{+\infty} \lambda h(n+1) \exp[-\lambda] \frac{\lambda^n}{n!} - \sum_{n=0}^{+\infty} nh(n) \exp[-\lambda] \frac{\lambda^n}{n!} \\ &= \sum_{n=1}^{+\infty} h(n) \exp[-\lambda] \frac{\lambda^n}{(n-1)!} - \sum_{n=1}^{+\infty} h(n) \exp[-\lambda] \frac{\lambda^n}{(n-1)!} = 0.\end{aligned}$$

Conversely if the identity holds for every function $h(x)$, then setting $h(x) = \delta_{x-k}$ for some k, by assumption we have

$$\mathbb{E}_P[\lambda h(x+1) - xh(x)] = \lambda P(x = k-1) - kP(k) = 0,$$

which implies

$$\frac{P(x=k)}{P(x=k-1)} = \frac{\lambda}{k} = \frac{\text{Poisson}_\lambda(x=k)}{\text{Poisson}_\lambda(x=k-1)}.$$

Since it holds for all $k \in \mathbb{N}^+$, we conclude that x is Poisson distributed. □