UNIVERSITY OF CALIFORNIA,
IRVINE

On Priors for Bayesian Neural Networks

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Eric Thomas Nalisnick

DISSERTATION Committee:
Professor Padhraic Smyth, Chair
Chancellor's Professor Pierre Baldi
Associate Professor Babak Shahbaba

2018

DEDICATION

To my parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This dissertation would not have been possible without the support and encouragement of my advisor, Padhraic Smyth. Padhraic gave me a tremendous amount of intellectual freedom throughout my time at UCI. Whenever I brought him my latest half-baked idea (e.g. stick-breaking autoencoders), he listened intently and sharpened my thinking through our discussions. In this way, Padhraic taught me a good deal about communicating science and translating my unruly thoughts into coherent whiteboard arguments. Moreover, no matter where in the world a conference was being held, Padhraic always encouraged me to submit my work, saying he would find a way for me to attend upon acceptance—and he always did.

I also thank the rest of my committee: Pierre Baldi and Babak Shahbaba. Each suggested essential improvements to this dissertation in addition to providing helpful feedback at my public defense. I especially thank Babak for allowing me to attend his geometry, math, and statistics seminar. The discussions with attendees Hongkai Zhao, Kyle Cranmer, Jeff Streets, Alex Vandenberg-Rodes, Andrew Holbrook, and others were immensely influential and formative early in my graduate studies. I also am grateful to Anima Anandkumar, who served as a second advisor during my second and third years in the program and introduced to me interesting topics such as score matching and the continuation method. Past and present members of the Smyth DataLab have also been essential to making my time at UCI productive as well as enjoyable: Jimmy Foulds, Nick Navaroli, Kevin Bache, Moshe Lichman, Zach Butler, Homer Strong, Chris Galbraith, Dimitrios Kotzias, Jihyun Park, Disi Ji, Efi Karra Taniskidou, Casey Graff, and Robby Logan. I have also learned much from the mentors, collaborators, and friends with whom I have crossed paths during my stints in industry: Rich Caruana, Nick Craswell, Bhaskar Mitra, Vijai Mohan, Eiman Elnahrawy, Mihir Bhanot, Madhu Kurup, Eric Reel, Hugo Larochelle, Sachin Ravi, Harm De Vries, Ilija Bogunovic, Jake Snell, Mengye Ren, and others.

On a personal note: My time at UCI would not have been as successful without the lasting friendships I have made here. My knowledge of statistics and machine learning was expanded (in addition to my waistline) through weekly lunches with Brian Vegetabile and Andrew Holbrook. Regular pub visits with Brian, Andrew, David Stenning, Chris, Lars Hertel, Garren Gaut, Maricela Cruz, Micah Jackson, Lingge Li, and others provided some much needed liquid therapy. And last but not least, Dimitrios and Dylan Conroy dragged me away from my desk and on many fun adventures throughout southern California.

The work in this dissertation was supported by the National Science Foundation under awards IIS-1320527, NRT-1633631, CNS-1730158, and DGE-1633631 and by the National Institutes of Health under award NIH-1U01TR001801.

CURRICULUM VITAE

Eric Thomas Nalisnick

EDUCATION

Doctor of Philosophy in Computer Science — **2018**
University of California, Irvine — *Irvine, California*

Master of Science in Computer Science — **2013**
Lehigh University — *Bethlehem, Pennsylvania*

Bachelor of Science in Computer Science and English Lit. — **2012**
Lehigh University — *Bethlehem, Pennsylvania*

RESEARCH EXPERIENCE

Graduate Research Assistant — **2013–2018**
UCI Smyth DataLab — *Irvine, California*

Applied Scientist Intern — **Fall 2016**
Amazon — *Irvine, California*

Research Intern — **Summer 2016**
Twitter — *Cambridge, Massachusetts*

Research Intern — **Summer 2015**
Microsoft — *Redmond, Washington*

Research Scientist Intern — **Summer 2014**
Amazon — *Seattle, Washington*

REFEREED CONFERENCE PUBLICATIONS

E. Nalisnick and P. Smyth. **Learning Priors for Invariance**. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics* (AISTATS), 2018.

E. Nalisnick and P. Smyth. **Learning Approximately Objective Priors**. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence* (UAI), 2017.

E. Nalisnick and P. Smyth. **Stick-Breaking Variational Autoencoders**. In *Proceedings of the 5th International Conference on Learning Representations* (ICLR), 2017.

E. Nalisnick, B. Mitra, N. Craswell, and R. Caruana. **Improving Document Ranking with Dual Word Embeddings**. In *Proceedings of the 25th World Wide Web Conference* (WWW), 2016.

E. Nalisnick and H. Baird. **Character-to-Character Sentiment Analysis in Shakespeare's Plays**. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (ACL), 2013.

E. Nalisnick and H. Baird. **Extracting Sentiment Networks from Shakespeare's Plays**. In *Proceedings of the 12th International Conference on Document Analysis and Recognition* (ICDAR), 2013.

TEACHING EXPERIENCE

Teaching Assistant — **University of California, Irvine**
CS175: Projects in Artificial Intelligence — *Winter 2018*

Reader — **University of California, Irvine**
CS274A: Probabilistic Learning — *Winter 2017*

ACADEMIC REVIEWING

Conference on Neural Information Processing Systems (NIPS) — **2016–2018**
Best Reviewer Award (2017)
International Conference on Machine Learning (ICML) — **2018**
International Conference on Learning Representations (ICLR) — **2018**
Machine Learning — **2017**
Data Mining and Knowledge Discovery — **2017**

ABSTRACT OF THE DISSERTATION

On Priors for Bayesian Neural Networks

By

Eric Thomas Nalisnick

Doctor of Philosophy in Computer Science

University of California, Irvine, 2018

Professor Padhraic Smyth, Chair

Deep neural networks have bested notable benchmarks across computer vision, reinforcement learning, speech recognition, and natural language processing. However, neural networks still have deficiencies. For instance, they have a penchant to over-fit, and large data sets and careful regularization are needed to combat this tendency. Using neural networks within the Bayesian framework has the potential to ameliorate or even solve these problems. Shrinkage-inducing priors can be used to regularize the network, for example. Moreover, test set evaluation is done by integrating out uncertainty and using the posterior predictive distribution. Marginalizing the model parameters in this way is not only a natural regularization mechanism [MacKay, 1992a] but also enables uncertainty quantification—which is increasingly important as machine learning is deployed in ever more consequential applications.

Bayesian inference is characterized by specification of the prior distribution, and unfortunately, choosing priors for neural networks is difficult [Buntine and Weigend, 1991]. The primary obstacle is that the weights have no intuitive interpretation and seemingly sensible priors can induce unintended artifacts in the distribution on output functions. This dissertation aims to help the reader navigate the landscape of neural network priors. I first survey the existing work on priors for neural networks, isolating some key themes such as the move towards heavy-tailed priors [Neal, 1994]. I then describe my own work on broadening the class of priors applicable to

Bayesian neural networks. I show that introducing multiplicative noise to the hidden layer computation induces a Gaussian scale mixture prior, suggesting links between *dropout regularization* [Srivastava et al., 2014] and previous work on heavy-tailed priors. I then turn towards priors with frequentist properties. *Reference priors* [Bernardo, 1979] cannot be analytically derived for neural networks so I propose an algorithm to approximate them. Similarly, it is hard to derive priors that make the model invariant under certain input transformations. To make progress, I use an algorithm inspired by my work on objective priors to learn a prior that makes the model approximately invariant. Lastly, I describe how to give Bayesian neural networks an adaptive width by placing *stick-breaking priors* [Ishwaran and James, 2001] on their latent representation. I end the dissertation with a discussion of open problems, such as incorporating structure into priors while still maintaining efficient inference.

Chapter 1

Introduction

> *To call in the statistician after the experiment is done may be no more than asking him to perform a post-mortem examination: he may be able to say what the experiment died of.*
>
> R. A. Fisher

Imagine you are a scout for a major league baseball franchise and are tasked with finding the next Ted Williams.[1] For any player you encounter, you ask the natural question: *what is the chance he will register a 'hit' in any given at-bat?* This chance is known within baseball as a player's *batting average*: a real number between zero and one (inclusive) representing the chance–or more precisely, probability—that the player will score a hit. Of course, it is impossible to know a player's innate batting average, but we can take steps toward an answer through making observations, by tracking the results of his at-bats. The hope is that if we track the player's batting results for long enough, we can eventually make a good estimate of his true batting

[1] Theodore Samuel Williams (August 30, 1918 – July 5, 2002) was the last major league baseball player to record a batting average of $.400$ or higher over a season, which he accomplished in 1941.

potential (assuming the player's skill and difficulty of the opposing pitchers remain constant).

These batting observations can be thought of as data: an at-bat results in a realization of a random variable $X \in \{0, 1\}$. Say we observe N total at-bats for a given player, in turn producing a data set $\mathbf{x} = \{x_1, \ldots, x_i, \ldots, x_N\}$ with each x_i being a realization of X, equal to one if the player achieved a hit in the ith at-bat and zero if not. The total number of hits is $\sum_{i=1}^{N} x_i = k$. A probabilistic model for binary data of this type is the *binomial distribution*. The probability of a data set can be computed under the model as follows:

$$p(\mathbf{x}; \pi) = \text{Binomial}(k; N, \pi) = \binom{N}{k} \pi^k (1 - \pi)^{N-k}$$

where $\pi \in [0, 1]$ is known as a model parameter and, in the case of the binomial applied to batting results, represents the probability of a hit—or in other words, the player's batting average. In fact, if we assume this posited binomial model is indeed a useful model of how hits are generated, then batting results for the player can be thought of as draws from the model, i.e. $\hat{x} \sim \text{Binomial}(1, \pi^*)$ with π^* being the player's true hit probability. Making these modeling assumptions has turned the problem of player evaluation into one of *statistical inference*: *what is the value of a given player's Bernoulli parameter π^*?* If we can somehow obtain π^* or closely approximate it, then we know the player's innate batting average, which then allows us to evaluate him as a prospect.

A *frequentist* approach to inference relies on two fundamental principles: repeated observation and maximum likelihood estimation. In our example of player evaluation, a frequentist scout would collect as large of a data set as possible. Then, given the (hopefully large) data set $\mathbf{x}$, maximum likelihood estimation finds the parameter estimate $\hat{\pi}$ that results in the data having its highest probability under the model. Moreover, as $N \to \infty$, maximum likelihood estimation recovers the true parameter (under assumptions) [Casella and Berger, 2002]. Solving the simple maximization problem for the binomial model yields the formula $\hat{\pi}_{\text{MLE}} = \frac{k}{N}$. The result is intuitively satisfying as it is simply the number of hits divided by the number of at-bats. Readers

familiar with how major league baseball (as well as most other levels of play) reports statistics will notice that official batting averages are obtained via the frequentist paradigm, i.e. maximum likelihood estimates of the Bernoulli parameter. If the frequentist scout does not feel a sufficiently large data set has been collected as to trust the maximum likelihood estimate, uncertainty in the estimate can quantified via a confidence interval, which represents how widely $\hat{\pi}$ might vary under alternative data sets of the same size.

While the frequentist approach is undoubtedly elegant in its simplicity, its reliance on an increasing amount of data becoming available is unrealistic in some cases. Using $\hat{\pi}_{\mathsf{MLE}}$ to quantify a professional baseball player's hitting ability is not problematic because professionals play over 160 games and usually bat at least three times per game, but consider a scout attempting to evaluate amateur players by observing their performance in college or high school games. These players will take only a fraction of the number of at-bats a professional does. In that case, it may seem as if statistical inference is doomed from the outset for how can we glean knowledge from data without having much of it?

1.1 Bayesian Inference

Bayesian inference is an alternative to the frequentist paradigm, and it assumes the model parameters are fundamentally random. Instead of obtaining a point estimate $\hat{\pi}$ and controlling for how it varies under data resampling, Bayesians' fundamental quantity of interest is a probability distribution in its own right. Continuing with the baseball scout example, given the binomial model $p(\mathbf{x}|\pi)$[2] and a *prior* distribution $p(\pi)$, the scout uses Bayes' theorem [Bayes, 1763] to compute

$$p(\pi|\mathbf{x}) = \frac{p(\mathbf{x}|\pi)p(\pi)}{p(\mathbf{x})} \tag{1.1}$$

[2]I have changed the notation from $p(\mathbf{x};\pi)$ to $p(\mathbf{x}|\pi)$ to emphasize that π is now considered a random variable.

where $p(\mathbf{x}) = \int_\pi p(\mathbf{x}|\pi)p(\pi)d\pi$, which is known as the model evidence, and $p(\pi|\mathbf{x})$ is the conditional distribution of the parameter given the data and is known as the Bayesian *posterior*.

Before turning discussion towards the posterior, let us first consider the prior. Before any player data is observed, the Bayesian scout must specify $p(\pi)$, and doing so is "the most important step" in the Bayesian approach in that it can "drastically alter the subsequent inference" [Robert, 2001]. At first glance, Bayesian inference may seem non-rigorous for why would we want to let the modeler—with all of her human bias and subjectivity—exert such a degree of control over the parameter estimates? The criticism of using 'personalistic probability' [Savage, 1972, Jaynes, 1968] has been slung against Bayesians since the infancy of statistical inference [Fisher, 1922]. Yet, as Jaynes [1968] elegantly clarifies, priors, when specified properly, are not 'personalistic' in the sense that they contain information available only to one person. Rather, priors should be based on information that is scientifically testable and therefore available to anyone who may perform the appropriate experiment [Jaynes, 1968]. In this view, priors are perhaps no more subjective than the data model, which needs to be specified no matter the inference philosophy.

Returning to the problem posed above—how can we evaluate players without many batting observations?—the Bayesian scout can still hope to make accurate inferences by specifying a useful prior. If the player has batting performance data available for a previous season, that information could serve as a reasonable prior. Or for another example, perhaps out-of-game information such as the speed with which he swings the bat or the distance his hits travel during practice has been recorded. A player with a high bat speed and who hits many home runs during practice should likely be given a prior that favors a higher estimate of π. Note that in the these examples, the information on which the prior is based is all testable.

Now with $p(\pi)$ chosen, the Bayesian scout is free to collect data, again observing N at-bats $\mathbf{x} = \{x_1, \ldots, x_N\}$, and then compute the posterior $p(\pi|\mathbf{x})$. Assume that the prior has been

chosen to be the *Beta distribution*, which has density function

$$p(\pi; \alpha_0, \beta_0) = \frac{\pi^{\alpha_0 - 1}(1-\pi)^{\beta_0 - 1}}{\mathsf{B}(\alpha_0, \beta_0)}$$

where $\alpha_0, \beta_0 > 0$ are the parameters (chosen by the modeler during prior specification) and $\mathsf{B}(\cdot, \cdot)$ is the Beta function. For the Bernoulli model, the Beta distribution is said to be its *conjugate prior* [Casella and Berger, 2002] since the posterior distribution is again a Beta:

$$p(\pi|\mathbf{x}) = \mathsf{Beta}\left(\pi; \alpha = k + \alpha_0, \beta = (N-k) + \beta_0\right) = \frac{\pi^{k+\alpha_0-1}(1-\pi)^{(N-k)+\beta_0-1}}{\mathsf{B}\left(k + \alpha_0, (N-k) + \beta_0\right)}. \tag{1.2}$$

Notice how the prior's parameters interact with the data via addition. This simple posterior form enables us to think of α_0 and β_0 as *pseudo-counts* (or *pseudo-data*) contributing to the number of successes (k) and total number of at-bats (N) respectively. Knowing that the posterior will have this intuitive form enables the prior to be specified even more precisely since the scout can reason about the prior's parameters in terms of the data. For instance, if the prior is set according to the player's bat speed, then the scout can be explicit in how many extra hits (α_0) or failures (β_0) any given speed is worth. Having a concrete translation between physical or observed quantities and parameters further elucidates prior assumptions, allowing them to be reformulated if shown over time to be misguided. From this perspective we can see that priors provide nothing for free and are perhaps better thought of as a mechanism through which already observed data and the corresponding inferences can be brought mathematically to bear in a new inferential task.

Once it is time for the Bayesian scout to report her evaluation of the player, point estimates such as the posterior mean—which for the Beta is $\hat{\pi}_{\mathsf{MPE}} = (k + \alpha_0)/(N + \beta_0 + \alpha_0)$—or the posterior mode—which is $\hat{\pi}_{\mathsf{MAP}} = (k + \alpha_0 - 1)/(\alpha_0 + N + \beta_0 - 2)$—can be computed. If the scout wishes to incorporate some notion of uncertainty, then she can also report a posterior credible interval. Unlike confidence intervals, credible intervals have the intuitive interpretation that the parameter has the specified level of probability of falling within the range. Lastly, a satisfying

aspect of the Bayesian methodology is that it naturally lends itself to continual inference. If the scout is tasked with re-evaluating a player, such as in the subsequent season, then the current posterior can serve as the future prior and the whole framework remains coherent.

