Department of Computer Science
Series of Publications A
Report A-2021-1

From Approximations to Decisions

Joseph Sakaya

Doctoral dissertation, to be presented for public examination with the permission of the Faculty of Science of the University of Helsinki, in Hall A129, Chemicum, Kumpula on the 17th of February, 2021 at 4 p.m.

University of Helsinki
Finland

Supervisor
Arto Klami, University of Helsinki, Finland

Pre-examiners
Markus Heinonen, Aalto University, Finland
Matti Vihola, University of Jyväskylä, Finland

Opponent
José Miguel Hernández-Lobato, University of Cambridge, United Kingdom

Custos
Petri Myllymäki, University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Pietari Kalmin katu 5)
FI-00014 University of Helsinki
Finland

Email address: info@cs.helsinki.fi
URL: http://cs.helsinki.fi/
Telephone: +358 2941 911

ISSN 1238-8645
ISBN 978-951-51-6999-0 (paperback)
ISBN 978-951-51-7000-2 (PDF)
Helsinki 2021
Unigrafia

From Approximations to Decisions

Joseph Sakaya

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
joseph.sakaya@cs.helsinki.fi

PhD Thesis, Series of Publications A, Report A-2021-1
Helsinki, January 2021, 115 + 54 pages
ISSN 1238-8645
ISBN 978-951-51-6999-0 (paperback)
ISBN 978-951-51-7000-2 (PDF)

Abstract

Bayesian models capture the intrinsic variability of a data-generating process as a posterior distribution over the parameters of the model for the process. Decisions that are optimal for a user-defined loss are obtained by minimizing expectation of the loss over the posterior. Because posterior inference is often intractable, approximations of the posterior are obtained either via sampling with Monte Carlo Markov chain methods or through variational methods which minimize a discrepancy measure between an approximation and the true posterior. Probabilistic programming offers practitioners tools that combine easy model specification with automatic approximate inference techniques. However, these techniques do not yet accommodate posterior calibrations that yield decisions that are optimal for the expected posterior loss.

This thesis develops efficient and flexible variational approximations as well as density function transformations for flexible modeling of skewed data for use in probabilistic programs. It also proposes extensions to the Bayesian decision framework and a suite of automatic loss-sensitive inference techniques for decision-making under posterior approximations. Briefly, we make four concrete contributions: First, we exploit importance sampling to approximate the objective gradient and show how to speed up convergence in stochastic gradient and stochastic average gradient descent for variational inference. Next, we propose a new way to model skewed data in probabilistic programs by prescribing an improved version of the Lambert W distribution amenable to gradient-based inference. Lastly, we propose

two new techniques to better integrate decision-making into probabilistic programs – a gradient-based optimization routine for the loss-calibrated variational objective, specifically for the challenging case of continuous losses, and an amalgamation of learning theory and Bayesian decision theory that utilizes a separate decision-making module to map the posterior to decisions minimizing the empirical risk.

Computing Reviews (2012) Categories and Subject Descriptors:

Mathematics of computing → Probability and statistics → Probabilistic reasoning algorithms → Variational methods
Computing methodologies → Machine learning → Machine learning approaches
Computing methodologies → Modeling and simulation → Model development and analysis → Uncertainty quantification

General Terms:
Approximate inference, Bayesian inference, Bayesian decision theory, flexible approximations, probabilistic programming, optimization, gradient descent

Additional Key Words and Phrases:
Importance sampling, Lambert function, loss calibration, risk minimization, Monte Carlo methods, reparameterization gradients

Acknowledgements

The words of Longfellow "Art is long, time is fleeting", ring true as a journey begun seven years ago draws to an end – and I have much to be grateful for. I am hugely indebted to Professor Arto Klami for taking a chance on me first as a research assistant and again as a doctoral student. As a mentor and supervisor, he has been amply generous with his time, patience, and feedback. I want to especially acknowledge the stress-free (almost to a fault) and pleasant working environment he has provided. This thesis would not have been possible under any other circumstance.

I thank my pre-examiners, Professor Matti Vihola and Dr. Markus Heinonen, for painstakingly reviewing the thesis and providing feedback. I thank Professor José Miguel Hernández-Lobato for consenting to act as my opponent. I am grateful to Pirjo Moen for guiding me through the graduation process.

I acknowledge and appreciate the financial support offered me primarily by the Doctoral Programme in Computer Science (DoCS) in addition to the Academy of Finland and Scalable Probabilistic Analytics project funded by Business Finland.

I thank Krista Longi and Aditya Jitta for all the long conversations we have had about life, work and other things. I owe more than I let on to Tomasz Kúsmierczyk for being such an amazing postdoc and coauthor and for reviewing parts of the thesis – you are probably the biggest factor in its completion – and for that I am thankful. I thank my coauthor Jarkko Lagus for also providing the occasionally comical takes on research and work. Maksym Gabielkov, Sara Ramazanian, Marcelo Hartmann, Joonas Miettinen and Chen He: you all have been such great people to know and be around.

To my friend Siddharth Sainath, thank you for putting me up for a whole year and looking out for me. To Han Xiao, who has walked the same path that I have, thank you for reminding me to count my blessings. To Grace and Paul Choo for first receiving me in Finland and helping me get started with life here, I am deeply grateful.

I thank my parents Lucia Puthota and Sakaya Milton for letting me tread off the beaten path. I am truly fortunate to have parents like you. I am indebted to my in-laws Rajini Jesuraj and Jesuraj Vedamuthu for trusting me with their daughter when life's prospects still loom uncertain. Annie and Glen Jasper, I take comfort in knowing that you are always looking out for me.

To my wife Jenny and my daughter Maven, I give thanks for the big and little joys in life – you both are all the world to me.

Chicago, January 2021
Joseph Sakaya

We cannot kindle when we will
The fire which in the heart resides;
The spirit bloweth and is still,
In mystery our soul abides.
But tasks in hours of insight will'd
Can be through hours of gloom fulfill'd.

With aching hands and bleeding feet
We dig and heap, lay stone on stone;
We bear the burden and the heat
Of the long day, and wish 'twere done.
Not till the hours of light return,
All we have built do we discern.

Matthew Arnold, *Morality*

Contents

Original Publications

This thesis is based on the following publications. They are reprinted at the end of the thesis.

I. Joseph Sakaya and Arto Klami. Importance Sampled Stochastic Optimization for Variational Inference. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017*, 2017.

II. Arto Klami, Jarkko Lagus, and Joseph Sakaya. Lambert Matrix Factorization. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Proceedings, Part II*, volume 11052 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 2019.

III. Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. Variational Bayesian Decision-making for Continuous Utilities. In *Advances in Neural Information Processing Systems 32*, pages 6395–6405. Curran Associates, Inc., 2019.

IV. Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. Correcting Predictions for Approximate Bayesian inference. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, volume 34, pages 4511–4518. AAAI Press, 2020.

Author contribution

All four papers were written collaboratively with my coauthors. Paper II will also appear in the PhD thesis of Jarkko Lagus. None of the other articles have been included in any other theses.

Paper I. The theory for the importance-sampling estimator was developed jointly, as was the planning and analysis of the experi-

ments. I implemented the algorithms and carried out all experiments.

Paper II. The modified Lambert distribution was developed together by all three authors. All authors contributed to the planning and analysis of experiments. I carried out two experiments, namely, the scalable implementation of Lambert Matrix Factorization for word representations, as well as the illustrative toy experiment.

Paper III. The general theory for automatic loss calibration was developed jointly by all three authors. The experiments were planned and analyzed jointly. Tomasz Kuśmierczyk conducted the final experiments. I implemented and carried out the experiment on amortized optimization (described in Section 4.1.6 of this introduction) included in early drafts of this paper, that was eventually left out of the final version.

Paper IV. The overall theory was developed together by all authors, but I did not contribute to the details of the implicit priors. All experiments were planned and analyzed jointly. I carried out the experiment on model faithfulness, while Tomasz Kuśmierczyk carried out the rest.

Chapter 1

Abstractions for Modeling Uncertainty

Machine learning embodies a broad spectrum of algorithms and models that serve to analyze data, detect patterns, predict future outcomes and make decisions under uncertainty. There are two broad schools of thought as to what constitutes learning. Probabilistic machine learning (Ghahramani, 2015) views learning as the act of inferring plausible models for a data-generating process. Since the data is consistent with a number of models, the uncertainty over models, encoded as distributions over its parameters, is propagated as uncertainty over predictions and decisions. Deep learning (LeCun et al., 2015), in contrast, learns representations of the data with multiple levels of abstraction via optimization of computer models consisting of multiple processing layers. The primary characteristic that sets apart the Bayesian approach from the representation learning approach is the use of marginalization to account for model uncertainty (Wilson, 2020). Whereas marginalization over the model parameters yields predictions that represent an average over all plausible models for the data, representation learning often relies on a single hypothesis, typically *maximum a posteriori* (MAP) estimates of the model parameters, for prediction tasks. Previous works (MacKay, 1992; Neal, 1996) that merge these two schools of thought exist. More recent literature such as Osawa et al. (2019) offer practical Bayesian approaches to deep learning that prevent model overfitting and produce well-calibrated predictive distributions with marginalization.

This thesis takes a special interest in the use of Bayesian models to make decisions under uncertainty. The uncertainty we deal with is of two kinds. Our ignorance about which of the many plausible models generated the observable data, also known as epistemic uncertainty (Kendall and Gal, 2017), is expressed as posterior distributions over the parameters of the

model. Such uncertainty can often be explained away as the model sees more data. The inherent noisiness in the data-generating process, known as aleatoric uncertainty (Kendall and Gal, 2017), can also be modeled as probability distributions that describe the noise with variance parameters. Domain knowledge about the data is incorporated as Bayesian priors over the model parameters. These parameters are connected to the data through the likelihood function which is an interpretable approximation of the true data-generating process. Together, the likelihood and prior specify our knowledge of the Bayesian model before observing any data. The Bayesian posterior obtained by conditioning the parameters on the observed data, constitutes the act of learning or inference. It tends to favor simple and constrained solutions, even for models that are over-specified and complex, performing automatic model selection in conformity with Occam's razor. Models can also be used to make decisions under posterior uncertainty. Bayes optimal decisions for a user-defined loss over the model predictions can be obtained by minimizing the expectation of the loss over the posterior, also known as the expected posterior loss or Bayes risk (Berger, 1985).

Progress in machine learning, as in any scientific undertaking, depends on the generalizability of the learning algorithms and the ease of model specification. Machine learning platforms that abstract away learning and model-building have deeply transformed our capacity to model and process data. This is evidenced by the explosion of interest (He, 2019) in deep learning and its applications with the advent of high-level systems like Tensorflow (Abadi et al., 2016) and abstractions built atop them, such as Keras (Chollet et al., 2015). Rapid prototyping and model iteration in deep learning has been made possible by use of gradient-based optimization methods that utilize gradients determined by automatic differentiation. Automatic differentiation tools (Baydin et al., 2015) evaluate derivatives of numeric functions expressed as programs, thereby automating the tedious and manual process of deriving gradients of the objective function for optimization. Standard engineering practices such as flow diagrams, code reuse and composition of complex systems as modular layers have become the norm in deep learning. Although the learning paradigm utilized in deep learning is primarily based on optimization of an objective function to yield MAP estimates of the parameters, the tools developed for such automated gradient-based optimization can be extended with some effort for use in Bayesian inference.

Despite the several advantages going hand in hand with modeling uncertainty, Bayesian inference is hard. This is because posterior inference is intractable for all but the simplest models and requires that practitioners

use inference techniques to approximate the posterior on account of computational constraints. The high cost of expertise and skill involved in developing such inference algorithms for Bayesian models has so far forestalled its widespread adoption. This inconvenient dependency between inference and modeling has led to a systematic confusion that mixes up the laborious process of deriving model-specific inference procedures with the task of model building itself. This means that considerations and assumptions in modeling are informed (wrongly so) by the ability of the practitioner to run inference on such models. *Probabilistic programming* aims to break this impasse by clearly delineating the task of model specification from inference by providing a language for formal model specification and a model-agnostic inference procedure for posterior inference. Advances in approximate inference techniques have allowed adapting tools used in gradient-based optimization for deep learning, such as automatic differentiation, for use in model-agnostic inference for probabilistic models. Probabilistic programming systems such as `Stan` (Carpenter et al., 2017), `Pyro` (Bingham et al., 2018), `PyMC3` (Salvatier et al., 2016) and `Tensorflow Probability` (Tran et al., 2016) seek to recreate for Bayesian modeling the popularity achieved by deep learning frameworks for neural networks. That is, these systems seek to incorporate sound engineering practices into Bayesian learning for rapid model prototyping, iteration and criticism.

1.1 Probabilistic Programming

Probabilistic programming is commonly understood as leveraging programming paradigms in computer science to solve statistical inference problems (van de Meent et al., 2018). Figure 1.1 shows how probabilistic programming lies at the intersection of programming languages and statistics drawing from such ideas as semantics and compilers in computer science and inference algorithms in statistics and machine learning. The classical view of probabilistic programming defines it as the development of "syntax and semantics for languages that denote conditional inference problems and evaluators or solvers that computationally characterize the denoted conditional distribution" (van de Meent et al., 2018). `Anglican` (Le et al., 2017) and `Church` (Goodman et al., 2012) are prime examples of this. A more utilitarian characterization of probabilistic programming, as understood in this thesis, is that of an abstraction framework that builds on elements like automatic differentiation tools and Monte Carlo methods to provide model-agnostic inference for Bayesian models. `Stan` (Carpenter et al., 2017) is an archetypal example of this view.

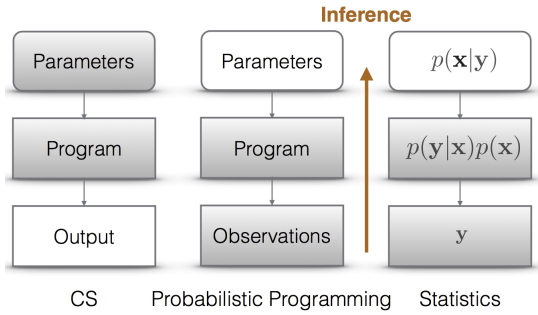

Figure 1.1: Probabilistic programming refers to use of tools in computer science to do statistics. Computer programs are written to accept variables and arguments as parameters to generate useful outputs. Statistical inference deals with inference of model parameters $\mathbf{x}$, under the observations $\mathbf{y}$ for the generative model $p(\mathbf{y}, \mathbf{x})$. Probabilistic programming performs statistical inference with the help of tools from computer science. It provides a language for model specification and an inference engine that infers the conditional distribution of the parameters under the given observations. Figure from van de Meent et al. (2018).

Probabilistic programming delivers the compositionality probabilistic graphical models promised and provides tools to speed up model iteration and support development of a wide variety of models. A probabilistic programming system should address at least four aspects of Bayesian modeling. First, it should provide a language for easy and precise specification of Bayesian models. Second, it should contain an automated inference engine for model fitting and prediction that chooses the appropriate inference tools based on the model specification and the degree of approximation required. Third, it should offer diagnostic tools for convergence and warn the user of convergence failures, if any. And fourth, it should contain tools that allow posterior analysis and model criticism with posterior predictive checks. A final aspect of Bayesian modeling that is often not addressed is the strong relationship between inference and decision-making. This follows from a flavor of Bayesianism that is partial to a clean delineation between inference and decisions (MacKay, 1991). But as we shall see in Chapter 4, we can improve our inference tools by incorporating decision-making and targeting regions of the posterior that have substantial impact on the decisions.

Probabilistic programs simplify the task of probabilistic modeling by eliminating the biggest obstacle to efficient Bayesian modeling – model-dependent inference procedures. While the principle behind the inference

procedure is straightforward, typical model-independent procedures such as Metropolis-Hastings (Hastings, 1970) and rejection sampling (Bishop, 2006) require fine-tuning of parameters for efficient exploration of the parameter space. Alternatively, model-dependent methods such as Gibbs sampling (Bishop, 2006), mean-field variational inference (Jordan et al., 1999) and expectation propagation (Minka, 2001) that exploit conjugate distributions within the exponential family are fairly laborious to implement, even if there are fewer parameters to tune. With probabilistic programming, the practitioners can turn their attention away from the question of how to perform inference to the modeling task itself and concern themselves with the question of what to model. Combining model-agnostic inference algorithms with a descriptive modeling language has brought a paradigm shift in the Bayesian modeling community, by making Bayesian modeling accessible to a broad community of lay users and domain experts. Practitioners are not constrained in their choice of models by the limitations of inference techniques, but are free to experiment with complex, non-conjugate models that describe the data-generating process accurately. There is active research aimed at improving and scaling inference techniques and diagnostic tools for complex, non-differentiable models (Yao et al., 2018; Rezende and Mohamed, 2015; Figurnov et al., 2018; Lee et al., 2018).

1.2 Approximate Inference

For many models, the computation of the (Bayesian) posterior distribution is analytically intractable. We therefore adopt approximate inference techniques that rely on computationally tractable approximations to serve as proxies to the true posterior. There are a diverse class of approximation techniques with varying trade-offs between computation time and accuracy of approximation. On one hand, classic Monte Carlo Marko chain (MCMC) methods rely on samples from the posterior for approximating it. Although they asymptomatically converge to the true posterior, practical time budgets lead to their categorization as approximate techniques. However, recent work by Seita et al. (2017) have provided rigorous theoretical bounds for scaling up MCMC techniques. More efficient MCMC methods like the Hamiltonian Monte Carlo (HMC) (Neal, 2011) and No-U-Turn Sampler (NUTS) (Hoffman and Gelman, 2014) take advantage of gradient information from the joint likelihood of the model to speed up convergence and decrease correlation between samples.

Distributional approximations, on the other hand, posit a parametric family of distributions $q_\lambda(\theta)$, known as the variational approximation and

optimize the variational parameters λ to minimize the variational objective equivalent to a discrepancy measure between the variational approximation and the true posterior. This view of inference as optimization also allows using automatic gradient descent methods from the deep learning toolchain for performing approximate inference. The discrepancy measure is often a subset of f-divergence (Csiszar, 1967) such as the Kullback-Leibler divergence. The two most popular strategies for achieving this are variational inference (Blei et al., 2017) and expectation propagation (Minka, 2001). Techniques such as the stochastic Langevin dynamics (Welling and Teh, 2011) and other works by Ruiz and Titsias (2019) and Salimans et al. (2015) seek to bridge the gap between distributional approximations and sampling methods. Recent work has greatly increased the flexibility and expressive power of approximate inference tools: Normalizing flows (Rezende and Mohamed, 2015) use composable layers of transformations to approximate complex, multi-modal posteriors; generalized Bayesian inference (Knoblauch et al., 2019) extend variational inference for posteriors over arbitrary loss functions; optimizers for Bayesian deep learning (Khan et al., 2018) show how gradient-descent strategies can be modified to provide uncertainty estimates over model parameters.

1.3 COVID-19: A Motivating Example

To motivate the importance of rapid model prototyping and iteration with probabilistic programming we present COVID-19 as an illustration where rapid prototyping of epidemiological models have proven indispensable. The COVID-19 pandemic has officially neared two million confirmed cases[1] as of April 2020 and has increasingly stretched thin the world's medical and healthcare resources (World Health Organization, 2020). Countries have come up with a variety of measures to contain the spread of the disease: Sweden has imposed limited quarantine of the vulnerable and elderly; the United States has initiated strict social distancing measures and shutdown of non-essential services (Centers for Disease Control and Prevention, 2020); India's unplanned and disastrous lockdown has inflicted misery on huge masses of its population. The prevalence of such drastic, blanket measures with severe impairments to the economy is partly due to lack of testing equipment to quickly test and effectively contain the infected population.

[1] As I finalize the thesis in January 2021, there are 96.2 million confirmed cases globally with 2.06 million deaths. (Johns Hopkins University, 2020). Even as Pfizer-BioNTech and Moderna have developed vaccines that have been authorized (Centers for Disease Control and Prevention, 2021) to combat this disease, variants of the SARS-CoV-2 virus (World Health Organization, 2021) have caused further public health concern.

Modeling of infectious diseases is useful for predicting outcomes of pandemics and helps define public health response. The response should be informed by models that are up-to-date and account for a host of secondary factors such as underreporting, confounding variables and preventive measures. Such modeling needs to be done rapidly and needs to evolve to adapt to new data and information sources as well as the progress of the epidemic. Exploration of different models that accommodate different sets of assumptions (for example, a fixed population size, density or age and gender-specific mortality ratio) and alternative models should include preventative measures that curb the pandemic. This is essential for simulating the spread of the epidemic under different conditions and assumptions. The models should also be reliable – assumptions should be made explicit and inference should have convergence diagnostics that inform us of the trustworthiness of the model predictions. The data received is typically noisy and there may be a dearth of data for comorbidities or due to low testing capacity – a model that captures uncertainty in its predictions and latent variable estimates allows us to base the strength of our predictions on the uncertainty of our estimates. Lastly, the models should provide a basis for making decisions that define public health response to the crises. Probabilistic programming addresses these concerns directly by defining a language for clearly specifying models and model assumptions and by providing inference tools for rapid model refinement. We now elaborate on three settings where Bayesian modeling uses probabilistic programs to estimate outcomes and measure covariate association for COVID-19.

As of April 2020, the age-specific mortality ratio remains unknown partly due to underreporting of confirmed cases and partly due to delay between onset of disease and death. Estimates of mortality ratios are essential for assisting in mitigation efforts and planning for healthcare logistics. Hauser et al. (2020) present a simplified epidemiological model that simulates the transmission dynamics of COVID-19 using data from Hubei, China and Northern Italy and provides estimates of age-specific mortality ratios while also adjusting for bias introduced by underreporting and delay between disease onset and death. The use of `Stan` allowed for rapid model iteration and fitting[2] to quickly churn out mortality estimates that are critical for emergency healthcare planning.

Kubinec (2020) presents a `Stan` model to measure association between covariates such as geopolitical and health factors and the test and case

[2]One of the authors of Hauser et al. (2020) state in Andrew Gelman (2020) that an optimized implementation of the model reduced the computation time from 3 days to 2 hours.

counts of the disease. For instance, as of April 2020, increasing infection rates in the United States are concentrated in areas with larger youth populations, smaller economies, less public health spending and fewer Trump voters. Air quality, number of immigrants, smoking and cardiovascular deaths have no significant effect on the infection rate. The unobserved infection rate, which is established as a confounding factor is accounted for by introducing an informative prior estimated from epidemiological models. Such rapid modeling tasks would prove impracticable without probabilistic programming.

A final illustration of the effectiveness of Bayesian modeling is the unconventional use of pooled design to estimate the spatial map of the prevalence of COVID-19 (Lakeland, 2020). For example, k people can be swabbed and randomly split into l pools where $l \ll k$. Let us assume a testing mechanism that tests accurately for the disease from the combined material of k/l swabs. The probability of a random pool testing positive with at least one positive case is $\hat{\theta} = 1 - \text{Bin}(0|l, \theta)$, where θ is the frequency of the disease in the population and Bin is the binomial distribution. The likelihood that N pools test positive is therefore $\text{Bin}(N|k/l, \hat{\theta})$. Given N, we can estimate the prevalence θ of the disease among the population k with just k/l tests. Replicating this over in different geographical locations gives a posterior estimate of θ with tight error bars. This allows for more targeted, district-wise lockdown measures allowing for restart of the economy especially in countries with low testing capacity.

The Bayesian models described here are successful in modeling critical facets of the pandemic. However, they do not lay out clear decision principles that help make public health choices or provide any means of comparing the implications of each of these choices. Decisions should be based on a well-considered trade-off between the confidence of the model predictions and the cost associated with each of the decisions. An erroneous decision incurs a cost that depends on the kind of error made. For example, an underestimate of the number of ventilators required during the peak of the epidemic curve is more disastrous from the public health viewpoint than an overestimate. The models discussed so far have failed to provide an explicit decision-theoretic loss function that captures the costs associated with different kinds of errors. The conceit of this thesis is that such loss functions are necessary for calibrating posterior inference to yield loss-aware decisions that incur a lower expected loss.

1.4 Overview

Papers I and II constitute the first part of the thesis which focuses on distributional approximations, in particular variational inference, as the tool of choice for fast and flexible modeling. In Paper I (Sakaya and Klami, 2016, 2017), we present techniques for improving the convergence rate of automatic variational inference with simple modifications to gradient evaluation of the variational objective. Paper II (Sakaya and Klami, 2017) prescribes a differentiable transformation of density functions to model skewed versions of standard distributions for use in gradient-based variational inference.

Papers III and IV, covered in the second part of the thesis, consider decision-making in the context of approximate inference in probabilistic programming. Paper III (Kuśmierczyk et al., 2019b) demonstrates that incorporating the user-defined loss as part of the inference process yields decisions that lowers the Bayes risk when compared to generic inference tools agnostic to the loss. Paper IV (Kuśmierczyk et al., 2020) introduces a novel framework that combines empirical risk minimization with Bayesian inference, and shows how post-inference objects from arbitrary probabilistic programs can be mapped into decisions that are optimal with respect to the empirical risk.

We now briefly present the new elements of the thesis below:

1. Automatic variational inference for probabilistic programs utilizes gradient-based approaches to optimize the variational objective to obtain approximations to the posterior (Kucukelbir et al., 2017; Salvatier et al., 2016). The Monte Carlo gradient of the objective consists of two parts – a model-dependent component that is expensive to compute and an approximation-dependent component that is significantly cheaper to evaluate. We exploit importance sampling to approximate the objective gradient by reusing the expensive model-specific gradients from past iterations and removing the bias so introduced with importance weights calculated from the new and old approximations. We show how to speed up convergence in stochastic gradient descent and cheaply stabilize older gradients in stochastic average gradient descent for linear convergence. This method is broadly applicable to any Monte Carlo expectation over a differentiable function and therefore finds application in many practical use cases.

2. Distributions modeling skewed data are often expensive to compute and not widely supported in probabilistic programming libraries. The flexible Lambert $W \times F$ family of skewed distributions is obtained

by skewing standard base distributions $F_Z(z)$ in the location-scale family with a differentiable, non-linear forward transformation (Goerg, 2011). The backward Lambert W transformation is problematic since it is not bijective and lacks a closed-form solution. We prescribe modifications with a memoized, piece-wise extension of the Lambert W function for gradient-based inference, enabling straightforward extension of existing tools to model skewed data and sample from skewed versions of standard distributions by simply defining the forward skewing function and modified backward Lambert W transform.

3. Since approximations are designed to capture only general properties of the Bayesian posterior without accounting for the decision task, decisions made by optimizing the *expected posterior risk or utility* over the approximate posteriors are not always Bayes optimal. Loss-calibrating inference compounds the variational objective with the expected log utility to produce approximations that yield utility-aware decisions (Lacoste-Julien et al., 2011). We delineate a gradient-based optimization routine for the loss-calibrated objective, specifically for the challenging case of continuous losses, and suggest practical transformations to convert from losses to utilities and vice versa as well as techniques to maximize the calibration effect. We recognize and minimize the risk of violating Bayesian decision principles arising from the tight coupling of inference and decision-making.

4. Lastly, we propose a novel amalgamation of learning theory and Bayesian decision theory that furnishes an alternative approach to optimal decision-making by minimizing *empirical risk*. Rather than calibrating the posterior approximation, we characterize the posterior with summary statistics or quantiles fed to a separate decision-making module, typically a neural network, that minimizes the empirical risk even as we retain the interpretability of Bayesian modeling. The generalized Bayesian framework stipulates valid posterior updates for decisions connected to observations via arbitrary loss functions allowing us to operate in terms of belief distributions over decisions. In conjunction with bootstrapping, the framework facilitates setting empirical priors over the decisions and provides a coherent means of specifying the trade-off between risk minimization and model faithfulness

The layout of the chapters are as follows: The background in Chapter 2 discusses crucial aspects of Bayesian inference, decision theory, automatic

inference techniques and importance sampling. Chapter 3 develops the importance-sampled gradient-based procedures for variational inference and the Lambert W transformation for modeling skewed data efficiently. The two loss calibration techniques designed for easy integration into probabilistic programs are discussed in Chapter 4. We conclude in Chapter 5 by discussing the limitations of our contributions, as well as some recent developments that expand on the ideas we presented.

Chapter 2

On Approximations and Decisions

Practical applications of machine learning often involve making decisions with the goal of optimizing a user-defined evaluation criterion. Decisions appear in a variety of settings: in buying or selling stocks and bonds in finance; in the diagnosis and treatment of diseases in medicine; and in implementation of welfare schemes in public policy. The consequences and penalties incurred when making decisions are captured by the evaluation criteria specified as a confusion matrix (for discrete decisions) or as a utility or loss function. It is a fairly common error for decisions to be reported without expressly stating the evaluation criteria they are optimizing. For example, the decision of a scientific article to report the sample mean $\bar{\theta}$ of a parameter θ as the representative statistic betrays an implicit assumption of the mean squared error as the criterion to optimize. However, as shown in Figure 2.1, when the distribution of θ is symmetric, the mean and median coincide and it becomes impossible to precisely identify the evaluation criteria as either squared or absolute error. Therefore, it becomes difficult to evaluate and compare different decisions without the specification of an evaluation criterion.

As an illustration of binary decision-making, we describe a fairly commonplace medical decision task that involves diagnosing lupus – as everyone who has watched *House, M.D.* is aware, lupus is a notoriously hard disease to identify. A loss function $\ell(\theta, h)$ such as in Table 2.1 captures the decision-making principles that guide the diagnosis. When the decisions h are the same as the true but unknown observations θ, the diagnosis is correct and there is no penalty attached. A decision $h = \texttt{Lupus}$ is *false positive* when the unknown true state $\theta = \neg\texttt{Lupus}$. It has a cost associated with it since treating a perfectly healthy subject with immunosuppressants is deleterious. A diametric *false negative* diagnosis of $h = \neg\texttt{Lupus}$ when the patient truly has lupus ($\theta = \texttt{Lupus}$) may prove fatal and hence is penal-

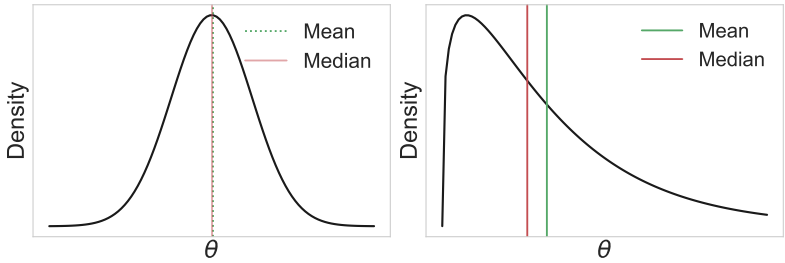

Figure 2.1: Decision-making with symmetric and asymmetric distributions: The choice of the optimal decision is a function of the user-defined loss. When choosing a squared loss, $\ell(\theta, h) = (\theta - h)^2$, the decision h that minimizes the loss is the mean. For the absolute loss, $\ell(\theta, h) = |\theta - h|$, the median is the optimal decision. In the asymmetric distribution, the decision reported varies depending on the choice of loss function. When the distribution is symmetric, however, the choice of loss function is not readily apparent from the decision.

ized heavily. In practice though, the evaluation criteria as defined through the loss matrix is hard to determine and achieve consensus on since the decisions have indeterminable real-world consequences. We remark that we have not yet incorporated the uncertainty surrounding the decision into the decision-making process: any decision should be a carefully considered trade-off between the uncertainty surrounding the decision and the cost associated with being wrong.