1.2 Priors for Prediction

For much of the discipline's history, statistics has concerned itself with asking inferential questions like the one posed in the previous section. There is some random variable that we wish to hone in upon by observing data generated as a function of that same variable. In the example of a baseball scout, the random variable is the batting average (π), the data is the batting results, and the model is the binomial distribution. Once the quantity has been estimated to a satisfying degree of accuracy and precision, then the task is essentially complete, and decisions can be made based on the outcome. The scout reports the estimated batting average to the franchise and the leadership decides whether to sign the player to a contract or not. Situations such as determining the effectiveness of a pharmaceutical drug or validating the outcome of a scientific experiment would transpire similarly.

However, there are alternative forms of probabilistic reasoning. The one this dissertation is primarily concerned with is model-based *prediction*. Instead of seeking to determine the value of a particular model parameter, prediction's goal is to estimate the parameters of the model so that the model's output itself is useful. I will demonstrate the difference via another example from baseball. Consider a data set that contains the coordinates of a player's hits, and denote this data set as $\mathbf{X} = \{\mathbf{x}_i = (x_i, y_i)\}_{i=1}^{N}$ whereby $\mathbf{x}_i = (x_i, y_i)$ are the $x - y$ coordinates of the ith hit and N is the total hits observed. If a team's statistician has such a data set for opposing players, she can build a model to predict where a given player may hit the ball. The team's coach can then use the model to position his players in the field so they may make a catch[3]. A *Gaussian*

[3]If a batted ball is caught by an opposing player before it touches the ground, then it is counted as an 'out.' The

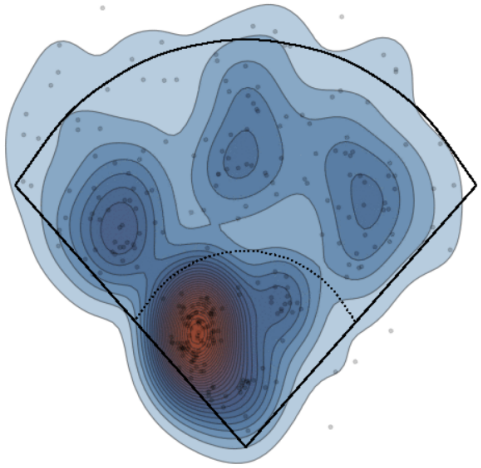


Figure 1.1: *Gaussian Mixture Model Fit to Hit Locations.* Graphic made with *baseballr* [Petti, 2016].

mixture model is one model that can make such predictions; it has the form

$$p\left(\mathbf{x}; \boldsymbol{\pi}, \{\boldsymbol{\mu}_k\}_{k=1}^K, \{\boldsymbol{\Sigma}_k\}_{k=1}^K\right) = \sum_{k=1}^{K} \pi_k \text{ Normal}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{1.3}$$

where $\boldsymbol{\pi}$ is a K-dimensional vector of mixture weights that sum to one, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean (row) vector and covariance matrix of a Gaussian (a.k.a. normal) distribution, and k indexes the K components that form the mixture. The Gaussian densities have the functional form: $\text{Normal}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \exp\left\{\frac{-1}{2}(\mathbf{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})^T\right\} / \sqrt{|2\pi\boldsymbol{\Sigma}|}$. After performing maximum likelihood estimation to find the weights, means, and covariances, the model's output can be visualized by plugging every position on the baseball field into Equation 1.3. Doing this with an example data set yields the density shown in Figure 1.1. The utility of Figure 1.1 is immediate, even from a cursory glance. The red shading—the area of highest probability density—clearly shows that the player's hits favor the left side of the field. Thus, the coach should signal to the third baseman to expect a hit his way and perhaps should even have the other players take a few steps to their right.

The crucial difference between the batting average example and the hit density example above is that there is no specific parameter of interest in the latter model. Of course, we need the

offensive player does not reach base and leaves the field of play.

parameter estimation procedure—in the example above, maximum likelihood—to work well in order to build a model faithful to the data, but in no way are we concerned with recovering some true mean or covariance. Rather, we simply want the aggregate density, which is formed by all parameters in concert, to reflect the batter's future behavior. To rephrase the distinction, a coach has a clear interpretation of what the batting average estimate $\hat{\pi}$ means in the absence of the binomial model. Yet, she has no immediate use for, say $\hat{\boldsymbol{\mu}}_{k=2}$, the mean of the second component. It is only the model output $p\left(\mathbf{x}; \hat{\boldsymbol{\pi}}, \{\hat{\boldsymbol{\mu}}_k\}_{k=1}^K, \{\hat{\boldsymbol{\Sigma}}_k\}_{k=1}^K\right)$ that is useful.

Another way to think about the difference between prediction and inference is of the former subsuming the later. The goal of *artificial intelligence* (AI) is to build autonomous systems that can perform useful tasks without any human supervision. And if one assumes these decisions should be based on data, then a well-performing AI must encapsulate both the role of the statistician—building models, performing inference—and the person making the decision based on the inference [Ghahramani, 2008]. In the example of deciding whether to give a player a contract or not, a franchise could build an AI program to make that decision itself, giving it the raw batting data as input. The AI ostensibly would need to perform inference for the Bernoulli parameter on its own as a sub-task, which is the role of the statistician / scout, and then generate the final decision, the role of the franchise leadership. Robust, general AI is far off, but I include the above argument to show that inference and prediction are not necessarily disparate.

Thinking back to the Bayesian paradigm for inference, one may wonder how useful it is for prediction tasks. If we do not have a strong concern for the values of the model parameters, how can we set the prior distribution? For the Gaussian mixture model, it is possible to incorporate some considerations about the player for which we wish to build the model. For instance, if the player has a large and powerful physique, the statistician make place priors on the means such that they are located closer to the outfield fence than would be done for, say, a player known for his smaller stature and quickness. However, the Gaussian mixture posterior does not have the additive interactions between the prior and data that the beta-Bernoulli model does and

therefore priors may behave in unexpected ways.

Setting priors for the Gaussian mixture is relatively easy compared to the the models with which this dissertation is concerned: *neural networks* [Hebb, 1949, McCulloch and Pitts, 1943, Rosenblatt, 1958, Hopfield, 1982, Hinton, 1986, LeCun et al., 1998, Goodfellow et al., 2016]. Neural networks are hierarchical models that transform some input by composing several pairs of linear transformations followed by element-wise non-linear functions. In a Bayesian treatment, the priors are (usually) placed on the parameters of the linear transformation. Setting priors for these parameters is an especially challenging task that is only magnified at the deeper levels of composition. The immediate answer to this problem is that we should not be thinking about priors in terms of the distribution they induce on parameters but instead as the distribution they induce on *output functions*. Since output is the model artifact the user truly cares about, then the priors should be specified in that context. Of course, such a statement is easier written than done well. Over the course of this dissertation, I will discuss a range of priors for neural networks, including ones that are *objective*, *subjective*, and *nonparametric*.

1.3 Preliminaries

Before starting the dissertation's technical material, I give its outline and highlight the primary research contributions. I then introduce the notation used throughout the dissertation.

1.3.1 Outline and Contributions

This dissertation is organized into a background chapter (2), a chapter surveying the work on neural network priors (3), four chapters of original work (4, 5, 6, 7), and a concluding chapter (8). The background chapter briefly reviews neural networks in their supervised and autoencoder formulations before discussing Bayesian versions of each model. Posterior inference by way

of both Markov chain Monte Carlo and variational inference is also covered. Since there are many well-written textbooks and tutorials on this foundational material, my treatment aims to be high-level and to draw connections between concepts. Detailed explanations can be found in the relevant citations.

Moving on to Chapter 3, the dissertation surveys other work on priors for neural networks (again, for both supervised and unsupervised models). The dissertations of MacKay [1992c] and Neal [1994] established much of the research bedrock underlying Bayesian neural networks, and many of their contributions are still relevant today. I summarize their ideas and then discuss how they have been built upon in the deep learning era. This chapter is the most current survey of neural network priors of which I am aware. In fact, Robinson [2001] and Lee [2004] are the only previous surveys I could find, and they are now considerably out of date.

I begin discussing my own work in Chapter 4, starting with an analysis of multiplicative noise in neural networks [Nalisnick et al., 2015]. I show that regularizing a network with multiplicative noise—as is done with *dropout* [Srivastava et al., 2014]—is equivalent to placing a Gaussian scale mixture prior on the parameters and integrating out the scale with a Monte Carlo approximation. I then derive a variational EM update for the noise / scale parameter, revealing its shrinkage mechanism. Interestingly, the derivation shows that the mechanics of noise regularization conflict with the commonly used signal-to-noise weight pruning heuristic. I then reformulate the heuristic so that it is in agreement with noise-based regularization. Experiments validate the analysis, showing that my proposed heuristic outperforms the signal-to-noise ratio in several pruning tasks.

In the subsequent chapter, I turn to my work on approximating objective Bayesian priors [Nalisnick and Smyth, 2017b]. Objective priors are said to be noninformative in the sense that they result in posteriors with frequentist properties—for example, credible intervals nearly match confidence intervals [Irony and Singpurwalla, 1997]. These priors, however, are intractable to compute for neural networks. To bypass the obstacle, I derive an algorithm for learning reference prior

approximations for a wide class of models that includes neural networks. In the experiments, I show the proposed approach is superior to two previously proposed methods for approximating reference priors and use it to reveal the variational autoencoder's objective prior.

One notable hole in the previous work on Bayesian neural networks is the lack of subjective priors. Inspired by the above work on reference priors, I fill in this gap for a subclass of *invariant* priors [Nalisnick and Smyth, 2018]. The method measures how the model output changes under input perturbations and optimizes the prior so that the model output is left invariant. In supervised settings, this method does not depend on the labels being present and thus is amenable to semi-supervised learning. The prior can be optimized on the unlabeled data and then used in the fully-supervised model as usual. Experiments show that learning invariant priors improves model performance, allowing Bayesian methods to become competitive to non-Bayesian approaches.

The final original work contained in the dissertation is described in Chapter 7. I propose placing Bayesian nonparametric priors on the latent variables of unsupervised Bayesian neural networks [Nalisnick et al., 2016, Nalisnick and Smyth, 2017c]. This allows their width to become adaptive and enables their latent space to inflate its dimensionality as the data necessitates. I show two variants of this model—one with just stick-breaking latent variables and another with a Dirichlet process mixture latent space. Experiments confirm these priors allow for richer latent representations: we see factors of variation divide into different subspaces or regions of latent space (depending on the model version).

Finally, chapter 8 concludes the dissertation. I discuss several directions for future research, concentrating particularly on structured and discrete priors that do not admit analytic differentiation. Such priors are especially challenging to apply to neural networks since they do not admit backpropagation, but they have the potential to make neural network computation more efficient and to extend Bayesian reasoning to the level of network architectures.

1.3.2 Notation

I use the following notation throughout the dissertation except where noted otherwise. Matrices are denoted with upper-case and bold letters (e.g. $\mathbf{X}$), vectors with lower-case and bold (e.g. $\mathbf{x}$), and scalars with lower-case and no bolding (e.g. x). Data are represented as row vectors $\mathbf{x} \in \mathcal{X}$ with $\mathcal{X}$ representing the underlying population. I assume N independently and identically distributed draws from $\mathcal{X}$ are observed, and these draws constitute the empirical data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. When $\mathbf{X}$ is the variable I wish to model, I define the likelihood function to be $p(\mathbf{x}|\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ are the model parameters taking values in some space $\boldsymbol{\Theta}$. Furthermore, let the data set likelihood be denoted $p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{\theta})$ with the factorization following from the i.i.d. assumption. In the case of prediction or supervised learning tasks in which $\mathbf{X}$ are covariates (features) that are predictive of another variable $\mathbf{y} = \{y_1, \ldots, y_N\}$, the likelihood function is a conditional model of the form $p(y|\mathbf{x}, \boldsymbol{\theta})$, and the data set likelihood is $\prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$. When a distribution is conditioned on a random variable, I write it as $p(\mathbf{X}|\boldsymbol{\theta})$, but when those parameter are considered fixed or a hyperparameter, I denote the distribution as $p(\mathbf{X}; \boldsymbol{\theta})$. I overload the notation for Bayesian inference, using the semi-colon to demarcate variables without priors. For example, writing $p(\mathbf{X}|\boldsymbol{\theta}; \boldsymbol{\alpha})$ means that the reader should expect there to be a prior $p(\boldsymbol{\theta})$ placed on $\boldsymbol{\theta}$. On the other hand, $\boldsymbol{\alpha}$ does not have a prior and will be given a point estimate when performing any posterior inference.

Chapter 2

Bayesian Neural Networks

> `Radford Neal:` *I don't necessarily think that the Bayesian method is the best thing to do in all cases...*
> `Geoff Hinton:` *Sorry Radford, my prior probability for you saying this is zero, so I couldn't hear what you said.*
>
> An exchange at the 2004 CIFAR workshop

When performing prediction, the goal of model building is to obtain parameter estimates that extract generalizing patterns from the data. To make the model as accurate as possible, it is natural to want to make it as 'big' as possible. 'Big,' in this sense, means it has the capacity to represent a wide range of predictive functions. *Neural networks* are one variety of high-capacity model that have achieved notable success as of late. The family of neural-network-based techniques known under the umbrella term *deep learning* [LeCun et al., 2015, Goodfellow et al., 2016, Deng and Yu, 2014, Bengio et al., 2013a, Schmidhuber, 2015] has made clear empirical progress on certain tasks in the subfields of computer vision [Krizhevsky et al., 2012], speech processing [Dahl et al., 2012],

natural language processing [Manning, 2016], and reinforcement learning [Mnih et al., 2015]. In this chapter, I define neural networks in both their supervised and unsupervised variants. I then motivate and introduce their Bayesian formulation, establishing the necessary background for the dissertation's eventual discussion of various classes of priors.

2.1 Neural Networks

I introduce neural networks by starting with linear models in order to demonstrate how the former can be composed from the latter. Assume we have data of the form $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with $\mathbf{x}_i$ being a row vector of covariates predictive of the corresponding label or response y_i. Recall the definition of a *generalized linear model* (GLM) [Nelder and Baker, 1972], which parametrizes the expected value of the dependent variable by way of a transformed inner product between data and parameters:

$$\mathbb{E}[y_i|\mathbf{x}_i] = g^{-1}(\mathbf{x}_i\mathbf{w} + b) \tag{2.1}$$

where $\mathbb{E}[y|\mathbf{x}]$ denotes the conditional expectation of y, $g : \mu \mapsto \mathbb{R}$ is the *link function*, $\mathbf{w} \in \mathbb{R}^d$ is a column vector of real-valued parameters, and $b \in \mathbb{R}$ is a scalar bias variable. Crucially, the link function maps the mean of the distribution to the real line, and its specification depends on the distributional assumptions placed on y. For instance, if we assume $y \sim$ Poisson$(\lambda = g^{-1}(\mathbf{x}\mathbf{w} + b))$, then g^{-1} is the exponential function, i.e. $\lambda = \exp(\mathbf{x}\mathbf{w} + b)$.

A common problem is that while $\mathbf{x}$ may contain information about its corresponding y, that information is not laid bare in the original feature set—or at least not exposed enough to be extracted by a linear function. One solution is to define a new feature vector based on a transformation, i.e. $\tilde{\mathbf{x}} = h(\mathbf{x})$ with $h(\cdot)$ denoting the transformation function. A polynomial expansion is one simple way to define $h(\cdot)$; if $\mathbf{x}$ is assumed two-dimensional, then $h(\mathbf{x}) = (x_1, x_2, x_1^2, x_1x_2, x_2^2)$.

Yet, choosing a specific feature transformation can be time consuming, and one workaround is to parametrize the transformation, i.e. $\tilde{\mathbf{x}} = h(\mathbf{x}; \boldsymbol{\phi})$ with $\boldsymbol{\phi}$ being the parameters. Combining this formulation with the GLM definition, we have what is known as *adaptive basis function regression*:

$$\mathbb{E}[y_i|\mathbf{x}_i] = g^{-1}(h(\mathbf{x}_i; \boldsymbol{\phi})\mathbf{w} + b). \tag{2.2}$$

This model is termed *adaptive* since the model itself can choose how to transform the features (up to the expressivity of $h(\cdot)$). As the model's choice is guided by the optimization objective, the transformation should be one that enables a good fit of the conditional model. While passing the burden of feature design from modeler to model is sensible and attractive, it comes with a trade-off in that the new feature representation may not be interpretable to the user. All of its ties to the original semantic designations and physical quantities may be lost. Yet, as our primarily goal is optimal predictive accuracy, many modelers are willing to sacrifice interpretability if the predictive gains are worthwhile. The successes of deep learning have been won by taking this bet.

2.1.1 Conditional Models

Neural networks (NNs) are simply adaptive basis function regressors with the basis function being a series of stacked GLMs. To demonstrate, a one-hidden-layer NN is defined as follows:

$$\mathbb{E}[y_i|\mathbf{x}_i] = g^{-1}(h(\mathbf{x}_i; \mathbf{W}_1, \mathbf{b}_1)\mathbf{w}_2 + b_2), \quad h(\mathbf{x}_i; \mathbf{W}_1, \mathbf{b}_1) = f(\mathbf{x}_i\mathbf{W}_1 + \mathbf{b}_1) \tag{2.3}$$

where the equality on the LHS is the same as Equation 2.2 but with the basis function parameters made explicit, i.e. $\boldsymbol{\phi} = \{\mathbf{W}_1, \mathbf{b}_1\}$. The equation on the RHS is the feature transformation; it takes the form of our previous GLM definition except that it produces a vector output. Accordingly, the parameters $\mathbf{W}_1 \in \mathbb{R}^{d \times d'}$ and $\mathbf{b}_1 \in \mathbb{R}^{d'}$ are a matrix and a vector respectively, and they map from

the original feature space's dimensionality d to a new dimensionality d'. The function $f(\cdot)$ acts element-wise and is known as the *activation* function. Sometimes $f(\cdot)$ is chosen to be a particular link function—the logistic function used to be a common choice—but it is important to note that, unlike $g(\cdot)$, $f(\cdot)$'s form is in no way dictated by y's support. Instead, $f(\cdot)$ is essentially free for the modeler to choose. Yet, in order to perform gradient-based optimization, $f(\cdot)$ is all but always chosen to be differentiable and non-saturating (at least in one direction). The *rectified linear unit* (ReLU) defined as $f(z) = \max(z, 0)$ is a popular choice [Nair and Hinton, 2010, Goodfellow et al., 2016]. Readers familiar with *spline regression* [De Boor et al., 1978, Schumaker, 2007] will notice this as the 'hockey stick' function.

Extending a one-hidden-layer network to a *deep* network of multiple layers simply means that $h(\mathbf{x}; \cdot)$ is comprised of a composition of several GLM-like transformations. Define a L-layer NN's basis function recursively as

$$h_l\left(\mathbf{x}_i; \{\mathbf{W}_j\}_{j=1}^{l}, \{\mathbf{b}_j\}_{j=1}^{l}\right) = f_l(h_{l-1}\left(\mathbf{x}_i; \{\mathbf{W}_j\}_{j=1}^{l-1}, \{\mathbf{b}_j\}_{j=1}^{l-1}\right)\mathbf{W}_l + \mathbf{b}_l), \quad h_0\left(\mathbf{x}_i\right) = \mathbf{x}_i \tag{2.4}$$

for $l \in [0, L]$. The deep NN output is then a GLM defined on the last layer of features: $\mathbb{E}[y_i|\mathbf{x}_i] = g^{-1}(h_L(\mathbf{x}_i; \{\mathbf{W}_l\}_{l=1}^{L}, \{\mathbf{b}_l\}_{l=1}^{L})\mathbf{W}_{L+1} + \mathbf{b}_{L+1})$. For simplicity, from here forward I suppress the functional notation and instead write the intermediate hidden representations as

$$\mathbf{h}_{i,l} = h_l\left(\mathbf{x}_i; \{\mathbf{W}_j\}_{j=1}^{l}, \{\mathbf{b}_j\}_{j=1}^{l}\right).$$

In order to perform probabilistic inference, we need to write the deep NN log likelihood. It is:

$$\begin{aligned}\mathcal{L}\left(\mathcal{D}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}\right) &= \log p\left(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}\right) \\ &= \sum_{i=1}^{N} \log p\left(y_i|\mathbf{x}_i, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}\right)\end{aligned} \tag{2.5}$$

with the model parameters being the collection of weights and biases.

2.1.2 Autoencoders

NNs can also be used without supervision, and these models are known as *autoencoders* (AEs) (a.k.a. *diablo networks* or *auto-associators*) [Cottrell et al., 1989, Baldi and Hornik, 1989, Bourlard and Kamp, 1988, Hinton and Salakhutdinov, 2006, Bengio et al., 2013a]. The AE takes in an observation $\mathbf{x}_i$ as input, computes one or more hidden representations $\mathbf{h}_{i,1}, \ldots, \mathbf{h}_{i,L}$, and then tries to predict the observation back from the latent representation:

$$\mathbb{E}[\mathbf{x}_i|\mathbf{x}_i] = g^{-1}(\mathbf{h}_{i,L}\mathbf{W}_{L+1} + \mathbf{b}_{L+1}), \quad \mathbf{h}_{i,l} = f_L(\mathbf{h}_{i,l-1}\mathbf{W}_l + \mathbf{b}_l), \quad \mathbf{h}_{i,0} = \mathbf{x}_i \tag{2.6}$$

where $g^{-1}(\cdot)$ is again the inverse link function that maps the inner product to the proper domain, $f_l(\cdot)$ is the hidden activation, such as a ReLU, that produces a non-linear latent representation, and the parameters to be estimated are the weights $\{\mathbf{W}_l\}_{l=1}^{L+1}$ and biases $\{\mathbf{b}_l\}_{l=1}^{L+1}$. One-hidden-layer AEs are usually thought of as being comprised of two separate components: the *encoder* $f(\mathbf{x}_i\mathbf{W}_1 + \mathbf{b}_1)$ and the *decoder* $g^{-1}(\mathbf{h}_i\mathbf{W}_2 + \mathbf{b}_2)$. The AE optimization objective is equivalent to that of *principal components analysis* (PCA) when *(i)* we assume the observations are Gaussian, *(ii)* the non-linear activation functions are removed (or equivalently, assumed to be the identity function), and *(iii)* the encoding and decoding parameters are tied [Baldi and Hornik, 1989]. Just as one could do with the latent representations produced by PCA, $\mathbf{h}_i$ can be used as a compressed representation of $\mathbf{x}_i$ and incorporated into downstream tasks such as fast retrieval [Hinton and Salakhutdinov, 2006].

The data set log likelihood under an AE is written as:

$$\begin{aligned}\mathcal{L}\left(\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}\right) &= \log p\left(\mathbf{X}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}\right) \\ &= \sum_{i=1}^{N} \log p\left(\mathbf{x}_i|\mathbf{x}_i, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}\right).\end{aligned} \tag{2.7}$$

Notice that the basic AE is not a principled probabilistic model because the distribution on $\mathbf{x}_i$

is conditioned on itself via the latent representation $\mathbf{h}_{i,L}$. Yet, the model is made practical by choosing $\mathbf{h}$ to have a dimensionality less than that of $\mathbf{x}$, resulting in a 'bottleneck' that prevents the model from learning a spurious identity map. Ideally, the information lost should be the observation noise, and $\mathbf{h}_{i,\cdot}$ should retain the underlying, useful semantic information. Some work has been done to reformulate AEs to be proper generative models. Bengio et al. [2013c] accomplishes this by adding noise to $\mathbf{x}$ before it is passed into the model. Then the noise model and AE can be composed to form a transition operator of a Markov chain (subject to assumptions and conditions).

2.2 Bayesian Neural Networks

Frequentist methods—such as maximum likelihood for parameter estimation, penalized objectives for regularization, cross-validation for model selection, and bagging [Breiman, 1996] as a means of ensembling—have predominantly been used to fit NNs. Bayesian treatments have been less common with the primary obstacle being mathematical: the analytical intractabilities that result from the NN's composition of non-linear functions. However, Bayesian NNs have the potential to be immune to several of the problems known to accompany deep learning. For instance, NNs are 'data hungry,' requiring training data sets of perhaps millions of labeled instances. Bayesian NNs may get around this gluttony by, for example, using well specified priors, similarly to how I described them being used in Chapter 1 for the data scarce baseball setting. Or for another example, the denominator in Bayes' theorem, the model evidence term $p(\mathbf{x})$, integrates over the uncertainty in the parameters and thus is a natural regularization mechanism [MacKay, 1992a]. Using the evidence and related quantities such as the posterior predictive distribution may make deep NNs robust even in situations without much available data.

2.2.1 Conditional Models

NNs can be given a Bayesian treatment by placing priors on the weights and biases. Writing Bayes theorem with a conditional NN model, we have

$$p(\{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}) \prod_{l=1}^{L+1} p(\mathbf{W}_l)p(\mathbf{b}_l)}{p(\mathbf{y}|\mathbf{X})} \tag{2.8}$$

where the model evidence term is computed as

$$p(\mathbf{y}|\mathbf{X}) = \int_{\mathbf{W},\mathbf{b}} p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}) \prod_{l=1}^{L+1} p(\mathbf{W}_l)p(\mathbf{b}_l)\, d\mathbf{W}d\mathbf{b}.$$

One glance at the posterior distribution above provokes the question: *why is it useful?* In the baseball example from Chapter 1, the posterior $p(\pi|\mathbf{x})$ is conspicuously useful since it quantifies uncertainty about the player's batting average. There is no analogous interpretation for $p(\{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}|\mathbf{y}, \mathbf{X})$. $\mathbf{W}_l$ is just one of many (perhaps billions) of parameters and has no inherent meaning in isolation. In fact, NNs are invariant under permutation—that is, the incoming and outgoing weights of the hidden representations can be switched and (with appropriate book keeping) still the network will have the same output [Goodfellow et al., 2016]. Thus, even if two NNs have exactly the same architecture and produce exactly the same outputs, their parameter estimates will not necessarily correspond. This fact makes clear that it will be exceptionally hard for the modeler to glean an interpretation from $p(\{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}|\mathbf{y}, \mathbf{X})$ alone.

Recalling that our goal is to generate predictions for a newly observed point $\mathbf{x}^*$, the *posterior predictive distribution* is the quantity we truly desire:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{y}, \mathbf{X}) = \int_{\mathbf{W},\mathbf{b}} p(\mathbf{y}^*|\mathbf{x}^*, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1})\, p(\{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}|\mathbf{y}, \mathbf{X})\, d\mathbf{W}d\mathbf{b} \tag{2.9}$$

where $p(\mathbf{y}^*|\mathbf{x}^*, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1})$ is the likelihood evaluated on the new feature vector and $p(\{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1}|\mathbf{y}, \mathbf{X})$ is the posterior as computed on the training set. Notice that this

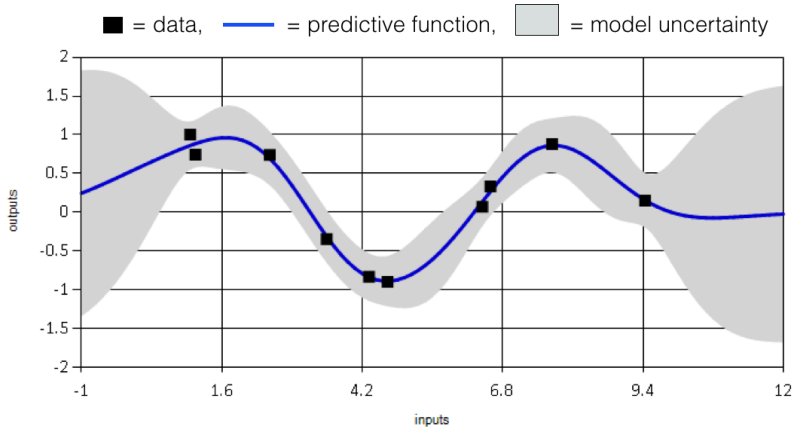


Figure 2.1: *Example of a Posterior Predictive Distribution.*

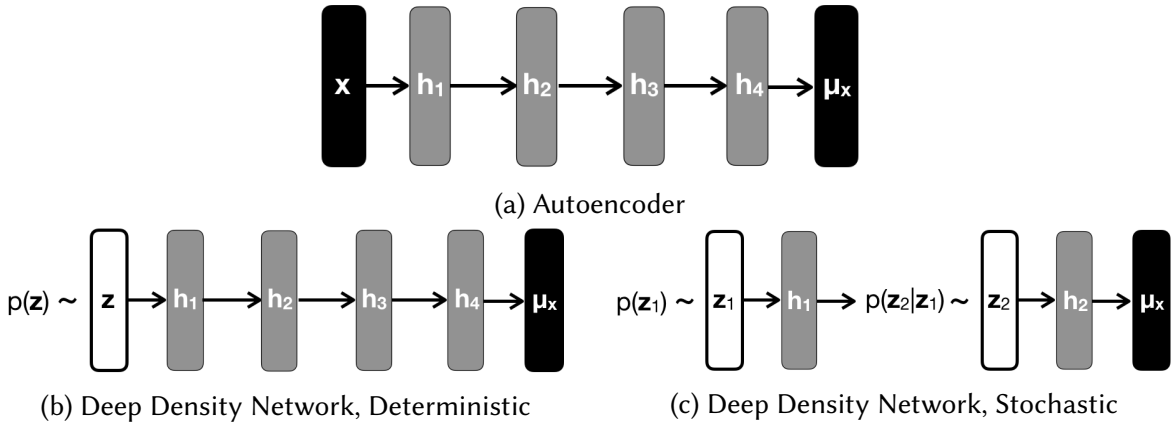
distribution is defined on the data space and therefore has much more of a hope of being interpretable than any distribution on the NN parameters. Figure 2.1 shows an example predictive distribution. The black squares denote training observations, the blue line represents the predictive function, and the gray regions mark the model's posterior predictive uncertainty. Observe how the uncertainty inflates in the areas without data.

2.2.2 Generative Models

AEs could be given a Bayesian formulation in much the same way as the conditional model above, by simply placing priors on the weight matrices and biases. However, this would not fix the problem of AEs being improper generative models. One solution is to reformulate the AE into a latent variable model akin to *factor analysis* (FA) [Bartholomew, 1987], where the latent representation is a random variable. MacKay and Gibbs [1999] call this model a *density network*, and its one-hidden-layer version is defined as follows:

$$\mathbb{E}[\mathbf{x}_i|\mathbf{z}_i] = g^{-1}(\mathbf{h}_i\mathbf{W}_2 + \mathbf{b}_2), \quad \mathbf{h}_i = f(\mathbf{z}_i\mathbf{W}_1 + \mathbf{b}_1), \quad \mathbf{z}_i \sim p(\mathbf{z}) \tag{2.10}$$

where $\mathbf{h}_i$ represents a *deterministic* latent representation (like in the AE) and $\mathbf{z}_i$ denotes a *random* latent variable drawn from the shared (across all i) prior $p(\mathbf{z})$. Priors can also be placed on the



(a) Autoencoder

(b) Deep Density Network, Deterministic

(c) Deep Density Network, Stochastic

Figure 2.2: *Unsupervised Neural Network Models.* Computation diagrams for the various unsupervised neural network models. Black rectangles denote observed quantities, gray rectangles denote deterministic hidden units, white rectangles denote stochastic latent variables.

weights and biases. Density networks are properly generative since $\mathbf{z}$ appears on the RHS of the conditional expectation and not $\mathbf{x}$. Thus, *ancestral sampling* can be used to draw from the model: $\hat{\mathbf{z}} \sim p(\mathbf{z})$, $\hat{\mathbf{x}} \sim p(\mathbf{x}|\hat{\mathbf{z}}; \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{b}_l\}_{l=1}^{L+1})$. Readers familiar with FA will first notice that removing the deterministic hidden layer and setting the link to the identity results in FA's traditional formation, i.e. $\mathbb{E}[\mathbf{x}_i|\mathbf{z}_i] = \mathbf{z}_i\mathbf{W} + \mathbf{b}$, with $\mathbf{W}$ being known as the *loading matrix*. Hence, the density network model can be thought of as a particular implementation of non-linear FA [Gibson, 1960, McDonald, 1962]—specifically, non-linear in both the parameters and the factors, as was first introduced by Amemiya [1993]. *Generative adversarial networks* [Goodfellow et al., 2014] have a density network as their underlying generative model; however, the generator network is not trained through the usual likelihood-based approaches, making the coming discussion of inference methods not fully applicable.