Decision-making relies on explicit identification of two factors: the evaluation criteria to optimize, defined either in terms of loss or utility, and a statistical model for the underlying process based on which the decisions are made. The formal framework that axiomizes these assumptions is known as *decision theory*. *Bayesian decision theory* extends this framework for rational decision-making under model uncertainty with well-defined properties. Both classical and Bayesian decision theory are similar insofar as they operate by minimizing an expected loss $\mathbb{E}[\ell(\theta, h)]$. They, however, differ on the quantity the expectation is over: the Bayesian approach averages over the model parameters conditioned on the observed data, whereas classical decision theory defines the expected loss as an average over data sampled i.i.d. from the statistical model (Murphy, 2012).

In this chapter, we outline the principles upon which Bayesian inference and Bayesian decision theory build. We begin with an overview of Bayes' theorem and its components to provide the basis of uncertainty quantification and model selection. We follow up with a discussion of classical

Table 2.1: An example of asymmetric loss functions that are widespread in many decision-making applications. The columns correspond to the decisions, h, made by the model while the rows are the classes of true, but unknown, observations θ. The correct diagnosis along the diagonal carry no penalty. Since the cost associated with the decision is a function of the nature of the error made, false positives and false negatives are associated with varying penalties.

	h	
θ	`Lupus`	$\neg$ `Lupus`
`Lupus`	0	-100
$\neg$ `Lupus`	-1	0

inference techniques that approximate the true Bayesian posterior in Section 2.2. In particular, we expand on automatic variational inference techniques (Carpenter et al., 2017; Titsias and Lázaro-Gredilla, 2014) which use stochastic gradient descent to optimize the variational objective. This discussion helps build up the contributions we make in Chapters 3 and 4. We next lay down the principles of Bayesian decision theory and concretize notions of *Bayes optimal* decisions that optimize the posterior *risk* or *gain*. We show that substituting the true posteriors with their approximations yields decisions that are sub-optimal with respect to the true posteriors and argue that loss-agnostic approximate inference techniques should be abandoned in favor of inference methods that are aware of the loss or utility function. Lastly, on a different note, we briefly cover the importance sampling necessary for developing theory in Chapter 3.

2.1 Bayes' theorem

Bayes' theorem forms the cornerstone of Bayesian inference and provides a basis for uncertainty quantification and a means to update our beliefs about model parameters. Statistical models are useful simplifications of data-generating mechanisms that allow us to capture the data using probability distributions. In the Bayesian context, assumptions about the data-generating process are encoded via the joint density composed of the likelihood function and the prior. By conditioning the joint density on the observed data, we get posteriors over our model parameters that reflect our model uncertainty after having seen the data. Such posteriors can be used to simulate data points to compare with the observed data to test

our model assumptions or to make decisions that take account of model uncertainty. The Bayesian view of learning as posterior inference makes it very versatile and it is hard to overstate its importance.

We now formalize the notation used throughout this thesis. The data or observations that are produced by the data-generating mechanism are denoted as $\mathcal{D}$. In supervised setups such as classification and regression, the data, $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$, consists of N input-output pairs where $x \in \mathcal{X}$ refers to the covariates or features from the feature space $\mathcal{X}$ and $y \in \mathcal{Y}$ are the observations drawn from the observation space $\mathcal{Y}$, which is discrete for classification tasks and real-valued for regression tasks. In unsupervised cases, such as clustering or k-nearest neighbors, the data $\mathcal{D} = \{x_n\}_{n=1}^N$ includes only the covariates $x \in \mathcal{X}$. A statistical model is represented as a joint density $p(\mathcal{D}, \theta)$ where $\theta \in \Theta$ refers to the unknown model parameters of interest and Θ is the parameter space. The joint density can be decomposed as $p(\mathcal{D}, \theta) = p(\mathcal{D}|\theta)p(\theta)$ where $p(\mathcal{D}|\theta)$ is the conditional density of the data for a given parameter, also known as the likelihood or the sampling distribution, and $p(\theta)$ represents our prior uncertainty about the model parameters θ. A simple application of the rules of conditional probability, results in the Bayes' theorem which yields the posterior density $p(\theta|\mathcal{D})$ of the model parameters conditioned on the data as

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \tag{2.1}$$

Breaking the equation down, the likelihood $p(\mathcal{D}|\theta)$, which is a function of θ, is typically a probability density from a well-known parametric family of distributions – for instance, the exponential family (Morris, 1982). Bissiri et al. (2016) generalizes the Bayesian update framework to cases where the likelihood can be an arbitrary loss function. We use this generalized Bayesian framework to develop techniques that minimize the empirical risk in Chapter 4. Since the denominator $p(\mathcal{D})$ is constant with respect to θ, Equation 2.1 also yields the unnormalized posterior density

$$p^*(\theta|\mathcal{D}) = p(\mathcal{D}|\theta)p(\theta) \propto p(\theta|\mathcal{D}).$$

The denominator $p(\mathcal{D})$ is the normalization constant that ensures $p(\theta|\mathcal{D})$ is a density that integrates to one. However, $p(\mathcal{D})$ is typically hard to calculate as it requires the integral

$$p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)\ d\theta. \tag{2.2}$$

Bayes' theorem permits us to test our model assumptions by predicting objectively testable and observable quantities, known as predictive inference (Gelman et al., 2013). In the context of supervised learning, the *prior*

predictive distribution is the distribution of an unknown but observable $\mathcal{D}$ marginalized over the prior density

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

This has the same form as Equation 2.2, but is a function of $\mathcal{D}$. The *posterior predictive distribution* provides a distribution of an unknown and unobservable $\widehat{\mathcal{D}} = \{\hat{x}, \hat{y}\}$, after conditioning θ on the observed $\mathcal{D}$, as

$$p(\widehat{\mathcal{D}}|\mathcal{D}) = \int p(\widehat{\mathcal{D}}, \theta|\mathcal{D})\ d\theta = \int p(\widehat{\mathcal{D}}|\theta)p(\theta|\mathcal{D})\ d\theta.$$

We will frequently refer to the posterior predictive distribution while discussing decision-making strategies in this chapter and in Chapter 4. We point the reader to Gelman et al. (2013) for a comprehensive discussion of model evaluation, model comparison and data analysis from the Bayesian perspective.

2.2 Approximate Inference

Bayes' theorem offers a coherent, *theoretical* basis for conditioning the model parameters θ on the observations $\mathcal{D}$, but the model evidence $p(\mathcal{D})$ that is crucial for posterior density evaluation and model averaging often causes practical problems. The computational difficulty of $p(\mathcal{D})$ scales exponentially with the dimensionality of θ – when θ is high-dimensional, as it is in most models of practical interest, the evaluation of $p(\mathcal{D})$ becomes analytically intractable. In such cases, the model evidence can be approximated using numerical methods. However, naive Monte Carlo approximation techniques for estimating $p(\mathcal{D})$ suffer from high variances depriving the estimates of any practical significance. Other more sophisticated sampling techniques such as bridge sampling (Gronau et al., 2017) produce more reliable estimates of the model evidence for a limited set of high-dimensional models. Fourment et al. (2019) lists different ways with varying accuracy of approximating the model evidence including, but not limited to, variational inference and path sampling.

Our interest, however, is in inferring the posterior distribution $p(\theta|\mathcal{D})$ and estimating the model evidence is simply a means to that end. A careful choice of priors and likelihood such that the posterior lies in the same *parametric family* as the prior allows estimating the posterior by avoiding evaluation of $p(\mathcal{D})$ altogether. This property, known as *conjugacy*, allows the normalized posterior density to be estimated analytically in closed-form.

Although conjugate priors enjoy popular use because of their simplifying computation and ease of interpretation, they restrict the choice of models that we can run inference on by limiting our options to a narrow set of compatible likelihoods and priors, forcing us to make modeling assumptions that are sometimes inconsistent with the observed data. This shortcoming opens up opportunities for more generic and powerful approximate inference techniques, that have the potential to run inference on a wider variety of non-conjugate models with assumptions that are not constrained by our ability to run inference. Approximate inference methods, as introduced in Chapter 1, are a broad class of techniques that approximate the posterior either with Monte Carlo approximations or via minimization of a divergence measure between the posterior and its proxy.

2.2.1 Sampling-based methods

Sampling-based methods refer to a family of algorithms that generate samples from the posterior, $\theta_s \sim p(\theta|\mathcal{D})$, to estimate quantities of interest such as the posterior predictive. The expectation of a function f representing our quantity of interest, such as the loss, over the posterior is approximated as

$$\mathbb{E}_{p(\theta|\mathcal{D})}[f] \approx \frac{1}{S}\sum_{s=1}^{S} f(\theta_s).$$

The accuracy of sampling-based methods is a function of the effective sample size obtained after accounting for correlation between samples. Although asymptotically exact in theory, sampling-based methods can be classified as approximate methods because the number of samples is finite and depends on computational and practical constraints. Moreover, there exists no definitive guarantee of the samples accurately representing the posterior distribution. The precise method of generating the posterior samples θ_s varies depending on the properties of the model, giving rise to a diverse class of techniques known as Monte Carlo Markov Chain (MCMC) methods that sample efficiently from high-dimensional spaces.

Gibbs sampling (Geman and Geman, 1984), a standard choice for hierarchical Bayesian models, samples each component θ^i of the parameters θ iteratively, conditional on the rest of the components $\theta^{\setminus i}$. When the models are conditionally conjugate, that is $p(\theta^i|\theta^{\setminus i}, \mathcal{D})$ is conjugate, and the posteriors have weak dependencies, we can sample conditionally in closed-form making Gibbs sampling a very efficient sampling strategy. In point of fact, Gibbs sampling is a special case of the Metropolis-Hastings algorithm (Metropolis et al., 1953), the workhorse of MCMC methods. It is a

Algorithm 1: The Metropolis-Hastings algorithm

Initialize θ_0
for $s = 0, 1, \ldots S$ **do**
  $\theta_{s+1} \sim q(\theta_{s+1}|\theta_s)$
  Compute acceptance probability r from Equation 2.3.
  $u \sim \text{Uniform}(0, 1)$
  $\theta_{s+1} = \begin{cases} \theta_{s+1}, & \text{if } u < r \\ \theta_s, & \text{if } u \geq r \end{cases}$
end

generic method to construct a Markov chain that is ergodic and stationary with respect to the posterior density, and therefore generates samples from the posterior. It uses an asymmetric proposal kernel $q(\theta_{s+1}|\theta_s)$ to propose samples θ_{s+1} with acceptance probability

$$r = \min\left(1, \frac{p^*(\theta_{s+1}|\mathcal{D})/q(\theta_{s+1}|\theta_s)}{p^*(\theta_s|\mathcal{D})/q(\theta_s|\theta_{s+1})}\right) \tag{2.3}$$

where $p^*(\theta|\mathcal{D})$ refers to the unnormalized posterior density. Variations to the Metropolis-Hastings algorithm retain its general algorithmic flavor shown in Algorithm 1, while modifying the proposal θ_{s+1} so as to increase the probability of acceptance. Hamiltonian Monte Carlo techniques such as the NUTS sampler (Hoffman and Gelman, 2014), are improvements of Metropolis-Hastings algorithms which make use of first-order gradient information from the log joint density $\log p(\mathcal{D}, \theta)$ to minimize random walk behavior and correlation between successive samples, paving way for quicker convergence to the posterior density.

MCMC methods historically suffered from a lack of definitive criteria for assessing convergence. Convergence diagnostics such as $\hat{R}$ (Gelman and Rubin, 1992; Vehtari et al., 2019), which measure convergence by checking if chains have mixed well by comparing between- and within-chain estimates, have addressed this drawback. MCMC methods also scale poorly since the computational complexity grows with the size of the data set $|\mathcal{D}|$. Several works seek to remedy this shortcoming, the most prominent of which is Seita et al. (2017) which outlines a theoretically sound variant of acceptance tests for minibatches that account for the variance in minibatch statistics leading to a scalable variant of Metropolis-Hastings. Finally, we note that characterizations of sampling-based methods using summary statistics or quantiles can be used in the context of post-hoc calibration of the posterior density in Chapter 4. Since in-depth discussion of

sampling-based methods is not necessary for developments in this thesis, we point the reader to Brooks et al. (2011) and Robert and Casella (2005) as comprehensive sources of sampling-based methods.

2.2.2 Optimization-based methods

Optimization-based methods operate by casting the problem of posterior inference as one optimization of a divergence measure between a distributional approximation and the posterior. The distributional approximation $q_\lambda(\theta)$ is a parametric family of distributions characterized by the variational parameters λ. These parameters are tuned to find a member of the variational family that minimizes the divergence measure $\mathbb{D}(q\|p)$ and is thereby the closest approximate of the posterior $p(\theta|\mathcal{D})$. Such a divergence measure $\mathbb{D}(q\|p)$ measures the difference between the approximation and the posterior such that $\mathbb{D}(q\|p) \geq 0$ with equality if and only if $p(\theta|\mathcal{D}) = q_\lambda(\theta)$. One of the broadest classes of divergence measures is the f-divergence (Csiszar, 1967)

$$\mathbb{D}_f(q\|p) = \int q_\lambda(\theta) f\left(\frac{p(\theta|\mathcal{D})}{q_\lambda(\theta)}\right) d\theta,$$

where $f(t)$ is concave and $f(1) = 0$. Commonly used divergence measures are obtained as special cases of the f-divergence: the χ^2 divergence is obtained when $f(t) = (t-1)^2$; Rényi divergence (Li and Turner, 2016) when $f(t) = 4(1 - t^{(1+\alpha)/2})/(1 - \alpha^2)$; and forward and reverse KL-divergence when $f(t) = t \log t$ and $f(t) = -\log t$, respectively.

The KL divergence (Kullback and Leibler, 1951) is perhaps the most commonly used divergence measure in machine learning literature (Bishop, 2006). Also known as information gain or mutual entropy, the KL divergence between two distributions is defined as

$$\mathrm{KL}(q\|p) = \int q_\lambda(\theta) \log \frac{q_\lambda(\theta)}{p(\theta|\mathcal{D})} \, d\theta. \tag{2.4}$$

Since $\mathrm{KL}(q\|p) \neq \mathrm{KL}(p\|q)$, the KL divergence is an asymmetric measure yielding two different approximate inference techniques depending on the ordering of $q_\lambda(\theta)$ and $p(\theta|\mathcal{D})$. Expectation propagation (Minka, 2001) optimizes the forward KL divergence $\mathrm{KL}(p\|q)$ to generate global approximations of $p(\theta|\mathcal{D})$ via local moment-matching. Variational inference, in contrast, optimizes the reverse KL divergence $\mathrm{KL}(q\|p)$ to produce more compact, local approximations that tend to underestimate the posterior variance. In this thesis, we develop modifications to variational inference. Since the contributions in this thesis primarily deal with automatic variational approximations (Kucukelbir et al., 2017; Titsias and Lázaro-Gredilla,

2014; Kingma and Welling, 2014; Ranganath et al., 2014), we devote Section 2.3 to expanding on stochastic optimization techniques with different types of Monte Carlo estimators used in automating variational inference.

Throughout this thesis we work with fully-factorized approximations with the mean-field assumption that model parameters θ_i are all independent:

$$q_\lambda(\theta) = \prod_i q_{\lambda_i}(\theta_i).$$

Although the mean-field assumption is computationally efficient it underestimates posterior variance (MacKay, 2003) and does not model posterior dependencies. While the mean-field assumption makes the most independence assumptions, the technique of factorizing approximation can be generalized to partly factorized approximations that make posterior dependency assumptions between a subset of the model parameters as well as full-rank approximations that model dependencies between every parameter.

For the sake of completeness, we briefly outline structured and flexible variational approaches that better model the posterior and capture posterior dependencies. *Hierarchical variational inference* (Ranganath et al., 2016b) posits a two-level hierarchical approximation that draws the mean-field variational parameters λ from a multivariate prior $q_\alpha(\lambda)$. This induces a family of distributions with the variational parameters marginalized out: $q_\alpha(\theta) = \int q_\alpha(\lambda) q_\lambda(\theta) d\lambda$. The marginalization allows the approximation to capture posterior dependencies. *Boosting variational inference* (BVI) (Guo et al., 2017; Locatello et al., 2018) extends the variational framework such that the approximation is a member of the set of all possible finite mixtures of a parametric base distribution. Inspired by gradient boosting (Friedman, 2000), BVI successively improves accuracy of the approximation by adding a new variational component $q^k_\lambda(\theta)$ from the base distribution to produce efficient and flexible approximations to multi-model posteriors.

Amortized approaches such as *variational autoencoders* (VAE) (Kingma and Welling, 2014; Kingma et al., 2016) and *normalizing flows* (Rezende and Mohamed, 2015) amortize the cost of inference by sharing variational parameters across data points in $\mathcal{D}$. VAEs approximate the posteriors via an amortized mean-field approximation conditional on the datapoints: $q_\lambda(\theta|\mathcal{D}) = \prod_{i=1}^{|\mathcal{D}|} q_\lambda(\theta_i|\mathcal{D}_i)$. The variational parameters λ are parameters of a neural network that learns a non-linear mapping from the data point $\mathcal{D}_i$ to θ_i. Normalizing flows construct variational approximations to complex posteriors by passing a simple parameter-free density function through a sequence of invertible transformations until the approximation attains the

desired level of complexity. We refer the reader to Zhang et al. (2019) for an overview of recent trends and innovations in variational inference.

2.3 Variational Inference

Variational inference posits a variational family $q_\lambda(\theta)$ and selects a member of this family to approximate the posterior $p(\theta|\mathcal{D})$ by minimizing the reverse KL divergence $\mathrm{KL}(q\|p)$ in Equation 2.4. The approximation $q_\lambda(\theta)$ is typically under-parameterized, resulting in variational inference being categorized as approximate since the true posterior $p(\theta|\mathcal{D})$ is often not a member of the variational family. Since it is impractical to directly minimize $\mathrm{KL}(q\|p)$, we instead maximize a functionally equivalent quantity, the evidence lower bound $\mathcal{L}(\lambda)$, and prove that maximization of the lower bound corresponds to the minimization of the KL-divergence. To show this equivalence we first derive the evidence lower bound $\mathcal{L}(\lambda)$.

$$\begin{aligned}
\log p(\mathcal{D}) &= \log \int p(\mathcal{D}, \theta)\ d\theta, \\
&= \log \int \frac{q_\lambda(\theta)}{q_\lambda(\theta)} p(\mathcal{D}, \theta)\ d\theta, \\
&\geq \int q_\lambda(\theta) \log \frac{p(\mathcal{D}, \theta)}{q_\lambda(\theta)}\ d\theta, \qquad (2.5) \\
&= \mathcal{L}(\lambda).
\end{aligned}$$

The inequality appearing in Equation 2.5 arises from Jensen's rule. From the definition of $\mathrm{KL}(q\|p)$,

$$\begin{aligned}
\mathrm{KL}(q\|p) &= \int q_\lambda(\theta) \frac{q_\lambda(\theta)}{p(\theta|\mathcal{D})}\ d\theta, \\
&= \int q_\lambda(\theta) \frac{q_\lambda(\theta)}{p(\theta, \mathcal{D})}\ d\theta + \log p(\mathcal{D}), \\
&= -\mathcal{L}(\lambda) + \log p(\mathcal{D}).
\end{aligned}$$

This leads to the equivalence

$$\mathrm{KL}(q\|p) + \mathcal{L}(\lambda) = \log p(\mathcal{D}).$$

Figure 2.2 shows a schematic that provides a visual intuition of how maximizing $\mathcal{L}(\lambda)$ is functionally equivalent to minimizing the $\mathrm{KL}(q\|p)$. The lower bound $\mathcal{L}(\lambda) \leq \log p(\mathcal{D})$ with equality at $\mathrm{KL}(q\|p) = 0$. The variational objective to optimize is therefore the evidence lower bound $\mathcal{L}(\lambda)$

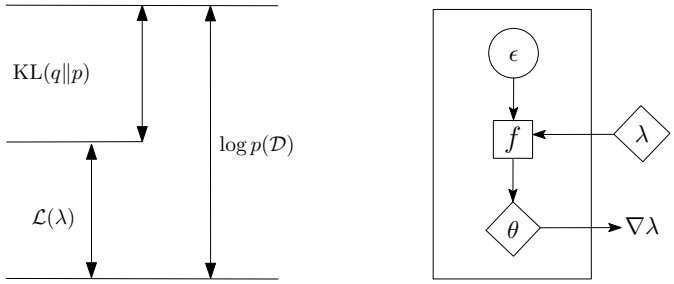

Figure 2.2: Schematics for variational inference (left) and the reparameterization trick (right). The model evidence $p(\mathcal{D})$ is the sum of the evidence lower bound $\mathcal{L}(\lambda)$ and $\mathrm{KL}(q\|p)$. Maximizing the lower bound produces optimal variational parameters λ^* that yields an approximation $q_{\lambda^*}(\theta)$ which minimizes $\mathrm{KL}(q\|p)$. The lower bound is optimized via gradient descent. However, the gradient of the lower bound $\nabla_\lambda \mathcal{L}(\lambda)$ with respect to λ cannot be calculated easily as the expectation in Equation 2.6 is over $q_\lambda(\theta)$. By reparameterizing $\theta = f_\lambda(\epsilon)$ with a deterministic transformation, the expectations over $q_\lambda(\theta)$ can be rewritten as expectations over a parameter-free density $q_0(\epsilon)$ as shown in Equation 2.11. This allows $\nabla_\lambda \mathcal{L}(\lambda)$ to be easily evaluated via Monte Carlo methods.

defined as

$$\begin{aligned}\mathcal{L}(\lambda) &= \int q_\lambda(\theta) \log \frac{p(\mathcal{D},\theta)}{q_\lambda(\theta)}\, d\theta, \\ &= \mathbb{E}_q \underbrace{[\log p(\mathcal{D},\theta) - \log q_\lambda(\theta)]}_{g(\theta)}, \\ &= \mathbb{E}_q [\log p(\mathcal{D},\theta)] - \underbrace{\mathbb{E}_q [\log q_\lambda(\theta)]}_{\text{Entropy term: } \mathbb{H}(q)},\end{aligned} \tag{2.6}$$

where we abbreviate $q_\lambda(\theta)$ as q for clarity and

$$g(\theta) = \log p(\mathcal{D},\theta) - \log q_\lambda(\theta). \tag{2.7}$$

Thus, $\mathrm{KL}(q\|p)$ is minimized with optimal variational parameters λ^* obtained as

$$\lambda^* = \arg\max_\lambda \mathcal{L}(\lambda).$$

Gradient-based optimization is the tool of choice for maximizing $\mathcal{L}(\lambda)$. Traditional methods use coordinate ascent to optimize $\mathcal{L}(\lambda)$ analytically in closed form (Blei et al., 2017; Jordan et al., 1999). The mean-field assumption allows the fully-factorized variational approximation to be updated

Algorithm 2: SGD for variational inference

input : data $\mathcal{D}$, model $p(\mathcal{D}, \theta)$, the variational approximation $q_\lambda(\theta)$, batches B
output: optimal variational parameters λ^*
Initialize counter $t = 0$.
Set rate schedule ρ_t as per Robbins-Monro condition.
Initialize λ.
while $\mathcal{L}(\lambda)$ *has not converged* **do**
  Select a minibatch $\mathcal{D}_b$ from $\mathcal{D}$
  Estimate $\nabla_\lambda \mathcal{L}(\lambda)$ for $\mathcal{D}_b$ using Equation 2.10 or 2.12
  $\lambda_{t+1} \leftarrow \lambda_t + \rho_t \nabla_\lambda \mathcal{L}(\lambda_t)$
  Increment iteration counter t
end
Return $\lambda^* \leftarrow \lambda$

successively until convergence as

$$q_\lambda^i(\theta) \propto \exp(\mathbb{E}_{q^{\setminus i}}[\log p(\mathcal{D}, \theta)]),$$

where $q^{\setminus i}$ refers to all factors excluding $q_\lambda^i(\theta)$. As long as the model is conditionally conjugate, the expectations over $q^{\setminus i}$ can be obtained analytically. Although they converge rapidly, coordinate ascent techniques require updates that are tedious to derive and prone to error. The inference procedure is model-specific and non-conjugate models require local variational approximations to obtain analytical expectations. A comprehensive review of mean-field techniques for exponential families can be found in Jordan et al. (1999) and Blei et al. (2017).

The advent of automatic differentiation frameworks (Abadi et al., 2016; Salvatier et al., 2016; Kucukelbir et al., 2017) has ushered in more generic inference schemes to solve a broader variety of models. Gradient-based optimization of the variational objective $\mathcal{L}(\lambda)$ requires computing the gradient

$$\nabla_\lambda \mathcal{L}(\lambda) = \nabla_\lambda \mathbb{E}_q \left[g(\theta) \right]. \tag{2.8}$$

Assuming the gradient $\nabla_\lambda \mathcal{L}(\lambda)$ can be estimated through black-box methods, it is straightforward to devise a stochastic gradient descent procedure (Robbins and Monro, 1951), with a rate schedule that follows the Robbins-Monro condition, shown in Algorithm 2. This updates λ with stochastic estimates of $\nabla_\lambda \mathcal{L}(\lambda)$ to obtain optimal variational parameters λ^*. We now outline three schemes of varying accuracy that produce unbiased Monte Carlo estimates of Equation 2.8.

2.3.1 Score function gradients

The score function gradient (Cox and Hinkley, 1979; Ranganath et al., 2014) uses the log derivative trick to estimate $\nabla_\lambda \mathcal{L}(\lambda)$. It is based on the general identity for gradients of logarithms

$$\nabla_\lambda \log q_\lambda(\theta) = \frac{\nabla_\lambda q_\lambda(\theta)}{q_\lambda(\theta)}.$$

Using this identity, we obtain the gradient

$$\begin{aligned}
\nabla_\lambda \mathcal{L}(\lambda) &= \nabla_\lambda \mathbb{E}_q \left[g(\theta) \right], \\
&= \int \nabla_\lambda g(\theta) q_\lambda(\theta) \; d\theta, \\
&= \underbrace{\int q_\lambda(\theta) \nabla_\lambda g(\theta) \; d\theta}_{(a)=0} + \int g(\theta) \nabla_\lambda q_\lambda(\theta) d\theta, \\
&= \int \frac{q_\lambda(\theta)}{q_\lambda(\theta)} g(\theta) \nabla_\lambda q_\lambda(\theta), \quad \because \frac{q_\lambda(\theta)}{q_\lambda(\theta)} = 1, \\
&= \int q_\lambda(\theta) g(\theta) \nabla_\lambda \log q_\lambda(\theta), \quad \because \nabla_\lambda \log q_\lambda(\theta) = \frac{\nabla_\lambda q_\lambda(\theta)}{q_\lambda(\theta)}, \\
&= \mathbb{E}_q \left[g(\theta) \nabla_\lambda \log q_\lambda(\theta) \right]
\end{aligned} \tag{2.9}$$

where $g(\theta)$ is the shorthand notation in Equation 2.7. Unwrapping the equation, we first change the order of integration and differentiation and take the derivative of the product using the product rule. We now show how the quantity (a) is 0. From the definition of $g(\theta)$ in Equation 2.7 we observe that both terms reduce to 0: $\nabla_\lambda \log p(\mathcal{D}, \theta) = 0$ since it is constant with respect to λ and by application of the log deriviative trick and using the fact that $q_\lambda(\theta)$ is a density function integrating to 1, we conclude that $\mathbb{E}_q[\nabla_\lambda \log q_\lambda(\theta)] = 0$. The crucial observation in Equation 2.9 is that the gradient is over the variational approximation and not over $g(\theta)$. Therefore, as long as the approximation is differentiable, it does not matter if the log density $g(\theta)$ is differentiable. We can obtain Monte Carlo estimates of the gradient with M samples as

$$\nabla_\lambda \mathcal{L}(\lambda) \approx \frac{1}{M} \sum_{m=1}^{M} g(\theta_m) \nabla_\lambda \log q_\lambda(\theta_m), \quad \theta_m \sim q_\lambda(\theta) \tag{2.10}$$

This makes the approach very general: as long as we pick an approximation for which derivatives are easy to compute we can perform variational inference for any model.