Density networks can be made 'deep' in at least two ways. The first is to insert multiple deterministic representations $\mathbf{h}_{i,\cdot}$ between $\mathbf{z}_i$ and the model output, just as one would make an AE deep. The second is slightly more interesting: use multiple stochastic latent variables and parametrize the prior at layer l in terms of the variables at layer $l-1$. For a concrete example,

consider parametrizing the mean of a Gaussian with the preceding latent variable:

$$p(\mathbf{z}_l|\mathbf{z}_{l-1};\boldsymbol{\theta},\boldsymbol{\Sigma}_l) = \mathsf{N}(\mathbf{z}_l;\boldsymbol{\mu}_l = \psi(\mathbf{z}_{l-1};\boldsymbol{\theta}),\boldsymbol{\Sigma}_l)$$

where $\psi(\mathbf{z}_{l-1};\boldsymbol{\theta})$ is a function with parameters $\boldsymbol{\theta}$ that maps the latent variable to the mean. Rezende et al. [2014] consider such a model and use NN transformations for the maps (which could include multiple deterministic layers). Figure 2.2 shows computation diagrams for the unsupervised NN models discussed: the AE and the two variants of a deep density network. Subfigure (b) shows how depth is added to the density network via deterministic transformations and no additional stochastic latent variables. Subfigure (c) shows multiple levels of stochastic variables with the one at an earlier layer parametrizing the prior for one at a later layer. There has also been work on fusing density networks with general graphical model structures [Johnson et al., 2016, Lin et al., 2018].

Lastly, I show how to write Bayes' theorem for the density network. In order to make the expression most general, I write it for the deep stochastic version with L layers of (random) latent variables. I use $\boldsymbol{\theta}_l$ to denote the parameters of the transformation that maps $\mathbf{z}_{l-1}$ to the parameters of the distribution on $\mathbf{z}_l$:

$$p(\{\mathbf{z}_{i,l}\}_{l=1}^{L}|\mathbf{X};\{\boldsymbol{\theta}_l\}_{l=2}^{L+1}) = \frac{\prod_{i=1}^{N} p(\mathbf{x}_i|\mathbf{z}_{i,L};\boldsymbol{\theta}_{L+1})p(\mathbf{z}_{i,1})\prod_{l=2}^{L} p(\mathbf{z}_{i,l}|\mathbf{z}_{i,l-1};\boldsymbol{\theta}_l)}{p(\mathbf{X};\{\boldsymbol{\theta}_l\}_{l=2}^{L+1})} \tag{2.11}$$

where $p(\mathbf{X};\{\boldsymbol{\theta}_l\}_{l=2}^{L+1})$ is the model evidence[1] and is computed by integrating over all $\{\mathbf{z}_{i,l}\}_{l=1}^{L}$ for all i. A fully Bayesian treatment would also place priors $\prod_{l=1}^{L+1} p(\boldsymbol{\theta}_l)$ on all the transformation parameters, but practical implementations commonly use point estimates [Kingma and Welling, 2014b, Rezende et al., 2014].

[1] Note that since there is no transformation associated with $\mathbf{z}_1$, the index on $\boldsymbol{\theta}_l$ begins at two.

2.3 Posterior Inference

While I have given the analytical expression for the posterior distribution of both types of Bayesian NNs discussed (Equations 2.8 and 2.11), computing the posterior is not easily done. It is made complicated by the model evidence terms $p(\mathbf{y}|\mathbf{X})$ and $p(\mathbf{X}; \{\boldsymbol{\theta}_l\}_{l=2}^{L+1})$. These integrals are intractable, generally, because *(i)* NN's composition of non-linear functions prevents analytical solutions and *(ii)* the usual high-dimensionality of the parameter space causes naive numerical approaches, i.e. *quadrature*, to fail. There are two ways to get around computing the model evidence. The first is to work with posterior ratios so the model evidence term cancels out; this is the idea behind Markov chain Monte Carlo. The second is to optimize a bound on the evidence; this is the idea behind variational inference. I describe both ideas below, and to do so most generally, I use $\boldsymbol{\pi}$ to denote the random variables and $\mathbf{X}$ to denote the data. After introducing the methods, I discuss the details for each Bayesian NN type.

2.3.1 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) generates posterior samples via a random walk through the posterior density. A random walk is simply a probabilistic transition between two points, which I denote $p(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)$, and in order to guarantee proper sampling, this walk's stationary distribution must be the posterior:

$$\frac{p(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)}{p(\boldsymbol{\pi}_t|\boldsymbol{\pi}_{t+1})} = \frac{p(\boldsymbol{\pi}_{t+1}|\mathbf{X})}{p(\boldsymbol{\pi}_t|\mathbf{X})} \tag{2.12}$$

Although the RHS of the expression looks problematic, as it involves the very posterior we are unable to compute, it is not since the evidence terms cancel, i.e.

$$\frac{p(\boldsymbol{\pi}_{t+1}|\mathbf{X})}{p(\boldsymbol{\pi}_t|\mathbf{X})} = \frac{p(\mathbf{X}|\boldsymbol{\pi}_{t+1})\, p(\boldsymbol{\pi}_{t+1})\, \cancel{p(\mathbf{X})}}{p(\mathbf{X}|\boldsymbol{\pi}_t)\, p(\boldsymbol{\pi}_t)\, \cancel{p(\mathbf{X})}}.$$

This simplification is the key property that makes MCMC suitable for posterior inference. Turning back to Equation 2.12, constructing the LHS, the transition probabilities, is actually the challenging part. We can make progress by decomposing the true transition into a proposed transition and an accept probability [Hastings, 1970]: $p(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t) = q(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)A(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)$ where $q(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)$ is known as the *proposal distribution* and $A(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)$ as the *accept distribution*. With these two quantities in hand, we can write:

$$\frac{p(\boldsymbol{\pi}_{t+1}|\mathbf{X})}{p(\boldsymbol{\pi}_t|\mathbf{X})} = \frac{p(\mathbf{X}|\boldsymbol{\pi}_{t+1})p(\boldsymbol{\pi}_{t+1})}{p(\mathbf{X}|\boldsymbol{\pi}_t)p(\boldsymbol{\pi}_t)} = \frac{p(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)}{p(\boldsymbol{\pi}_t|\boldsymbol{\pi}_{t+1})} = \frac{q(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)A(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)}{q(\boldsymbol{\pi}_t|\boldsymbol{\pi}_{t+1})A(\boldsymbol{\pi}_t|\boldsymbol{\pi}_{t+1})}. \tag{2.13}$$

The user specifies the proposal distribution thus leaving $A(\cdot|\cdot)$ as the only unknown quantity above. Re-writing Equation 2.13 by moving the proposals to the same side as the likelihood and prior, so that all the known quantities are together, we have

$$\frac{A(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)}{A(\boldsymbol{\pi}_t|\boldsymbol{\pi}_{t+1})} = \frac{p(\mathbf{X}|\boldsymbol{\pi}_{t+1})p(\boldsymbol{\pi}_{t+1})}{p(\mathbf{X}|\boldsymbol{\pi}_t)p(\boldsymbol{\pi}_t)}\frac{q(\boldsymbol{\pi}_t|\boldsymbol{\pi}_{t+1})}{q(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)}. \tag{2.14}$$

The second ratio on the RHS is known as the *Hastings correction* since it adjusts for any asymmetry in the proposal and will cancel out if the proposal is symmetric.

From Equation 2.14, we see the problem of posterior sampling has been distilled into the problem of computing the accept probability $A(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)$. Or in other words, given a seed point $\hat{\boldsymbol{\pi}}_t$ and a new point sampled from the proposal $\hat{\boldsymbol{\pi}}_{\text{prop}} \sim q(\boldsymbol{\pi}_{t+1}|\hat{\boldsymbol{\pi}}_t)$, does $\hat{\boldsymbol{\pi}}_{\text{prop}}$ have a high enough probability under $p(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)$ such that we can treat it as a sample from the chain and hence from the posterior? The difficulty is we do not have access to $A(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)$ directly, only the surrogate ratio on the RHS of Equation 2.14. We can work in terms of the RHS by observing that there are only two outcomes when evaluating

$$r(\hat{\boldsymbol{\pi}}_{\text{prop}}, \hat{\boldsymbol{\pi}}_t) = \frac{p(\mathbf{X}|\hat{\boldsymbol{\pi}}_{\text{prop}})p(\hat{\boldsymbol{\pi}}_{\text{prop}})}{p(\mathbf{X}|\hat{\boldsymbol{\pi}}_t)p(\hat{\boldsymbol{\pi}}_t)}\frac{q(\hat{\boldsymbol{\pi}}_t|\hat{\boldsymbol{\pi}}_{\text{prop}})}{q(\hat{\boldsymbol{\pi}}_{\text{prop}}|\hat{\boldsymbol{\pi}}_t)}. \tag{2.15}$$

If $r(\hat{\boldsymbol{\pi}}_{\text{prop}}, \hat{\boldsymbol{\pi}}_t) \geq 1$, then that implies $A(\hat{\boldsymbol{\pi}}_{\text{prop}}|\hat{\boldsymbol{\pi}}_t) \geq A(\hat{\boldsymbol{\pi}}_t|\hat{\boldsymbol{\pi}}_{\text{prop}})$, meaning that the proposed

point is at least as probable under the chain as the previous point. If this is the case, we should accept the transition. On the other hand, if $r(\hat{\boldsymbol{\pi}}_{\text{prop}}, \hat{\boldsymbol{\pi}}_t) < 1$, then $A(\hat{\boldsymbol{\pi}}_{\text{prop}}|\hat{\boldsymbol{\pi}}_t) < A(\hat{\boldsymbol{\pi}}_t|\hat{\boldsymbol{\pi}}_{\text{prop}})$, which means that the new point is less probable under the chain. But this does not mean that we want to reject for then the algorithm would get trapped at modes. We instead want to sample an outcome (move or not) from $A(\hat{\boldsymbol{\pi}}_{\text{prop}}|\hat{\boldsymbol{\pi}}_t)$. Noticing that $A(\hat{\boldsymbol{\pi}}_t|\hat{\boldsymbol{\pi}}_{\text{prop}})$ is an upper bound on the probability of accepting $\hat{\boldsymbol{\pi}}_{\text{prop}}$ makes the ratio of accept distributions amenable to *rejection sampling*, which we can perform on the surrogate: sample $\hat{u} \sim \text{Uniform}[0, 1]$ and if $r(\hat{\boldsymbol{\pi}}_{\text{prop}}, \hat{\boldsymbol{\pi}}_t) > \hat{u}$, we accept the move, setting $\hat{\boldsymbol{\pi}}_{t+1} = \hat{\boldsymbol{\pi}}_{\text{prop}}$ [Hastings, 1970]. Otherwise, we sample a new proposed transition and re-evaluate. After many iterations, the algorithm returns a collection of samples $\{\hat{\boldsymbol{\pi}}_0, \ldots, \hat{\boldsymbol{\pi}}_t, \ldots, \hat{\boldsymbol{\pi}}_T\}$. Usually the early samples from time $t \in [0, B]$ are discarded because they were sampled when the chain was simply making its way from the initial point to an area of high density; this is known as the *burn-in phase*. The samples $\{\hat{\boldsymbol{\pi}}_{B+1}, \ldots, \hat{\boldsymbol{\pi}}_T\}$ are then treated as being from the posterior and used to calculate quantities of interest, such as moments: $\mathbb{E}[f(\boldsymbol{\pi})] \approx \frac{1}{T-B-1} \sum_{t=B+1}^{T} f(\hat{\boldsymbol{\pi}}_t)$.

While MCMC has proven successful, its base implementation described above can fail as dimensionality increases. One reason is that the proposal distribution must be set well in order to effeciently explore the high dimensional space. However, as dimensionality increases, human intuition degrades, making it hard to set the very proposal distribution that is so critical. One way to fix this is to incorporate model information into $q(\boldsymbol{\pi}_{t+1}|\boldsymbol{\pi}_t)$. *Hamiltonian Monte Carlo* (HMC) [Neal, 2011] does just this by using the model's gradient to compute the next candidate point. HMC operates on an augmented system with *Hamiltonian function*

$$H(\boldsymbol{\pi}, \mathbf{v}) = U(\boldsymbol{\pi}) + \frac{1}{2}\mathbf{v}^T\mathbf{M}^{-1}\mathbf{v}, \quad U(\boldsymbol{\pi}) = -\sum_{i=1}^{N} \log p(\mathbf{x}_i|\boldsymbol{\pi}) - \log p(\boldsymbol{\pi}), \quad \mathbf{v} \sim \text{N}(\mathbf{0}, \mathbf{M}) \tag{2.16}$$

with $\mathbf{v}$ being an auxiliary momentum variable and $\mathbf{M}$ being a mass matrix. The system then moves with dynamics $\{d\boldsymbol{\pi} = \mathbf{M}^{-1}\mathbf{v}dt, \quad d\mathbf{v} = -\nabla U(\boldsymbol{\pi})dt\}$ for a user-defined number of 'leapfrog' steps to propose new $\hat{\mathbf{v}}_{\text{prop}}$ and $\hat{\boldsymbol{\pi}}_{\text{prop}}$ variables. Once this is done, the usual HM accept-

reject step is performed with the ratio

$$r_{\text{HMC}}(\hat{\boldsymbol{\pi}}_{\text{prop}}, \hat{\boldsymbol{\pi}}_t, \hat{\mathbf{v}}_{\text{prop}}, \hat{\mathbf{v}}_t) = \exp\{H(\hat{\boldsymbol{\pi}}_{\text{prop}}, \hat{\mathbf{v}}_{\text{prop}}) - H(\hat{\boldsymbol{\pi}}_t, \hat{\mathbf{v}}_t)\} \tag{2.17}$$

with $\hat{\boldsymbol{\pi}}_{\text{prop}}$ being accepted as $\hat{\boldsymbol{\pi}}_{t+1} = \hat{\boldsymbol{\pi}}_{\text{prop}}$ if $r_{\text{HMC}}(\hat{\boldsymbol{\pi}}_{\text{prop}}, \hat{\boldsymbol{\pi}}_t, \hat{\mathbf{v}}_{\text{prop}}, \hat{\mathbf{v}}_t) > \hat{u}$.

2.3.1.1 Conditional Models

Now I discuss performing MCMC for Bayesian NNs in particular. To reduce notational clutter, I drop the bias variables from here forward. As most practical NNs are deep and wide, the high dimensionality of their parameter space complicates MCMC from the outset, and using HMC is all but required [Neal, 1993]. HMC requires the model joint be differentiable in order to calculate $\nabla U(\{\mathbf{W}_l\}_{l=1}^{L+1})$ to run the Hamiltonian dynamics. Fortunately, this is not an issue with NNs, as it can be for some models, since they are designed with gradient-based optimization (i.e. backpropagation) in mind. However, notice that $U(\{\mathbf{W}_l\}_{l=1}^{L+1})$ (Equation 2.16) requires a sum over the whole data set *for each leapfrog step*, and this $\mathcal{O}(N)$ dependence can present an obstacle to scaling HMC to large data sets. One natural idea is to use a subset of the data to compute an approximation:

$$\begin{aligned} U(\{\mathbf{W}_l\}_{l=1}^{L+1}) &= -\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \{\mathbf{W}_l\}_{l=1}^{L+1}) - \log p(\{\mathbf{W}_l\}_{l=1}^{L+1}) \\ &\approx -\frac{N}{M}\sum_{m=1}^{M} \log p(y_m|\mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1}) - \log p(\{\mathbf{W}_l\}_{l=1}^{L+1}) \end{aligned} \tag{2.18}$$

for some subset or mini-batch of the data $\{(y_1, \mathbf{x}_1), \ldots, (y_M, \mathbf{x}_M)\}$, $M << N$. Chen et al. [2014] propose such a scheme, calling it *stochastic gradient HMC*, but Betancourt [2015] points out subsampling the data introduces "an irreducible bias that devastates the scalable[2] performance" of HMC.

[2] *Scalable* here refers to scalable w.r.t the dimensionality of the parameter space.