Although the score function estimator is unbiased, it suffers from a high variance that robs the estimates of any practical value leading to small step sizes and slow convergence. The variance can be reduced by using *control variates* (Boyle, 1977). A control variate $h(\theta)$ is a family of functions that shares the same expectation. It can be added to the score function estimator to reduce the variance while retaining the expectations. Consider, for instance, the control variate parameterized by a

$$\hat{g}(\theta) = g(\theta) - a\left(h(\theta) - \mathbb{E}_q[h(\theta)]\right).$$

A prudent choice of a yields a function $\hat{g}(\theta)$ that has $\mathbb{E}_q[g] = \mathbb{E}_q[\hat{g}]$ with a reduced variance $\mathrm{Var}_q[\hat{g}] < \mathrm{Var}_q[g]$. The value of a depends on the choice of h. Applying the variance operator across the equation yields

$$\mathrm{Var}_q[\hat{g}(\theta)] = \mathrm{Var}_q[g(\theta)] + a^2\mathrm{Var}_q[(h(\theta)] - 2a\mathrm{Cov}_q(g, h).$$

We see that a choice of a and h that maximizes $2a\mathrm{Cov}_q(g, h)$ results in maximal variance reduction in $\hat{g}(\theta)$. To get the optimal a^*, for instance, we differentiate with respect to a and set the equation to zero to obtain $a^* = \mathrm{Cov}_q(g, h)/\mathrm{Var}_q[(h(\theta)]$. While control variates help reduce the variance of score function estimators, reparameterization methods, discussed next, yield unbiased estimators with much lower variances

2.3.2 Reparameterization methods

A low-variance estimator of $\nabla_\lambda \mathcal{L}(\lambda)$ can be obtained by representing the model parameters θ as deterministic transformations of samples from a standard, parameter-free distribution, $\epsilon \sim q_0(\epsilon)$ (Opper and Archambeau, 2009; Salimans and Knowles, 2013; Kingma and Welling, 2014; Titsias and Lázaro-Gredilla, 2014; Kucukelbir et al., 2017). The gradient $\nabla_\lambda \mathbb{E}_q[g(\theta)]$ is difficult to compute since the variational parameters are tied to the expectation via $q_\lambda(\theta)$, thereby preventing the gradient from passing through the expectation. This issue can be rectified by reparameterizing θ and changing the expectation to be over a parameter-free distribution $q_0(\epsilon)$ that is independent of the variational parameters λ. The reparameterization trick samples $\epsilon \sim q_0(\epsilon)$ and transforms it to samples $\theta \sim q_\lambda(\theta)$ through the deterministic transformation $\theta = f_\lambda(\epsilon)$. The transformation $f_\lambda(\epsilon)$ must be differentiable and invertible. The inverse transformation $\epsilon = s_\lambda(\theta)$ is known as the standardization function as it transforms θ back to samples from the underlying noise distribution. By a change of variables, the expectation over $q_\lambda(\theta)$ can be transformed into an expectation over the parameter-free

distribution $q_0(\epsilon)$ as

$$\begin{aligned}\nabla_\lambda \mathbb{E}_q[g(\theta)] &= \nabla_\lambda \mathbb{E}_{q_0}[g(f_\lambda(\epsilon)) + \log|\det_{J_f}(\epsilon,\lambda)|], \\ &= \mathbb{E}_{q_0}\left[\nabla_\lambda \left(g(f_\lambda(\epsilon)) + \log|\det_{J_f}(\epsilon,\lambda)|\right)\right], \end{aligned} \tag{2.11}$$

where $g(\theta)$ is the shorthand notation in Equation 2.7. The log determinant of the Jacobian $|\det_{J_f}(\epsilon,\lambda)|$ accounts for the change in volume introduced by the transformation $f_\lambda(\epsilon)$. Since the expectation $\mathbb{E}_{q_0}[g(f_\lambda(\theta))]$ no longer depends on λ, the gradient ∇_λ can be brought into the expectation term. However, for the reparametrization trick to work, $g(\theta)$ should be differentiable with respect to the variational parameters. This stands in contrast to score function estimators that require only the variational approximation to be differentiable. The Monte Carlo estimates of the gradient with M samples is

$$\nabla_\lambda \mathcal{L}(\lambda) \approx \frac{1}{M}\sum_{m=1}^{M} \nabla_\lambda \left(g(f_\lambda(\epsilon_m)) + \log|\det J_f(\epsilon_m,\lambda)|\right), \quad \text{where } \epsilon \sim q_0(\epsilon). \tag{2.12}$$

The Monte Carlo estimates based on reparameterization are unbiased and have been empirically shown to have low variance (Kucukelbir et al., 2017) which is ascribed to the gradients of $g(\theta)$ being better informed about the direction of the posterior mode. Although the entropy term in Equation 2.6 can be estimated analytically, evaluating entropy with Monte Carlo samples provides estimates with low variance as the approximation approaches the true posterior. Roeder et al. (2017) show that the reparameterization gradient can be decomposed into a path derivative and a score function. By dropping the score function as the approximation converged, they obtained a lower variance estimator of the gradient.

Illustration: Normal Approximations

For further clarity, we provide a practical example of using reparameterization gradients in the context of full-rank Gaussian variational approximations, following Titsias and Lázaro-Gredilla (2014). Full-rank variational inference approximates the posterior $p(\theta|\mathcal{D})$ as a multivariate normal distribution $\mathcal{N}(\theta;\mu,L)$ where μ is the mean and L is the Cholesky decomposition of the covariance matrix Σ. We reparameterize θ as a deterministic transformation of a parameter-free base distribution $q_0(\epsilon)$. This can be done by choosing as the base distribution the standard normal, and drawing samples $\epsilon \sim \mathcal{N}(0,1)$. The draws ϵ can be transformed to θ with the deterministic location-scale transformation $\theta = L\epsilon + \mu$. For this choice of transformation

the lower bound can be written as

$$\begin{aligned}\mathcal{L}(\mu, L) &= \int q(\theta; \mu, L) \log \frac{p(\mathcal{D}, \theta)}{q(\theta; \mu, L)} d\theta \\ &= \int \mathcal{N}(\epsilon; 0, I) \log \frac{p(\mathcal{D}, L\epsilon + \mu)|L|}{\mathcal{N}(\epsilon; 0, I)} \, d\epsilon \\ &= \mathbb{E}_{\mathcal{N}(\epsilon;0,I))} \left[\log p(\mathcal{D}, L\epsilon + \mu) + \log |L|\right] + \mathbb{H}(\mathcal{N}(\epsilon; 0, I)),\end{aligned}$$

where $\mathbb{H}(.)$ is the entropy and the Jacobian of the reverse transformation $\epsilon = L^{-1}(\theta - \mu)$ is $|L|$ and accounts for the change of variables. Since the expectation is now over the standard distribution $\mathcal{N}(\epsilon; 0, 1)$ that does not depend on the parameters μ or L of the approximation, we can evaluate derivatives with respect to them. Using the chain rule with the identities $\nabla_\mu(L\epsilon + \mu) = 1$ and $\nabla_L(L\epsilon + \mu) = \epsilon$,

$$\mathbb{E}\left[\nabla_\mu \log p(\mathcal{D}, L\epsilon + \mu)\right] = \mathbb{E}\left[\nabla_\theta \log p(\mathcal{D}, \theta)\right], \tag{2.13}$$

$$\mathbb{E}\left[\nabla_L \log p(\mathcal{D}, L\epsilon + \mu)\right] = \mathbb{E}\left[\nabla_\theta \log p(\mathcal{D}, \theta)\epsilon^T\right]. \tag{2.14}$$

We drop $\mathbb{H}$ as it is a constant and use the identity $\nabla_L \log |L| = \text{diag}(L)^{-1}$ where diag retrieves the vector of diagonals elements. We could also consider more complex transformations from ϵ to θ by successively applying the change of variables technique. For example, by setting $\theta = \exp(L\epsilon + \mu)$ we could constrain θ to take only positive values.

Implicit Reparameterization

Reparameterization gradients require that the variational approximation be expressible as a deterministic transformation $f_\lambda(\epsilon)$. However, this clause is quite restrictive since it precludes a number of standard distributions such as the Gamma, Beta or Dirichlet distribution from use as variational approximations owing to the lack of a simple deterministic transformation. Some workarounds, such as Ruiz et al. (2016), extend the reparameterization gradient to a wider class of approximations by using transformations that lead to a base distribution $q_0(\epsilon; \lambda)$, which weakly depend on the variational parameters.

Although the cumulative density function (CDF) of the approximation serves as a universal standardization function $s_\lambda(\theta)$, inverting it to obtain the transformation $f_\lambda(\epsilon)$ is difficult and finding derivatives of the inverted CDF is hard. Figurnov et al. (2018) proposed the following alternative reparameterization scheme that takes advantage of implicit gradients and does not require inversion of the standardization function.

$$\nabla_\lambda \mathbb{E}_q\left[g(\theta)\right] = \mathbb{E}_q\left[\nabla_\theta g(\theta) \nabla_\lambda \theta\right]; \quad \nabla_\lambda \theta = \nabla_\lambda f_\lambda(\epsilon)|_{\epsilon = s_\lambda(\theta)}.$$

Figurnov et al. (2018) observe that the gradient $\nabla_\lambda \theta$ can be calculated by implicit differentiation. Applying the total derivative to the equality $s_\lambda(\theta) = \epsilon$ and using the chain rule gives: $\nabla_\theta s_\lambda(\theta) \nabla_\lambda \theta + \nabla_\lambda s_\lambda(\theta) = 0$, yielding the equivalence

$$\nabla_\lambda \theta = \frac{-\nabla_\lambda s_\lambda(\theta)}{\nabla_\theta s_\lambda(\theta)} \, .$$

Note that the gradient $\nabla_\lambda \theta$ requires access only to the standardization function $s_\lambda(\theta)$ and there is no need for $f_\lambda(\theta)$ to be explicitly defined or analytically tractable. When $f_\lambda(\theta)$ is a location-scale transformation, both implicit and explicit reparameterization gradients produce the same results. However, when no location-scale transformation exists and $f_\lambda(\theta)$ cannot be defined, we can use a universal standardization function such as the CDF to estimate the gradients. A CDF such as $s_\lambda(\theta) = F_\lambda(\theta) \sim \text{Uniform}(0, 1)$. is continuously differentiable and

$$\nabla_\lambda \theta = \frac{-\nabla_\lambda F_\lambda(\theta)}{q_\lambda(\theta)}$$

where the denominator $\nabla_\theta s_\lambda(\theta) = \nabla_\theta F_\lambda(\theta) = q_\lambda(\theta)$. The numerator $\nabla_\lambda F_\lambda(\theta)$ requires differentiating the CDF with respect to λ. When the CDF is intractable, as in the case of Gamma and von Mises distributions (Figurnov et al., 2018), one can simply use forward differentiation of numerical approximations – such as Bhattacharjee (1970) – of the CDF. Therefore, as long as the CDF is *numerically* tractable, implicit reparameterization yields low-variance estimates of the gradient. Practical implementations of implicit reparameterization routines is available in `Tensorflow Probability` (Dillon et al., 2017)

This concludes our discussion of automatic methods for optimizing the variational objective. Now that we have covered the necessary elements of approximate Bayesian inference, in the next section we turn our attention to a statistical framework for decision-making with Bayesian posteriors.

2.4 Decision Theory

A statistical decision problem can be thought of as a game between a statistical model and the data-generating mechanism (Murphy, 2012). In this game, the data-generating mechanism picks a parameter $\theta^* \in \Theta$ which is hidden from the model, and generates a set of features $x \in \mathcal{X}$ that are observed by the model as well as a set of outputs $y \in \mathcal{Y}$ that are unknown to the model and the dataset $\mathcal{D} = \{x, y\}$. The task of the model then is to

produce a decision, $h \in \mathcal{H}$, from a set of plausible decisions. The nature of decision h and the decision space $\mathcal{H}$ depends upon the kind of task we intend to solve. In standard *estimation* tasks, the decision h amounts to estimating a parameter that minimizes a loss $\ell(\theta^*, h)$ that measures the compatibility of h with the true, hidden θ^*. This is achieved by minimizing the expected loss, known as the *risk*, over the sampling distribution $p(\widehat{\mathcal{D}}|\theta^*)$,

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{p(\widehat{\mathcal{D}}|\theta^*)} \left[\ell(\theta^*, h(\widehat{\mathcal{D}})) \right]$$

where we use the notation $h(\widehat{\mathcal{D}})$ to imply that the parameter estimate is obtained from the data $\widehat{\mathcal{D}}$ sampled from "nature's distribution" $p(\widehat{\mathcal{D}}|\theta^*)$ parameterized by the true parameter θ^*. However, this formulation is not very practical — the risk cannot be evaluated since it requires access to the true data-generating distribution $p(\widehat{\mathcal{D}}|\theta^*)$. This classical decision theoretic formulation can still be of use in the standard predictive settings to predict unknown outcomes y via empirical risk minimization.

In *predictive* tasks, we are interested in predicting the observable but unknown outcome $y \in Y$. The decisions h are chosen to minimize the loss function $\ell(y, h)$ which measures the compatibility of the decisions h with the outcomes y. Standard examples of such losses include the squared loss $\ell(y, h) = (y - h)^2$ and absolute loss $\ell(y, h) = |y - h|$ when $\mathcal{Y}$ is continuous. In discrete binary classification tasks, the loss of choice is the $0-1$ loss $\ell(y, h) = \mathbf{1}(y \neq h)$. The optimal predictions h^* that minimize the expected loss in predictive setups is defined as

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(x,y) \sim p(\widehat{\mathcal{D}}|\theta^*)} \left[\ell(y, h(x)) \right]$$

where $h(x)$ signifies the decision h is a prediction based on x. While the true distribution $p(\widehat{\mathcal{D}}|\theta^*)$ is still unknown, we can now use simple empirical approximations based on the data set $\mathcal{D}$ to approximate the expected loss as

$$\mathbb{E}_{(x,y) \sim p(\widehat{\mathcal{D}}|\theta^*)} \left[\ell(y, h(x)) \right] \approx \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, h(x_n))$$

where $N = |\mathcal{D}|$ and we have defined the empirical distribution to be

$$p(\mathcal{D}|\theta^*) \approx \frac{1}{N} \sum_{n=1}^{N} \delta_{x_n}(x) \delta_{y_n}(y)$$

where δ is the Dirac delta function. *Empirical risk minimization* finds

optimal decisions h^* to minimize the empirical risk

$$h^*_{\text{emp}} = \arg\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, h(x_n)).$$

We utilize the empirical risk for post-hoc calibration of Bayesian posteriors in Section 4.2. Next, we discuss the Bayesian approach to decision making which provides an alternative definition of the expected loss. Instead of defining the expectation over the true data-generating distribution, Bayesian decision theory defines the expected loss over the posterior of the model parameters $p(\theta|\mathcal{D})$. This allows the Bayesian decision framework to be applied effortlessly in both estimation and prediction tasks.

2.4.1 Bayesian decision theory

Bayesian decision theory outlines sound principles of decision-making under posterior uncertainty $p(\theta|\mathcal{D})$. It depends on the determination of three components:

1. a loss function $\ell(\theta, h)$ that measures the compatibility of the decision h with θ,
2. the conditional distribution $p(\mathcal{D}|\theta)$ that defines the likelihood of the observations $\mathcal{D}$ under the parameters θ,
3. a prior $p(\theta)$ on the parameter θ.

Any linear transformation of the loss function $\ell(\theta, h)$

$$\tilde{\ell}(\theta, h) = \alpha \cdot \ell(\theta, h) + \beta, \quad \alpha > 0.$$

can serve in its place (Berger, 1985). The optimal decision h remains the same for this family of loss function since any h that minimizes $\ell(\theta, h)$ also minimizes $\tilde{\ell}(\theta, h)$. It is sometimes more natural to formulate a decision problem in terms of a utility function $u(\theta, h)$ that measures the reward associated with predicting h when the true state is θ. Upon constructing $u(\theta, h)$, a simple linear transformation such as

$$\ell(\theta, h) = -u(\theta, h)$$

yields an equivalent loss function. The maximization of the utility can therefore be recast as the minimization of the loss and vice versa. Without loss of generality, the loss function is assumed to be non-negative for convenience's sake; and hence, it naturally follows that $u(\theta, h)$ is negative. However, derivations for the loss-calibrated variational bound in Section 4.1,

require defining $u(\theta, h)$ to be greater than 0. In order to define a coherent utility function in such cases, we can use an alternative loss-to-utility transformation

$$u(\theta, h) = M - \ell(\theta, h)$$

where $M \geq \sup_{\theta,h} \ell(\theta, h)$ yielding $u(\theta, h) \geq 0$. Although the assumption that the loss $\ell(\theta, h)$ is bounded from above by M limits the transformation to losses that are bounded, we develop empirical heuristics in Subsection 4.1.4 that allow transformation to positive utilities even in cases where the loss is not bounded.

We adopt an unconventional definition of $u(\theta, h)$ that allows our discussion of decision theory to be applicable to both estimation and prediction tasks simultaneously. This generic utility is defined as

$$u(\theta, h) = \begin{cases} \int \tilde{u}(y, h) p(y|\theta, \mathcal{D}) \, dy & \text{in predictive settings,} \\ \tilde{u}(\theta, h) & \text{in estimation settings.} \end{cases} \tag{2.15}$$

The utility for estimation tasks is by definition simply $\tilde{u}(\theta, h)$ for the parameters θ over the dataset $\mathcal{D}$. Defining the utility function for predictive tasks is a little more involved. First, we do not model the marginal distribution $p(x)$ of x. Second, since the decisions h are conditional on x, we make pointwise decisions $h(x)$ for every $x \in \mathcal{X}$. Thus, the conditional utility $u(\theta, h)$ defined in terms of $\tilde{u}(y, h)$ is

$$u(\theta, h) = \int \tilde{u}(y, h) p(y|\theta, \mathcal{D}) \, dy.$$

Having developed a language for utility specification we can define the *gain* or the *expected posterior utility* as

$$\mathcal{G}_u^p(h) = \int p(\theta|\mathcal{D}) u(\theta, h) \, d\theta. \tag{2.16}$$

An equivalent formulation can be obtained by defining the *risk* or the *expected posterior loss* as

$$\mathcal{R}_\ell^p(h) = \int p(\theta|\mathcal{D}) \ell(\theta, h) \, d\theta. \tag{2.17}$$

Of course, since the posterior expectation is rarely computed analytically, we resort to simple Monte Carlo approximations of $\mathcal{G}_u(h)$. For estimation tasks,

$$\mathcal{G}_u^p(h) \approx \frac{1}{M} \sum_{m=1}^{M} \tilde{u}(\theta_m, h), \quad \theta_m \sim p(\theta|\mathcal{D}).$$

The definition of Bayesian risk as the expected loss over the posterior distribution $p(\theta|\mathcal{D})$ allows evaluating risk in estimation settings. For predictive tasks, the gain is approximated by

$$\mathcal{G}_u^p(h) \approx \frac{1}{SM} \sum_{s=1}^{S} \sum_{m=1}^{M} \tilde{u}(y_{s,m}, h), \quad \theta_m \sim p(\theta|\mathcal{D}), \quad y_{s,m} \sim p(y|\theta_m, \mathcal{D}).$$

We use S samples from the posterior distribution and SM samples from the posterior predictive.

Rational decision making is defined in terms of the maximum expected utility principle. That is, the Bayes optimal decision that maximizes the gain is obtained as

$$h^* = \arg\max_{h \in \mathcal{H}} \mathcal{G}_u^p(h) \equiv \arg\min_{h \in \mathcal{H}} \mathcal{R}_\ell^p(h).$$

As a final point, as we have discussed before, the posterior $p(\theta|\mathcal{D})$ is intractable and substituting the approximation $q(\theta)$ in place of $p(\theta|\mathcal{D})$ does not always preserve the integrity of Bayesian decision making.

2.4.2 Approximate Decision Theory

The Bayesian decision framework operates under the assumption that the posterior $p(\theta|\mathcal{D})$ provides necessary and sufficient information to make Bayes optimal decisions. It can however be argued (Lacoste-Julien et al., 2011) that the decisions made under approximate posteriors $q(\theta)$ are not optimal with respect to $p(\theta|\mathcal{D})$ unless the true posterior itself is a member of the approximation family. We will develop techniques and heuristics that calibrate the approximate posteriors for addressing this discrepancy between the decisions in Chapter 4.

The major drawback of approximation methods is that they view posterior inference in isolation and ignore the wider context of their applicability in optimal decision making. Classical approximation techniques discussed in Section 2.3 focus on approximating the general properties of the Bayesian posterior such as its moments and skew. While this is essential for faithful representation of the posterior, these techniques are trained in a loss-agnostic manner and do not necessarily minimize the expected decision-theoretic loss over the posterior $p(\theta|\mathcal{D})$. The standard routine for handling decision-making with approximations is to infer $q(\theta)$ as a proxy to the true posterior $p(\theta|\mathcal{D})$ and choose a decision that minimizes the expected *approximate loss*. But this two-step breakdown of inference and

decision-making yields suboptimal decisions. Consider, for instance, the q-risk, which is the expected loss under the approximation $q(\theta)$,

$$\mathcal{R}_\ell^q(h) = \int q(\theta)\ell(\theta, h)\ d\theta. \tag{2.18}$$

The optimal decision that minimizes the q-risk $R_\ell^q(h)$, referred to as the q-optimal action h_q (Lacoste-Julien et al., 2011), then is

$$h_q = \arg\min_{h\in\mathcal{H}} \mathcal{R}_\ell^q(h).$$

However, h_q is not optimal with respect to the p-risk $\mathcal{R}_\ell^p(h)$ over the posterior, and

$$h_q \neq h_{\mathrm{p}} = \arg\min_{h\in\mathcal{H}} \underbrace{\int p(\theta|\mathcal{D})\ell(\theta, h)\ d\theta}_{\mathcal{R}_\ell^p(h)}.$$

Approximate decision theory reformulates risk minimization with a different objective: we couple inference and decision-making by choosing a suitable approximation $q(\theta)$ that in addition to minimizing a discrepancy measure $\mathbb{D}(q\|p)$ also yields q-optimal decision h_q that minimizes the posterior risk $\mathcal{R}_\ell^p(h_q)$

$$q_{\mathrm{opt}} = \arg\min_{q\in\mathcal{Q}} \mathbb{D}(q\|p) + \mathcal{R}_\ell^p(h_q),$$

where $\mathcal{Q}$ is the variational family. Although the quantity $\mathcal{R}_\ell^p(h_q)$ is still intractable, practical algorithms outlined in Chapter 4 minimize an approximation to this quantity.

2.5 Importance Sampling

In Chapter 3, we will need techniques that help us approximate the expectation of a function $f(\theta)$ over a target density $p(\theta)$ when such an expectation cannot be evaluated analytically. The expectation we wish to evaluate is given as

$$\mathbb{E}_p[f] = \int f(\theta)p(\theta)d\theta.$$

We can obtain a Monte Carlo estimate of this integral by approximating it with a set of samples θ_s, where $s = 1, \ldots, S$, drawn independently from the density $p(\theta)$. A finite sum estimator of the expectation is therefore

$$\widehat{f} = \frac{1}{S}\sum_{s=1}^{S} f(\theta_s), \quad \text{where } \theta_s \sim p(\theta).$$

The estimator $\widehat{f}$ is consistent and converges in mean to $\mathbb{E}_p[f]$ such that $\mathbb{E}[\widehat{f}] = \mathbb{E}_p[f]$. The variance of $\widehat{f}$ is

$$\text{Var}[\widehat{f}] = \frac{1}{S}\mathbb{E}_p\left[(f - \mathbb{E}_p[f])^2\right].$$

The estimator $\widehat{f}$ is valid only so long as we are able to draw independent samples from $p(\theta)$.

Importance sampling (Hesterberg, 1988; Bishop, 2006) is a technique to approximate the expectation $\mathbb{E}_p[f]$, when we lack a mechanism for efficiently sampling from $p(\theta)$. A proposal density $q(\theta)$, from which we are able to sample efficiently, is used to approximate the expectation as

$$\begin{aligned}\mathbb{E}_p[f] &= \int f(\theta)p(\theta)d\theta \\ &= \int \frac{q(\theta)}{q(\theta)} f(\theta)p(\theta)d\theta \\ &\approx \frac{1}{S}\sum_{s=1}^{S} \frac{p(\theta_s)}{q(\theta_s)} f(\theta_s), \quad \text{where } \theta_s \sim q(\theta).\end{aligned}$$

The fractions $p(\theta_s)/q(\theta_s)$ are the importance weights that correct the bias introduced by sampling from the wrong distribution $q(\theta)$. The samples θ_s are appropriately weighting depending on how likely they are under the target distribution $p(\theta_s)$ — samples that fall in the high-density regions of $p(\theta)$ are weighted up while those that lie in its low-density regions are effectively discarded as the importance weights tend to be close to zero.

The target density $p(\theta)$ is not always normalized and sometimes is known only up to normalization constant Z_p. In such cases, we have access only to the unnormalized density $\tilde{p}(\theta)$ such that $p(\theta) = \tilde{p}(\theta)/Z_p$. With importance sampling, we can utilize $\tilde{p}(\theta)$ to evaluate $\mathbb{E}_p[f]$. We use a proposal distribution with similar structure as $p(\theta)$ such that $q(\theta) = \tilde{q}(\theta)/Z_q$ where $\tilde{q}(\theta)$ and Z_q are the unnormalized proposal density and the normalization constant respectively of $q(\theta)$. We can then write the expectation as

$$\begin{aligned}\mathbb{E}_p[f] &= \frac{1}{Z_p}\int f(\theta)\tilde{p}(\theta)d\theta \\ &= \frac{Z_q}{Z_p}\int \frac{\tilde{p}(\theta)}{\tilde{q}(\theta)} f(\theta)q(\theta)d\theta \\ &\approx \frac{Z_q}{Z_p}\frac{1}{S}\sum_{s=1}^{S} \frac{\tilde{p}(\theta_s)}{\tilde{q}(\theta_s)} f(\theta_s) \quad \text{where } \theta_s \sim q(\theta)\end{aligned}$$

Here, $r_s = \tilde{p}(\theta_s)/\tilde{q}(\theta_s)$ are the *unnormalized* importance ratios. We can use these weights to approximate the ratio Z_p/Z_q using the following identity

$$\begin{aligned}\frac{Z_p}{Z_q} &= \frac{1}{Z_q}\int \tilde{p}(\theta)d\theta \\ &= \int \frac{\tilde{p}(\theta)}{\tilde{q}(\theta)} q(\theta)d\theta \\ &\approx \frac{1}{S}\sum_{s=1}^{S} r_s. \end{aligned} \tag{2.19}$$

Using this identity expectation $\mathbb{E}_p[f]$ can be estimated with the unnormalized $\tilde{p}(\theta)$ as

$$\begin{aligned}\mathbb{E}_p[f] &\approx \frac{Z_q}{Z_p}\frac{1}{S}\sum_{s=1}^{S}\frac{\tilde{p}(\theta_s)}{\tilde{q}(\theta_s)} f(\theta_s) \\ &\approx \left(S\frac{1}{\sum_{t=1}^{S} r_t}\right)\frac{1}{S}\sum_{s=1}^{S} r_s f(\theta_s) \\ &= \sum_{s=1}^{S}\frac{r_s}{\sum_t r_t} f(\theta_s).\end{aligned}$$

The normalized importance weights are defined to be $w_s = r_s/\sum_t r_t$. It remains important to carefully choose $q(\theta)$ to match the target density $p(\theta)$. A poor choice of $q(\theta)$ will yield a set of importance weights w_s that are dominated by a few weights leading to reduced effective sample size. Even worse, if none of the samples from $q(\theta)$ fall in regions where $f(\theta)p(\theta)$ is large, the sample variances of r_s and $r_s f(\theta_s)$ may be small even if the estimate of the expectation is wrong (Bishop, 2006). This has the potential to produce estimates that have arbitrary error without access to any diagnostic information. This shortcoming has been recently rectified by Pareto-smoothed importance weighting (Vehtari et al., 2015) which stabilizes importance weights by fitting a generalized Pareto distribution to the upper tail of the importance ratios to provides stable estimates of effective sample size, Monte Carlo errors estimates and convergence diagnostics.

Estimating Risk with Importance Sampling

Although our practical algorithms in Chapter 4 minimize the risk by modifying either the posterior approximation (Section 4.1) or a representation of

the posterior predictive (Section 4.2), practitioners may sometimes prefer to simply substitute the q-risk

$$\mathcal{R}_\ell^q(h) = \int q(\theta)\ell(\theta, h)\ d\theta$$

for the p-risk. This could be due to computational or time constraints or lack of access to data among other reasons. In such cases, instead of using the q-risk as a naive plug-in, we recommend an alternative strategy to obtain an unbiased approximation of the p-risk using importance sampling,

$$\begin{aligned}
\mathcal{R}_\ell^p(h) &= \int p(\theta|\mathcal{D})\ell(\theta, h)\ d\theta \\
&\propto \int p^*(\theta|\mathcal{D})\ell(\theta, h)\ d\theta \\
&= \int \left[\frac{p^*(\theta|\mathcal{D})}{q(\theta)}\ell(\theta, h)\right] q(\theta)\ d\theta \\
&\approx \frac{1}{S}\sum_{s=1}^{S} \frac{p^*(\theta_s|\mathcal{D})}{q(\theta_s)}\ell(\theta, h) \quad \text{where } \theta_s \sim q(\theta).
\end{aligned}$$

The ratio $r_s = p^*(\theta|\mathcal{D})/q(\theta)$ can be estimated as long as the posterior density $p(\theta|\mathcal{D})$ can be evaluated up to a constant. Rainforth et al. (2020) extend this framework by using an amortized proposal $q(\theta)$ that is loss-aware and amortized over a family of loss functions.

2.6 Summary

We discussed the implication of the Bayes theorem in posterior uncertainty quantification and showed how the Bayesian posterior is intractable. We examined two major approximation methods: firstly, sampling-based methods that rely on samples from the joint density $p(\mathcal{D}, \theta)$ to approximate the posterior; secondly, optimization-based methods that minimize a divergence between the posterior and a distributional approximation. In particular, we expanded on variational inference and specified two gradient-based optimization techniques that rely on modern automatic differentiation tools to minimize the variational bounds, namely, score function gradients and reparameterization gradients. We then explored decision making under the Bayesian framework and how it compares with classical decision theory. We reformulated rational decision-making principle under approximate posteriors and showed that we benefit from integrating decision-making and approximate inference. We finally discussed importance sampling as a means

of approximating expectations over target densities from which we cannot efficiently draw independent samples. This concludes our discussion of the background required for building the contributions in the thesis. In Chapters 3 and 4 we discuss the novel contributions of this thesis.

Chapter 3

Faster Inference and Skewed Densities

We introduced stochastic, gradient-based optimization techniques for variational inference in Section 2.3. These techniques rely on a combination of automatic differentiation (Baydin et al., 2015) and Monte Carlo estimators to yield stochastic approximations of the gradient of the variational objective in Equation 2.6,

$$\mathcal{L}(\lambda) = \mathbb{E}_q \underbrace{\left[\log p(\mathcal{D}, \theta) - \log q_\lambda(\theta)\right]}_{g(\theta)}.$$

The gradient estimates of the lower bound $\mathcal{L}(\lambda)$ can be plugged into standard gradient descent algorithms to retrieve the parameters maximizing the lower bound. Estimators of the gradient rely on an appropriate characterization of $\mathbb{E}_q[g(\theta)]$ to obtain unbiased, computationally tractable estimates of the gradient. Depending on the type of characterization, these estimators are broadly classified into *score function estimators* (Subsection 2.3.1) and *reparametrization methods* (Subsection 2.3.2).

In this chapter, we introduce two novel contributions that operate within the framework of Monte Carlo gradient estimators. Firstly, we elaborate on ways of speeding-up gradient computation by using importance-sampling to reuse parts of the gradient that are expensive to evaluate and demonstrate how they fit within the broader context of stochastic optimization routines. Our contribution helps improve the convergence speed of probabilistic programs by offering a simple, yet effective tweak to existing stochastic gradient algorithms that form the backbone of automatic inference engines. Secondly, we introduce a differentiable transformation to skew standard distributions from the location-scale family for use in building differentiable

models for skewed data as well as variational approximations that utilize reparameterization to model skewed posteriors.

3.1 Importance Sampled Gradients

Research in optimization-based inference primarily focuses on obtaining flexible and amortized approximations of Bayesian posteriors (Rezende and Mohamed, 2015; Kingma and Welling, 2014; Ranganath et al., 2016a) or on the generalization of gradient estimators to arbitrary distributions (Ruiz et al., 2016; Naesseth et al., 2017; Figurnov et al., 2018). Most such methods discussed in Subsection 2.2.2 rely on stochastic gradient descent as the tool of choice to solve the resulting optimization problem. Since it is such a powerful tool, studying the stochastic optimizer in the specific context of variational inference yields interesting insights into speeding up convergence. For instance, Roeder et al. (2017) show that reduction in variance of the gradient by removing part of the derivative as it approaches the optimum results in quicker convergence. In this section, we construct an importance-sampling estimator to speed-up gradient computation and propose ways in which the structure of the gradient can be exploited to obtain faster estimates of the gradient. We then outline novel ways to employ such gradient estimates in standard optimization routines such as stochastic gradient descent (SGD) and stochastic average gradient descent (SAG) (Schatz et al., 2014).

To further illustrate this point, consider the typical reparameterization gradient estimate with the transformation $\theta = f_\lambda(\epsilon)$: the computationally intensive part of $g(\theta)$ is $\log p(\mathcal{D}, \theta)$ and its gradient with respect to the variational parameters is written as

$$\nabla_\lambda \log p(\mathcal{D}, \theta) = \underbrace{\nabla_\theta \log p(\mathcal{D}, \theta)}_{\text{model gradients}} \underbrace{\nabla_\lambda f_\lambda(\epsilon) + \nabla_\lambda \log \left|\det J_f(\epsilon_m, \lambda)\right|}_{\text{transformation gradients}}.$$

The decomposition is obtained by simple application of the chain rule and consists of two conceptual parts, namely, the model gradients and the transformation gradients. The computational cost of the model gradients is a function of the model complexity and the data size $|\mathcal{D}|$ because for i.i.d data we have

$$\log p(\mathcal{D}, \theta) = \sum_{n=1}^{|\mathcal{D}|} \log p(d^n, \theta)$$

where d_n is the n^{th} data point in $\mathcal{D}$. The cost of transformation gradients depends on the complexity of the transformation $f_\lambda(\epsilon)$ which is often a

simple linear or non-linear transformation. In a standard reparameterization setup with SGD, the bulk of the computational cost is expended on evaluating the model gradient. The question then arises whether we could simply reuse the model gradients to obtain new estimates of the gradient $\nabla_{\lambda'} \log p(\mathcal{D}, \theta)$ for $\lambda' = \lambda + \delta$, by recomputing the transformation gradients alone to account for the change in the variational parameters. We show that when the transformation $f_\lambda(\epsilon)$ is invertible and one-to-one, importance sampling can be utilized to estimate the new gradient $\nabla_{\lambda'} \log p(\mathcal{D}, \theta))$ by recomputing it only partially.

3.1.1 The Importance-Sampled Estimator

Practical implementations of gradient descent to optimize the variational objective rely on two kinds of stochastic approximations. Firstly, it uses minibatches of data to approximate the gradient stochastically as

$$\nabla_\theta \log p(\mathcal{D}, \theta) = \frac{\mathcal{D}}{|\mathcal{D}_b|} \sum_{d \in \mathcal{D}_b} \nabla_\theta p(d, \theta).$$

Secondly, it uses Monte Carlo integration to estimate the expectations occurring in the lower bound and their gradients as summarized in Equation 2.10 and Equation 2.12. We introduce a third dimension of stochasticity to speed up gradient computation when the gradient is over an expectation approximated via Monte Carlo integration. While derived for the specific instance of optimizing the lower bound, we note that optimization of any differentiable function $g(\theta)$ with parameters θ, where the objective is presented as an expectation $\mathbb{E}_q[g(\theta)]$ over an arbitrary distribution $q(\theta)$, can also be sped up using the techniques outlined below.

We begin by recalling that importance sampling helps approximate the expectation when there is no mechanism to sample from the target distribution as explained in Section 2.5. Consider two variational approximations $q_\lambda(\theta)$ and $q_{\lambda'}(\theta)$ where the variational parameters λ and λ' differ by a (infinitesimally) small value such that $|\lambda' - \lambda| \leq \delta$ for some small δ. We can write the expectation of $\mathbb{E}_{q'}[g(\theta)]$ under the approximation $q_{\lambda'}(\theta)$ as

$$\mathbb{E}_{q'}[g(\theta)] = \int \frac{q_\lambda(\theta)}{q_\lambda(\theta)} g(\theta) q_{\lambda'}(\theta)\ d\theta \approx \sum_{m=1}^{M} \frac{q_{\lambda'}(\theta_m)}{q_\lambda(\theta_m)} g(\theta_m),$$

where the samples are drawn as $\theta_m \sim q_\lambda(\theta)$ and $w_m = q_{\lambda'}(\theta_m)/q_\lambda(\theta_m)$ are the importance weights. In the parlance of importance sampling, $q'_\lambda(\theta)$ is the target distribution and $q_\lambda(\theta)$ is the proposal distribution that has

similar support as the target as long as $|\lambda' - \lambda| \leq \delta$. Although, the classic use cases for importance sampling are for estimating expectations when we cannot easily draw samples from the target density and for variance reduction, in our case We are interested in taking samples $\theta_m \sim q_\lambda(\theta)$ that were previously used for estimating $\mathbb{E}_q[g(\theta)]$ and reusing them for approximating $\mathbb{E}_{q'}[g(\theta)]$ so as to avoid recomputing $g(\theta)$ for new values of θ. Therefore, although we could sample from the target $q_{\lambda'}(\theta)$ for the purposes of this estimator, we choose not to.

The importance-sampled estimator suffers from high variance when the density of $q_{\lambda'}(\theta)$ is very different from $q_\lambda(\theta)$, which here happens during early stages of optimization. There are several ways to remedy this. For instance, when θ is high-dimensional, the importance weights w_m approach zero quite rapidly. This can be avoided by applying the mean-field assumption and applying the importance-sampling technique on each factor or dimension independently, each with its own separate weights. This is especially true when computing gradients since $\nabla_\theta \log p(\mathcal{D}, \theta_m)$ can be computed over each factor separately even if $\log p(\mathcal{D}, \theta)$ cannot. This is because the gradient $\nabla_\theta p(\mathcal{D}, \theta_m)$ can be evaluated over each dimension of θ separately irrespective of the assumptions of model factorization. That is, if the number of dimensions in θ is d, we have

$$\nabla_\theta \log p(\mathcal{D}, \theta_m) = \begin{bmatrix} \frac{\partial \log p(\mathcal{D},\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial \log p(\mathcal{D},\theta)}{\partial \theta_d} \end{bmatrix}.$$

We can also monitor the weights w_m on each dimension to determine at which point the importance-sampled estimate is no longer accurate and proceed to draw a new set of samples θ_m to recompute the gradient from scratch. This can be achieved stochastically by using a parameter $\rho \sim \text{Uniform}(0, 1)$ and specifying a threshold p, so that when $\rho < p$, we recompute the gradient fully with a fresh batch of samples.

Reparameterization Gradients

We now apply the technique discussed to derive importance-sampled estimators for reparameterization gradients. Under the approximation $q_\lambda(\theta)$, the reparameterization gradient is estimated as

$$\nabla_\lambda \mathcal{L}(\lambda) = E_{q_0}\left[\nabla_\theta \log p(\mathcal{D}, \theta) \nabla_\lambda f_\lambda(\epsilon) + \nabla_\lambda \log \left|\det J_f(\epsilon, \lambda)\right|\right], \tag{3.1}$$

where the expectation is under the parameter-free distribution $q_0(\epsilon)$. We then take a gradient step towards the direction pointed at by $\nabla_\lambda \mathcal{L}(\lambda)$ to

Algorithm 3: Importance-sampled gradients

input : Samples θ_m and ϵ_m, gradient $\nabla_\theta p(\mathcal{D}, \theta_m)$, standardization function $s(.)$, current approximation λ'

output: Importance sampled gradients $\nabla_\lambda \widetilde{\mathcal{L}}(\lambda)$

$\epsilon'_m \leftarrow s_{\lambda'}(\theta_m)$

$w_m \leftarrow \frac{q_0(\epsilon'_m)}{q_0(\epsilon_m)}$

Calculate $\nabla_\lambda \widetilde{\mathcal{L}}(\lambda)$ using Equation 3.2;

obtain the new approximation $q_{\lambda'}(\theta)$. Since we have already evaluated the computationally expensive term $\nabla_\theta \log p(\mathcal{D}, \theta)$, we wish to reuse it when evaluating the gradient under $q_{\lambda'}(\theta)$. This is done by retaining the set of samples θ_m drawn from $q_\lambda(\theta)$ and using them to evaluate the expectation under the approximation $q_{\lambda'}(\theta)$. We proceed by first estimating for $q_{\lambda'}(\theta)$ the values ϵ'_m that would have generated θ_m under the new transformation $\theta = f_{\lambda'}(\epsilon')$ using the standardization function $\epsilon'_m = s_{\lambda'}(\theta_m)$. We then re-estimate the transformation gradients at λ' for ϵ'. Given these quantities we can use importance sampling to derive the importance-sampled gradient estimate as

$$\begin{aligned}
\nabla_\lambda \widetilde{\mathcal{L}}(\lambda) &= \int \left[\nabla_\theta \log p(\mathcal{D}, \theta) \nabla_\lambda f'_\lambda(\epsilon') + \nabla'_\lambda \log \left|\det J_f(\epsilon', \lambda')\right|\right] q_0(\epsilon') \; d\epsilon \\
&= \int \frac{q_0(\epsilon)}{q_0(\epsilon)} \left[\nabla_\theta \log p(\mathcal{D}, \theta) \nabla_\lambda f_\lambda(\epsilon) + \nabla_\lambda \log \left|\det J_f(\epsilon, \lambda)\right|\right] q_0(\epsilon') \; d\epsilon \\
&\approx \frac{1}{M} \sum_{m=1}^{M} w_m \left[\underbrace{\nabla_\theta \log p(\mathcal{D}, \theta_m)}_{\text{model gradient}} \underbrace{\nabla_\lambda f'_\lambda(\epsilon'_m) + \nabla_\lambda \log \left|\det J_f(\epsilon'_m, \lambda')\right|}_{\text{transformation gradients}} \right],
\end{aligned} \tag{3.2}$$

where $w_m = q_0(\epsilon')/q_0(\epsilon)$ are the importance weights, θ_m are the samples originally drawn from $q_\lambda(\theta)$ and ϵ'_m are the samples obtained from $s_{\lambda'}(\theta_m)$. In practice, the estimator here does not involve drawing ϵ' from $q_0(\epsilon')$, but rather reasons back from previously drawn θ_m from $q_0(\epsilon)$ to infer the values of ϵ that would have resulted in θ_m under the new approximation $q_{\lambda'}(\theta)$. We also see that since the model gradient at θ_m was previously computed and stored for reuse, and the transformation gradients do not contribute significantly to the computational cost, the estimator is efficient. A schematic of this is show in Figure 3.1. A brief algorithm that demonstrates the application of importance-sampled estimates on the reparameterization gradient is outlined in Algorithm 3.

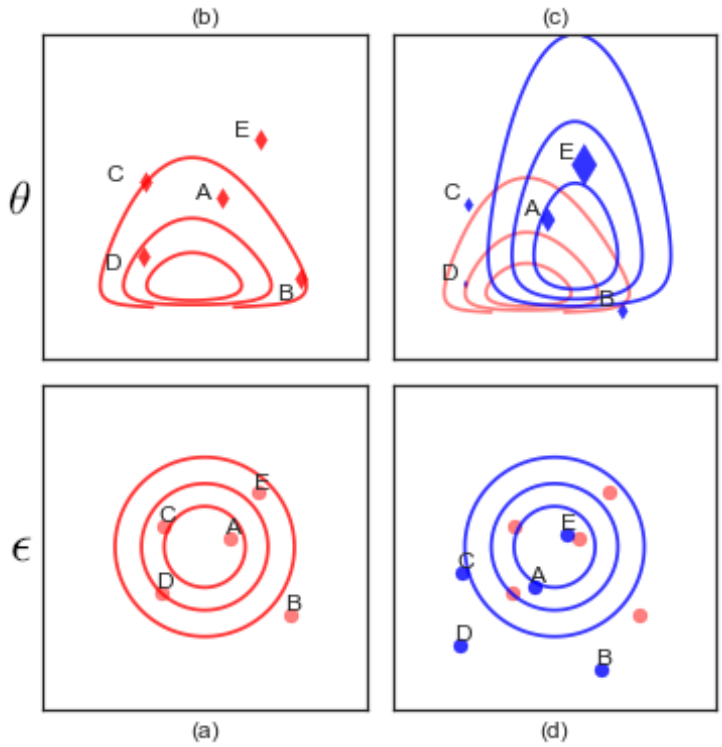

Figure 3.1: Monte Carlo estimates of the lower bound $\mathcal{L}(\lambda)$ are obtained by drawing M samples θ_m from the variational approximation, $q_\lambda(\theta)$ (top left panel). Reparameterization methods draw samples ϵ_m from the parameter-free distribution $q_0(\epsilon)$ (bottom left panel) and convert them to the parameter space as samples $\theta_m \sim q_\lambda(\theta)$ via the invertible transformation $\theta_m = f_\lambda(\epsilon_m)$. Under the updated variational approximation $q_{\lambda'}(\theta)$ (top right panel), importance-sampled estimates of the lower bound are obtained by retaining the samples θ_m and estimating the values $\epsilon' = s_{\lambda'}(\theta)$ that would have been required to produce θ_m under the updated approximation (bottom right panel). The likelihoods $\log p(\mathcal{D}, \theta)$ evaluated under the previous approximation can be reweighted with the importance weights $w_m = q_0(\epsilon')/q_0(\epsilon)$ to obtain importance-sampled estimates of the lower bound under the updated approximation. Note how the samples of θ_m, such as E, which are more probable under the updated approximation are shown bigger whereas those less likely to have been drawn from the updated approximation, such as D, are shown smaller. The same train of thought can be applied to using importance sampling to evaluate gradients of the lower bound $\nabla_\lambda \mathcal{L}(\lambda)$. Figure reproduced from Paper I (Sakaya and Klami, 2017).

Illustration: Normal Approximation

We build upon the illustration of normal approximation with reparameterization gradients in Subsection 2.3.2 to further clarify the derivations for importance-sampled gradients. Normal approximations reparameterize the model parameters using the location-scale transformation

$$\theta = f_\lambda(\epsilon) = L\epsilon + \mu,$$

where L is the Cholesky decomposition of the covariance and μ is the mean both of which constitute the variational parameters $\lambda = \{\mu, L\}$. We draw ϵ from the parameter-free distribution $q_0(\epsilon)$ which, in this case, is the standard normal distribution $\mathcal{N}(0, I)$. The standardization function for the transformation $f_\lambda(\epsilon)$ is

$$\epsilon = s_\lambda(\theta) = L^{-1}(\theta - \mu).$$

Given a set of model parameters θ_m drawn from $\mathcal{N}(\theta; \mu, L)$, importance-sampling helps us compute the gradient at the updated variational approximation $\mathcal{N}(\theta; \mu', L')$ where $|\mu' - \mu| \leq \delta$ and $|L' - L| \leq \delta$. Given the draws θ_m, the standardization function helps us infer that $\epsilon'_m = L'^{-1}(\theta_m - \mu')$ and the importance weights can then be readily evaluated as

$$w_m = \frac{\mathcal{N}(\epsilon; 0, I)}{\mathcal{N}(\epsilon'; 0, I)}.$$

In order to compute the importance-sampled gradient, we reuse the gradients $\nabla_\theta p(\mathcal{D}, \theta)$ while recomputing $\nabla_\lambda f_{\lambda'}(\epsilon')$. We begin with Equation 2.13 and 2.14 to obtain

$$\begin{aligned}
\mathbb{E}\left[\nabla_\mu \log p(\mathcal{D}, L'\epsilon + \mu')\right] &= \mathbb{E}\left[\frac{q_0(\epsilon')}{q_0(\epsilon)} \nabla_\theta \log p(\mathcal{D}, \theta)\right] \\
&\approx \frac{1}{M} \sum_{m=1}^{M} \left[w_m \underbrace{\nabla_\theta \log p(\mathcal{D}, \theta_m)}_{\text{reused}} \right] \\
\mathbb{E}\left[\nabla_L \log p(\mathcal{D}, L'\epsilon + \mu')\right] &= \mathbb{E}\left[\frac{q_0(\epsilon')}{q_0(\epsilon)} \nabla_\theta \log p(\mathcal{D}, \theta) \epsilon'^T\right] \\
&\approx \frac{1}{M} \sum_{m=1}^{M} \left[w_m \underbrace{\nabla_\theta \log p(\mathcal{D}, \theta_m)}_{\text{reused}} \epsilon'^T_m \right].
\end{aligned}$$

Note that $\nabla_L \mathcal{L}(\mu', L')$ also includes the gradient of the Jacobian $\nabla_L |\log L'|$, the exact form of which depends on the assumptions made on L'. In our

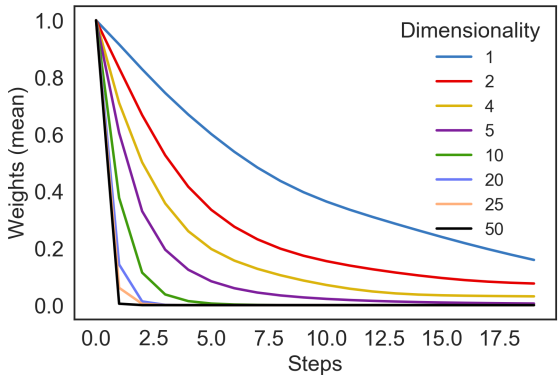

Figure 3.2: Illustration of the importance weight decay rate as a function of the dimensionality of θ during optimization. The approximation $q(\theta|\lambda)$ is over a 100-dimensional space. When fully factorized into 100 single-dimensional factors $q(\theta|\lambda) = \prod_{i=1}^{100} q(\theta_i|\lambda_i)$, the weight decay is gradual and the importance-sampled estimates are reliable for as many as 20 steps On the other hand, when factorized into 2 factors each of 50 dimensions $q(\theta|\lambda) = \prod_{i=1}^{2} q(\theta_i|\lambda_i)$, the importance weights become zero almost instantaneously rendering the importance-sampled estimate unusable.

case, since L' is a lower-triangular matrix, the gradient of the Jacobian is the sum of the inverse of the diagonal elements in L'. We also note that the mean-field assumption with variational approximations that are univariate normal over each dimension of θ, implies that the importance-sampling is done independently over each dimension. In such cases, if the variational approximation is factorized into single-dimensional factors, the weight decay in the importance sampled estimates tends to be more gradual than if the factors had higher dimensions as demonstrated in Figure 3.2.

We also note that the importance-sampling estimate, in addition to reusing the gradients $\nabla_\theta \log p(\mathcal{D}, \theta)$ adjusted using the importance weights, the transformations converting ϵ' to the space of the variational parameters λ could also involve non-linear functions, such as the Softplus transformation. In such cases, the importance-sampled gradients follow a non-linear trajectory that cannot be replicated by simply adjusting the step-size of the gradient descent optimizer. An example of the non-linear trajectory followed by the importance-sampled estimates is shown in Figure 3.3.

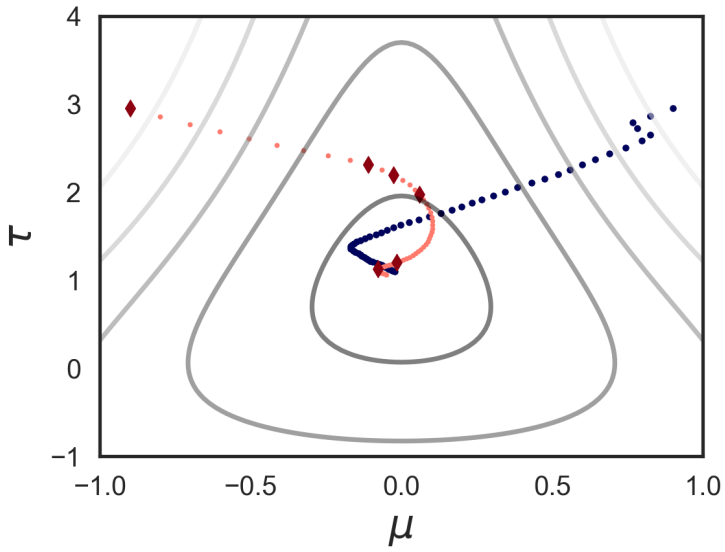

Figure 3.3: We contrast convergence of the importance-sampled estimates with regular gradient descent. While regular gradient descent (in blue) needs to compute the gradient from scratch for every update including the expensive model gradient $\nabla_\theta \log p(\mathcal{D}, \theta)$, the importance sampled gradient computes the model gradient only once every ten steps (shown as red diamonds) resulting in quicker convergence. In the importance-sampled steps (in pink), we evaluate only the transformation gradients while reusing the most recently evaluated model gradients. This allows convergence of the importance-sampled version to be approximately 9 times as fast as the conventional version since the transformation gradients are evaluated at a fraction of the cost required for model gradients. Note that although we reuse the samples from the previous iterations, the variational parameters μ and τ still follow a non-linear trajectory towards convergence showing empirically that the importance-sampling gradients can take the loss landscape into account. If the trajectory were linear, this behavior could have been replicated by careful adaptation of learning rates. Figure reproduced from Paper I (Sakaya and Klami, 2017).

Score Function Gradients

For the sake of completeness, we also show how to incorporate importance sampling into score function estimators. The computational speed-up associated with using importance-sampling in conjunction with score function gradients is usually meager unless the log probability of the model $\log p(\mathcal{D}, \theta)$ dominates the variational approximation. This can be clearly derived starting with the formulation of a score function estimate of the gradient under the approximation $q_\lambda(\theta)$

$$\nabla_\lambda \mathcal{L}(\lambda) = \mathbb{E}_q \left[g(\theta) \nabla_\lambda \log q_\lambda(\theta) \right].$$

Given this estimator, the importance-sampled estimate of the gradient under the new approximation $q_{\lambda'}(\theta)$ can be derived as follows:

$$\begin{aligned}
\nabla_\lambda \widetilde{\mathcal{L}}(\lambda) &= \int \left[g(\theta) \nabla_\lambda \log q_{\lambda'}(\theta) \right] q_{\lambda'}(\theta) \; d\theta \\
&= \int \frac{q_\lambda(\theta)}{q_\lambda(\theta)} \left[g(\theta) \nabla_\lambda \log q_{\lambda'}(\theta) \right] q_{\lambda'}(\theta) \; d\theta \\
&\approx \frac{1}{M} \sum_{m=1}^{M} \underbrace{\frac{q_{\lambda'}(\theta_m)}{q_\lambda(\theta_m)}}_{w_m} \left[g(\theta_m) \underbrace{\nabla_\lambda \log q_{\lambda'}(\theta_m)}_{\text{expensive}} \right],
\end{aligned}$$

where $\theta_m \sim q_\lambda(\theta)$ and $w_m = q_{\lambda'}(\theta_m)/q_\lambda(\theta_m)$. Even though the log probability $\log p(\mathcal{D}, \theta_m)$ and $g(\theta_m)$ are available from the previous iteration, the gradient of the new approximation $\nabla_\lambda \log q_{\lambda'}(\theta_m)$ usually dominates the computation time. However, since the gradient of the approximation has to be recomputed from scratch at λ' it is difficult to tease out computational savings unless $\log p(\mathcal{D}, \theta)$ and by implication $g(\theta)$ dominates the computation. Next, we discuss how the importance-sampled gradients introduced integrate into stochastic optimizers such as SGD and SAG and incorporate a novel mechanism that ensures that such gradient estimates continue to remain valid.

3.1.2 Stochastic Gradients and Averaging

Gradient descent methods take advantage of automatic differentiation tools to minimize the finite-sum objective of the form

$$\mathcal{L}(\lambda) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathcal{L}^i(\lambda)$$

where $\mathcal{L}^i(\lambda)$ is the lower bound per data point. Apart from the expensive gradient calculations involved in evaluating $\nabla_\lambda \mathcal{L}^i(\lambda)$ discussed in Section 3.1, the key limiting aspect of gradient descent tools is the size of the data set $|\mathcal{D}|$. When the objective function is strongly convex and differentiable, one can design stochastic methods that leverage the sum structure of the objective that are independent of the size of the dataset $|\mathcal{D}|$ and exhibit linear time convergence (Schmidt et al., 2017).

The full gradient method which dates back to Cauchy (Cauchy, 1847) evaluates the gradients by averaging over the whole dataset and defines the gradient updates as

$$\lambda^{t+1} = \lambda^t + \underbrace{\frac{\alpha^t}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \nabla \mathcal{L}^i(\lambda)}_{\text{full batch gradient}}$$

where α^t is an appropriate learning schedule such as a constant step-size. Although expensive to calculate, under strong concavity conditions, full gradients enjoy a linear convergence rate (Nesterov, 2014). The appeal of stochastic gradient methods (Robbins and Monro, 1951; Bottou and LeCun, 2004) lies in the fact that the iteration costs are independent of the size of the data set $|\mathcal{D}|$. The full batch gradients are replaced with a randomly drawn minibatch $\mathcal{D}_b \in [1 \in B]$. The gradient estimate using the random minibatch is unbiased and the parameters λ are updated as

$$\lambda^{t+1} = \lambda^t + \underbrace{\frac{\alpha^t}{|\mathcal{D}_b|} \sum_{b=1}^{|\mathcal{D}_b|} \nabla \mathcal{L}^b(\lambda)}_{\text{stochastic gradient}}$$

where $\nabla \mathcal{L}^b(\lambda)$ refers to the gradient of the minibatch. As long as the learning schedule follows the Robbins-Monro condition (Robbins and Monro, 1951) the algorithm converges to a local optimum. The convergence rate of stochastic gradient descent methods was originally thought to be sublinear until several variance reduction methods proved that under certain conditions (such as strongly convex functions), linear convergence could be achieved (Schatz et al., 2014). The basic principle of such variance reduction methods involves the idea of gradient averaging. For instance, the simplest such technique is the gradient average method that averages over

all historical gradients,

$$\lambda^{t+1} = \lambda^t + \underbrace{\frac{\alpha}{t}\sum_{b=1}^{t}\nabla\mathcal{L}^b(\lambda)}_{\text{gradient averaging}} .$$

We can take advantage of the shorter gradient window to drop the older and staler gradients to obtain gradient estimates with reduced bias. However, the convergence in these cases still tends to remain sublinear. More recent methods such as the stochastic average gradient (Schmidt et al., 2017) and stochastic variance reduced gradient (Johnson and Zhang, 2013) utilize gradient averages and show that stochastic methods that utilize gradient information from the full batch, even if the gradients are not up-to-date, result in linear convergence. These methods tend to have (greatly) reduced variance at the cost of increased bias. In this section, we adapt SAG for use in variational inference and in particular for applying the importance-sampling estimator to reduce bias in estimating the full-batch gradients.

3.1.3 Importance Sampled Stochastic Gradient Descent

In Section 2.3, we described stochastic gradient approaches for maximization of the variational objective in Equation 2.6. We now show how we can incorporate importance-sampled estimators to reuse gradients specifically in the context of stochastic gradient descent. SGD evaluates gradients based on a minibatch drawn from the data and takes a step in the direction of the gradient, using adaptive learning rates for better convergence, such as Adam (Kingma and Ba, 2015) or Adagrad (Duchi et al., 2010). Importance sampled SGD (ISGD) behaves in much the same way as SGD, except that it reuses the gradient evaluated at the minibatch to take several steps in the direction of the importance-sampled gradient estimates evaluated using Equation 3.2 until a criterion to resample a minibatch to calculate new gradients is met. The criterion to determine the number of importance-sampled step involves two competing aspects. Firstly, the original approximation and the updated approximation should have common support and as long as such overlap in support exists, importance-sampled estimates can be relied on to point in the direction of steepest descent. Secondly, since the gradient is based on a minibatch and is therefore noisy, taking too many steps due to an overlap between the original and updated approximation is also ill-advised. In Algorithm 4, we propose a criterion that helps us strike the right trade-off between these conflicting interests.

Algorithm 4: Importance Sampled SGD

input : data $\mathcal{D}$, model $p(\mathcal{D}, \theta)$, the variational approximation $q_\lambda(\theta)$, threshold p
output: variational parameters λ^*

while $\mathcal{L}(\lambda)$ *has not converged* **do**
  $\rho \sim \text{Uniform}(0, 1)$
  if $\rho < p$ **then**
    Retrieve recently cached θ_m, ϵ_m and $\nabla_\theta \log p(\mathcal{D}, \theta_m)$
    Update $\nabla_\lambda \mathcal{L}(\lambda)$ using Algorithm 3
  else
    Draw mini-batch $\mathcal{D}_b$ from $\mathcal{D}$
    $\epsilon_m \sim q_0(\epsilon)$
    $\theta_m \leftarrow f_\lambda(\epsilon_m)$
    Evaluate $\nabla_\lambda \mathcal{L}(\lambda)$ using Equation 2.12
    Cache θ_m, ϵ_m and $\nabla_\theta \log p(\mathcal{D}, z_m)$
  end
  Update $\lambda \leftarrow \lambda + \alpha \nabla_\lambda \mathcal{L}(\lambda)$
end

Algorithm 4 outlines the principle behind ISGD. Initially, an estimate of the gradient is obtained by evaluating it from scratch either using reparameterization gradients or score function estimators. We then draw a sample ρ from the uniform distribution between 0 and 1 and set the importance-sampling threshold to p that determines whether we sample a new minibatch or we use an importance-sampled estimate of the last evaluated gradient. For all values of $\rho > p$, we resample a new minibatch and evaluate the gradient again. When $\rho < p$ we reuse the previously evaluated gradient to obtain the importance-sampled gradient estimates. Except for the expensive evaluation at the beginning of a minibatch, the evaluation of the subsequent importance-sampled gradients can be performed significantly faster. After either the variational approximation has shifted significantly causing the importance weights to become 0 or if the threshold $\rho > p$ has been reached, we move on to a fresh minibatch and repeat the process until convergence. After having passed through the entire dataset $\mathcal{D}$, we have evaluated $\log p(\mathcal{D}, \theta)$ at every point in $\mathcal{D}$, but have taken significantly more gradient steps than SGD. An illustration that compares SGD to ISGD to approximate a univariate normal distribution is shown in Figure 3.3.

3.1.4 Importance Sampled Stochastic Average Gradient

We described the stochastic average gradient (SAG) algorithm as an optimization technique that preserves the low iteration cost associated with stochastic gradients while retaining the linear convergence rate of full gradients without resorting to expensive gradient evaluations. Importance-sampled estimators can be put to excellent use in such settings in order to remove stale gradients as well as to cheaply modify and reuse gradients from the previous iterations, thereby yielding less biased gradient estimates than can be acheived by SAG. We now define the SAG algorithm in more concrete terms to use as a starting point to construct an algorithm for the importance-sampled stochastic average gradient (ISAG).