When HMC is made impractical by the data set size, *stochastic gradient Langevin dynamics* (SGLD) [Welling and Teh, 2011] and its sister method *stochastic gradient Fisher scoring* [Ahn et al., 2012] (SGFS) can be used instead. SGLD generates a posterior sample at time t, denoted $\hat{\mathbf{W}}_t$, via the equation:

$$\hat{\mathbf{W}}_l^{t+1} = \hat{\mathbf{W}}_l^t + \frac{\alpha}{2}\mathbf{C}\,\nabla_{\mathbf{W}_l}\left[\frac{N}{M}\sum_{m=1}^{M}\log p(y_m|\mathbf{x}_m, \{\mathbf{W}_l^t\}_{l=1}^{L+1}) - \log p(\{\mathbf{W}_l^t\}_{l=1}^{L+1})\right] + \hat{\boldsymbol{\epsilon}} \tag{2.19}$$
$$\hat{\boldsymbol{\epsilon}} \sim \mathsf{N}(\mathbf{0}, \alpha\mathbf{C})$$

where α is again the step size, $\mathbf{C}$ is a preconditioning matrix, and $\boldsymbol{\epsilon}$ is a Gaussian random variable. Notice that the expression inside the brackets is the subsampled $U(\{\mathbf{W}_l\}_{l=1}^{L+1})$ from above. Intuitively, SGLD generates posterior samples via 'noisy' stochastic gradient ascent on the log model joint. The Gaussian noise serves a similar role to the rejection sampling component of the Metropolis-Hastings step in it allows the chain to transition out of local modes and explore neighboring regions of lower posterior probability. Hence, the SGLD step size must be properly annealed in order to perform unbiased sampling from the posterior [Welling and Teh, 2011]. SGFS uses the same base update as SGLD but improves upon the chain's mixing time by setting the gradient preconditioning as:

$$\frac{\alpha}{2}\mathbf{C} \to 2\left[\frac{N+M}{N}\hat{\mathbf{F}} + \frac{4}{\alpha}\mathbf{B}\right]^{-1}$$

where $\mathbf{B}$ is any symmetric positive-definite matrix and $\hat{\mathbf{F}}$ is the running empirical Fisher information matrix (online estimate of the covariance of the gradients). The Gaussian distribution is changed as well, to $\boldsymbol{\epsilon} \sim \mathsf{N}(\mathbf{0}, \frac{4}{\alpha}\mathbf{B})$. Experiments [Welling and Teh, 2011, Ahn et al., 2012] have demonstrated that SGLD and SGFS are competitive alternatives to HMC, especially when computational costs are factored in.

The choice of MCMC variant not withstanding, the unidentifiability, symmetries, and invariances innate to the NN model definition can prevent mixing [Müller and Insua, 1998, Vehtari

et al., 2000, Pourzanjani et al., 2017]. For instance, as noted earlier in the chapter, NNs are invariant under permutation, and this leads to multiple, equally-likely posterior modes. MCMC is liable to explore all of these modes but the effort is essentially wasted since the modes represent practically equivalent solutions. This permutation invariance can be broken by imposing a strict ordering on the biases at each hidden layer [Pourzanjani et al., 2017]. ReLU units present another model pathology because they have the scaling symmetry $\text{ReLU}(\mathbf{hw}) = 1/c \cdot \text{ReLU}(c \cdot \mathbf{hw})$ for a scalar c. Constraining the incoming weight vectors to have unit norm breaks this symmetry [Pourzanjani et al., 2017]. In addition to these details of model specification, implementation details such as the starting value and the number of chains can have a substantial effect on the results [Vehtari et al., 2000].

2.3.1.2 Density Networks

Performing MCMC for density networks involves all the parametrization-based difficulties encountered with conditional models—scale invariance, equivalence under permutation—but these are magnified by the fact that we need to perform MCMC for each per-data-point latent variable $\mathbf{z}_i$. The $\mathcal{O}(N)$ dependence has moved from the inner loop, the proposal step, to the outer loop of running the whole chain. In the paper that proposed density networks, MacKay and Gibbs [1999] performed posterior inference by Monte Carlo importance sampling and by HMC. However, the data sets used were relatively small by today's standards in both number of points and in dimensionality.

Hoffman [2017] revisits MCMC for GPU-era density networks, proposing a hybrid strategy in which a model $q(\mathbf{z}_i^{\text{init}}|\mathbf{x}_i; \boldsymbol{\phi})$ is used to sample an initialization and then HMC is run starting from the sample. The number of necessary HMC steps, presumably, should be reduced since the initialization model can be optimized to give HMC a significant 'head start.' The initialization model is also helpful when encountering new data points since the training set can be leveraged to expedite inference. This general strategy—incorporating a parametrized distribution into MCMC

and optimizing it—is called *variational Monte Carlo* [De Freitas et al., 2001].

2.3.2 Variational Inference

Variational inference (VI) [Peterson and Anderson, 1987, Saul et al., 1996, Jordan et al., 1999, Blei et al., 2017] is a model- and optimization-based approach to computing the posterior. Instead of obtaining a finite set of samples as MCMC does, VI fits a distribution—call it $q(\boldsymbol{\pi};\boldsymbol{\phi})$ with parameters $\boldsymbol{\phi}$—to approximate the true posterior: $p(\boldsymbol{\pi}|\mathbf{X}) \approx q(\boldsymbol{\pi};\boldsymbol{\phi})$. VI bypasses the problematic evidence term by bounding it, usually from below, and optimizing the bound w.r.t. the variational parameters. The derivation is as follows:

$$\begin{aligned}
\log p(\mathbf{X}) &= \log \int_{\boldsymbol{\pi}} p(\mathbf{X},\boldsymbol{\pi}) d\boldsymbol{\pi} = \log \int_{\boldsymbol{\pi}} \frac{q(\boldsymbol{\pi};\boldsymbol{\phi})}{q(\boldsymbol{\pi};\boldsymbol{\phi})} p(\mathbf{X},\boldsymbol{\pi}) d\boldsymbol{\pi} \\
&= \log \mathbb{E}_{q(\boldsymbol{\pi})}\left[\frac{p(\mathbf{X},\boldsymbol{\pi})}{q(\boldsymbol{\pi};\boldsymbol{\phi})}\right] \quad \text{(importance integral)} \\
&\geq \mathbb{E}_{q(\boldsymbol{\pi})}\left[\log p(\mathbf{X},\boldsymbol{\pi}) - \log q(\boldsymbol{\pi};\boldsymbol{\phi})\right] = \mathbb{E}_{q(\boldsymbol{\pi})}\left[\log p(\mathbf{X},\boldsymbol{\pi})\right] + \mathbb{H}_{q(\boldsymbol{\pi})}[\boldsymbol{\pi}] \\
&= \mathcal{J}_{\text{ELBO}}(\mathbf{X},\boldsymbol{\phi})
\end{aligned} \tag{2.20}$$

where $\mathbb{H}_{q(\boldsymbol{\pi})}[\boldsymbol{\pi}]$ is the entropy of the approximation. The inequality is from Jensen's. This objective is known as the *evidence lower bound* (ELBO) [Jordan et al., 1999], and it is optimized by maximizing w.r.t. $\boldsymbol{\phi}$. Readers familiar with the *expectation-maximization* (EM) algorithm [Dempster et al., 1977] may notice similarities. In fact, the relationship runs deep, as pointed out by Neal and Hinton [1998]. Firstly, the expectation in the ELBO is the same as the one computed in the E-step of EM except that in VI it is taken w.r.t. a model of our choosing, not a conditional posterior. Secondly, the entropy term does not appear in EM since the approximation is just a point estimate with constant entropy. Turning back to VI's implementation, often a *mean-field* assumption [Peterson and Anderson, 1987, Saul et al., 1996] is invoked that assumes the posterior approximation factorizes over some or all variables, i.e. $q(\boldsymbol{\pi};\boldsymbol{\phi}) = \prod_{d=1}^{D} q(\pi_d;\phi_d)$. This simplification makes computing the ELBO's expectations easier. After the approximation has

been fit, $q(\boldsymbol{\pi}; \boldsymbol{\phi})$ can be used to compute posterior quantities of interest, such as expectations: $\mathbb{E}_{p(\boldsymbol{\pi}|\mathbf{X})}[f(\boldsymbol{\pi})] \approx \int_{\boldsymbol{\pi}} q(\boldsymbol{\pi}; \hat{\boldsymbol{\phi}}) f(\boldsymbol{\pi}) d\boldsymbol{\pi}$.

Further intuition about the ELBO's workings can be gleaned by separating the model joint into likelihood and prior:

$$\begin{aligned}
\mathcal{J}_{\text{ELBO}}(\mathbf{X}, \boldsymbol{\phi}) &= \mathbb{E}_{q(\boldsymbol{\pi})} \left[\log p(\mathbf{X}, \boldsymbol{\pi}) - \log q(\boldsymbol{\pi}; \boldsymbol{\phi}) \right] \\
&= \mathbb{E}_{q(\boldsymbol{\pi})} \left[\log p(\mathbf{X}|\boldsymbol{\pi}) + \log p(\boldsymbol{\pi}) - \log q(\boldsymbol{\pi}; \boldsymbol{\phi}) \right] \\
&= \mathbb{E}_{q(\boldsymbol{\pi})} \left[\log p(\mathbf{X}|\boldsymbol{\pi}) \right] - \text{KLD} \left[q(\boldsymbol{\pi}; \boldsymbol{\phi}) \,||\, p(\boldsymbol{\pi}) \right]
\end{aligned} \tag{2.21}$$

where $\text{KLD}\left[\cdot \,||\, \cdot\right]$ denotes the *Kullback-Leibler divergence* (KLD), which is defined as $\text{KLD}\left[q \,||\, p\right] = \int_z q(z)[\log q(z) - \log p(z)]dz$. From this view, we see that the ELBO is akin to regularized maximum likelihood: the first term is simply the expected log likelihood under the variational posterior and the second term, the KLD regularizer, quantifies how far the approximation has deviated from the prior. Since the posterior is an interpolation between the MLE and prior, we see that both of these terms are accounted for when fitting $q(\boldsymbol{\pi}; \boldsymbol{\phi})$.

Lastly, an important question but one I have yet to address is: how accurate is the approximation $p(\boldsymbol{\pi}|\mathbf{X}) \approx q(\boldsymbol{\pi}; \boldsymbol{\phi})$? An alternative derivation of the ELBO unveils the approximation gap:

$$\begin{aligned}
\text{KLD}\left[q(\boldsymbol{\pi}; \boldsymbol{\phi}) \,||\, p(\boldsymbol{\pi}|\mathbf{X})\right] &= \int_{\boldsymbol{\pi}} q(\boldsymbol{\pi}; \boldsymbol{\phi}) \frac{q(\boldsymbol{\pi}; \boldsymbol{\phi})}{p(\boldsymbol{\pi}|\mathbf{X})} d\boldsymbol{\pi} = \int_{\boldsymbol{\pi}} q(\boldsymbol{\pi}; \boldsymbol{\phi}) \frac{q(\boldsymbol{\pi}; \boldsymbol{\phi}) p(\mathbf{X})}{p(\mathbf{X}, \boldsymbol{\pi})} d\boldsymbol{\pi} \\
&= -\mathbb{E}_{q(\boldsymbol{\pi})} \left[\log p(\mathbf{X}, \boldsymbol{\pi}) - \log q(\boldsymbol{\pi}; \boldsymbol{\phi}) \right] + \log p(\mathbf{X}) \\
&= -\mathcal{J}_{\text{ELBO}}(\mathbf{X}, \boldsymbol{\phi}) + \log p(\mathbf{X}), \quad \text{rearranging terms yields...} \\
\log p(\mathbf{X}) &= \mathcal{J}_{\text{ELBO}}(\mathbf{X}, \boldsymbol{\phi}) + \text{KLD}\left[q(\boldsymbol{\pi}; \boldsymbol{\phi}) \,||\, p(\boldsymbol{\pi}|\mathbf{X})\right].
\end{aligned} \tag{2.22}$$

This derivation shows that maximizing the ELBO is equivalent to minimizing the KLD between approximation and posterior. The gap between ELBO and model evidence is exactly the bits lost in the approximation, as quantified by the KLD. If the approximation becomes exact, then $\text{KLD}\left[q(\boldsymbol{\pi}; \boldsymbol{\phi}) \,||\, p(\boldsymbol{\pi}|\mathbf{X})\right] = 0$, making the ELBO equivalent to the log evidence. Unfortunately,

while this derivation reveals the intuition underlying the approximation gap, it does not allow us to usefully quantify the gap because the expression depends on the hard-to-compute posterior that necessitates the use of VI in the first place. Developing metrics to assess VI's goodness-of-fit is an open problem; see Yao et al. [2018] for work in the area. Using alternative optimization objectives is also an area of active research [Minka, 2001, Burda et al., 2016, Li and Turner, 2016, Ranganath et al., 2016, Liu and Wang, 2016, Bouchard and Lakshminarayanan, 2015, Dieng et al., 2017, Chen et al., 2015].

2.3.2.1 Conditional Models

For Bayesian NNs, unfortunately, the ELBO's expectations do not have a closed-form except in simplistic cases (such as with identity or linear activation functions). Therefore, the major challenge in performing VI is how to approximate these expectations efficiently and within a scalable optimization framework [Hinton and Van Camp, 1993, Graves, 2011]. Due to stochastic gradient ascent/descent being just about the only viable method for NN optimization, we aim to compute the gradients w.r.t. the parameters of the mean-field approximation $p(\{\mathbf{W}_l\}_{l=1}^{L+1}|\mathbf{y}, \mathbf{X}) \approx q(\{\mathbf{W}_l\}_{l=1}^{L+1}; \boldsymbol{\phi}) = \prod_{l=1}^{L+1} q(\mathbf{W}_l; \boldsymbol{\phi}_l)$:

$$
\begin{aligned}
&\nabla_{\boldsymbol{\phi}} \mathcal{J}_{\text{NN-ELBO}}(\mathbf{X}, \boldsymbol{\phi}) \\
&\quad= \sum_{i=1}^{N} \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q(\{\mathbf{W}_l\}_{l=1}^{L+1})} \left[\log p(y_i|\mathbf{x}_i, \{\mathbf{W}_l\}_{l=1}^{L+1})\right] - \sum_{l=1}^{L+1} \nabla_{\boldsymbol{\phi}} \text{KLD}\left[q(\mathbf{W}_l; \boldsymbol{\phi}_l) \mid\mid p(\mathbf{W}_l)\right] \\
&\quad\approx \sum_{m=1}^{M} \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q(\{\mathbf{W}_l\}_{l=1}^{L+1})} \left[\log p(y_m|\mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1})\right] - \sum_{l=1}^{L+1} \nabla_{\boldsymbol{\phi}} \text{KLD}\left[q(\mathbf{W}_l; \boldsymbol{\phi}_l) \mid\mid p(\mathbf{W}_l)\right] \\
&\quad= \nabla_{\boldsymbol{\phi}} \tilde{\mathcal{J}}_{\text{NN-ELBO}}(\tilde{\mathbf{X}}, \boldsymbol{\phi})
\end{aligned}
\tag{2.23}
$$

where $\tilde{\mathbf{X}}$ denotes a mini-batch $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ ($M << N$) of the training set. Again, I have dropped the bias terms to simplify the expressions. We assume the KLD gradient can be computed in closed form, which is usually the case, but if not, techniques similar to the ones I describe

below can be used for approximation. Thus, the primarily difficulty is how to compute the gradient of the expected log likelihood. There are essentially three well-known techniques that I now summarize.

1. **Delta Method:** A straight-forward but expensive and usually inaccurate approximation is to linearize around the mean of the variational posterior [Tierney et al., 1989, MacKay, 1992b, Wang and Blei, 2013]:

$$
\begin{aligned}
&\mathbb{E}_{q(\{\mathbf{W}_l\}_{l=1}^{L+1})}\left[\log p(y_m|\mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1})\right] \\
&\approx \mathbb{E}_{q(\{\mathbf{W}_l\}_{l=1}^{L+1})}\Bigg[\log p(y_m|\mathbf{x}_m, \{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}\}_{l=1}^{L+1}) \\
&\qquad + \sum_{l=1}^{L+1}(\mathbf{W}_l - \boldsymbol{\mu}_l^{\boldsymbol{\phi}})\nabla_{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}} \log p(y_m|\mathbf{x}_m, \{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}\}_{l=1}^{L+1})^T \\
&\qquad + \frac{1}{2}\sum_{l=1}^{L+1}(\mathbf{W}_l - \boldsymbol{\mu}_l^{\boldsymbol{\phi}})\nabla^2_{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}} \log p(y_m|\mathbf{x}_m, \{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}\}_{l=1}^{L+1})(\mathbf{W}_l - \boldsymbol{\mu}_l^{\boldsymbol{\phi}})^T\Bigg] \\
&= \log p(y_m|\mathbf{x}_m, \{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}\}_{l=1}^{L+1}) \\
&\qquad + \sum_{l=1}^{L+1}(\mathbb{E}_{q(\mathbf{W}_l)}\left[\mathbf{W}_l\right] - \boldsymbol{\mu}_l^{\boldsymbol{\phi}})\nabla_{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}} \log p(y_m|\mathbf{x}_m, \{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}\}_{l=1}^{L+1})^T \\
&\qquad + \frac{1}{2}\sum_{l=1}^{L+1}\mathbb{E}_{q(\mathbf{W}_l)}\left[(\mathbf{W}_l - \boldsymbol{\mu}_l^{\boldsymbol{\phi}})\nabla^2_{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}} \log p(y_m|\mathbf{x}_m, \{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}\}_{l=1}^{L+1})(\mathbf{W}_l - \boldsymbol{\mu}_l^{\boldsymbol{\phi}})^T]\right].
\end{aligned}
$$

Solving the expectations then yields the final gradient approximation:

$$
\begin{aligned}
&\nabla_{\boldsymbol{\phi}}\mathbb{E}_{q(\{\mathbf{W}_l\}_{l=1}^{L+1})}\left[\log p(y_m|\mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1})\right] \\
&\approx \nabla_{\boldsymbol{\phi}}\left[\log p(y_m|\mathbf{x}_m, \{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}\}_{l=1}^{L+1}) + \frac{1}{2}\sum_{l=1}^{L+1}\text{trace}\left\{\boldsymbol{\Sigma}_l^{\boldsymbol{\phi}}\nabla^2_{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}} \log p(y_m|\mathbf{x}_m, \{\boldsymbol{\mu}_l^{\boldsymbol{\phi}}\}_{l=1}^{L+1})\right\}\right]
\end{aligned}
\tag{2.24}
$$

where $\boldsymbol{\mu}_l^{\boldsymbol{\phi}}$ is the first moment of $q(\{\mathbf{W}_l\}_{l=1}^{L+1})$ and $\boldsymbol{\Sigma}_l^{\boldsymbol{\phi}}$ is the second. I use the subscript $\boldsymbol{\phi}$ to emphasize these are (or are functions of) the variational parameters. Thus, they have been exposed and gradients can be calculated as usual.