The estimate of the full batch gradient is obtained by summing up estimates of each individual minibatch gradient

$$\nabla_\lambda \mathcal{L}(\lambda) = \frac{1}{B} \sum_{b=1}^{B} \nabla_\lambda \mathcal{L}^b(\lambda),$$

where the gradients $\nabla_\lambda \mathcal{L}^b(\lambda)$ are stored in memory for each epoch. During each iteration t in an epoch e, a random minibatch $\mathcal{D}_{b_k}$, where $b_k \in [1 \dots B]$, is picked and the gradients previously stored for minibatch $\mathcal{D}_{b_k}$ are replaced with fresh gradients evaluated at the current approximation characterized by $\lambda^{(t)}$. This implies that the full gradient is approximated by summing gradients of the minibatches evaluated at different time points with different parameter values, which leads to reduced variance at the cost of biased estimates. More specifically, if $\nabla^b_\lambda \mathcal{L}(\lambda)^{(t)}$ is the gradient for the minibatch $\mathcal{D}_b$ with respect to the variational parameters λ at the t^{th} iteration then, the update for the variational parameters λ at time step t is as follows

$$\lambda^{(t+1)} = \lambda^{(t)} + \frac{\alpha^{(t)}}{|\mathcal{D}_b|} \underbrace{\sum_{b=1}^{|\mathcal{D}_b|} \delta_b^e}_{\nabla_\lambda \mathcal{L}(\lambda)}.$$

At each iteration t, we pick a random minibatch b_k to update its gradient while leaving the gradients of the other minibatches untouched,

$$\delta_b^{(e)} = \begin{cases} \nabla^b_\lambda \mathcal{L}(\lambda)^{(t)} & \text{if } b = b_k, \\ \delta_b^{(e-1)} & \text{if } b \text{ not picked in epoch } e \text{ yet}, \\ \delta_b^{(e)} & \text{otherwise.} \end{cases}$$

Under strong convexity, similar to batch gradient descent, the gradient accounts for the whole dataset $\mathcal{D}$ and exhibits a linear convergence rate

(Schatz et al., 2014). Although SAG converges to the optimum, the convergence is impeded by stale gradients that accumulate into and bias the estimate of the full gradient.

We rectify this bias by using importance-sampling to obtain less biased estimates of the stochastic average gradient with importance sampling. There are three kinds of minibatches in SAG: one, the randomly chosen minibatch $\mathcal{D}_{b_k}$ for which the gradients have been freshly evaluated; two, minibatch gradients that have already been evaluated at the current epoch, but have since become stale owing to changes in the variational approximation; and three, gradients that have not yet been evaluated at the current epoch. We use importance sampling to re-weight and update the latter two gradients so that they are more accurate under the current approximation characterized by $\lambda^{(t)}$. That is to say, we choose $\widetilde{\delta}_b^{(e)}$ such that,

$$\widetilde{\delta}_b^{(e)} = \begin{cases} \nabla_\lambda \mathcal{L}^b(\lambda)^{(t)} & \text{if } b = b_k, \\ \nabla_\lambda \widetilde{\mathcal{L}}^b(\lambda)^{(t)} & \text{evaluated with Algorithm 3, otherwise.} \end{cases}$$

All gradients except the current one b_k are updated using importance sampling $\nabla_\lambda \widetilde{\mathcal{L}}^b(\lambda)^{(t)}$. This idea is outlined in Algorithm 5.

Because the gradients have been importance-wieghted and transformed, the full-batch gradient is no longer heavily biased by stale gradients yielding better convergence properties in comparison to SAG – even if this comes at the cost of the extra computational time involved in re-estimating δ_b^e under I-SAG. This reduced bias is a direct consequence of the importance-sampling property that requires an overlap in densities to yield non-zero weights. When the current approximation has a completely different support from the approximation under which the stale gradients were calculated, the importance weights effectively become zero resulting in the discarding of the stale gradients.

We finally comment on the time and memory complexity of the algorithm. The computational overhead associated with obtaining importance-sampled estimates of past gradients is not insignificant, since it is a linear function of the number of minibatches in the dataset with time complexity $\mathcal{O}(B)$. However, the computational burden for ISAG is limited to the complexity of the transformation $\theta = f_\lambda(\epsilon)$, which is a linear function in most cases, allowing it to outperform naive implementations of SAG. When faced with a large number of minibatches, an adequate solution is to consider an alternative to SAG, such as the stochastic running average (SRA), a gradient averaging algorithm that limits the gradient window size to the past k gradients, for an arbitrarily chosen k. Imposing such gradient windows

Algorithm 5: Importance sampled SAG

input : data $\mathcal{D}$, model $p(\mathcal{D}, \theta)$, the variational approximation $q_\lambda(\theta)$, batches B
output: variational parameters λ^*

while $\mathcal{L}(\lambda)$ *has not converged* **do**
  foreach *mini-batch* $\mathcal{D}_b$ *in* $\mathcal{D}$ **do**
    $\epsilon_m^b \sim \phi(\epsilon)$
    $\theta_m^b \leftarrow f(\epsilon_m^b, \lambda)$
    Calculate $\nabla_\lambda \mathcal{L}^b(\lambda)$ using Equation 2.12
    Store θ_m^b, ϵ_m^b and $\nabla_\theta \log p(\mathcal{D}_b, \theta_m)$
    foreach *mini-batch* $\mathcal{D}_c$ *in* $\mathcal{D} \backslash \{\mathcal{D}_b\}$ **do**
      Retrieve θ_m^c, ϵ_m^c and $\nabla_\theta \log p(\mathcal{D}_c, \theta_m)$
      Update $\nabla_\lambda \mathcal{L}^c(\lambda)$ using Algorithm 3
    end
    $\nabla_\lambda \mathcal{L}(\lambda) \leftarrow \sum_{b=1}^{B} \nabla_\lambda \mathcal{L}^b(\lambda)$
    Update $\lambda \leftarrow \lambda + \rho \nabla_\lambda \mathcal{L}(\lambda)$
  end
end
Return optimal variational parameters λ^*

limits the computational overhead of the importance-sampled estimates to a factor of k. Finally, the memory overhead for ISAG is the same as that of SAG. While SAG requires maintaining a table of the most recent update of the gradient to evaluate δ_b^e, ISAG requires that the model gradients be stored for each minibatch to evaluate $\widetilde{\delta}_b^e$.

3.1.5 Summary

Monte Carlo-based estimators of the gradients of the variational objective have opened up approximate inference techniques to a broad variety of models. These methods, broadly classified into reparameterization gradients and score function estimators, rely on using stochastic Monte Carlo methods to evaluate expectations of the form $\mathbb{E}_q[g(\theta)]$. Another element of stochasticity is introduced by utilizing stochastic gradient descent as the algorithm of choice in optimization. In Section 3.1, we introduce a third element of stochasticity that uses importance sampling to speed-up gradient calculations. The driving force behind importance-sampling estimators involves the computational efficiency achieved by storing computation-heavy parts of the gradient previously evaluated for a given approximation $q_\lambda(\theta)$

and reusing them to estimate the gradients under the updated approximation $q_{\lambda'}(\theta)$ by adjusting for the change in approximation using importance weights. The parts of the gradient that directly depend on the approximation are easy to evaluate yielding an efficient estimator for the gradient under the updated approximation. Lastly, we outline an importance-sampling estimator that can be easily incorporated into stochastic gradient descent algorithms and provide a novel method of estimating stochastic average gradients that automatically down-weights stale gradients in addition to re-weighting and updating previous gradients to more accurately reflect their estimate the current approximation. This concludes our discussion of importance-sampled estimators. In the next section, we expand on a differentiable transformation that allows straightforward incorporation of skewed distributions into standard probabilistic programs.

3.2 Modelling Skewed Data

Many data-generating processes generate data that are skewed in one direction with one of the tails much heavier than the other. Such skewed distributions occur naturally in many settings:

- Household income distribution has a high positive skew (McDonald et al., 2011), with the mass of distribution concentrated in the lower income regions and a right tail in the higher income regions.

- Stock markets returns exhibit a negative skew centered around its current value that should be taken into account when calculating risk (Jarjir, 2004). A high negative skew implies that even if the probability of the stock's value decreasing is much lower than the probability of it increasing, in the event it does go down, because of the heavy left tail, the stock value tends to drop much steeper.

- The distribution of the age of retirement (in, for instance, the UK) exhibits a negative skew as well, since most of the population retires between 65 and 68, while others retire as early as 40.

- Monotonic transformations of symmetric distributions tend to skew them towards the left or right. For instance, log transforming count data (Feng et al., 2014) (such as population data, word bigram counts) reduces its skew to make it appear normally distributed.

The Bayesian modeling community sometimes makes simplifying assumptions about such skewed, real-world data-generating processes because of

inference and/or modeling constraints and has a tendency to adapt symmetric distributions with well-established statistical properties such as the normal distribution or heavier tailed distributions such as the Cauchy or the Student-T distribution to model skewed data. To compensate for the choice of wrong distribution, appropriate transformations on the data so as to reduce its skew such that it can be captured using standard, symmetric distributions. However, this process is unwieldy, involving intimate knowledge of the data and is not easy to automate. Probabilistic programming implementations of skewed distributions such as the univariate skew-normal by `PyMC3` and `Stan`, while addressing the need for modeling skewed data, cannot be easily extended to other symmetric distributions. A more flexible family of skewed distribution to capture such skewed data was proposed by Goerg (2011). Called the Lambert $W \times F$ distribution, it combines an arbitrary base distribution $F_Z(z)$ from the location-scale family with a forward transformation $g(z) = ze^{\gamma z}$ that skews the base distribution. The corresponding inverse transformation $g^{-1}(x)$ uses the Lambert function from which the name Lambert $W \times F_Z$ distribution derives. Since the degree and direction of the skew is controlled by γ, with an arbitrary base distribution, the Lambert $W \times F$ distribution promises a general purpose tool to flexibly model skewed data that can be easily adapted for use in probabilistic programs.

3.2.1 Lambert W function

The Lambert function is defined as the inverse function of the forward transformation

$$x = g_0(z) = ze^z$$

and is generally used to solve functions where the unknown z appears inside and outside of the exponential function. The Lambert function is a multivalued function with more than one solution which cannot be expressed in terms of elementary functions. More specifically, it has two real branches of interest to us in the interval

$$x \in \left]-\frac{1}{e}, 0\right[$$

meeting at $x = 1/e$. This means that for any value of x in the interval, $W(x)$ yields two solutions. This is shown in the top left illustration in Figure 3.4, where the principal branch $W_0(x)$ is shown in black for $x \geq \frac{-1}{e}$ and the negative branch $W_{-1}(x)$ is shown in red for $x \in [-1/e, 0)$. As x approahces 0, the solution $W_{-1}(x) \to -\infty$. The principal branch $W_0(x)$ grows gradually towards infinity as $x \to \infty$.

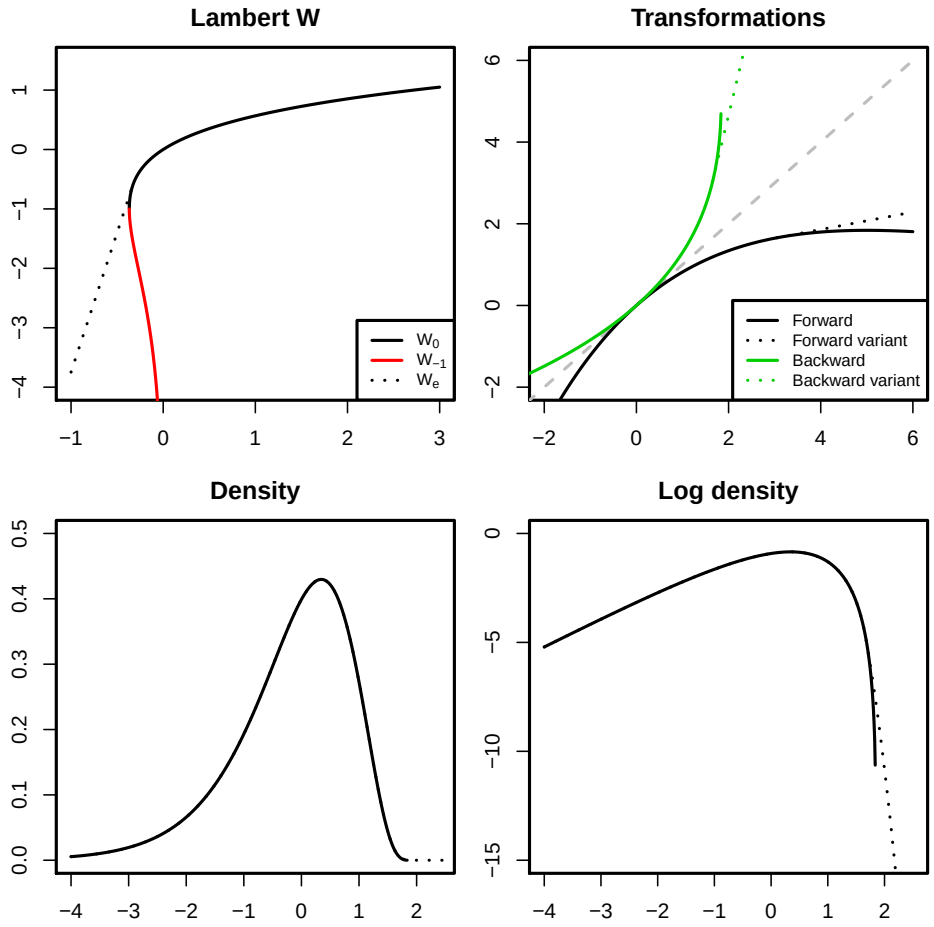

Figure 3.4: *The modified Lambert distribution.* The Lambert W distribution consists of the two branches in the interval $-\frac{1}{e} < x < 0$, the principal branch $W_0(x)$ and the negative $W_{-1}(x)$ which meet at $x = -\frac{1}{e}$. The modified Lambert W replaces the $W_{-1}(x)$ branch with a linear continuation of $W_0(x)$. This linearizes sections of the forward and reverse transformations with a large skew of $\gamma = 0.2$. With a standard normal distribution as the base distribution $F_Z(z)$, only 0.1% of the samples fall into the linearized region and the linearized density and log density functions are almost identical to the original density functions. Figure reproduced from Paper II (Klami et al., 2019).

Since the Lambert function cannot be represented with elementary functions, numerical approximations are needed to obtain solutions to the Lambert W. The simplest is to use Newton's approximation starting with an initial guess $w_0 = W(x)$ and successively approximating $w_j = W(x)$ as

$$w_{j+1} = w_j - \frac{w_j e^{w_j} - z}{e^{w_j} + w_j e^{w_j}}.$$

A better approximation is obtained with Halley's method (Corless et al., 1993),

$$w_{j+1} = w_j - \frac{w_j e^{w_j} - z}{e^{w_j}\left(w_j + 1\right) - \frac{\left(w_j + 2\right)\left(w_j e^{w_j} - z\right)}{2w_j + 2}}$$

Careful choice of the initial guess w_0 is needed to end up with real values in the principal branch. Highly optimized and tested implementations of numerical approximations to the Lambert W function make it a practical choice of skewing function.

3.2.2 Lambert $W \times F$ distribution

The Lambert $W \times F$ family is defined as a transformation of a family of location-scale base distributions $F_Z(z)$ with zero mean and unit variance. The latent inputs $z \sim F_Z(z)$ are transformed with a skew function $g(z)$ into,

$$x = g(z) = ze^{\gamma z}, \tag{3.3}$$

where $g(z) = g_0(\gamma z)/\gamma$ and γ controls the degree and direction skew. x is skewed towards the left when $\gamma > 0$ and to the right when $\gamma < 0$ as shown in Figure 3.5. At $\gamma = 0$, the transformation reduces to identity. Since the forward transformation is differentiable it can be conveniently plugged into any standard probabilistic programming framework that relies on gradient-based inference techniques. The skew γ, is a free parameter, and the probabilistic inference module is able to learn the skew over any given base distribution F_Z, allowing for flexible and interpretable modeling.

Using the change of variables technique, the probability density function of the skewed x can be expressed in terms of $p(z)$ as

$$p_X(x|\gamma) = p_Z(g^{-1}(x)) \left| \frac{dg^{-1}(x)}{dx} \right|$$

The absolute value of the Jacobian accounts for change of volume under the non-linear transformation $g(z)$. The inverse transformation $g^{-1}(x)$ is

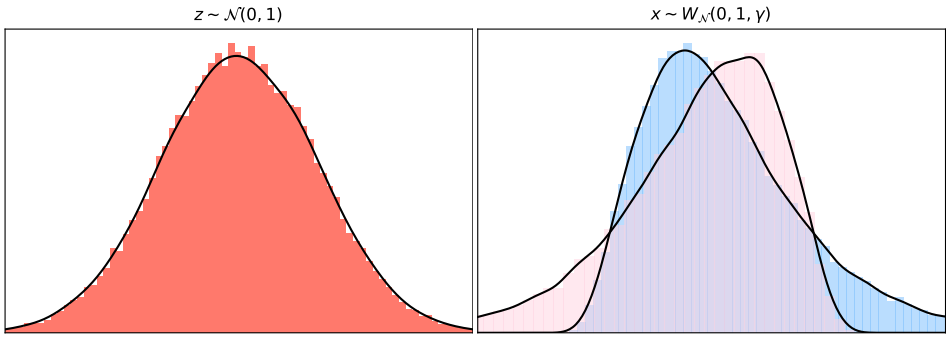

Figure 3.5: Generative story of the skewed normal distribution. The latent input variable z is first sampled from the standard normal distribution. The distribution shown on the left is then passed through the skewing transformation $g(z)$ to obtain the skewed Lambert $W \times \mathcal{N}$ distributions shown on the right. A skew of $\gamma = 0.2$ gives rise to a distribution with a right skew (in blue) and a skew of $\gamma = -0.2$ yields a left-skewed distribution (in pink). Note that large values of skew are fairly atypical and it is common for most values of γ to fall between -0.3 and 0.3.

defined as,

$$z = g^{-1}(x) = \frac{W_0(\gamma x)}{\gamma},$$

where $W_0(.)$ is the principal branch of the Lambert W function which gives its name to the Lambert $W \times F$ distribution.. The gradient of the Lambert W function for all of its branches $W(x)$ at $x \neq -\frac{1}{e}$ is given as,

$$\frac{dW(x)}{dx} = \frac{1}{x + e^{W(x)}}.$$

The density function $p_X(x|\gamma)$ with the change of variables consequently is

$$p_X(x|\gamma) = p_Z\left(\frac{W(\gamma x)}{\gamma}\right)\left|\frac{1}{\gamma x + e^{W(\gamma x)}}\right|.$$

It can be seen that a parametric family of skewed distributions can be created by applying the non-linear transformation to convert base random variable Z into a Lambert W random variable X with skew γ.

Illustration

We illustrate this for the case where the base distribution F is normal and produces a data set that has a skewed normal distribution. The skewed

Lambert $W \times F$ distribution which is zero mean and unit variance, can be further transformed with a location-scale transformation as

$$y = \sigma x + \mu \tag{3.4}$$

Thus, y is obtained by a series of two transformations, first the non-linear skewing transformation of z drawn from the base distribution $F_Z(z)$ in Equation 3.3 and then applying the location-scale transformation in Equation 3.4. Applying the change of variables formula twice, the density function of y is simply given as

$$p_Y(y|\mu, \sigma, \gamma) = \frac{p_Z\left(\frac{1}{\gamma}W\left(\gamma\frac{(y-\mu)}{\sigma}\right)\right)}{\gamma(y-\mu) + \sigma e^{W\left(\gamma\frac{(y-\mu)}{\sigma}\right)}}.$$

This density is denoted by $\mathcal{W}_F(\mu, \sigma, \gamma)$ where F refers to the base distribution under consideration, in this particular case, the standard normal distribution. A visual illustration of this transformation is shown in Figure 3.6 where the base normal distribution is skewed towards the left (in red) and the right (in blue) prior to being transformed with the location-scale transformation.

Maximum likelihood estimation of the skew and the moments of $\mathcal{W}_F$ can be performed via the iterated generalized method of moments outlined in Goerg (2011) which derives closed form estimates of the moments of $\mathcal{W}_F$ for the case of $F = \mathcal{N}(0, 1)$. In the next section, we introduce the modified Lambert $W \times F$ distribution that can be used together with gradient-based techniques for inferring the moments of $\mathcal{W}_F$ for an arbitrary F from the location-scale family.

3.2.3 Modified Lambert $W \times F$ distribution

As we have already seen, the reverse transformation of the Lambert $W \times F$ has only finite support. For negative skew $\gamma < 0$, the support is

$$\left(-\infty, -\frac{1}{\gamma e}\right)$$

and for positive skew $\gamma > 0$ the support is

$$\left(-\frac{1}{\gamma e}, \infty\right).$$

This results in undue constraints on the values that γ, μ and σ can take and places restrictions on the inference process since a naive application of

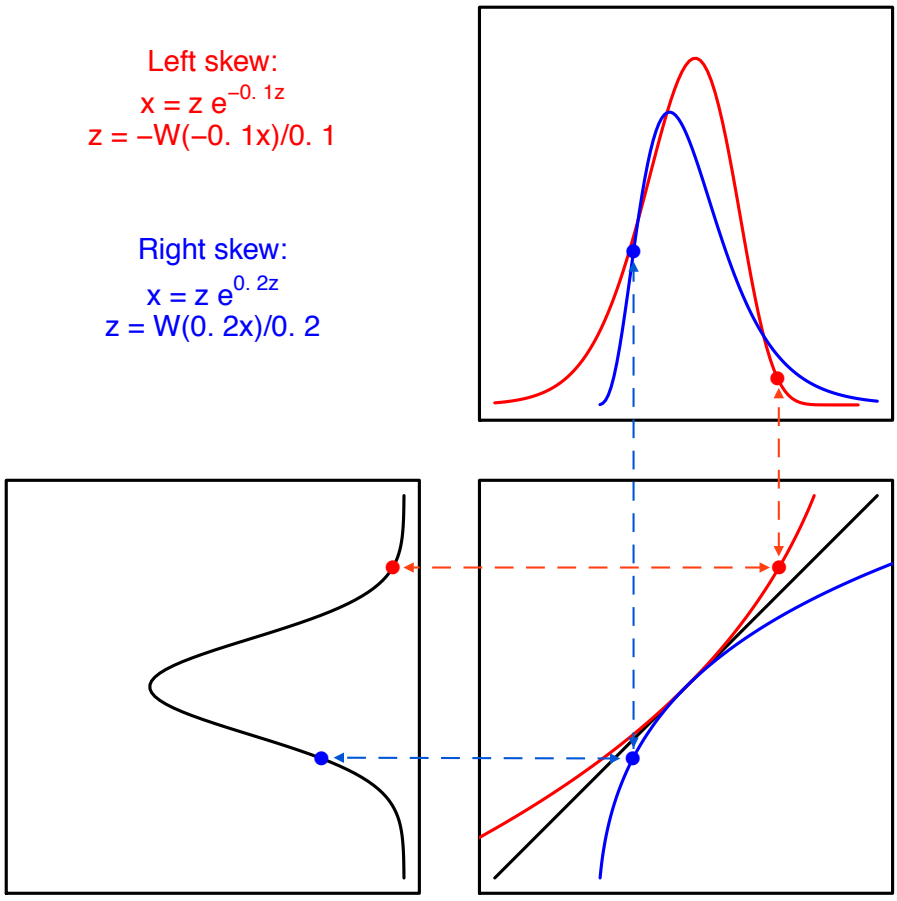

Figure 3.6: **Illustration of the Lambert $W \times F$ distribution.** Draws from a unit normal distribution $z \sim \mathcal{N}(0, 1)$ are transformed with the skewing function $g(z) = ze^{\gamma z}$ with skew controlled by γ. For $\gamma < 0$ (shown in red), the convex transformation results in a left skew whereas for $\gamma > 0$ (shown in blue) the concave transformation results in a right skew. The skewed density can be made more flexible by passing it through a location-scale transformation. This succession of two differentiable transformations, namely, the skewing and the location-scale transformation, allow the transformed distributions to be easily integrated into probabilistic programs that rely on gradient based techniques for inference. Figure reproduced from Paper II (Klami et al., 2019).

unconstrained optimization would result in values of μ, σ or γ that would fall outside the support of the Lambert $W \times F$ distribution.

This can be avoided by applying constrained optimization to ensure that the skew γ never becomes so large as to cause x to fall outside the support of $W(\gamma x)$ resulting in zero probability. However, this precludes the Lambert $W \times F$ distribution from having broad applicability in probabilistic programs that used constraint-free gradient-based optimization for inference.

An alternative is to modify the Lambert $W \times F$ distribution itself so that this limitation imposed by the Lambert W function is circumvented by extending support across the real line causing the skewed density to have support outside of the original density formulation. The modified linear extension of the Lambert W function, which we call the *modified Lambert W function*. $W_e(x)$ is defined as follows

$$W_e(x) = \begin{cases} W(x) & \text{if } x \geq d \\ W(d) + \frac{x-d}{x+e^{W(d)}} & \text{if } x < d \end{cases} \tag{3.5}$$

where a suitable choice of d can be defined between $-1/e < d < 0$. Practically we found a choice of $d = -0.9/e$ works well although any values close to $-1/e$ should yield similar behavior. The modified Lambert W, as illustrated in Figure 3.4, replaces the secondary negative branch $W_{-1}(x)$ and a section of the principal branch $W_0(x)$ with a linear extension that matches the principal branch and its derivative at d.

3.2.4 Lambert $W \times F$ for Probabilistic Programming

The modified Lambert $W \times F$ distribution is bijective, with forward and reverse transformations that are differentiable making it straightforward to plug into probabilistic programs with gradient-based inference. A transformed distribution can be obtained by combining a base distribution with a series of differentiable transformations. We use the forward transformation to skew the base distribution followed by a location-scale transformation as follows

$$\begin{aligned} x &= g(z), \\ y &= \sigma x + \mu. \end{aligned}$$

The transformed distribution has several uses within the context of probabilistic programs. The simplest and the most widely applicable use for the Lambert $W \times F$ is as the likelihood function for modeling any family of

skewed distribution. That is, we model the skewed data x as

$$\log p(x|\mu, \sigma, \gamma) = \log \mathcal{W}_F(x; \mu, \sigma, \gamma).$$

The second application is its use as a variational approximation of skewed posteriors. Because the Lambert $W \times F$ distribution is fully reparametrizable with a base distribution that is parameter free and transformations that are fully deterministic, it can serve as a variational approximation parameterized by μ, σ and the skew γ. For example, we could define a variational approximation to model skewed posteriors as

$$q_\lambda(\theta) = \mathcal{W}_F(\mu, \sigma, \gamma)$$

where the variational parameters are $\lambda = \{\mu, \sigma, \gamma\}$. Standard reparameterization estimates of the gradient can be obtained by defining parameter-free base distribution $q_0(\epsilon)$ as $F_Z(z)$ followed by two deterministic transformations,

$$\theta = f_{\mu,\sigma}(g_\gamma(\epsilon)).$$

Practical usage of these techniques in the context of probabilistic programming requires further attention. Although we resolved most optimization issues by introducing the modified Lambert W function, the computation of W still lacks a closed-form solution. Although efficient implementations of numerical solutions, such as Halley's method, exist in many standard scientific libraries, and ten iterations of Halley's method are often sufficient to give solutions with accuracy adequate for practical purposes, the computational cost of such iterative methods are still too expensive and prevent the skewing transformation from scaling up. We propose a simple but effective solution by taking advantage of the fact that both $W(x)$ and its gradient are scalar functions and need to be typically evaluated only over a narrow range of inputs. Armed with this knowledge, we precompute and cache solutions to $W_e(x)$ and its gradients in a look-up table with a pre-configured granularity that is application-specific. This lets us scale the Lambert W function to arbitrary precision enabling scalable, gradient-based learning.

For further clarity, we provide code snippets that we utilized to implement the modified Lambert $W \times F$ distribution in `autograd` (Maclaurin et al., 2015). Integration of the skewed transformations of base distribution can be done seamlessly in terms of the syntax of the probabilistic program itself without meddling with the inference engine. We first import the modules we require:

```
import autograd.numpy as np
from scipy.special import lambertw
from autograd.extend import primitive, defvjp
from autograd import grad
```

The `scipy.special` module contains an iterative solution of the Lambert function that uses Halley's method. We now define the `lambertw` function as a primitive (which informs the automatic differentiation engine that the function is a basic unit of operation) and its gradient for the Lambert function:

```
lambertw = primitive(lambertw)
# gradient of the primitive
defvjp(lambertw,
    # gradient w.r.t the first input x of lambertw
    lambda ans, x: lambda g:  g * 1./ (x + np.exp(ans)),
    # gradient w.r.t the second input k of lambertw
    None )
```

The log density of the modified Lambert $W \times F$ distribution that uses the transformation in Equation 3.5 as illustrated in Figure 3.4 is defined in terms of the Lambert function $W(x)$ with linearization:

```
def modified_lambertw_logpdf(y, loc, scale, skew):
    u = (y - loc)/scale
    if skew != 0:
        # the cutoff is d in Equation(3.5)
        # we also incorporate skew
        cutoff = -0.9 * 1/(np.e * skew)
        # define condition for piecewise extension
        cond = u >= cutoff if skew < 0 else u <= cutoff
        wc = lambertw(skew*u)
        wc[cond] = wtmp = lambertw(skew * cutoff)
        gc = 1./(np.exp(wc) + skew*u)
        gc[cond] = 1./(np.exp(wtmp) + skew*cutoff)
        # Equation(3.5):W(x), x >= d
        z = wc/skew
        # Equation(3.5): W(d) = W(d) + (x-d)/(x+exp(W(d))),
        # u < d
        z[cond] = wc[cond]/skew +\
           (u[cond] - cutoff) * gc[cond]
        return norm.logpdf(z) + np.log(gc) - np.log(scale)
```

```
    else:
        # return the base distribution
        return norm.logpdf(u) - np.log(scale)
```

We can then incorporate the precomputed Lambert function by taking advantage of function closures:

```
def precompute_lambertw(a, b, resolution):
    y = np.linspace(a, b, resolution * (b - a) + 1)
    y[y < -tol/np.e] = -tol/np.e
    z = np.array(lambertw(y).real).astype(np.float32)
    def lambert(x):
        sel = ((x - a) *     resolution).astype(np.int32)
        sel[sel < a] = 0
        return z[sel]
    return lambert
precomputed_lambertw = precompute_lambert(a, b, resolution)
```

The `precomuted_lambertw` can now be used in place of the Lambert function `lambertw`. We define the objective function to optimize as the negative log likelihood and can use its gradient with respect to the function parameters in any gradient-based inference routine:

```
def objective(parameters, y):
    loc, scale, skew = unpack(parameters)
    return -np.sum(
    modified_lambertw_logpdf(y, loc, scale, skew)
    )
gradient = grad(objective)
```

We finally comment on two practical implications involving implementation of inference for the skew parameter γ. First, one needs to account for the possibility that when $\gamma = 0$, the reverse transformation involves a division by zero. This can be handled as a special case, by defining transformed distribution to be same as the base distribution when $\gamma = 0$. Second, in most practical cases, the relatively small values of the skew γ tend to be the most useful. It is typically centered around zero and very rarely exceeds ± 0.3. In order to constrain the values of γ, we reparameterize it as

$$\gamma = \gamma_m \tanh(\hat{\gamma})$$

such that while $\hat{\gamma}$ takes any value in the real line, $|\gamma| \leq \gamma_m$. We set $\gamma_m = 0.3$ to capture our intuition about the values that the skew can take.

3.2.5 Summary

Since there exists a preponderance of skewed data generated by naturally occurring process, capturing such data with an appropriate skewed distribution is crucial for accurate modeling. The Lambert $W \times F$ is a flexible family of skewed distributions that skews a base distribution from the location-scale family through a differentiable skewing transformation $g(z)$. The Jacobian of the inverse skewing transformation required involves evaluating the Lambert W function which suffers from several drawbacks: Firstly, $W(x)$ does not have a closed-form solution; instead, it requires numerical approximations using iterative algorithms such as the Halley's method. Secondly, the Lambert W transformation consists of two branches, namely the primary branch $W_0(x)$ and the secondary branch $W_{-1}(x)$ in the interval $-1/e < x < 0$, which implies that the transformation is not one-to-one. Additionally, $W(x)$ has limited support which does not extend across the real line, ruling out unconstrained optimization. We proposed the modified Lambert $W \times F$ distribution to address these concerns. We first modify the Lambert function by replacing the secondary negative branch $W\_1(x)$ with a piece-wise linear extension of $W_0(x)$ allowing a one-to-one transformation. This incidentally also extends the support of the Lambert function to the real line by introducing a low-probability tail and, consequently, the parameters of the Lambert $W \times F$ distribution are no longer restricted by the limited support of $W(x)$. Since $W(x)$ and its gradient are required only over a narrow range of values, we pre-calculate and cache these values and circumvent the expensive numerical calculations to speed up the inference process. The modified Lambert W function is thus easily incorporated into constraint-free gradient-based optimization routines in probabilistic programs.

Chapter 4

Loss-Aware Approximations

Automatic inference engines in probabilistic programs favor a delineation between inference and decisions, viewing posterior inference as a standalone exercise agnostic to the decision tasks it is meant to solve and corresponding user-defined losses or utilities that it is intended to optimize. The objective of the Bayesian methodology, however, is often to use the posteriors for making (optimal) decisions under uncertainty quantified by said posteriors for a user-defined decision-theoretic loss or utility within the Bayesian decision framework developed in Subsection 2.4.1. More specifically, the optimal decisions under posterior uncertainty are defined as

$$h_p = \arg\min_{h \in \mathcal{H}} \underbrace{\int p(\theta|\mathcal{D})\ell(\theta, h)\ d\theta}_{\mathcal{R}_\ell^p(h)}.$$

where $\mathcal{R}_\ell^p(h)$ is the expected posterior risk defined in Equation 2.17. As outlined in Subsection 2.2.2, ongoing research efforts are rightly aimed at developing flexible, efficient and accurate approximations $q(\theta)$ to the posterior distribution $p(\theta|\mathcal{D})$ (Rezende and Mohamed, 2015; Guo et al., 2017; Locatello et al., 2018; Knoblauch et al., 2019; Hoffman and Blei, 2015). As such, most of these approximate posteriors could serve as a proxy for the true posterior, especially when the decision-theoretic loss is undefined, if the loss is trivial and symmetric or if the model is simple enough such that the true posterior is a member of the variational family. We could, for instance, substitute $q(\theta)$ to obtain,

$$h_q = \arg\min_{h \in \mathcal{H}} \underbrace{\int q(\theta)\ell(\theta, h)\ d\theta}_{\mathcal{R}_\ell^q(h)}.$$

In most machine learning applications, however, since the decision-theoretic losses are non-trivial, asymmetric or skewed, substituting the approximation for the true posterior would yield decisions h_q that are not guaranteed to be the same as the Bayes optimal decision h_p, potentially resulting in unintended and possibly disastrous consequences. Instead of the naive substitution of $q(\theta)$ for $p(\theta|\mathcal{D})$, a better approximation of the risk can be obtained by using the importance sampling estimates discussed in Section 2.5,

$$R_\ell^p(h) = \int \left[\frac{p^*(\theta|\mathcal{D})}{q(\theta)}\ell(\theta, h)\right] q(\theta)\ d\theta,$$

As long as the posterior density $p^*(\theta|\mathcal{D})$ can be evaluated up to a constant and the support of $q(\theta)$ is the same as that of the posterior, the importance-sampling method produces an unbiased estimate of the risk.

The primary reason why approximate inference methods often fail to yield the Bayes optimal decision is that they by design ignore the decision task and instead expend their limited computational budget on better approximating general features of the Bayesian posterior such as its mode, skew or other higher order moments at the cost of sub-optimal decisions h_q. This implies that the approximate posterior seldom captures the regions of the true posterior that are of importance in the decision-making task. This is easily seen in the example of the nuclear power plant described in (Lacoste-Julien et al., 2011) The posterior temperature $p(\theta|\mathcal{D})$ of the nuclear plant measured from readings $\mathcal{D}$ yields a multi-modal distribution as shown in black in Figure 4.1. When the plant runs into the danger of overheating, two discrete decisions h can be taken: the plant is left *'on'* or *'off'*. Shutting the plant down when $\theta < T_{crit}$ incurs a moderate loss $\ell(\theta < T_{crit}, \text{'off'}) = C$ for some constant C. Leaving the plant switched on when $\theta > T_{crit}$ incurs a massive loss $\ell(\theta > T_{crit}, h = \text{'on'}) \gg C$. Consider now how various approximate inference methods minimize the posterior risk and estimate the Bayes optimal decision illustrated in Figure 4.1. If $q(\theta)$ is the approximation, minimizing $\mathrm{KL}(q\|p)$ results in an approximation q_{VI} (dotted, blue curve) that is mode-seeking and concentrated around the major mode completely failing to capture the second mode. Minimizing $\mathrm{KL}(p\|q)$ via moment matching results in the approximation q_{EP} (dashed, pink curve) that matches all moments resulting in a more global approximation, albeit one that does not accurately cover the second mode. Both approximations lead to a suboptimal Bayes decision owing to their failure to capture the second mode appropriately. An optimal approximation such as the normal approximation in q_{opt} (dashed-dotted, pink curve), while still failing to model the bimodal posterior faithfully, nonetheless captures the regions of the posterior that are critical for the decision-making task

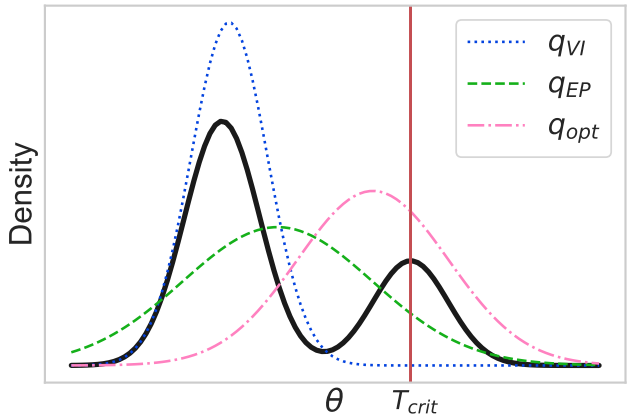

Figure 4.1: The true posterior $p(\theta|\mathcal{D})$, show in black, is bimodal and approximated by a normal approximation. Minimizing $\mathrm{KL}(q\|p)$ results in q_{VI}, which is mode-seeking and fits the major mode. Minimizing $\mathrm{KL}(p\|q)$ results in q_{EP} obtained via moment matching for a more global fit. Both q_{VI} and q_{EP} fail to capture the second mode, the region where the T_{crit} lies, accurately. An accurate fit q_{opt} under constraints imposed by normal approximation can be obtained by co-opting the decision-task into the inference process. The normal approximation with limited expressive power to capture the region of the posterior (here, the minor mode) that is critical to the decision task. The estimates of the Bayes optimal decision for q_{opt} are more in line with the decisions obtained from the posterior $p(\theta|\mathcal{D})$.

prompting accurate estimates of the Bayes optimal decisions. Note that a bimodal distribution would here capture the true posterior faithfully and yield optimal decisions. However, in many practical cases having a sufficiently flexible approximation that captures multi-modal posterior is difficult to guarantee.

An approximate posterior such as q_{opt} that minimizes the risk to produce a Bayes optimal decision in line with the true posterior $p(\theta|\mathcal{D})$ can be realized in multiple ways depending on the stage at which calibration of the approximate posterior takes place. We can, for instance, co-opt the decision-making task into the inference process so that the limited expressive power of the approximation and its computational budget are expended on the identifying and modeling regions of the posterior that are important to the task such as q_{opt}. An alternate approach leaves the inference process

untouched, instead taking a characterization of the suboptimal posterior distribution such as q_{VI} or q_{EP}, to learn a function that maps to the Bayes optimal decision minimizing the risk. In either case, Bayesian decision theory provides a means of directing our modeling efforts on criteria that are most relevant to the decision task. The rest of the chapter elaborates on each of these approaches, loss-calibrated variational inference and post-hoc corrections.

4.1 Loss Calibrated Variational Bayes

Loss-calibrated inference modifies the inference process itself to capture the regions of the posterior that are relevant to the decision making task under the constraints imposed by the posterior approximation. The crux of loss calibration lies in maximizing the expected utility computed over the posterior approximation, instead of just maximizing the approximation accuracy. The idea of loss-calibrating the inference procedure was first introduced by Lacoste-Julien et al. (2011) in which a modification of the variational lower bound that accounted for the decision-making task was proposed. It relied on an EM-style procedure to alternately optimize the decisions and the variational parameters. A similar algorithm for MCMC methods was proposed by Abbasnejad et al. (2015). Cobb et al. (2018) extended the work of Lacoste-Julien et al. (2011) in the context of variational dropout that used reparametrization techniques and Monte Carlo approximations to loss-calibrate discrete utilities. We discuss modifications to the loss calibration procedure by introducing an automated decision module in Section 4.1.

4.1.1 Loss-Calibrated Expected Lower Bound

In Section 2.4, we laid out the principles of Bayesian decision theory and explained how the maximum expected utility principle forms the basis of rational decision making. The Bayes optimal decision h_p is one that maximizes the posterior gain $\mathcal{G}_u^p(h)$ defined in Equation 2.16 or equivalently, minimizes the posterior risk $\mathcal{R}_\ell^p(h)$ defined in Equation 2.17. However, since variational approximations $q_\lambda(\theta)$ are loss-agnostic, the standard variational lower bound is designed to capture only the general properties of the posterior, and the resulting approximation does not guarantee that the decisions h_q minimize $\mathcal{R}_\ell^p(h)$ and therefore is not Bayes optimal. To rectify this, in Subsection 2.4.2 we develop an alternative formulation of risk minimization, which we call approximate decision theory, that involves

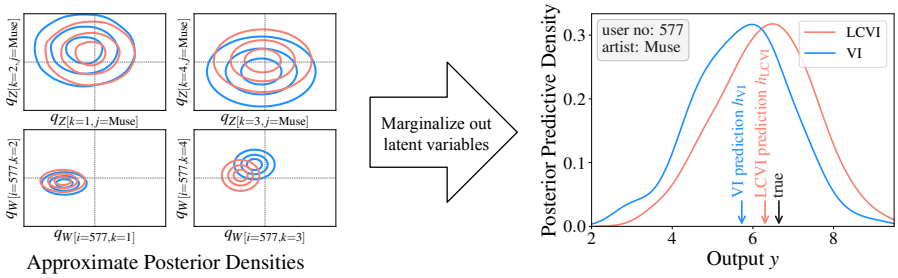

Figure 4.2: The loss calibration procedure transforms the standard variational approximations (blue) into loss-calibrated posterior approximations (red) in such a manner that the Bayes optimal decisions for the posterior predictive are optimal with respect to the user-defined loss (in this example, the squared error loss). The loss-calibrated posterior, however, still characterizes the standard approximation accurately. The shift in the loss-calibrated posterior predictive (red) is obtained by marginalizing out the latent variable with Monte Carlo samples from the loss-calibrated posterior. Figure reproduced from Paper III (Kuśmierczyk et al., 2019b).

finding a member of the variational family parameterized by λ to yield decisions h_q with maximal risk reduction within the constraints imposed by the variational family $\mathcal{Q}$,

$$q_{\text{opt}} = \arg\min_{q \in \mathcal{Q}} \mathcal{R}_\ell^p(h_q).$$

where $\mathcal{R}_\ell^p(h_q)$ is the q-risk defined in Equation 2.18. When the posterior $p(\theta|\mathcal{D})$ is a member of the variational family $\mathcal{Q}$, h_q indeed becomes the Bayes optimal decision. With this in mind, we develop a modified variational lower bound that yields an approximation $q_\lambda(\theta)$ that is loss-aware, and unlike classical variational inference, spends its computational budget on choosing a member of the variational family that minimizes the posterior risk in addition to approximating the posterior. Figure 4.2 demonstrates how the modified lower bound shifts a suboptimal approximation to improve the decisions.

The framework for configuring or calibrating the variational posterior for the decision making task with a user-specified decision-theoretic loss or utility was laid out by Lacoste-Julien et al. (2011). Since decoupling inference and decision-making in the context of approximate posteriors might lead to suboptimal decisions, they rightly pointed out that including the

utility to calibrate approximate posterior yields decisions which, while not necessarily optimal with respect to the true posterior, would nevertheless maximize the expected posterior gain under the constraints imposed by the approximation. The calibrated posterior is obtained by defining a lower bound to the logarithmic gain, as defined in Equation 2.16, using Jensen's inequality:

$$\begin{aligned}
\log \mathcal{G}_u^p(h) &= \log \int p(\theta|\mathcal{D}) u(\theta, h)\ d\theta \\
&= \log \int \frac{q_\lambda(\theta)}{q_\lambda(\theta)} p(\theta|\mathcal{D}) u(\theta, h)\ d\theta \\
&\geq \int q_\lambda(\theta) \log \frac{p(\theta|\mathcal{D}) u(\theta, h)}{q_\lambda(\theta)}\ d\theta \quad \text{(by Jensen's inequality)} \\
&= -\mathrm{KL}(q\|p) + \underbrace{\int q_\lambda(\theta) \log u(\theta, h) d\theta}_{\mathcal{U}(\lambda,h) \text{ - utility-dependent term}}. \qquad (4.1)
\end{aligned}$$

This augmented lower bound to the logarithmic gain $\log \mathcal{G}_u^p(h)$ consists of two parts. The first is the negative KL divergence between the variational approximation and the true posterior which can be replaced by the more familiar expected lower bound $\mathcal{L}(\lambda)$ to the model evidence derived in Section 2.3. The second term calibrates the variational posterior to account for the decision making task with the user-defined utility $u(\theta, h)$. The utility-dependent term is independent of the data and only depends on the variational approximation $q_\lambda(\theta)$ calibrating the approximation to optimize the utility. Since we will be referring to the utility-dependent term quite often, we denote it as $\mathcal{U}(\lambda, h)$. This leads to a new loss-calibrated lower bound

$$\begin{aligned}
\log \mathcal{G}_u^p(h) &\geq -\mathrm{KL}(q, p) + \mathcal{U}(\lambda, h) \\
&\geq \mathcal{L}(\lambda) + \mathcal{U}(\lambda, h) = \mathcal{L}_{\mathrm{LCVI}}(\lambda, h).
\end{aligned}$$

Since most decision-theoretic tasks involve estimating the Bayes optimal decisions over the output y or the model parameters θ, we do not model the marginal distribution of the input $p(x)$ and instead restrict our analysis to the discriminative setting. For the dataset $\mathcal{D} = \{(x_i, y_i)_{i=1}^N\}$, we assume y_i to have been generated independently from $p(y|x_i, \theta)$. With the assumption that $\mathcal{D}$ is independent and identically distributed, predictive models exhibiting additive log-gains capture most practical scenarios. The loss-calibrated lower bound can therefore be rewritten as a summation over

individual data points as

$$\mathcal{L}_{\text{LCVI}}(\lambda, \{h\}) = \sum_{i \in [\mathcal{D}]} \left(\mathcal{L}^i(\lambda) + \mathcal{U}(\lambda, h_i) \right) \leq \log \mathcal{G}_u^p(\{h\}).$$

$\mathcal{L}^i_{\text{VI}}(\lambda)$ accounts for the lower bound per data point y_i for which the hypothesis that maximizes the utility is h_i. The term $\{h\} = \{h_1 \dots h_{|\mathcal{D}|}\}$ serves to emphasize that the optimization is performed over each individual h_i. We drop the subscript i for brevity in the rest of the derivation.

Since finding the optimal hypotheses h and fitting variational parameters λ are inter-dependent, we resort to an expectation-maximization algorithm (Dempster et al., 1977) for optimization. We alternate between learning the optimal variational parameters λ to fit the approximation using estimates of the optimal decisions h from the previous iteration and finding the decisions h that maximize the gain for the current approximation $q_\lambda(\theta)$ in the maximization step. For a given set of decisions $\{h\}$, the expectation step finds the variational approximation that maximizes the loss-calibrated bound:

$$\lambda = \arg\max_{\lambda} \mathcal{L}_{\text{LCVI}}(\lambda, \{h\}).$$

The maximization step finds the optimal hypotheses under the approximation characterized by λ:

$$\{h\} = \arg\max_{\{h\}} \mathcal{L}_{\text{LCVI}}(\lambda, \{h\}) \equiv \arg\max_{\{h\}} \ \mathcal{U}(\lambda, \{h\}).$$

Lacoste-Julien et al. (2011) used closed-form analytic updates for λ that relies on contemporaneous coordinate ascent techniques and likelihoods with conjugate priors. This makes incorporating the utility-dependent term difficult at best and impracticable at worst, limiting their framework to a narrow range of probabilistic models. Notwithstanding this limitation, they were able to demonstrate marked improvements in posterior gain when using loss calibration for classification with Gaussian processes for discrete h. Since then, as explained in Section 2.3, rapid progress has been made in approximate inference techniques, and automatic gradient descent has become the de facto standard for optimizing the variational objective. (Kingma and Welling, 2014; Titsias and Lázaro-Gredilla, 2014; Rezende and Mohamed, 2015; Kucukelbir et al., 2017). More recently, Cobb et al. (2018) addressed the limitations imposed by mean field variational inference with coordinate ascent techniques for Bayesian models in Lacoste-Julien et al. (2011) by deriving a loss-calibrated variational bound in the context of Bayesian neural networks with discrete decisions h. Their optimization routine for λ

uses gradient descent for optimization and is based on variational dropout (Gal and Ghahramani, 2016). Since it is more generic, Cobb et al. (2018) is broadly applicable to a variety of utility-dependent terms so long as the hypotheses $\{h\}$ are discrete. In both cases, the challenges in the maximization step can usually be resolved simply via explicit enumeration and summation of the decision space. However, this cannot be readily extended to continuous spaces.

The rest of the chapter develops a suite of techniques to handle the complications that arise with the utility-dependent term $\mathcal{U}(\lambda, \{h\})$ in the case of non-trivial losses with continuous decisions. In particular, we suggest recipes for transformations of the utility functions that secure maximal calibration of the posterior and a variety of practical methods for optimizing the loss-calibrated bound depending on factors such as the type of decision-theoretic loss, the manner of transformation of losses to utilities and the size of the dataset $|\mathcal{D}|$ among other things. We also show how to trade memory usage for improved convergence speed with joint inference of variational parameters and the decisions.

4.1.2 Two Types of Decisions

As we elaborated in Section 2.4, depending on the object of interest, decision-making can be classified into estimation tasks and predictions tasks. In estimation settings, the decisions h are chosen to maximize the utility $\tilde{u}(\theta, h)$ over parameters or latent variables θ of the model. For example, we could estimate the rate parameter in a Poisson-Gamma model to minimize the squared error loss. In the predictive setup, on the other hand, the decisions $\{h\}$ are predictions that minimize the pointwise utility $\tilde{u}(y, h)$ over model outputs $y \sim p(y|\theta, \mathcal{D})$ simulated from the model. In the context of the Poisson-Gamma model, this means that we predict y in order to minimize the count values sampled from the Poisson distribution. Using notation from Equation 2.15, for estimation tasks, the utility-dependent term is defined as

$$\mathcal{U}(\lambda, h) = \int q_\lambda(\theta) \log \underbrace{\tilde{u}(\theta, h)}_{u(\theta,h)} d\theta,$$

whereas for predictive tasks it becomes

$$\mathcal{U}(\lambda, h) = \int q_\lambda(\theta) \log \underbrace{\int p(y|\theta, \mathcal{D}) \tilde{u}(y, h) dy}_{u(\theta,h)} d\theta. \tag{4.2}$$

The latter is computationally more difficult due to the nested integration. We focus on this more challenging family of decisions, but note that scenarios with decisions on θ can be handled analogously while being able to skip many of the technical complications. Since we deal only with the predictive setting, in the rest of the discussion, we use $\tilde{u}(y, h)$ and $u(y, h)$ interchangeably to refer to the utilities over the outputs in prediction tasks without loss of clarity.

4.1.3 Attaining maximal calibration

Optimal decisions derived from sound Bayesian decision principles are invariant to linear transformations of the utility or loss in standard Bayesian decision theory (Berger, 1985). More specifically, a decision h^* that is optimal with respect to $\mathcal{G}_u(h)$ with utility $u(y, h)$ is also optimal with respect to the linear transformation, $u'(y, h) = \alpha \cdot u(y, h) + \beta$ where $\alpha > 0$. Thus a decision $h^* = \arg\max_h \mathcal{G}_u(h)$ is also one that maximizes $\mathcal{G}_{u'}(h)$

Loss-calibrating the posterior introduces certain aberrations from such an invariance assumption since the optimal decisions and the variational parameters are invariably linked. This betrays the Bayesian decision framework that holds that the true posterior remain unaltered by the user-defined loss or utility. The degree of calibration, in such cases, is affected by the bias term β of the linear transformation. This becomes clear by rewriting the utility-dependent term as follows

$$\mathbb{E}_q\left[\log\left(\overbrace{\beta + \alpha\int p(y|\theta,\mathcal{D})u(y,h)dy}^{(a)}\right)\right]$$

$$= \underbrace{\mathbb{E}_q\left[\log\left(1 + \overbrace{\frac{\alpha}{\beta}\int p(y|\theta,\mathcal{D})u(y,h)dy}^{(b)}\right)\right]}_{\text{expectation term}} + \log\beta.$$

The equality holds as long as $\beta > 0$ and $\log\beta$ remains valid. Ignoring $\log\beta$, since it is independent of the variational parameters, we see that the effect of the expectation term is calibrated by the fraction α/β. When $\beta \to \infty$, the term $(b) \to 0$ and the expectation term $\mathbb{E}_q[\log 1] \to 0$ thereby negating the calibration effect. Alternatively, as $\beta \to 0$, the calibration effect of the expectation term becomes maximal. By setting $\beta = 0$ and manipulating

the term (a), we obtain

$$\mathbb{E}_q \left[\log \left(\int p(y|\theta, \mathcal{D}) u(y, h) dy \right) \right] + \log \alpha.$$

Thus, the expectation term is the independent of the scaling term α. Thus, as a general principle, the calibration effect of $\mathcal{U}(\lambda, \{h\})$ is maximized when $\inf_{y,h} u'(y, h) = 0$ $(\beta \to 0)$ which also ensures that the calibration effect is independent of the scaling constant α preserving at least part of the invariance to linear transformation principle outlined in Bayesian decision theory.

4.1.4 Converting between utilities and losses

Decision problems are typically formulated in terms of either minimization of the risk $\mathcal{R}^p_\ell(h)$ or maximization of the gain $\mathcal{G}^p_u(h)$ depending on the decision task. The loss-calibrated variational bound in Equation 4.1 is defined in terms of $\log \mathcal{G}^p_u(h)$ as both the model evidence and the log gain are easy to bound from below and maximizing the lower bound narrows the bound simultaneously on both terms. If using the log risk $\log \mathcal{R}^p_\ell(h)$, this would involve minimizing an upper bound to the risk and maximizing a variational lower bound to $\mathrm{KL}(q\|p)$. In order to calibrate for a decision-theoretic loss within the variational inference framework, it becomes necessary to convert the loss $\ell(y, h) \geq 0$ to a utility $u(y, h) \geq 0$. Bayes optimal decisions obtained from Bayesian decision principles should remain unaffected by transformation between losses and utilities as long as the decisions rely on the true posterior. However, as discussed before, because of integrating the utility term $\mathcal{U}(\lambda, \{h\})$ into the lower bound, the variational approximations are interlinked with the decision, causing the conversion of loss to the utility to affect the final variational approximation. This naturally alters the optimal decisions.

In Subsection 4.1.3, we showed how to minimize the impact of linear transformations on the Bayes optimal decisions. We now invoke the same principles to convert losses to utilities in such a manner as to retain the optimal decisions. A linear transformation preserving the Bayes optimal decisions and non-negative utilities is $u(h, y) = M - \ell(h, y)$ where $M \geq \sup_{y,h} \ell(h, y)$ with optimal calibration at equality. This solution works very well when the loss is either discrete or continuous and is bounded. The transformation does not hold when the loss, discrete or continuous, is unbounded or if there exist outliers (in a bounded) $\ell(y, h)$ so large that $M \gg \ell(y, h)$ for almost all configurations of $\ell(y, h)$ as illustrated in Figure 4.3. The net result of this is that large values of M tend to negate the

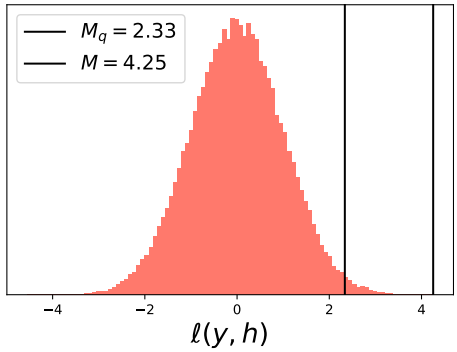

Figure 4.3: An extreme example of an outlier: The loss distribution of $\ell(y, h)$ is provided in the schematic above. Most of the values are concentrated around the mean, with outliers extending to the left and right. The choice of supremum $M \geq \sup_{y,h} \ell(y, h)$ yields a value $M \gg \ell(y, h)$ for most configurations of $\ell(y, h)$. When using the conversion $u(y, h) = M - \ell(y, h)$ this results in values of $u(y, h)$ that are all effectively zero, negating the effect of calibration. The same reasoning applies to unbounded losses, where $M \in [0, \infty)$. A more robust estimate of the maximum such as M_q avoids this problem by ensuring that for most configurations of $\ell(y, h)$, the conversion to the utility retains the effect of calibration.

calibration effect of the utility term since all the transformed utilities tend to be close to zero. To address this issue we propose two solutions that we found to be empirically sound.

Robust maximum For the linear transformation $u(y, h) = M - \ell(y, h)$, any value of $M \leq \sup_{y,h} \ell(y, h)$ will lead to negative utilities $u(\theta, h) \leq 0$ which causes the term $\log u(y, h)$ in Equation 4.2 to be undefined. This can be avoided by using a first order Taylor expansion of $\log u(\theta, h)$ around M and linearizing the log term as follows,

$$\log u(\theta, h) = \log(M - \ell(\theta, h)) = \log M - \frac{\ell(\theta, h)}{M} + \mathcal{O}\left(\frac{\ell(\theta, h)^2}{M^2}\right).$$

Ignoring the error term $\mathcal{O}\left(\frac{\ell(\theta,h)^2}{M^2}\right)$ we see that $\log u(\theta, h) \approx \log M - \frac{\ell(\theta,h)}{M}$. The expectation $\mathbb{E}_q[\log u(\theta, h)]$ can be linearized and rewritten as $\left(\log M - \frac{1}{M}\mathbb{E}_q\left[\ell(\theta, h)\right]\right)$. We drop the constant term $\log M$ leading to the following approximation

$$\mathbb{E}_q[\log u(\theta, h)] \approx -\frac{1}{M}\mathbb{E}_q[\log \ell(\theta, h)].$$

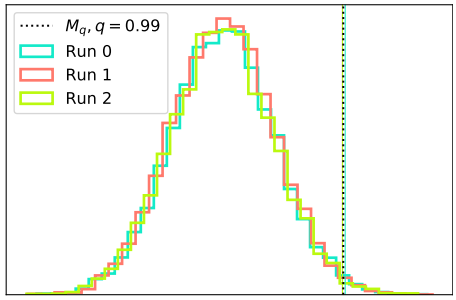

Figure 4.4: Robust estimation of M_q: The value of the quantile q is decided by running several instances of classic variational inference algorithm until convergence and plotting the distribution of loss $\ell(y, h)$ for each data point as a histogram. The values of all runs of the algorithm are sorted to obtain an estimate of the M_q that is robust to outliers. The loss distribution obtained under classical VI approximates the loss distribution under calibrated VI, albeit with a wider variance.

Now that $\ell(\theta, h)$, and by extension $u(\theta, h)$, no longer appears within the logarithm, M no longer needs to be a strict upper bound of $\ell(y, h)$. Instead, we replace it with an estimator that excludes the tail of loss distributions making it robust to outliers. We do this by searching for some value of $M < \sup_{y,h} \ell(y, h)$ such that M falls at the q^{th} percentile of the loss distribution, but other robust estimators would also be applicable.

Empirical experiments show a choice of M_q as the q^{th} quantile of the loss distribution results in strong calibration, the value of q that varies based on the experiments. More specifically, we obtain a robust estimate of M_q by running standard variational inference until convergence and compute the losses $\ell(y, h)$ calculated over the approximate posterior predictive for every data point. The resulting histogram of losses, shown in Figure 4.4, can be used to choose the quantile M_q such that $M_q \ll \sup_{y,h} \ell(y, h)$. For most practical experiments, a value of $q = 70\%$ yielded robust and reliable calibration.

Non-linear transformations Non-linear transformations sidestep the issue of having to choose an optimal value of M_q while still mapping the losses to non-negative utilities. For instance, the transformation

$$u(y, h) = e^{-\gamma \ell(y,h)} \tag{4.3}$$

ensures that the utilities never become negative.

Using the non-linear transformation in Equation 4.3, we can arrive at a robust estimate of γ that can be estimated from the empirical loss distribution. Consider an alternative representation of the utility as a function of $\ell(y, h)$ such that

$$u(\ell) = e^{-\gamma \ell},$$

where the variable ℓ takes arbitrary values from the loss function $\ell(y, h)$ and ℓ_0 refers to instantiations of y and h such that $\ell(y, h) = 0$. We linearize the utility $u(\ell)$ around ℓ_0 (which implies $u(\ell_0) = 1$) and find the value of γ for which the linearized utility is zero at the quantile M_q. Using a second-order Taylor series expansion around ℓ_0, we obtain

$$u(\ell) = u(\ell_0) + \frac{\partial u}{\partial \ell}(\ell - \ell_0) + \frac{1}{2}\frac{\partial^2 u}{\partial^2 \ell}(\ell - \ell_0)^2 + R_3(\ell)$$

where $R_3(\ell)$ is the remainder term. With linearization, one can drop the second order remainder to obtain the approximation,

$$u(\ell) \approx u(\ell_0) + \frac{\partial u}{\partial \ell}(\ell - \ell_0)$$

Since

$$\frac{\partial u}{\partial \ell} = -\gamma e^{\gamma \ell},$$

at ℓ_0,

$$\left.\frac{\partial u}{\partial \ell}\right|_{\ell=\ell_0} = -\gamma e^0 = -\gamma.$$

Finally solving for $u(M_q) = 0$, we obtain,

$$1 - \gamma M_q = 0 \quad \text{and,}$$
$$\gamma = \frac{1}{M_q}.$$

Thus a robust choice of rate parameter γ that also maximizes the calibration effect can be estimated from the quantiles of the empirical loss distribution as $\gamma = \frac{1}{M_q}$.