2. **Score Function Estimator:** The *score function* (or *likelihood ratio* or *REINFORCE*) estimator [Williams, 1992, Paisley et al., 2012] relies on an algebraic identity of the score function to re-write the gradient of the expectation as an expectation of the score function:

$$\begin{aligned}
&\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q(\{\mathbf{W}_l\}_{l=1}^{L+1})} \left[\log p(y_m | \mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1}) \right] \\
&\quad = \int_{\mathbf{W}} \log p(y_m | \mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1}) \; \nabla_{\boldsymbol{\phi}} q(\{\mathbf{W}_l\}_{l=1}^{L+1}; \boldsymbol{\phi}) \; d\mathbf{W} \\
&\quad = \int_{\mathbf{W}} \log p(y_m | \mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1}) \; \nabla_{\boldsymbol{\phi}} \log q(\{\mathbf{W}_l\}_{l=1}^{L+1}; \boldsymbol{\phi}) \; q(\{\mathbf{W}_l\}_{l=1}^{L+1}; \boldsymbol{\phi}) \; d\mathbf{W} \\
&\quad = \mathbb{E}_{q(\{\mathbf{W}_l\}_{l=1}^{L+1})} \left[\log p(y_m | \mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1}) \; \nabla_{\boldsymbol{\phi}} \log q(\{\mathbf{W}_l\}_{l=1}^{L+1}; \boldsymbol{\phi}) \right] \\
&\quad \approx \frac{1}{S} \sum_{s=1}^{S} \log p(y_m | \mathbf{x}_m, \{\hat{\mathbf{W}}_{l,s}\}_{l=1}^{L+1}) \; \nabla_{\boldsymbol{\phi}} \log q(\{\hat{\mathbf{W}}_{l,s}\}_{l=1}^{L+1}; \boldsymbol{\phi})
\end{aligned} \tag{2.25}$$

where the Monte Carlo expectation is calculated with S samples from the approximation, i.e. $\hat{\mathbf{W}}_{l,s} \sim q(\mathbf{W}_l)$. While this estimator is quite general, not even requiring the likelihood be differentiable w.r.t. $\mathbf{W}$, it suffers from high variance and must often be paired with *control variates* [Ross, 2006].

3. **Pathwise Derivative Estimator:** The *pathwise derivative estimator* (a.k.a. *infinitesimal perturbation analysis* or the *re-parametrization trick*) [Glasserman, 2004, Kingma and Welling, 2014b,a] uses a change of variables to write the integral as a Monte Carlo expectation under some fixed distribution, call it $p_0(\boldsymbol{\xi})$, whose samples can be deterministically transformed into samples from $q(\cdot; \boldsymbol{\phi})$, i.e. $\hat{\mathbf{W}} = g(\hat{\boldsymbol{\xi}}; \boldsymbol{\phi}), \; \hat{\boldsymbol{\xi}} \sim q(\cdot; \boldsymbol{\phi})$. For example, the Normal distribution can be sampled via its location-scale form: $\hat{x} = \mu_x + \sigma_x \odot \hat{\xi}, \; \hat{\xi} \sim \mathsf{N}(0, 1)$. *Inverse transform sampling* is another re-parametrization that would fall into this class. Given p_0

and $g(\cdot;\cdot)$, a Monte Carlo estimate of the gradient can be derived as:

$$
\begin{aligned}
&\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q(\{\mathbf{W}_l\}_{l=1}^{L+1})}\left[\log p(y_m|\mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1})\right] \\
&= \nabla_{\boldsymbol{\phi}} \int_{\mathbf{W}} \log p(y_m|\mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1})\; q(\{\mathbf{W}_l\}_{l=1}^{L+1}; \boldsymbol{\phi})\; d\mathbf{W} \\
&= \nabla_{\boldsymbol{\phi}} \int_{\mathbf{W}} \log p(y_m|\mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1}) \int_{\boldsymbol{\xi}} q(\{\mathbf{W}_l\}_{l=1}^{L+1}|\boldsymbol{\xi}; \boldsymbol{\phi}) p_0(\boldsymbol{\xi})\; d\boldsymbol{\xi}\; d\mathbf{W} \\
&= \nabla_{\boldsymbol{\phi}} \int_{\boldsymbol{\xi}} \int_{\mathbf{W}} \log p(y_m|\mathbf{x}_m, \{\mathbf{W}_l\}_{l=1}^{L+1}) \delta(\{\mathbf{W}_l\}_{l=1}^{L+1} - \{g(\boldsymbol{\xi}_l; \boldsymbol{\phi})\}_{l=1}^{L+1}) p_0(\boldsymbol{\xi})\; d\mathbf{W}\; d\boldsymbol{\xi} \\
&= \nabla_{\boldsymbol{\phi}} \int_{\boldsymbol{\xi}} \log p(y_m|\mathbf{x}_m, \{g(\boldsymbol{\xi}_l; \boldsymbol{\phi})\}_{l=1}^{L+1}) p_0(\boldsymbol{\xi})\; d\boldsymbol{\xi} \\
&= \mathbb{E}_{p_0(\boldsymbol{\xi})}\left[\nabla_{\boldsymbol{\phi}} \log p(y_m|\mathbf{x}_m, \{g(\boldsymbol{\xi}_l; \boldsymbol{\phi})\}_{l=1}^{L+1})\right] \\
&\approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\phi}} \log p(y_m|\mathbf{x}_m, \{g(\hat{\boldsymbol{\xi}}_{l,s}; \boldsymbol{\phi})\}_{l=1}^{L+1})
\end{aligned}
\tag{2.26}
$$

where again S samples are drawn $\hat{\boldsymbol{\xi}}_{l,s} \sim p_0(\boldsymbol{\xi})$. This estimator has been shown to have a manageable variance [Kingma and Welling, 2014a] and is currently the preferred choice in VI for Bayesian NNs [Blundell et al., 2015].

Once a gradient estimator has been chosen, performing VI for Bayesian NNs is simply a matter of performing updates $\boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}_t + \alpha\, \nabla_{\boldsymbol{\phi}} \tilde{\mathcal{J}}_{\text{NN-ELBO}}(\tilde{\mathbf{X}}, \boldsymbol{\phi}_t)$, with α denoting the learning rate, until convergence. Then the approximation can be used to compute quantities such as the posterior predictive distribution, which will require a Monte Carlo expectation:

$$
\begin{aligned}
p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{y}, \mathbf{X}) &\approx \int_{\mathbf{W}} q(\{\hat{\mathbf{W}}_{l,s}\}_{l=1}^{L+1}; \boldsymbol{\phi}_T)\; p(\mathbf{y}^*|\mathbf{x}^*, \{\mathbf{W}_l\}_{l=1}^{L+1})\; d\mathbf{W} \\
&\approx \frac{1}{S} \sum_{s=1}^{S} p(\mathbf{y}^*|\mathbf{x}^*, \{\hat{\mathbf{W}}_{l,s}\}_{l=1}^{L+1})
\end{aligned}
\tag{2.27}
$$

where $\hat{\mathbf{W}}_{l,s} \sim q(\mathbf{W}_l; \boldsymbol{\phi}_T)$. Notice that there are two steps of approximation. The first arises from using an approximation to the true posterior and the second is from the Monte Carlo expectation.

2.3.2.2 Density Networks

Moving on to VI for generative models, the ELBO for a one-stochastic-layer density network is given as:

$$\begin{aligned}\mathcal{J}_{\text{DN-ELBO}}(\mathbf{X}, \boldsymbol{\phi}) &= \mathbb{E}_{q(\{\mathbf{z}_i\}_{i=1}^N)}\left[\log p(\mathbf{X}|\{\mathbf{z}_i\}_{i=1}^N; \boldsymbol{\theta})\right] - \text{KLD}\left[q(\{\mathbf{z}_i\}_{i=1}^N; \boldsymbol{\phi}) \,||\, p(\mathbf{z})\right] \\ &= \sum_{i=1}^N \mathbb{E}_{q(\mathbf{z}_i)}\left[\log p(\mathbf{x}_i|\mathbf{z}_i; \boldsymbol{\theta})\right] - \text{KLD}\left[q(\mathbf{z}_i; \boldsymbol{\phi}_i) \,||\, p(\mathbf{z})\right]\end{aligned} \tag{2.28}$$

where again I have assumed a mean-field factorization of the approximation: $p(\{\mathbf{z}_i\}_{i=1}^N|\mathbf{X};;\boldsymbol{\theta}) \approx q(\{\mathbf{z}_i\}_{i=1}^N; \boldsymbol{\phi}) = \prod_{i=1}^N q(\mathbf{z}_i; \boldsymbol{\phi}_i)$. Whereas with conditional models the posterior factorizes over weight matrices, here the variational posterior factorizes across per-data-point latent variables, each with their own parameters $\boldsymbol{\phi}_i$. VI for density networks involves the same challenges described for conditional models in the previous subsection. Again, as stochastic gradient methods are the target optimization strategy, we require cheap gradients of the expected log likelihood $\mathbb{E}_{q(\mathbf{z}_i)}\left[\log p(\mathbf{x}_i|\mathbf{z}_i; \boldsymbol{\theta})\right]$. The same three techniques—the delta method, score function estimators [Mnih and Gregor, 2014], and pathwise derivatives [Kingma and Welling, 2014b] can all be used here—but the pathwise estimator has been the particular method of choice for density networks [Kingma and Welling, 2014b, Kingma et al., 2014, Burda et al., 2016]:

$$\mathbb{E}_{q(\mathbf{z}_i)}\left[\log p(\mathbf{x}_i|\mathbf{z}_i; \boldsymbol{\theta})\right] \approx \frac{1}{S}\sum_{s=1}^S \log p(\mathbf{x}_i|g(\hat{\boldsymbol{\xi}}_{i,s}; \boldsymbol{\phi}_i); \boldsymbol{\theta}), \quad \hat{\boldsymbol{\xi}}_s \sim p_0(\hat{\boldsymbol{\xi}}) \tag{2.29}$$

with p_0 again being the fixed distribution subject to transformation.

Calculating the expectations and their gradients is not the only obstacle with performing VI for density networks. There is an additional cost, an $\mathcal{O}(N)$ memory requirement for storing the variational parameters, which limits their application to large data sets. *Amoritzed VI* [Gershman and Goodman, 2014] alleviates this cost by using a model to predict the variational parameters

whenever they are required:

$$\boldsymbol{\phi}_i = \gamma(\mathbf{x}_i; \boldsymbol{\psi}) \tag{2.30}$$

where $\gamma(\cdot; \boldsymbol{\psi})$ is the *inference model* with parameters $\boldsymbol{\psi}$. NNs [Kingma and Welling, 2014b, Rezende et al., 2014] and Gaussian processes [Tran et al., 2016] both have been used successfully as inference models. Writing the ELBO with both the pathwise derivative function $g(\cdot; \cdot)$ and the amortization model $\gamma(\cdot; \cdot)$, we have

$$\begin{aligned}
\mathcal{J}_{\text{DN-ELBO}}(\mathbf{X}, \boldsymbol{\phi}) &= \mathcal{J}_{\text{DN-ELBO}}(\mathbf{X}, \boldsymbol{\psi}) \\
&= \sum_{i=1}^{N} \mathbb{E}_{p_0(\boldsymbol{\xi})} \left[\log p(\mathbf{x}_i | g(\hat{\boldsymbol{\xi}}_{i,s}; \boldsymbol{\phi}_i); \boldsymbol{\theta}) \right] - \text{KLD}\left[q(\mathbf{z}_i; \boldsymbol{\phi}_i) \mid\mid p(\mathbf{z}) \right] \\
&= \sum_{i=1}^{N} \mathbb{E}_{p_0(\boldsymbol{\xi})} \left[\log p(\mathbf{x}_i | g(\hat{\boldsymbol{\xi}}_{i,s}; \gamma(\mathbf{x}_i; \boldsymbol{\psi})); \boldsymbol{\theta}) \right] - \text{KLD}\left[q(\mathbf{z}_i; \gamma(\mathbf{x}_i; \boldsymbol{\psi})) \mid\mid p(\mathbf{z}) \right].
\end{aligned} \tag{2.31}$$

Notice that the local variational parameters $\boldsymbol{\phi}_i$ have all been replaced by the global (shared) parameters $\boldsymbol{\psi}$ of the inference model. Not only does amortization help with the model's memory footprint, but it also expedites inference for a new (test) point $\mathbf{x}^*$. Without the inference model, obtaining $\mathbf{x}^*$'s posterior would require fitting a new distribution $q(\mathbf{z}^*; \boldsymbol{\phi}^*)$ from some random initialization, usually requiring several optimization steps. But with amortization, new parameters $\boldsymbol{\phi}^*$ are produced by simply running the inference model. Note that using an inference model, no matter how deep or complex it is, does not necessarily result in a more flexible posterior approximation. Using independent parameters $\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N\}$ is actually the upper limit of flexibility since they are not constrained by a shared parametrization. Yet, amortization can lead to a better approximation in practice because the inference model can act as a regularizer and make optimization better conditioned.

Looking at Equation 2.31 and recalling the base AE definition in Equation 2.6, we see a striking similarity: $\mathbf{x}_i$ now appears on both sides of the conditional distribution $p(\mathbf{x}_i | g(\hat{\boldsymbol{\xi}}_{i,s}; \gamma(\mathbf{x}_i; \boldsymbol{\psi})); \boldsymbol{\theta})$.

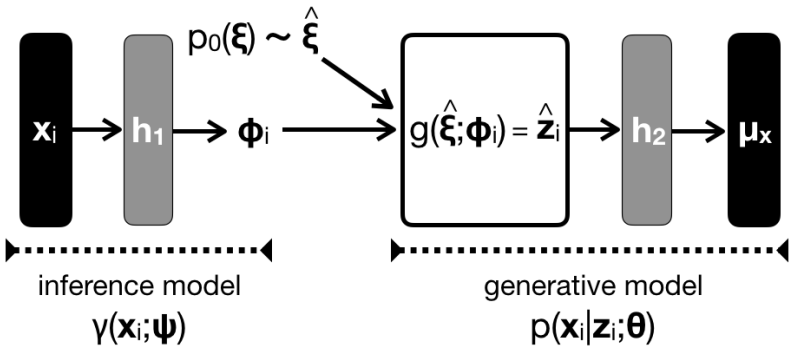


Figure 2.3: *Variational Autoencoder.*

The form is reminiscent of an AE, and this is why the composition of a density network with an inference model is known as a *variational autoencoder* (VAE) [Kingma and Welling, 2014b, Rezende et al., 2014]. A diagram of the VAE computation path is given in Figure 2.3. We see that there is a fully differentiable path between $\boldsymbol{\mu}_{\mathbf{x}_i}$ and $\mathbf{x}_i$, thus allowing gradients to be used to update both the generative ($\boldsymbol{\theta}$) and inference ($\boldsymbol{\psi}$) parameters simultaneously—just as we would take gradients w.r.t. both the decoder and encoder of a regular AE. Yet note that $\mathbf{x}_i$ being on both sides of the likelihood conditional does not make the model improperly generative. The RHS $\mathbf{x}_i$ is an artifact of the end-to-end composition of the generative and inference processes, and proper sampling is being done via the introduction of $\hat{\boldsymbol{\xi}}$. If we view the regular AE from this perspective, we can think of the encoder as parametrizing a degenerate distribution $\delta(\mathbf{z}_i - h(\mathbf{x}_i; \cdot))$, where $h(\mathbf{x}_i; \cdot)$ are the hidden units computed at some intermediate layer. As this distribution is just a point mass concentrated at $h(\mathbf{x}_i; \cdot)$, the only sample it can return is $h(\mathbf{x}_i; \cdot)$, resulting in a fully deterministic computation path.