4.1.5 Monte Carlo Approximation of $\mathcal{U}$

Loss-calibrated variational inference runs into practical challenges when estimating and optimizing the utility term

$$\mathcal{U}(\lambda, \{h\}) = \mathbb{E}_{q(\theta)}\left[\log u(\theta, \{h\})\right] = \mathbb{E}_{q(\theta)}\left[\log \mathbb{E}_{p(y|\theta,\mathcal{D})}[u(\theta, \{h\})]\right]$$

We start by noting that since we already reparametrized θ for optimization of $\mathcal{L}_{\text{VI}}(\lambda)$, we can use the same reparametrization technique to approximate the outer expectation

$$\begin{aligned}\mathcal{U}(\lambda, \{h\}) &\approx \frac{1}{S_\theta} \sum_{\theta \sim q_\lambda(\theta)} \log u(\theta, \{h\}) \\ &= \frac{1}{S_\theta} \sum_{\epsilon \sim q_0} \log u(f_\lambda(\epsilon), \{h\}) \end{aligned} \tag{4.4}$$

where q_0 is the parameter-free base distribution and the only stochastic element in an otherwise deterministic transformation. The deterministic mapping f transforms samples from $q_0(\epsilon)$ into samples from $q_\lambda(\theta)$, and S_θ is the number of samples for Monte Carlo integration.

Discrete Outputs

For discrete outputs, following Cobb et al. (2018), we can compute the inner expectation in Equation 4.2 by summing over all possible values of $y \in \mathcal{Y}$

$$\mathcal{U}(\lambda, \{h\}) = \frac{1}{S_\theta} \sum_{\epsilon \sim q_0} \log \sum_{y \in \mathcal{Y}} p(y | f_\lambda(\epsilon), \mathcal{D}) u(y, \{h\}) d\theta,$$

where $p(y|\theta, \mathcal{D})$ are closed-form functions differentiable with respect to θ (for example, softmax outputs from a neural network) and $u(y, h)$ are classification costs coming from a utility matrix. This makes $\mathcal{U}(\lambda, h)$ straightforward to optimize both with respect to λ via gradient ascent and h by simple enumeration of decisions. However, when the number of discrete classes is very large, this method of enumeration may prove too cumbersome and slow to be of any practical use as the complexity scales linearly with the number of discrete outcomes. Alternatives to this include introducing stochasticity over the classes, by using a subset of the classes to evaluate the *one-vs-each* bound (Titsias, 2016) which provides a rigorous lower bound to softmax probabilities as well as the *augment and reduce* (Ruiz et al., 2018) method to reduce the computational complexity using latent variable augmentation and stochastic variational inference to optimize a lower bound to the marginal likelihood of the data. Unlike the one-vs-each bound which is restricted to softmax, the augment and reduce can be easily extended to other categorical models such as the multinomial probit. Both methods, however, provide a tight lower bound to the exact test log likelihood and the accuracy over classification tasks.

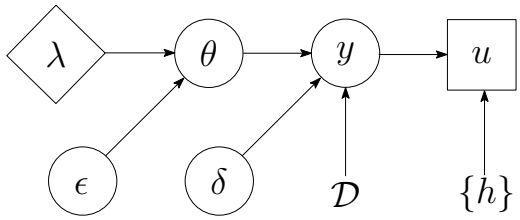

Figure 4.5: Illustration of the double parameterization scheme used in optimization of continuous-valued y: The predictive likelihood $p(y|\theta, \mathcal{D})$ is reparameterized, allowing us to preserve the differentiability with respect to both the variational parameters λ and decisions $\{h\}$. The model parameters θ are reparameterized by drawing samples ϵ and using the transformation $\theta = f_\lambda(\epsilon)$. The samples δ are drawn from a parameter-free distribution as $\delta \sim p_0(\delta)$ and the transformation $y = g_\theta(\delta; \mathcal{D})$ allows us to reparameterize arbitrary likelihoods $p(.)$ with implicit reparameterization gradients (Figurnov et al., 2018). Together y and $\{h\}$ can be used to evaluated the utility $u(y, \{h\})$ using the double reparameterization scheme.

Continuous Outputs

For continuous-valued y, however, the inner expectation in $\mathcal{U}(\lambda, \{h\})$ remains a challenge to approximate and optimize and we will outline a double reparametrization scheme to solve it. In addition to reparameterizing the approximation $q_\lambda(\theta)$ we are also reparameterizing the predictive likelihood $p(y|\theta, \mathcal{D})$. This allows us to find gradients of the inner expectation approximated with Monte Carlo samples

$$u(\theta, \{h\}) \approx \frac{1}{S_y} \sum_{\delta \sim p_0} u(g_{f_\lambda(\epsilon)}(\delta; \mathcal{D}), \{h\}), \tag{4.5}$$

to preserve differentiability with respect to both λ and $\{h\}$. The samples δ are drawn from the parameter-free distribution p_0 and transformed with the differentiable mapping $g(.)$ to samples $y \sim p(y|g_\theta(\delta; \mathcal{D}))$. The approximation relies on the ability to reparametrize (almost) arbitrary likelihoods p, which was made possible only recently thanks to implicit reparametrization gradients (Figurnov et al., 2018). The schematic of the double reparameterization technique is presented in Figure 4.5.

Nested integral estimation, especially in the form of Equation 4.2 where the inner expectation appears within the logarithm, is not straightforward. The simplest method of estimation is to replace the integrals in Equation 4.2 with their Monte Carlo estimates in Equations 4.4 and 4.5. While admittedly a biased estimator, it is suitable for most practical purposes.

Bias can, for instance, be reduced by using the Taylor series expansion of $\mathbb{E}_p[\log u]$ in a manner similar to (Teh et al., 2007), giving

$$\log \mathbb{E}_p[u] \approx \mathbb{E}_p[\log u] + \frac{\mathbb{V}_p[u]}{2\mathbb{E}_p[u]^2}.$$

For the log-linearized utility-term derived in Subsection 4.1.3, the log term disappears and one simply uses the unbiased estimator

$$\mathcal{U}(\lambda, \{h\}) \approx -\frac{1}{MS_\theta S_y} \sum_{\epsilon \sim q_0} \sum_{\delta \sim p_0} \ell(g_{f_\lambda(\epsilon)}(\delta; \mathcal{D}), \{h\}). \tag{4.6}$$

Despite its simplicity and usefulness, this estimator may violate the loss-calibrated bound in Equation 4.1. Even though the linearized estimator used by previous authors (Cobb et al., 2018; Lacoste-Julien et al., 2011) here benefits from optimal choice of M, it shows very little improvement in terms of risk when compared to standard VI. The naive estimator in Equation 4.6 fits the posterior better and results in consistent empirical improvement.

4.1.6 Optimization

The optimization of the variational parameters λ and the decisions h can be performed with alternating minimization. The variational parameters λ are optimized using gradient descent with a combination of standard reparametrization techniques and Monte Carlo estimates of the gradient. This is well covered in variational inference literature (Kucukelbir et al., 2017; Titsias and Lázaro-Gredilla, 2014). As for the decisions $\{h\}$, depending on the loss function $\ell(y, h)$ used, we can choose one of the following four techniques.

- **Closed-form optimization**: When utilities are log-linearized they can be expressed as loss functions. With trivial and symmetric loss functions, the Bayes estimate of the optimal decision h is evaluated analytically as statistic of the posterior predictive distribution. A small set of examples is shown in Table 4.1. The posterior predictive distribution, however, is usually unavailable in the closed form. In such cases, Monte Carlo approximation of posterior predictive is used to estimate the required statistics.

- **Numerical optimization**: When there is no closed form Bayes estimator for the loss or utility function, one can use standard black-box

Table 4.1: Selected losses and their closed-form analytic Bayes estimators that minimize the expected posterior risk. Non-trivial losses with no analytic estimates can be solved with black-box optimization routines such as L-BFGS (Liu and Nocedal, 1989). Derivative-free optimization methods can be used for non-differentiable losses.

Loss	Expression	Bayes estimator
Squared	$(h-y)^2$	$\mathbb{E}_p[y]$
LinEx	$e^{c(h-y)} - c(h-y) - 1$	$-\frac{1}{c}\log\int e^{-cy}p(y)dy$
Absolute	$\lvert h-y\rvert$	$\text{median}_p[y]$
Tilted	$\begin{cases} q\cdot\lvert h-y\rvert & y \geq h \\ (1-q)\cdot\lvert h-y\rvert & y < h \end{cases}$	q-percentile$_p[y]$
Imbalanced absolute	$\begin{cases} a\cdot\lvert h-y\rvert & y \geq h \\ b\cdot\lvert h-y\rvert & y < h \end{cases}$	$\frac{a}{a+b}$-percentile$_p[y]$

numerical optimization routines and maximize a Monte Carlo approximation of $\mathcal{U}(\lambda, \{h\})$ to obtaining the optimal decision. A straightforward implementation would result in solving $|\mathcal{D}|$ one-dimensional optimization problems:

$$\arg\max_{\{h\}} \frac{1}{S_\theta S_y} \sum_{\epsilon\sim q_0} \sum_{\delta\sim p_0} \log u(y_i, h_i).$$

While one could solve these in parallel, a better approach is to jointly optimize for all $\{h\}$ with the aggregated utility since the utility-dependent term is additive over the individual decisions:

$$\arg\max_{\{h\}} \frac{1}{|\mathcal{D}| S_\theta S_y} \sum_{i=0}^{|\mathcal{D}|} \sum_{\epsilon\sim q_0} \sum_{\delta\sim p_0} \log u(y_i, h_i).$$

Even though the estimation of gradient for each iteration is more expensive in the aggregated case, the number of iterations required to converge remains roughly the same. We also note that many of the interesting utilities (for example, derived from absolute and tilted loss) are not smooth functions of h, which could cause difficulties for gradient-based methods. However, the aggregated utility has been empirically shown to be smooth by Schaul and LeCun (2013) even if the utilities themselves are point-wise non-differentiable. Similarly,

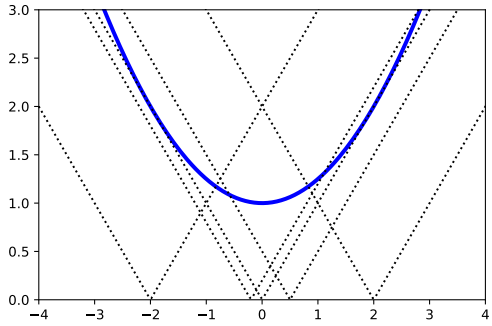

Figure 4.6: On the left we see several absolute error losses of the form $\ell(y, h) = |y - h|$ for different values of y. Schaul and LeCun (2013) empirically proved that the aggregated sum of theses losses, shown as the solid blue curve, is smooth even if the individual losses are themselves not. The same line of reasoning can be applied to evaluating the expectation $\log u(\theta, h)$.

the expectation $\log u(\theta, h)$ tends to be smooth when approximated by sampling even if $u(y, h)$ itself is non-differentiable (see Figure 4.6).

- **Joint optimization**: When using gradient-based methods for numerical optimization of the utility $\mathbb{U}(\lambda, \{h\})$, one can switch from using expectation-maximization Dempster et al. (1977) to instead optimizing the loss-calibrated bound $\mathcal{L}_{\text{LCVI}}(\lambda, \{h\})$ *jointly* with respect to both λ and $\{h\}$. In such cases $\nabla\mathcal{L}_{\text{LCVI}}$ is defined as

$$[\nabla_\lambda \mathcal{L}_{\text{LCVI}}, 0, \ldots, 0] + \left[\nabla_\lambda \mathbb{U}, \frac{\partial \mathbb{U}}{\partial h_1}, \ldots, \frac{\partial \mathbb{U}}{\partial h_{|\mathcal{D}|}}\right]$$

  This typically results in much quicker convergence and does not require full optimization of h for every update of λ, but comes with the disadvantage of having to store the decisions h for the dataset $\mathcal{D}$ even when using stochastic optimization methods with small batch sizes.

- **Amortized decision optimization** As part of the thesis, we introduce a novel, amortized approach to finding optimal decisions h, as originally presented in the first draft of Paper III that is available as Kuśmierczyk et al. (2019a). For extremely large $|\mathcal{D}|$, solving for the optimal $\{h\}$ may prove prohibitively expensive, due to the linear scaling of computation time (or required parallel cores) with respect to $|\mathcal{D}|$. However, this computational complexity can be alleviated by approximating h with a separate parametric decision-making module.

Table 4.2: Comparing numerical and amortized decision-makers on outputs generated by a polynomial regression model, with tilted loss ($q = 0.2$) and $S_y = 50$. The amortized variant learns a constant-time parametric model to predict the hypothesis with clearly sufficient accuracy, whereas the numerical optimizer has to solve h for $|\mathcal{D}|$ instances and is slow for $|\mathcal{D}| > 10^5$.

	Numerical		Amortized	
$\lvert\mathcal{D}\rvert$	MSE(e^{-3})	Time(s)	MSE(e^{-3})	Time(s)
10^3	1.41	0.04	5.35	4.43
10^4	1.49	0.42	4.95	4.23
10^5	1.51	3.44	2.60	4.27
10^6	1.52	39.3	4.92	4.69

This amortizes the cost of making the decisions, in a manner analogous to amortizing the inference in variational autoencoders (Kingma and Welling, 2014).

Given the input x_i and the prediction $y_i = g(\delta, f_\lambda(\epsilon), x_i)$ of the model $p(y|\theta, x_i)$, we train an arbitrary supervised model, in practice a deep neural network, to approximate the optimal decision. The network is trained on pairs of (x_i, y_i) to predict $h_i = \mathrm{NN}_\phi(x_i)$ minimizing directly the (linearized) objective

$$\arg\min_\phi \frac{1}{|\mathcal{D}| S_\theta S_y} \sum_{i=0}^{|\mathcal{D}|} \sum_{\epsilon \sim q_0} \sum_{\delta \sim p_0} \ell(y_i, h_i).$$

It learns to approximate optimal h_i for any $p(y_i|\theta)$, and after training can provide approximate h_i for all inputs. For sufficiently large $|\mathcal{D}|$, the computational cost of training the network is trumped by the computational saving of not needing to solve $|\mathcal{D}|$ decisions numerically.

Table 4.2 compares numerical optimization with amortized decision-making in the context of optimal Bayes decisions for a polynomial regression model, so that both are run until sufficiently low error is reached. For small problems optimizing h directly is clearly more efficient, but for problems larger than $|\mathcal{D}| \approx 10^5$ the amortized solution approximating the decisions with a separate neural network is faster.

4.1.7 Automatic Loss Calibration in Probabilistic Programs

We crystallize the previous ideas and repeat the algorithm description in Kuśmierczyk et al. (2019a) for convenience. We show how to calibrate variational approximations for a rich family of probabilistic models with continuous utilities with a simple running example. A problem specification consists of:

1. A *model* $p(y, \theta)$, typically specified using a probabilistic programming language, along with training data $\mathcal{D} = \{x_n, y_n\}$, $n = \{1 \dots N\}$. For instance, the Bayesian linear regression model is described in terms of weight parameters θ and variance σ^2 as

$$\begin{aligned} p(\theta) &= \mathcal{N}(\theta; 0, 1), \\ p(\sigma^2) &= \text{Gamma}(\sigma^2; 1, 1), \\ p(y; x, \theta, \sigma^2) &= \mathcal{N}(y; \theta^T x, \sigma^2). \end{aligned}$$

   The priors on θ have a unit normal distribution and on the variance σ^2 is a Gamma distribution with unit parameters. The parameter σ^2 lives in the constrained real space, $\sigma^2 \in \mathbb{R}^+$. T Since the Gaussian variational approximations typically operate in real space, it becomes expedient to transform $\tau = \log(\sigma^2) \in \mathbb{R}$ and operate on τ instead. To account for the change of variables, we introduce a Jacobian adjustment which is the derivative of inverse of the log transformation, viz. exp

$$p(\sigma^2) = \text{Gamma}(\exp(\tau); 1, 1) \exp(\tau)$$

   Most probabilistic programs have an internal mechanism that apply the transformations into real space and the concomitant Jacobian adjustments automatically.

2. A *user-defined decision-theoretic utility* $u(\theta, h)$ or $u(y, h)$ defined over parameters or predictions. User-defined losses can be converted into utilities as outlined in Subsection 4.1.3. In our example, we define the decision task in terms of losses and use techniques in Subsection 4.1.3 to convert these losses to utilities. A squared loss over the parameters θ can be defined as $\ell(\theta, h) = (\theta - h)^2$ in the case of estimation. In the predictive setting, we define a tilted loss over the output y as

$$\ell(y, h) = \begin{cases} q \cdot |h - y| & y \geq h \text{ and,} \\ (1 - q) \cdot |h - y| & y < h, \end{cases}$$

   where q controls the quantile. This allows the model to calibrate its predictions depending on the value of q.

3. An *approximating family of distributions* $q_\lambda(\theta)$. Standard probabilistic programming languages such as Stan (Carpenter et al., 2017) make the mean-field assumption as the default, making it straightforward to construct univariate Gaussian variational approximations on each parameter independently. Full-rank approximations that model correlations between the posteriors parameters can also be used (Dillon et al., 2017; Carpenter et al., 2017). In our running example, we make the mean field assumption with a normal variational approximation giving us,

$$q(\theta) = \mathcal{N}(\theta; \mu_\theta, \sigma_\theta^2),$$
$$q(\tau) = \mathcal{N}(\tau; \mu_\tau, \sigma_\tau^2).$$

   Together $\mu_\theta, \sigma_\theta^2, \mu_\tau$ and σ_τ^2 constitute the variational parameters λ.
4. Additionally, a more experienced user may also specify *convergence conditions*, e.g., error tolerance for values of gain $\mathcal{G}_u(h)$. This defaults to 10^{-2} in Stan (Carpenter et al., 2017) for example.

The objective in Equation 4.1 is maximized by alternating optimization of λ and h until convergence Monte Carlo evaluation of the bound is done with one of the alternatives presented in Subsection 4.1.5.

Optimization of λ Variational approximation parameters λ are optimized via gradient-based algorithms using reparametrization of θ and y as explained in Section 2.3 and Subsection 4.1.5.

Optimization of h The optimal decisions h are found using one of four strategies in Subsection 4.1.6, selected to suit the problem characteristics. Numerical optimization is the default option, while faster closed-form optimization is used for linearized estimators and when a closed-form Bayes estimator is available (Table 4.1). For complex losses and large $|\mathcal{D}|$, amortized decision-making is used. Since h changes slower than the parameters λ, it suffices to solve for h after every few iterations of updating λ.

This completes our discussion of the loss calibration procedure in the context of approximate inference. Although Bayesian decision theory tends to dissociate the inference phase from the decision-making phase, the restrictive limitations imposed by approximating the posterior with a family of distributional approximations necessitate a tight integration of the two phases. Automatic loss calibration renders the loss calibration framework user-friendly, practical, easy to integrate into existing Bayesian workflows and accessible for broader use in the Bayesian community. We elaborated

on the unavoidable complications that ensue when expressing the decision task in terms of losses in lieu of utilities especially for continuous, unbounded losses and proposed fixes to address these complications. We detailed unique ways of using Monte Carlo approximations for the utility-dependent terms. In Section 4.2 we discuss an alternate approach to loss calibration that is agnostic to the inference techniques and quality of approximations and loss-calibrates the approximate posterior by correcting the bias introduced by the approximation after the inference phase. As we shall see, this decoupling not only improves computational efficiency, but also allows straightforward application of Bayesian decision principles.

4.2 Post Hoc Corrections For Posteriors

We saw that loss-calibrating the approximate posterior during inference adapts the approximating distribution to maximize the gain for the decision-task under constraints imposed by the distributional limitations. The procedure is sound and leads to posteriors that are calibrated to minimize the decision-theoretic loss. It does, however, suffer from several drawbacks. Firstly, one needs to account for the fact the assumptions of Bayesian decision principles are violated. The final Bayes optimal decisions are not immune to linear transformation of the utilities or the manner of transformation of losses to utilities (See Subsections 4.1.3 and 4.1.4). Secondly, the alternating minimization of the variational parameters and decisions for non-trivial losses requires numerical optimization which is computationally expensive. It is not always easy to obtain marked enough empirical improvements in the gain to justify the computational costs. Lastly, the decision-theoretic losses might not be known beforehand, might be dynamic depending on the shifting use-cases, or there might be a number of losses that we wish to optimize for. This requires re-running the loss calibration repeatedly from scratch each time the loss changes or for every loss. As an alternative, we introduce an orthogonal approach to loss calibration that sidesteps many of the drawbacks. It helps evade the most problematic features of loss-calibrated variational inference, namely, the expensive inference and transformation of losses to utilities without breaking with Bayesian decision principles and negating the calibration effect of the utilities.

Keeping these considerations in mind, instead of modifying the inference procedure to calibrate the posterior, we modify only the decision-making phase. This is achieved by introducing a parametric function that transforms a suitable characterization of the approximate posterior to minimize

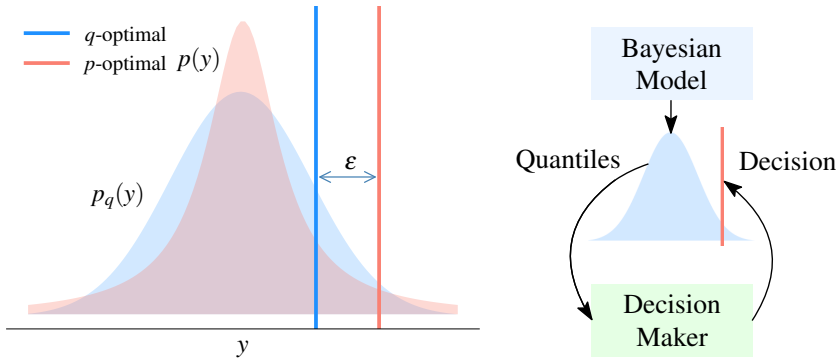

Figure 4.7: Bayesian decision theory equips us with a framework to make optimal decisions for a decision-theoretic loss under posterior uncertainty. The optimal decision under the true posterior predictive distribution $p(y)$ shown in red, can be obtained by closed-form analytic solutions for simple losses, or by numerical optimization for more complex losses if they do not have analytic solutions. However, the same decision rules cannot be used with the approximate posterior predictive distribution $p_q(y)$ is a normal approximation to the heavier-tailed $p(y)$, and q-optimal decision differs from the p-optimal, by a difference of ϵ. Our post hoc approach introduces a decision-making module that corrects for this error by transforming a characterization of the incorrect posterior distribution into the correct decision. Figure reproduced from Paper IV (Kuśmierczyk et al., 2020).

the *classical* or empirical risk or equivalently, maximize the empirical gain. We call this function the *decision-making module* as it outputs decisions that are optimal with respect to the empirical risk as showin in Figure 4.7. Our approach is broadly applicable to arbitrary probabilistic models and is agnostic to the approximate inference method used. The decision-making module needs only post-inference artifacts to correct for the posterior. This turns out to be surprisingly efficient as there is no need to revisit the inference phase. Accordingly, this constitutes a post hoc or after-the-event correction of the posterior, as the calibration is performed *after* the fact that the approximate posterior has been inferred independent of the decision making task.

4.2.1 Decision Making Modules

The discussion in this section is inspired by amortized optimization in Subsection 4.1.6. While the amortized technique learns to map covariates to decisions h for a dynamically calibrating $q(\theta)$, we now construct a decision-

making module that operates on a previously inferred, fixed $q(\theta)$ to yield optimal decisions h. The decision-making module, however, utilizes a characterization of the predictive distribution as its input. In practice, we sometimes incorporate the covariates when the posterior predictive characterization suffers from identifiability issues outlined in Subsection 4.2.6.

Here, as in Section 4.1, we focus on making optimal decisions for the outputs y – decisions on the parameters θ can be made in a similar fashion. For such supervised setups, decisions h are functions defined over data points (x, y) where the expected risk $\mathcal{R}(h)$ is the expectation of the decision-theoretic loss over the true data-generating distribution: $\mathcal{R}(h) = \mathbb{E}_{x,y\sim p(x,y)}\left[\ell(y, h(x))\right]$. Within the regime of discriminative learning, we are interested in faithfully modeling only $p(y|x, \theta)$ and do not model the marginal distribution $p(x)$. In such cases, $\mathcal{R}(h)$ could be approximated with the p-conditional risk $\mathcal{R}^p_\ell(h|x) = \mathbb{E}_{y\sim p(y|x)}\left[\ell(y, h))\right]$, where $p(y|x) = \int p(y|x, \theta)p(\theta|\mathcal{D})\ d\theta$. Since we lack access to the true data-generating distribution $p(x, y)$ as well as the true predictive distribution $p(y|x, \theta)$, we instead use an empirical approximation of the risk $\mathcal{R}(h)$ as the objective to minimize

$$\mathcal{R}(h) \approx \mathcal{E}\mathcal{R}(h) = \frac{1}{|\mathcal{D}|}\sum_{n=1}^{|\mathcal{D}|} \ell(y_n, h(x_n))$$

A decision h_q that minimizes the q-risk, $h_q = \arg\min_{h\in\mathcal{H}} \mathcal{R}^q_\ell(h|x)$, is not necessarily the same as the decision h that is optimal with respect to $\mathcal{R}(h)$. In order to correct for the sub-optimal decisions produced from the approximation $q(\theta)$, we take a characterization of the q-posterior predictive $p_q(y)$ and feed it into a mapping that transforms the characterization into the decisions h that are optimal with respect to the risk $\mathcal{R}(h)$. This amortized mapping $h = f(p_q(y); \omega)$, called the decision-making module, is typically a fully-connected neural network parameterized by weights ω, that are shared across all data points. The characterization can be in the form of a single statistic, such as the mean or median, or a set of quantiles estimated either using Monte Carlo sampling or analytical estimates from $p_q(y)$.

Since the decision-making module $f(p_q(y); \omega)$ is overly flexible as a universal approximator, there is the risk of completely overriding the underlying Bayesian model, with decisions h that, while minimizing $\mathcal{R}(h)$ would be improbable under the posterior $p(\theta|\mathcal{D})$. To avoid this, we implicitly regularize ω by setting normal priors $\mathcal{N}(\mu, \sigma^2)$ on the decisions h. These priors are learned by using bootstrap (Efron and Tibshirani, 1994). We control the trade-off between risk minimization and model faithfulness with a regularization parameter. Before we get into the particulars of optimizing this framework, we briefly introduce the generalized Bayesian inference

framework that allows us to incorporate the decision-making module as a post-inference adjustment of the posterior predictive.

4.2.2 Generalized Bayesian Inference

Generalized Bayesian inference (Bissiri et al., 2016) provides a rigorous basis for learning and updating prior distributions to the posteriors when the parameters of interest are linked to the observations via arbitrary loss functions rather than via a member of a family of parametric density functions. This paints a contrast to the probabilistic modeling approach that limits Bayesian inference to a highly restrictive set of parametric likelihood functions. Issues with classical Bayesian inference ensue when the parameters of interest η and the observations y are not connected via a density function $f(y;\eta)$ or if the form of $f(y;\eta)$ is unknown. In such cases, the generalized Bayesian inference approach readily defines belief updates for η on discrepancy measures between a proxy model and $f(y;\eta)$.

More concretely, given observations y and a parameter of interest η, if $p(\eta)$ represents the prior over η, a coherent update for the posterior exists, given by

$$p(\eta|y) \propto e^{-\ell(y,\eta)} p(\eta)^{\nu}$$

and is valid for any arbitrary loss function $\ell(y,\eta)$. A proof sketch for this is given in Bissiri et al. (2016) and relies on the fact that given a parameter η and a prior $p(\eta)$ an update $p(\eta|y)$ for η must exist as further information is gained about η from y via $\ell(y,\eta)$. The framework also supports use of partial information (where only part of the observations is relevant to the parameter) as well as non-stochastic information (such as expert opinions) to infer the posterior.

The prior $p(\eta)^{\nu}$ is a power prior where the parameter ν regulates the strength of the prior. When $\nu < 1$, the loss $\ell(y,\eta)$ is given more prominence when updating η allowing the data y to inform the posterior better. As $\nu \to 0$ this leads to maximum likelihood estimates of η. Alternatively, $\nu > 1$ minimizes the influence of the loss letting the prior assert itself more. One can make the inference more robust by specifying hyperpriors on ν, as is customary in hierarchical Bayesian models. This can be done by extending the loss function to include ν as an unknown parameter to be inferred,

Generalized Bayesian inference lets us make minimal assumptions about the loss, and concentrate on the parameter of interest. Observations from probability distributions are also accommodated with loss functions that are equivalent to negative log likelihoods. For a more in-depth discussion of generalized Bayesian inference in the context of variational inference see

Knoblauch et al. (2019). We now discuss how this framework relates to posterior inference on the decisions h for decision-making modules.

4.2.3 Decision Belief Distributions

Generalized Bayesian inference allows us to think directly in terms of the loss-calibrating of the model parameters θ, even though they tie only indirectly into the decision-theoretic loss function $\ell(y, h)$ for predictive problems. That is, we could define updates to the model parameters as

$$p(\theta|y) \propto e^{-\ell(h,y)} p(\theta)^{\nu}$$

The decisions h are obtained by marginalization of the model parameters to yield the q-posterior predictive $p_q(y)$. However, this proves too cumbersome to train as it involves repeatedly optimizing the decision-making module $f(p_q(y), \omega)$ which is a function of the approximation $q(\theta)$. This implies that each time $q(\theta)$ is updated during inference, the decision-making module has to be trained until convergence for that particular instantiation of $p_q(y)$.

An alternative is to consider *decision belief distributions* where the updates are defined in terms of the decisions themselves. This dissociates the inference of $q(\theta)$ from the decisions h that minimize the decision-theoretic loss. The posterior over h in such cases is,

$$p(h|y) \propto e^{-\ell(h,y)} p(h)^{\lambda} \tag{4.7}$$

where the decisions for different y are tied to the amortized decision-maker $h = f(p_q(y); \omega)$. The decision-maker parameterized by ω minimizes the loss $\ell(h, y)$. Understanding that learning the posterior over decisions h for each observation y is impractical, we take advantage of the amortization offered by the decision-making module and define a more efficient update rule. Since h is tied to the q-posterior predictive through the amortized weights ω, a belief distribution over ω in turn induces a belief distribution on the decisions. We can therefore learn decision belief distribution by simply defining a Bayesian update rule over the decision-making module parameters,

$$p(\omega|y) \propto e^{-\ell(f(p_q(y),\omega),y)} p(\omega)^{\nu}. \tag{4.8}$$

where we have explicitly replaced $\ell(h, y))$ with $\ell(f(p_q(y), \omega), y)$ to emphasize the dependence of the decisions h on the parameters ω of the decision-making module. Approximating the loss as a cumulative sum over a collection of N independent and identically distributed observations

$\mathcal{D} = \{x_i, y_i\}_{i=1}^N$, the posterior over ω is defined as,

$$\log p(\omega|\mathcal{D}) = \underbrace{-\frac{1}{N}\sum_{n=1}^{N} \ell(h_n, y_n)}_{\mathcal{ER}(h)} + \nu \log p(\omega) + Z,$$

where Z, the normalization constant, is independent of the parameters ω. The empirical risk $\mathcal{ER}(h)$ arises naturally from the definition of belief update rules and the posterior $p(\omega|\mathcal{D})$ minimizes the empirical risk.

Although this formulation technically allows full posterior inference over ω (by using a Bayesian neural network, for instance), since we are interested only in a point estimates decisions h that minimizes the empirical risk, it is computationally expedient to simply use MAP estimates of ω. The Bayesian update rules in Equations 4.7 and 4.8 help clarify the relationship between ω and h and broaches the possibility setting priors on ω to ensure that the decisions h remain plausible under posterior uncertainty $p(\theta|\mathcal{D})$ of the underlying Bayesian model.

4.2.4 Implicit Priors for Decision-making

Setting meaningful priors on the decision-making module through ω is non-intuitive since ω is often high-dimensional without any meaningful interpretation attached to it. The standard practice of setting weak normal priors on ω does little to inform the decision-maker of the range of values that h can take. A more sensible alternative (Sun et al., 2019) is to set priors on the function space, limiting the family of functions that the decision-maker can approximate, rather than the parameter space. This is not a viable strategy as we do not yet understand the relationship between functions the decision-maker can approximate and the Bayesian model. Given the ω plays in the amortized decision maker, we resort to a third alternative of setting implicit priors on ω by explicitly setting priors on the decisions h which are strongly tied to ω.

Regularization of h is carried out through a choice of priors that is informed by the relationship between the decisions and the underlying Bayesian model. First, we take inspiration from bootstrap techniques (Efron and Tibshirani, 1994) to obtain reasonable priors on the decisions h. This lets us gauge the mean of the q-optimal decisions and their variance from the empirical distribution of the decisions. Then, we use the regularization parameter ν from the power prior to determine the degree to which the prior influences the observation. Using power priors with optimal choice of ν ensures the decisions remain faithful to the underlying model

as well as preventing overfitting to the empirical risk that could undermine generalization of the decision-maker to the true risk $\mathcal{R}(h)$ evaluated from the data-generating distribution. A reasonable choice of ν can be obtained by cross-validation.

We now elaborate on the bootstrap procedure we use to define a prior $p(h)$ over the decisions and how this sets an implicit prior on ω. We begin with the q-optimal decision h_q^n for the n^{th} data point, obtained from the approximation $q(\theta)$. A normal prior on the decision h^n for the n^{th} point can be defined by

$$h^n \sim \mathcal{N}(h_q^n, 1),$$

with h^n centered on the q-optimal decisions with unit variance. The power prior parameter ν on $p(h)$ determines the degree to which the prior asserts itself in the update rule. The decisions are agnostic to the variance of the posterior predictive $p_q(y)$. A more informative prior can be obtained by allowing point-specific variation,

$$h^n \sim \mathcal{N}(\mu_q^n, \sigma_q^n),$$

where each decision h^n has a parameter controlling the variance σ_q^n independent of ν. Determination of the values of μ_q^n and σ_q^n is achieved with bootstrap (Efron and Tibshirani, 1994). In order to obtain bootstrap estimates of μ_q^n and σ_q^n, we sample with replacement L datasets $\mathcal{D}_l$ where $l = 0 \dots L$. The Bayesian model is trained on each $\mathcal{D}_l$, to obtain approximations $q^l(\theta)$ and q-optimal decisions h_{ql}. The sample mean can be estimated as

$$\hat{\mu}_h^n = \frac{1}{L} \sum_{l=1}^{L} h_{ql}^n.$$

The unbiased estimate of the sample standard deviation is

$$\hat{\sigma}_h^n = \frac{1}{L-1} \sum_{l=1}^{L} (h_{ql}^n - \hat{\mu}_h^n)^2.$$

Assuming point-wise independence over the decisions, we can derive the implicit prior for ω as

$$\log p(\omega) = \sum_{n=1}^{N} \log N(f(p_q(y_n), \omega) | \mu_h^n, \sigma_h^n).$$

When $\mu_h^n = h_q^n$ and $\sigma_h^n = 1$ all the decisions have the same amount of uncertainty allowing the uncertainty in the posterior estimates to be uniform

over data points. When $\mu_h^n = \hat{\mu}_h^n$ and $\sigma_h^n = \hat{\sigma}_h^n$, point-specific variance over the decisions are reflected in the posterior as well. For both alternatives ν controls the strength of the prior, although in the latter case, the strength of the prior varies by data point.

Now that we have described the generalized Bayesian update in the context of decision-making for the Bayesian model, we describe two empirical studies that best characterize the effect of posterior representation on the empirical risk and the effect of the power prior parameter ν on the faithfulness of the decisions to the underlying model.

4.2.5 Characterizing the Predictive Distribution

Bayesian decision theory operates under the principle that the Bayesian posterior is necessary and sufficient information for making optimal decisions. We stick to this principle and choose as our decision-making module a flexible fully-connected neural network with three hidden layers which maps an arbitrary characterization of posterior predictive $p_q(y)$ to the optimal decisions h. A selection of the optimal architecture also having a sufficient predictive power can be performed with cross-validation, by increasing neural network complexity starting from the most simple one. Overly complicated networks may overfit empirical risk while not being faithful to the underlying model.

For generalizing to a wide variety of models,we characterize the predictive distribution with S Monte Carlo samples y_s from the posterior predictive $p(y)$ under the posterior approximation $q(\theta)$. More specifically we draw, for each sample $s \in S$,

$$\begin{aligned} \theta_s &\sim q(\theta) \quad \text{and,} \\ y_s &\sim p(y; \theta_s). \end{aligned}$$

Such a collection of S draws is summarized into a finite statistic such as the mean, median or B evenly spaced empirical quantiles as illustrated in Figure 4.8. We demonstrate empirically how the empirical risk behaves as a function of the number of Monte Carlo samples S and quantiles B in Figure 4.9. We failed to see any improvement in the empirical risk for quantiles over $B = 20$ and for samples over $S = 100$.

4.2.6 Model Fidelity

We can also conduct experiments to study how implicit priors on the optimal decisions h control the fidelity of the optimal decisions h to the underlying Bayesian models. The decisions should only correct for errors that

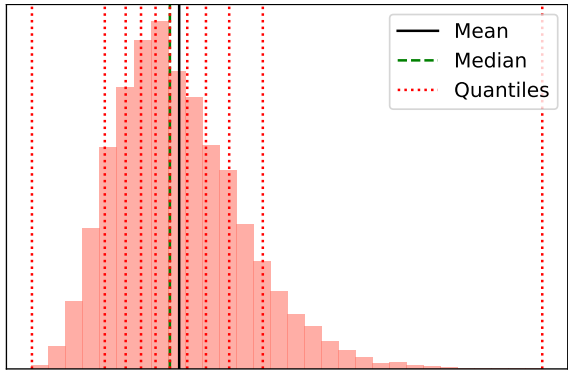

Figure 4.8: Characterization of the posterior. Draws from the posterior predictive distribution can be captured and summarized through sufficient statistics. This could be either the mean (black), the median (green) or a set of B evenly spaced quantiles (red). In this illustration, we set $B = 10$.

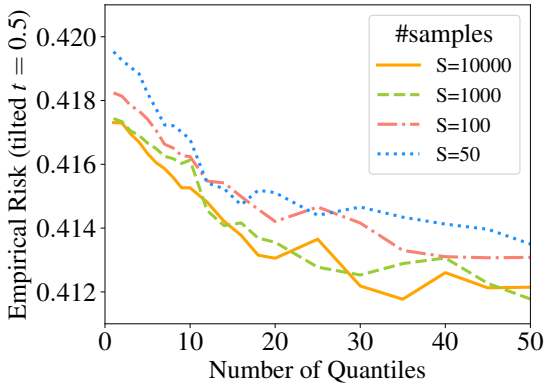

Figure 4.9: Effect of the number of samples and the number of quantiles on the empirical risk for the tilted loss. The risk was evaluated over a matrix factorization model on the test set of the *last.fm* data. The empirical risk reduction is negligible after the number of quantiles reaches 20 and the number of samples reaches 100. Figure reproduced from Paper IV (Kuśmierczyk et al., 2020).

arise out of poor posterior approximation and must not override the underlying Bayesian model. The practitioner requires that the decisions still be interpretable and in keeping with the underlying model. In Figure 4.10, we see the posterior predictive distribution of an intentionally misspecified model, Bayesian linear regression on non-linear data. We set the number of iterations for bootstrapping the prior to 10 and use the samples to estimate the mean and variance of the prior on the decisions $p(\omega)$. With power priors ν we control the degree of regularization. The decision-maker takes a characterization of the posterior predictive and with no regularization – that is, weak priors on ω – learns to model the underlying data distribution accurately to minimize the risk at the cost of ignoring the model. This is possible because the underlying injective mapping between the covariates and the posterior predictive allows the decision maker to implicitly learn from the covariates themselves to map to the non-linear data. When applying strong regularization, we create a strongly informative prior centered on the q-optimal decisions forcing the decision-maker to transition towards predicting the q-optimal decisions in keeping with the underlying Bayesian model.

The decision-maker takes a characterization of the posterior to infer the optimal decisions. Because of this, the data generated by the Bayesian model requires some sort of identifiability criterion. In this case, because the prediction has a slope $m \neq 0$, the slope provides enough context to disambiguate between various points of the predictive distribution. However, if the slope is zero, the characterizations of the predictive distribution are going to be similar across all the data points making it impossible for the decision-maker to learn a mapping to the optimal decisions. This shortcoming can be addressed, for example, by incorporating an identifiability criterion, such as the covariates x. However, this is not without risks, since the decision-maker can learn to ignore the predictive distribution completely and base its optimal decisions on the covariates x, thus overriding the Bayesian model.

4.3 Summary

We began this chapter by motivating the need for loss-aware approximate inference algorithms in order to yield optimal decisions with respect to the risk, both Bayesian and classical, specifically in the predictive setup. We discuss at length two alternative approaches to this end. The first is loss-calibrated variational inference that extends the work of (Lacoste-Julien et al., 2011). We adapt this framework for use in gradient-based,

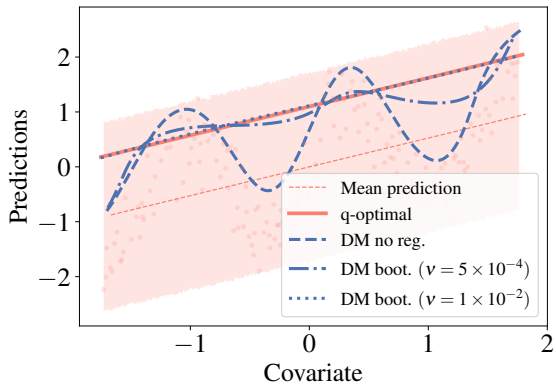

Figure 4.10: When $\nu = 0$, there is no regularization and the decision maker completely ignores the linear predictions (red). By setting priors on the decisions with bootstrap and selecting ν to specify the degree to which the decision maker should be faithful to the underlying model, we provide predictions that are compatible with the underlying model. Note, however, that in the case of linear regression, the q-optimal decisions are also p-optimal, because there is no approximation error. Figure reproduced from Paper IV (Kuśmierczyk et al., 2020).

model-agnostic variational algorithms and derive heuristics for maximizing the calibration and optimizing the utility function for the difficult case of incorporating continuous-valued utilities. More particularly, we first discuss techniques for utility transformation in order to maximize its calibration effect. We then specify routines to convert between losses and utilities that are robust to noise and take account of the empirical distribution of the utilities and losses to specify the parameters of the transformation. Finally, we outline a novel double-reparameterization scheme for Monte Carlo estimation of the utility-function, discuss the pros and cons of the Monte Carlo estimators and outline four optimization techniques that are tailored to perform well on specific types of loss functions and settings.

The second approach, called post-hoc correction, is an amalgam of learning theory and Bayesian decision theory that utilizes a characterization of the posterior distribution to minimize the empirical risk and thereby correct the posteriors that were inferred in a loss-agnostic fashion. This characterization, typically provided as quantiles of the posterior, is passed through a strongly regularized decision-maker that learns to predict the optimal decisions. The strong regularization prevents the decision-maker, often a neural network, from overriding the underlying Bayesian model and

ensures that the optimal decisions remain consistent the underlying model. The degree of regularization allows us to set our preference for the type of predictions: a highly regularized decision-maker focuses on optimal decisions being faithful to the Bayesian model even if it comes at the price of increased empirical risk; a decision-maker that is not regularized places its emphasis on minimizing the risk at the cost of model faithfulness. In order to regularize the decision-maker, we draw inspiration from generalized Bayesian inference (Bissiri et al., 2016) to outline a Bayesian framework that allows us to tie the decisions to the model parameters θ via the user-specified loss. This Bayesian setup allows specification of a prior on the decisions h, which we achieve by a combination bootstrapping. The priors on h serve to regularize the decision-maker by setting implicit priors on the parameters of the decision-maker such that the predictions of the decision-maker do not deviate too far from the model prediction. We finally note that it is important for the posterior characterization to be identifiable for the decision-maker to learn. This concludes our discussion of loss-aware Bayesian inference.

Chapter 5

Conclusion

This thesis makes novel contributions for improving automated inference mechanisms and incorporating loss-aware inference techniques for probabilistic programming. We briefly summarize the contributions below for the benefit of the reader:

I. An *importance-sampled estimator* for Monte Carlo gradients that speeds up variational inference by helping reuse the computation-heavy parts of the gradients for several consecutive iterations. We provide pointers on how to use this estimator in the context of stochastic gradient descent and stochastic gradient average descent.

II. The *modified Lambert $W \times F$ distribution* as a differentiable family of distributions to model skewed data as a non-linear transformation of symmetric distributions. The modification proposed extends the support of the Lambert function across the real line and ensures it is invertible, making it suitable for gradient-based optimization.

III. A strategy to automate loss-calibrated variational inference to yield variational approximations that minimize the posterior risk, in the specific context of *unbounded and continuous utilities* and losses. This includes heuristics to estimate a robust upper bound, conversion between losses and utilities and optimization techniques for different loss functions.

IV. A post-hoc loss calibration that borrows the language of classical and Bayesian decision theory for a framework that minimizes the empirical risk by learning a mapping from a *characterization of the posterior approximation* to decisions that minimize the *empirical risk*.

All our contributions can be incorporated into a probabilistic programming workflow, even though we did not attempt to explicitly do so in the original articles. The manner in which the contributions interact with the probabilistic programming framework outlined in Section 1.1 differs. Integrating Contribution I involves modifying the automatic inference engine of the probabilistic program. This can be done by inheriting from and extending inference classes such as the `normal_meanfield` class in `Stan` (Carpenter et al., 2017) or `tfp.monte_carlo.expectation` in Tensorflow Probability (Dillon et al., 2017) to define the importance-sampled estimator. Contribution II can be easily incorporated into most probabilistic programs as a simple transformation. For example, in `Stan`, skewed distributions can be achieved by using a combination of the `function` block to implement the modified Lambert function, the `transformed parameters` block to apply the Lambert transformation and the `model` block to incorporate the Jacobian adjustment. With `Tensorflow Probability`, we can remove the onus on the user to specify the transformation and instead utilize the `tfp.bijector.Bijector` interface to specify the transformation and the Jacobian adjustment to more elegantly define skewed distributions.

Contributions III and IV necessitate a more substantial departure from standard probabilistic programming syntax. This involves ensuring that decision-making and loss specification is adopted as a rigorous part of the Bayesian pipeline. For instance, the program blocks in `Stan` could be extended to accommodate a new block, called, for example, the `target` block, that allows specification of losses and utilities. Such a `target` block would then interact with the inference backend to allow incorporation of losses during inference. In this thesis, we developed a theory for generic, model and loss-agnostic loss calibration. However, more work is needed to bring this into more widespread use and generate interest for adopting the new Bayesian workflow specified in Section 1.1.

We now discuss three new articles that have come out since our original publications and relate to the contributions in this thesis. Tomasetti et al. (2019) propose a novel way of using importance-sampled estimators in the context of updated variational Bayes – a recursive version of variational inference that is useful in online settings. While we apply the estimator across iterations, they adapt it for use across several time steps, using only the data observed since the previous update to quickly obtain estimates of the new approximation. This allows a trade-off between computational speed and accuracy. Target-aware Bayesian inference (Rainforth et al., 2020) outlines a Bayesian framework that uses amortized approximations to directly model the posterior expectations to yield better Monte Carlo estimates.

This differs from loss calibration which aims to only approximate the posterior so as to reduce the expected posterior loss. Generalized variational inference (Knoblauch et al., 2019) extends standard variational inference to setups involving arbitrary likelihoods and divergence measures. Loss calibration can be thought of as a specific instantiation of generalized VI where the loss function is the loss-calibrated objective and the divergence measure is the KL-divergence.

The Bayesian community has begun to recognize that standard methods of Bayesian inference with well-specified likelihoods that are loss-agnostic are inefficient to calculate posterior expectations (Knoblauch et al., 2019). The target-aware Bayesian framework (Rainforth et al., 2020) that exploits knowledge of the target function to better estimate the posterior expectation is a step in the right direction. Given that there has been a steady drift towards using flexible, amortized approximations (Rezende and Mohamed, 2015; Guo et al., 2017; Locatello et al., 2018), we hope there is an ideological shift from accurate modeling of the posteriors to a direct modeling of posterior expectations.

References

Martin Abadi et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016.

Ehsan Abbasnejad, Justin Domke, and Scott Sanner. Loss-calibrated Monte Carlo Action Selection. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

Andrew Gelman. Coronavirus model update: Background, assumptions, and room for improvement. `https://bit.ly/39TDlgR`, 2020. [Online; accessed 16-April-2020].

Atilim Gunes Baydin, Barak A. Pearlmutter, and Alexey Andreyevich Radul. Automatic Differentiation in Machine Learning: a Survey. *arXiv:1502.05767 [cs.SC]*, 2015.

James O. Berger. *Statistical Decision Theory and Bayesian Analysis; 2nd edition.* Springer Series in Statistics. Springer, New York, 1985.

G. P. Bhattacharjee. Algorithm AS 32: The Incomplete Gamma Integral. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 19.3, 1970.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., 2006.

Pier G. Bissiri, Chris C. Holmes, and Stephen G. Walker. A General Framework for Updating Belief Distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5), 2016.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518), 2017.

Léon Bottou and Yann LeCun. Large Scale Online Learning. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

Phelim P. Boyle. Options: A Monte Carlo approach. *Journal of Financial Economics*, 4(3), 1977.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*, 76(1), 2017.

Augustin L. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes rendus des séances de l'Académie des sciences de Paris*, 25, 1847.

Centers for Disease Control and Prevention. Situation summary. `https://bit.ly/3bcHP3r`, 2020. [Online; accessed 16-April-2020].

Centers for Disease Control and Prevention. Interim clinical considerations for use of mrna covid-19 vaccines currently authorized in the united states. `https://bit.ly/3sMfWZ8`, 2021. [Online; accessed 21-Jan-2020].

François Chollet et al. Keras. `https://keras.io`, 2015.

Adam D. Cobb, Stephen J. Roberts, and Yarin Gal. Loss-Calibrated Approximate Inference in Bayesian Neural Networks. In *Theory of Deep Learning workshop, ICML*, 2018.

Robert M. Corless, Gaston H. Gonnet, David E.G. Hare, David J. Jeffrey, and Donald E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5(1), 1993.

David R. Cox and David V. Hinkley. *Theoretical Statistics*. Chapman and Hall, 1979.

Imre Csiszar. Information-Type Measures of Difference of Probability Distributions and Indirect Observations. *Studia Scientiarum Mathematicarum Hungarica*, 2, 1967.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39, 1977.

Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. TensorFlow Distributions. *arXiv:1711.10604 [cs.LG]*, 2017.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Technical report, EECS Department, University of California, Berkeley, 2010.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. CRC press, 1994.

Changyong Feng, Hongyue Wang, Naiji Lu, Tian Chen, Hua He, Ying Lu, and Xin M. Tui. Log-transformation and Its Implications for Data Analysis. *Shanghai Archives of Psychiatry*, 26, 2014.

Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit Reparameterization Gradients. *Advances in Neural Information Processing Systems 31*, 2018.

Mathieu Fourment, Andrew F. Magee, Chris Whidden, Arman Bilge, Frederick A. Matsen IV, and Vladimir N. Minin. 19 Dubious Ways to Compute the Marginal Likelihood of a Phylogenetic Tree Topology. *Systematic Biology*, 69(2), 2019.

Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29, 2000.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, 2016.

Andrew Gelman and Donald B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4), 1992.

Andrew Gelman, Hal S. Stern, John B. Carlin, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2013.

Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 1984.

Zoubin Ghahramani. Probabilistic Machine Learning and Artificial Intelligence. *Nature*, 521, 2015.

Georg M. Goerg. Lambert W random variables – a new family of generalized skewed distributions with applications to risk estimation. *The Annals of Applies Statistics*, 5(3), 2011.

Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: a language for generative models. *arXiv:1206.3255 [cs.PL]*, 2012.

Quentin F. Gronau, Alexandra Sarafoglou, Dora Matzke, Alexander Ly, Udo Boehm, Maarten Marsman, David S. Leslie, Jonathan J. Forster, Eric-Jan Wagenmakers, and Helen Steingroever. A Tutorial on Bridge Sampling. *arXiv:1703.05984 [stat.CO]*, 2017.

Fangjian Guo, Xiangyu Wang, Kai Fan, Tamara Broderick, and David B Dunson. Boosting Variational Inference. *arXiv:1611.05559 [stat.ML]*, 2017.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1), 1970.

Anthony Hauser, Michel J Counotte, Charles C Margossian, Garyfallos Konstantinoudis, Nicola Low, Christian L Althaus, and Julien Riou. Estimation of SARS-CoV-2 mortality during the early stages of an epidemic: a modelling study in Hubei, China and northern Italy. *medRxiv*, 2020.

Horace He. The State of Machine Learning Frameworks in 2019. `https://bit.ly/34QpBTe`, 2019. [Online; accessed 16-April-2020].

Timothy Classen Hesterberg. *Advances in Importance Sampling*. PhD thesis, Stanford University, 1988.

Matthew Hoffman and David M. Blei. Stochastic Structured Variational Inference. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015.

Matthew D. Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15, 2014.

Souad Lajili Jarjir. Size and Book to Market Effects vs. Co-Skewness and Co-Kurtosis in Explaining Stock Returns. *Social Science Research Network*, 2004.

Johns Hopkins University. COVID-19 Dashboard by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU). `https://bit.ly/2XZ60gV`, 2020. [Online; accessed 21-January-2021].

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, 2013.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2), 1999.

Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems 30*, 2017.

Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: a method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014.

Diederik P. Kingma, Tim Salimans, Rafal Józefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational autoencoders with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 2016.

Arto Klami, Jarkko Lagus, and Joseph Sakaya. Lambert Matrix Factorization. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Proceedings, Part II*, volume 11052 of *Lecture Notes in Artificial Intelligence*, pages 311–326. Springer, 2019.

Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. Generalized Variational Inference: Three arguments for deriving new Posteriors. *arXiv:1904.02063 [stat.ML]*, 2019.

Robert Kubinec. A Retrospective Bayesian Model for Measuring Covariate Effects on Observed COVID-19 Test and Case Counts, 2020.

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic Differentiation Variational Inference. *Journal of Machine Learning Research*, 18(14), 2017.

Solomon Kullback and Richard A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22(1), 1951.

Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. Variational Bayesian Decision-making for Continuous Utilities. *arXiv:1902.00792v1 [stat.ML]*, 2019a.

Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. Variational Bayesian Decision-making for Continuous Utilities. *Advances in Neural Information Processing Systems 32*, 2019b.

Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. Correcting Predictions for Approximate Bayesian Inference . In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

Simon Lacoste-Julien, Ferenc Huszár, and Zoubin Ghahramani. Approximate inference for the loss-calibrated Bayesian. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011.

Daniel Lakeland. For the cost of running 96 wells you can test 960 people. `https://bit.ly/39RGfm3`, 2020. [Online; accessed 16-April-2020].

Tuan Anh Le, Atılım Güneş Baydin, and Frank Wood. Inference Compilation and Universal Probabilistic Programming. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521, 2015.

Wonyeol Lee, Hangyeol Yu, and Hongseok Yang. Reparameterization Gradient for Non-differentiable Models. In *Advances in Neural Information Processing Systems 31*, 2018.

Yingzhen Li and Richard E. Turner. Rényi Divergence Variational Inference. In *Advances in Neural Information Processing Systems 29*, 2016.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45, 1989.

Francesco Locatello, Gideon Dresdner, Rajiv Khanna, Isabel Valera, and Gunnar Raetsch. Boosting Black Box Variational Inference. *Advances in Neural Information Processing Systems 31*, 2018.

David MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1991.

David MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3), 1992.

David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2003.

Dougal Maclaurin, David Duvenaud, Matthew Johnson, and Ryan P. Adams. Autograd: Reverse-mode Differentiation of Native Python. `http://github.com/HIPS/autograd`, 2015.

James B. McDonald, Jeff Sorensen, and Patrick A. Turley. Skewness and kurtosis properties of income distribution models. *The Review of Income and Wealth*, 59(2), 2011.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6), 1953.

Thomas P. Minka. Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, 2001.

Carl N. Morris. Natural Exponential Families with Quadratic Variance Functions. *Annals of Statistics*, 10(1), 1982.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020.

Christian A. Naesseth, Francisco J. R. Ruiz, Scott W. Linderman, and David M. Blei. Reparameterization gradients through acceptance-rejection sampling algorithms. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

Radford M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Publishing Company, Incorporated, 1 edition, 2014. ISBN 1461346916.

Manfred Opper and Cedric Archambeau. The variational Gaussian approximation revisited. *Neural computation*, 21(3), 2009.

Kazuki Osawa, Siddharth Swaroop, Mohammad Khan, Anirudh Jain, Runa Eschenhagen, Richard E. Turner, and Rio Yokota. Practical Deep Learning with Bayesian Principles. In *Advances in Neural Information Processing Systems 32*, 2019.

Tom Rainforth, Adam Golinski, Frank Wood, and Sheheryar Zaidi. Target-Aware Bayesian Inference: How to Beat Optimal Conventional Estimators. *Journal of Machine Learning Research*, 21(88), 2020.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black Box Variational Inference. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 2014.

Rajesh Ranganath, Dustin Tran, Jaan Altosaar, and David M. Blei. Operator variational inference. In *Advances in Neural Information Processing Systems*, 2016a.

Rajesh Ranganath, Dustin Tran, and David M. Blei. Hierarchical Variational Models. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016b.

Danilo Rezende and Shakir Mohamed. Variational Inference with Normalizing Flow. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3), 09 1951.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, 2005. ISBN 0387212396.

Geoffrey Roeder, Yuhuai Wu, and David Duvenaud. Sticking the landing: An asymptotically zero-variance gradient estimator for variational inference. *arXiv:1703.09194 [stat.ML]*, 2017.

Francisco Ruiz and Michalis Titsias. A Contrastive Divergence for Combining Variational Inference and MCMC. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Francisco Ruiz, Michalis Titsias, Adji Bousso Dieng, and David M. Blei. Augment and Reduce: Stochastic Inference for Large Categorical Distributions. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Francisco J. R. Ruiz, Michalis K. Titsias, and David M. Blei. The generalized reparameterization gradient. In *Advances in Neural Information Processing Systems*, 2016.

Joseph Sakaya and Arto Klami. Re-using gradient computation in automatic variational inference. *Advances in Approximate Bayesian Inference; NIPS 2016 Workshop*, 2016.

Joseph Sakaya and Arto Klami. Importance Sampled Stochastic Optimization for Variational Inference . *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017*, 2017.

Tim Salimans and David A. Knowles. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4), 2013.

Tim Salimans, Diederik P. Kingma, and Max Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, 2015.

John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science 2:e55*, 2016.

Thomas Schatz, Vijayaditya Peddinti, Xuan-Nga Cao, Francis R. Bach, Hynek Hermansky, and Emmanuel Dupoux. Evaluating speech features with the minimal-pair ABX task (II): resistance to noise. In *Annual Conference of the International Speech Communication Association*, 2014.

Tom Schaul and Yann LeCun. Adaptive learning rates and parallelization for stochastic, sparse, non-smooth gradients. In *Proceedings of the 1st International Conference on Learning Representations*, 2013.

Mark W. Schmidt, Nicolas Le Roux, and Francis R. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162, 2017.

Daniel Seita, Xinlei Pan, Haoyu Chen, and John F. Canny. An Efficient Minibatch Acceptance Test for Metropolis-Hastings. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017*, 2017.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional Variational Bayesian Neural Networks . *arXiv:1903.05779 [cs.LG]*, 2019.

Yee W Teh, David Newman, and Max Welling. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems 19*, 2007.

Michalis Titsias. One-vs-Each Approximation to Softmax for Scalable Estimation of Probabilities. In *Advances in Neural Information Processing Systems*, 2016.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly Stochastic Variational Bayes for non-Conjugate Inference. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Nathaniel Tomasetti, Catherine Forbes, and Anastasios Panagiotelis. Updating Variational Bayes: Fast sequential posterior inference. *arXiv:1908.00225 [stat.CO]*, 2019.

Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv:1610.09787 [stat.CO]*, 2016.

Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An Introduction to Probabilistic Programming . *arXiv:1809.10756*, 2018.

Aki Vehtari, Daniel Simpson, Andrew Gelman, Yuling Yao, and Jonah Gabry. Pareto Smoothed Importance Sampling. *arXiv:1507.02646 [stat.CO]*, 2015.

Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An

improved $\widehat{R}$ for assessing convergence of MCMC. *arXiv:1903.08008 [stat.CO]*, 2019.

Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011.

Andrew Gordon Wilson. The Case for Bayesian Deep Learning. *arXiv:2001.10995 [cs.LG]*, 2020.

World Health Organization. Coronavirus disease 2019 (COVID-19): situation report, 86. Technical documents, World Health Organization, 2020.

World Health Organization. COVID-19 Weekly Epidemiological Update. Technical documents, World Health Organization, 2021.

Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Yes, but Did It Work?: Evaluating Variational Inference. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 2018.

Cheng Zhang, Judith Butepage, Hedvig Kjellström, and Stephan Mandt. Advances in Variational Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8), 2019.