One may wonder if amortized inference is useful for models with global variables, such as conditional NNs. Upon first thought, the answer is 'no' since by definition global variables are already shared across all data points. However, upon further thought, amortization could be used to quickly generate posteriors over *multiple data sets*; this is the idea behind the *neural statistician* (NS) [Edwards and Storkey, 2017]. The NS is characterized by a statistic model $q(\cdot|\mathcal{D})$ that takes as input an entire data set $\mathcal{D}$ and produces some aggregate representation. Specifically, Ed-

wards and Storkey [2017] use mean pooling to get an exchangeable, fixed-dimension summary vector. I have done work (not described in this thesis) using amortization within the frequentist paradigm. The *amortized bootstrap* [Nalisnick and Smyth, 2017a] uses a shared model to generate the parameter estimates for each bootstrap replicate. This allows an unlimited number of parameter samples to be drawn at test time, providing a richer distribution than one restricted to the original K bootstrap replicates.

Chapter 3

Survey of Neural Network Priors

We demand rigidly defined areas of doubt and uncertainty!

Douglas Adams
The Hitchhiker's Guide to the Galaxy

Having covered the basics of Bayesian NNs and strategies for inferring their posterior, I now turn to the focal point of the dissertation: prior distributions for both conditional NNs and density networks. Surprisingly, a broad review of Bayesian NN priors has been performed by only Robinson [2001], which is now considerably out of date. Thus, in this chapter I survey the existing work on NN priors, some of which was performed in the early days of Bayesian NNs and therefore also discussed by Robinson [2001]. However, most of the work is recent, some having been conducted concurrently with my own work to be presented in the coming chapters.

NNs have been applied to a myriad of different problems over the past thirty years, and this of course makes it impossible to discuss every prior ever used for a NN. Instead, I attempt to summarize broad themes from the literature that pertain to core NN methodology. For instance, for conditional NNs, this discussion dates back to Neal [1993] and centers around the use of

Gaussian priors, their relationship to Gaussian processes, and why this relationship has provoked the use of heavy-tailed priors. Priors for density networks, on the other hand, have only recently become an active area of research. Accordingly, I point to various open questions.

3.1 Conditional Models

As conditional NNs were the predominant flavor of Bayesian NNs until of late, choosing their priors has been given the most thought. Regularization is usually of the foremost concern with conditional models, and therefore I will mainly discuss the work on shrinkage inducing priors, which is the most abundant. Yet, I will close with a brief discussion of more exotic priors such as discrete and noninformative ones.

3.1.1 Gaussian Priors

Not surprisingly, mostly due to its connection to *weight decay* [Plaut et al., 1986] and the *ridge penalty* [Hoerl and Kennard, 1970], the zero-mean Gaussian distribution was the first NN prior to be explored [Buntine and Weigend, 1991, MacKay, 1992c, Neal, 1993]: $p(\mathbf{W}_l) = \mathsf{N}(\mathbf{0}, \mathbf{\Sigma})$. The choice is a reasonable one since the zero-centering induces shrinkage and the Gaussian assumption implies smoothness. A thorough analysis of the prior by Neal [1994] showed that, assuming $\mathbf{h}_l$ is bounded, placing Gaussian priors on the output parameters and taking the number of hidden units to infinity results in the NN becoming a *Gaussian process* (GP) [Rasmussen, 2004]. To see this, I write the NN in terms of its last layer

$$\psi(\mathbf{x}_i) = \mathbf{h}_L(\mathbf{x}_i)\mathbf{w}_{L+1} = \sum_{k=1}^{H} w_{L+1,k} h_{L,k}(\mathbf{x}_i),$$

and place the Gaussian prior $w_{L,j} \sim \mathsf{N}(0, \sigma^2/H)$ (where H is the number of hidden units in the last hidden layer and σ^2 is some constant) on the hidden-to-output weights. It is easy to compute the first two moments, them being:

$$\mathbb{E}\left[\sum_{k=1}^{H} w_{L+1,k} h_{L,k}(\mathbf{x}_i)\right] = \sum_{k=1}^{H} \mathbb{E}\left[w_{L+1,k}\right] h_{L,k}(\mathbf{x}_i) = 0$$

and

$$\begin{aligned}\mathbb{E}\left[\left(\sum_{k=1}^{H} w_{L+1,k} h_{L,k}(\mathbf{x}_i)\right)^2\right] &= \mathbf{h}_L(\mathbf{x}_i)\mathbf{h}_L(\mathbf{x}_i)^T \sum_{k=1}^{H} \mathbb{E}\left[w_{L+1,k}^2\right] \\ &= \mathbf{h}_L(\mathbf{x}_i)\mathbf{h}_L(\mathbf{x}_i)^T H\sigma^2/H = \sigma^2 \mathbf{h}_L(\mathbf{x}_i)\mathbf{h}_L(\mathbf{x}_i)^T.\end{aligned}$$

From the central limit theorem, the distribution on the NN output is then

$$p(\psi(\mathbf{x}_i)) = \mathsf{N}(0, \sigma^2 \mathbf{h}_L(\mathbf{x}_i)\mathbf{h}_L(\mathbf{x}_i)^T).$$

To show the equivalence to a GP, we need to show the joint distribution between function evaluations is Normal: $p(\psi(\mathbf{x}_i), \psi(\mathbf{x}_j)) = \mathsf{N}(\mathbf{0}, \mathbf{K})$. The crucial quantity is the covariance between function evaluations, which Neal [1994] shows is:

$$\begin{aligned}\mathbb{E}\left[\psi(\mathbf{x}_i)\psi(\mathbf{x}_j)\right] &= \mathbb{E}\left[\left(\sum_{k=1}^{H} w_{L+1,k} h_{L,k}(\mathbf{x}_i)\right)\left(\sum_{k=1}^{H} w_{L+1,k} h_{L,k}(\mathbf{x}_j)\right)\right] \\ &= \sum_{k=1}^{H} \mathbb{E}\left[(w_{L+1,k})^2\right] h_{L,k}(\mathbf{x}_i) h_{L,k}(\mathbf{x}_j) = \sigma^2 \kappa(\mathbf{x}_i, \mathbf{x}_j).\end{aligned} \tag{3.1}$$

The multivariate CLT then implies $p(\psi(\mathbf{x}_i), \psi(\mathbf{x}_j)) = \mathsf{N}(\mathbf{0}, \mathbf{K})$ where $\kappa(\mathbf{x}_i, \mathbf{x}_i) = \mathbf{h}_L(\mathbf{x}_i)\mathbf{h}_L(\mathbf{x}_i)^T$ and $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{h}_L(\mathbf{x}_i)\mathbf{h}_L(\mathbf{x}_j)^T$. It is then natural to consider how placing priors on the preceding layers affects the GP equivalence, i.e. $w_{l,j,k} \sim \mathsf{N}(0, \sigma^2/H)$. Lee et al. [2018] show that the

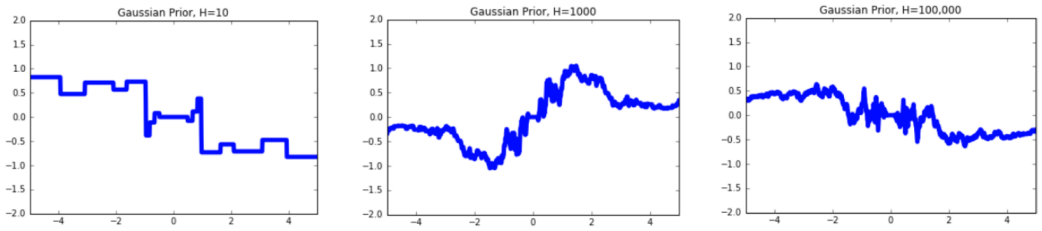
distribution at layer l can be characterized by the recurrence relation

$$\mathbb{E}\left[\psi_l(\mathbf{x}_i)\psi_l(\mathbf{x}_j)\right] = \sigma_l^2 \mathbb{E}_{\psi_{l-1}\sim \mathrm{GP}(\mathbf{0},\mathbf{K}_{l-1})}\left[f_l(\psi_{l-1}(\mathbf{x}_i))f_l(\psi_{l-1}(\mathbf{x}_j))\right]. \tag{3.2}$$

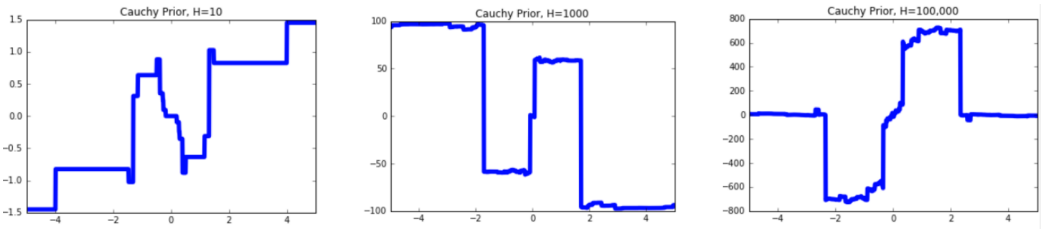
This quantity can be found in closed-form for ReLU functions [Lee et al., 2018].

Lastly, I recreate Neal [1994]'s simulations to show the GP behavior of NNs. The blue lines in Figure 3.1 are sampled from a one-hidden-layer NN with one input dimension (x-axis), H step-function (i.e. discrete) hidden units, and one output dimension (y-axis). They represent a draw of $\psi(\mathbf{x}_i)$. Subfigure (a) shows sampled functions when a Gaussian prior is placed on the weights and as the number of hidden units increases ($H = \{10, 1000, 100, 000\}$). When $H = 10$, one can see that there are 'jumps' in the regression line, meaning that the function can change drastically depending on which hidden units are active. If it is desired that the NN learn latent features, then these jumps are a good thing for the NN to exhibit. However, as $H \to \infty$, the line becomes Brownian motion: it wanders according to very small, independent increments. This is markedly different behavior than the jumps we saw when $H = 10$, and in the words of Neal [1994]: "the contributions of individual units are all negligible, and consequently, these units do not represent *hidden features* that capture important aspects of the data."

de G Matthews et al. [2018] show that this convergence to Brownian motion is a concern in practice, with Bayesian NNs taking on GP-like behavior when using as few as 20 hidden units. NN depth helps combat the behavior but only for a few 10's of units more. Yet, the convergence to GP behavior occurs for a wider class of priors than just Gaussian. In fact, the *central limit theorem* (CLT) underlies the phenomenon, and therefore even NNs with discrete weights will converge to GPs via the Lyapunov CLT if the hidden units are real valued. In order to stop the convergence, Bayesian NNs must be defined so that the CLT's underlying assumptions are broken [de G Matthews et al., 2018]. Doing so presents challenges, but I discuss priors that stop the onset of the CLT in the next section. While Bayesian NNs seem to insist on converging to



(a) Bayesian Neural Network with Gaussian Prior



(b) Bayesian Neural Network with Cauchy Prior

Figure 3.1: *Gaussian Process Behavior of Bayesian Neural Networks.*

GPs, this tendency can be a good thing for GP users as it means that Bayesian NNs can be used as plug-in approximations for GPs. Swapping a GP for a Bayesian NN results in considerable computational savings for large data sets since the user is trading the GP's $\mathcal{O}(N^3)$ cost of matrix inversions and determinants for a Bayesian NN's $\mathcal{O}(N)$ training data dependence. Snoek et al. [2015] use this very idea to perform scalable Bayesian optimization.

3.1.2 Heavy-Tailed Priors

One way to break the CLT is to draw the NN's weights from a distribution with infinite variance—such as from a member of the symmetric stable family with index $\alpha < 2$. These distributions have 'heavy,' sub-exponential tails, which allows "some of the hidden units in an infinite network [to] have output weights of significant size, allowing them to represent *hidden features*" [Neal, 1994]. The *Cauchy* distribution is in the stable family ($\alpha = 1$) and therefore meets the criterion. To demonstrate how the NN behavior changes under priors of this form, in Subfigure (b) of Figure 3.1 I sample functions from the same NN architecture but with a Cauchy prior[1]. We see

[1]This is another figure recreated from Neal [1994].

that the line exhibits large jumps even as $H \to \infty$, which means that the regression function is still dependent on which hidden units are active.

Unfortunately, heavy-tailed distributions are not easy to work with analytically or computationally. The Cauchy, for instance, does not have finite moments of any order. Neal [1994] recommends the *student-t* distribution as a compromise, which is CLT-breaking when its degrees of freedom parameter is less than two. A student-t can be conveniently represented as a Gaussian scale mixture of the form

$$w_{l,i,j} \sim \mathsf{N}(0, \sigma^2), \quad \sigma^2 \sim \Gamma^{-1}(\nu/2, s\nu/2)$$

where $\Gamma^{-1}(\cdot, \cdot)$ denotes the *inverse gamma* distribution. The marginal distribution is then a zero-centered student-t with scale s and ν degrees of freedom:

$$\text{student-t}(w; 0, s, \nu) = \int_\sigma \Gamma^{-1}(\sigma^2; \nu/2, s\nu/2)\mathsf{N}(w|0, \sigma^2)d\sigma.$$

Posterior inference for Bayesian NN's with student-t priors can be performed efficiently by using MCMC or VI to compute the expectation under the Gaussian and then by using Gibbs sampling steps to update the student-t's parameters (if desired). Decoupling inference for the weights and hyperparameters in this way allows the user to easily monitor and control the NN's CLT conditions.

In other work on heavy-tailed distributions, Laplacian priors were proposed by Williams [1995] and Goutte and Hansen [1997]. Like the student-t, the Laplace distribution can be represented as a Gaussian scale mixture but with the exponential distribution as the hyperprior:

$$w_{l,i,j} \sim \mathsf{N}(0, \sigma^2), \quad \sigma^2 \sim \text{Exponential}(b).$$

Williams [1995]'s argument for the prior is driven by Jaynes [1968]'s principles of transformation.

Since most NN architectures have symmetry w.r.t. the sign of the weights—as I discuss in Section 2.3.1.1—the prior should be a function of $|w_{l,i,j}|$. Gaussian priors, on the other hand, are a function of the signed weight and therefore do not reflect the symmetry. Both Williams [1995] and Goutte and Hansen [1997] present experiments using MAP estimates and demonstrate the ability to prune weights without suffering performance degradation. However, Kabán [2007] and Castillo et al. [2015] (among others) have shown that Laplace priors do not induce the same amount of useful sparsity in their posteriors as is found in their MAP estimates. Thus, Laplace priors may not provide enough regularization under fully Bayesian treatments of NNs.

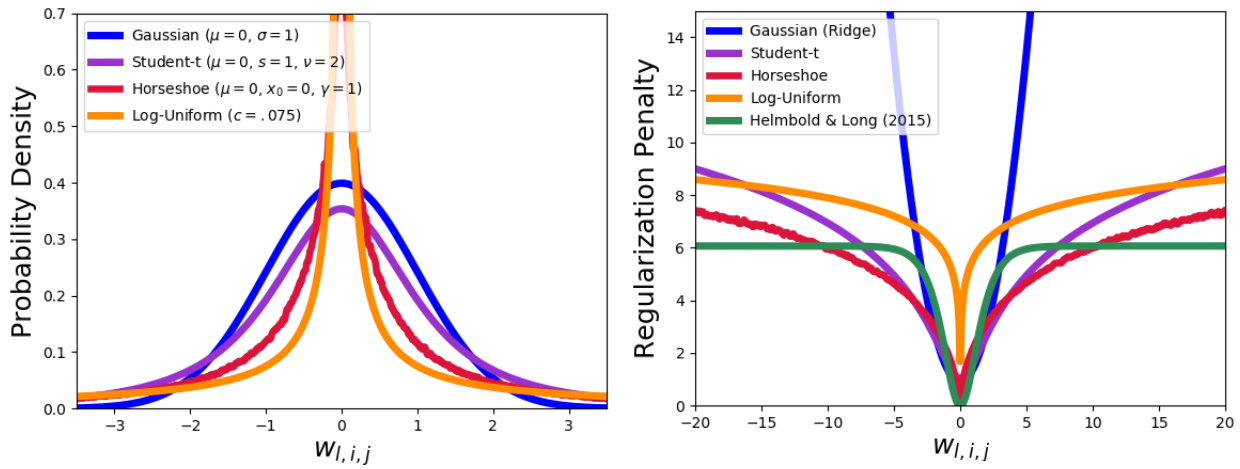
Recently, advances in VI have allowed more aggressive heavy-tailed priors to be considered, such as the *horseshoe prior* (HSP) [Carvalho et al., 2009]. It too has a convenient hierarchical form:

$$w_{l,i,j} \sim \mathsf{N}(0, \sigma^2), \quad \sigma \sim \mathsf{Cauchy}^{+}(x_0 = 0, \gamma = 1)$$

where $\mathsf{Cauchy}^{+}(\cdot, \cdot)$ denotes the *half-Cauchy* distribution—the Cauchy restricted to the positive real numbers. The HSP has no closed-form marginal expression, but it can be plotted, which I do in Figure 3.2 (a) using the red line. From the figure we see the HSP has heavy tails like the student-t, but unlike the student-t, it asymptotes to the left and right of zero. This means that it allows the NN weights to grow large, like the student-t, but has fierce shrinkage near zero, making the HSP a stronger regularizer than the Gaussian, Laplacian, or student-t densities. Ghosh and Doshi-Velez [2017] describe Bayesian NNs with HSPs placed on their pre-activation values in order to perform model selection over the number of hidden units.

Any time a hyperprior is placed on the first-level prior's scale parameter, as is the case for the student-t, Laplace, and HSP, then it is amenable to the *automatic relevance determination* (ARD) framework [MacKay, 1994, Neal, 1994]. ARD, when applied to NNs, ties the variance of all outgoing weights from a particular hidden unit:

$$w_{l,i,j} \sim \mathsf{N}(0, \sigma^2_{l,i,\cdot}), \quad \sigma_{l,i,\cdot} \sim p_\sigma(\lambda) \tag{3.3}$$



(a) Density Functions

(b) Regularization Penalties

Figure 3.2: *Heavy-Tailed Priors.*

where λ denotes the parameters of the scale hyperprior and $\sigma_{l,i,\cdot}$ signifies that all of the weights in the ith row of $\mathbf{W}_l$ share the same scale. Giving the prior this structure ensures that all the weights multiplied with hidden unit $h_{l-1,i}$ grow and shrink together. This is a form of group regularization: all weights within the same row must be shrunk together if any are shrunk. The end result is essentially feature / hidden unit selection since if all of a unit's outgoing weights are near zero, then the unit in inconsequential to the model output.

Lastly, heavy-tailed priors without hierarchical forms have been proposed by Williams [1999], Toussaint et al. [2006], and Kingma et al. [2015]. Beginning with the earliest work, Williams [1999] further developed his invariance arguments (used to motivate Laplace priors above) to derive the prior:

$$p(\mathbf{w}_i) \propto \frac{1}{||\mathbf{w}_i||_p^\gamma}$$

where $||\mathbf{w}_i||_p$ denotes the pth norm of the weight vector $\mathbf{w}_i$ and γ is a positive constant. Focusing on invariance in the activation functions in particular, Toussaint et al. [2006] proposed the following values for Williams [1999]'s constants: $p(\mathbf{w}_i) \propto 1/||\mathbf{w}_i||_{p=2}^{\gamma=d+1}$ where d is the number of elements in the vector. In the most recent work, a variational-inference-based analysis lead Kingma et al. [2015] to find that the log-uniform distribution is the Bayesian prior that best

mimics the *dropout*[2] [Srivastava et al., 2014] regularization mechanism:

$$p(w_{l,i,j}) \propto \frac{c}{\mid w_{l,i,j} \mid} \tag{3.4}$$

where c is a scalar constant and $\mid \cdot \mid$ denotes the absolute value. Interestingly, this is the same as Williams [1999]'s prior with $p = 1$, $\gamma = 1$ but factorized across the vector, i.e. $p(\mathbf{w}_{l,i}) = \prod_{j=1}^{d} p(w_{l,i,j})$. In Figure 3.2 (a), I plot the log-uniform density (orange line) against the Gaussian (blue), student-t (purple), and horseshoe (red) densities. I omitted the other priors mentioned to prevent clutter. We see that the log-uniform is similar to the HSP but has even stronger shrinkage and flatter tails. Molchanov et al. [2017] present experiments using the log-uniform prior and find that indeed it induces weight sparsity[3].

When finding a MAP estimate, the log-uniform prior corresponds to the regularization penalty $-\log\{1/ \mid w_{l,i,j} \mid\} = \log \mid w_{l,i,j} \mid$. Interestingly, Helmbold and Long [2015] analyze dropout from the perspective of optimizing a penalized loss and find a regularization penalty that looks qualitatively similar to $\log \mid w_{l,i,j} \mid$. In subfigure (b) of Figure 3.2, I plot Helmbold and Long [2015]'s function (their Equation 10) in green against the regularization penalties derived from the mentioned distributions by taking the negative logarithm of their density functions. From this figure, we see that all penalties except the Gaussian have the ability to 'turn off' in the sense that the penalty plateaus as the weight moves away from zero. This is from heavy-tailed densities' unique ability to distribute their mass away from their center. Although Helmbold and Long [2015]'s analysis is not within the Bayesian framework, their derived penalty is clearly similar in character to the densities discussed here, thus implying connections between dropout, heavy-tailed densities, and certain invariance properties. I explore connections between dropout and scale mixtures in the next chapter, but I conjecture there is more to be uncovered in this area, especially in regards to the invariances induced by dropout.

[2]Dropout will be defined and discussed at length in a later chapter.

[3]Although, Hron et al. [2017] argues that the log-uniform prior induces an ill-posed posterior that does not induce sparsity, in general, and that the observed sparsity is an optimization artifact.

3.1.3 Other Priors: Discrete and Noninformative

While the vast majority of work has been done on shrinkage-inducing priors, there are at least two more types of prior to note. The first is discrete. NNs with binary weights are defined via Bernoulli priors [Saad and Marom, 1990, Mayoraz and Aviolat, 1996, Soudry et al., 2014, Courbariaux et al., 2015, Hubara et al., 2016], and they are of interest for their computational efficiency [Soudry et al., 2014] and biological plausibility [Baldassi et al., 2007]. Unfortunately, binary NNs are exceptionally difficult to train since they are not amenable to backpropagation (gradients are undefined), and the majority of work has focused on inventing differentiable relaxations. The most widely studied solution is the *straight-through* estimator [Bengio et al., 2013b]: binary weights are sampled for the forward pass but their mean is used during the backward pass. Solutions relying on the CLT [Soudry et al., 2014, Shayar et al., 2018] have also been proposed.

The second prior type that deserves mention are so-called 'noninformative' priors. These will be discussed in further detail in Chapter 5, but to briefly summarize, noninformative priors are derived to be invariant under model re-parametrization. Jeffreys priors [Jeffreys, 1946] are the best known of this type and are defined as $p_{\text{JEFF}}(\boldsymbol{\pi}) \propto \sqrt{\det F(\boldsymbol{\pi})}$ where $\boldsymbol{\pi}$ are the model parameter and $F(\boldsymbol{\pi})$ is the Fisher information matrix. Lee [2005] derives the Jeffreys prior for a NN's output parameters and shows it produces results comparable to Neal [1994]'s student-t—with the added benefit of not having any hyperparameters to tune. However, deriving the Jefferys prior for the weights of any preceding layer is intractable, which is the problem my work addresses.

3.2 Density Networks

Gaussian densities were also the first priors investigated for density networks [MacKay and Gibbs, 1999], and the use of Gaussians has continued through to the modern work on VAEs [Kingma and Welling, 2014b, Rezende et al., 2014, Kingma et al., 2014, Burda et al., 2016]. The

widespread use of Gaussian priors seems sensible since density networks can use their NN layers to create complex transformations for warping the latent variables. Moreover, D. Kingma reported that preliminary experiments using Laplacian and scale mixture priors "did not result in better generative models."[4]

However, Hoffman and Johnson [2016] analyzed the VAE's ELBO and showed that it can be re-written by decomposing the KLD term as:

$$\mathcal{J}_{\text{DN-ELBO}}(\mathbf{X}, \boldsymbol{\phi}) = \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{z}_i)} \left[\log p(\mathbf{x}_i|\mathbf{z}_i; \boldsymbol{\theta})\right] - \mathbb{I}_{q(i,\mathbf{z}_i)} \left[i, \mathbf{z}_i\right] - \text{KLD} \left[q(\mathbf{z}_i) \mid\mid p(\mathbf{z})\right] \tag{3.5}$$

where $q(\mathbf{z}) = \frac{1}{N} \sum_{i=1}^{N} q(\mathbf{z}_i|\mathbf{x}_i)$ is the marginal distribution over latent variables and $\mathbb{I}_{q(i,\mathbf{z}_i)} \left[i, \mathbf{z}_i\right]$ is the mutual information between a latent variable and its index. Hoffman and Johnson [2016] found that the first two terms are optimized well by the basic VAE formulation, but the third term is often still considerably large even upon convergence, effectively decreasing the ELBO. Large values of $\text{KLD} \left[q(\mathbf{z}_i) \mid\mid p(\mathbf{z})\right]$ imply that the model is finding it difficult to match the aggregate posterior and the prior. The VAE wants to make the marginal posterior significantly richer but needs to sacrifice capacity in order to match the prior. Hoffman and Johnson [2016] recommend that "we should investigate multimodal priors that can meet $q(\mathbf{z})$ halfway."

There have been several proposed solutions to this marginal matching problem. Makhzani et al. [2016] were actually the first to highlight the pathology, and they proposed optimizing the marginal posterior directly via an adversarial loss. Unfortunately, this modifies the VI objective away from the foundationally-sound ELBO. Subsequent work by Chen et al. [2017] fix the problem without changing the ELBO by simply parametrizing the prior with an autoregressive transformation, which allows it to warp itself as necessary in order to meet the demands of the marginal posterior. Tomczak and Welling [2018] attack the problem more directly yet still within the ELBO framework. Firstly, they show that the prior that maximizes the ELBO is, unsurpris-

[4]Quote taken from D. Kingma's comment posted in *r/MachineLearning*: "Behind the scenes I did compare with sparse priors (Laplace and Gaussian scale mixtures) but these didn't result in better generative models."

ingly, the marginal posterior itself. However, setting the prior to the aggregated posterior can lead to overfitting and would be expensive to evaluate, requiring a sum over all the training data. In turn, Tomczak and Welling [2018] define the prior as the marginal posterior over K *pseudo-inputs*: $p(\mathbf{z}) = \frac{1}{K}\sum_{k=1}^{K} q(\mathbf{z}_k|\mathbf{u}_k)$ with $\mathbf{u}_k$ being the kth pseudo-input. The authors term this the *variational mixture of posteriors prior*, or *VampPrior* for short. The pseudo-inputs are optimized along with the other generative and variational parameters; yet this does not lead to overfitting since K is chosen to be much smaller than N[5]. Lastly, another path to solving the marginal matching problem is through re-scaling the KLD term in the ELBO. Higgins et al. [2017] propose training the VAE via

$$\mathcal{J}_{\beta-\text{VAE}}(\mathbf{X}, \boldsymbol{\phi}) = \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{z}_i)}\left[\log p(\mathbf{x}_i|\mathbf{z}_i;\boldsymbol{\theta})\right] - \beta\text{KLD}\left[q(\mathbf{z}_i;\boldsymbol{\phi}_i) \;||\; p(\mathbf{z})\right]$$

where $\beta > 0$. $\mathcal{J}_{\beta-\text{VAE}}$ was proposed originally for learning disentangled latent representations, but Hoffman et al. [2017] show that training with that objective is equivalent to regular ELBO optimization under the prior

$$r(\mathbf{z}) = \frac{q(\mathbf{z})^{1-\beta}p(\mathbf{z})^{\beta}}{Z(\beta)}, \quad Z(\beta) = \int_{\mathbf{z}} q(\mathbf{z})^{1-\beta}p(\mathbf{z})^{\beta}\, d\mathbf{z}$$

with $Z(\beta)$ being the normalizing constant. The implicit prior $r(\mathbf{z})$ interpolates between the aggregated posterior $q(\mathbf{z})$ and the explicit prior $p(\mathbf{z})$. We can think of $p(\mathbf{z})$ as multiplicatively regularizing $q(\mathbf{z})$ (with the regularization strength controlled by β), and therefore it serves the same role as choosing $K << N$ for the VampPrior.

Just as with conditional models, there is interest in using discrete priors on the latent space of density networks. In addition to the computational benefits noted earlier, discrete latent variables have the potential to be more interpretable than continuous ones. Discrete variables are clearly 'on' or 'off' and thus can be visualized with a grid of black-and-white boxes and the di-

[5]Experiments showed setting K to be between 200 and 400 to be sufficient.

mensions commonly active across a group can be readily identified. Yet, again because of a lack of available gradients, training density networks with discrete variables is difficult. Score function [Mnih and Gregor, 2014] and pass-through estimators could be used to attain gradients, but the most promising approach seems to be a continuous relaxation of the Gumbel-max 'trick' for sampling discrete variables. Instead of using the non-differential argmax operator, Maddison et al. [2017] and Jang et al. [2017] propose passing the Gumbel noise through the softmax operator so that the random variable is still amenable to pathwise gradients. Unfortunately, using this relaxation makes the Monte Carlo pathwise gradients biased, but follow-up work by Tucker et al. [2017] has shown the bias can be eliminated by using the Gumbel-softmax trick not as the primary estimator but as a control variate for a score function estimator.

Before ending my discussion of priors for density networks, I must note that using multiple stochastic layers induces complex priors on the subsequent variables, and this in turn muddles much of the discussion above. To elaborate, assume $\mathbf{z}_1$ is given a Gaussian prior and the prior on $\mathbf{z}_2$ is a Gaussian parametrized by $\mathbf{z}_1$, i.e. $p(\mathbf{z}_2|\mathbf{z}_1) = \mathsf{N}(\boldsymbol{\mu}(\mathbf{z}_1), \boldsymbol{\Sigma}(\mathbf{z}_1))$. While the prior on $\mathbf{z}_2$ is conditionally Gaussian, the marginal prior is decidedly not Gaussian: $p(\mathbf{z}_2) = \int_{\mathbf{z}_1} p(\mathbf{z}_1)\mathsf{N}(\mathbf{z}_2; \boldsymbol{\mu}(\mathbf{z}_1), \boldsymbol{\Sigma}(\mathbf{z}_1))d\mathbf{z}_1$. Performing this marginalization is intractable. Density networks with multiple stochastic latent variables have been found difficult to train [Rezende et al., 2014, Sønderby et al., 2016, Zhao et al., 2017], and hence, understanding their training dynamics—and how the choice of prior affects their performance—is an open problem.

Chapter 4

Multiplicative Noise as an Induced Prior

We have only begun to learn how to detect and measure,...finding in each Deeper Significance and trying to string them all together like terms of a power series hoping to zero in on the tremendous and secret Function whose name, like the permuted names of God, cannot be spoken...

Thomas Pynchon

Gravity's Rainbow

Training deep NNs (DNNs) under multiplicative noise—by introducing a random variable into the inner product between a hidden layer and a weight matrix—has led to significant improvements in predictive accuracy. Typically the noise is drawn from the Bernoulli distribution, which is equivalent to randomly dropping neurons from the network during training, and hence the practice has been termed *dropout* [Hinton et al., 2012, Srivastava et al., 2014]. Yet, Gaussian [Srivastava et al., 2014] and beta [Tomczak, 2013] noise have been shown to be just as effective. Despite its empirical success, regularization by way of multiplicative noise is not well understood

theoretically, especially for DNNs. The multiplicative noise term eludes analysis as a result of being buried within the DNN's composition of non-linear functions.

Adopting a Bayesian perspective, I show that we can develop closed-form analytical expressions that describe the effect of training under multiplicative noise[1]. When a zero-mean Gaussian prior is placed on the weights of the DNN, the multiplicative noise variable induces a *Gaussian scale mixture* (GSM), i.e. the variance of the Gaussian prior becomes a random variable whose distribution is determined by the multiplicative noise model. Conveniently, GSMs can be represented hierarchically with the scale mixing variable—in this case the multiplicative noise—becoming a hyperprior. This allows us to circumvent the problematic coupling of the noise and likelihood through reparametrization, making them conditionally independent. Once in this form, variational EM can be used to derive updates for the multiplicative noise terms and make the regularization mechanism explicit. While the GSM reparametrization and learning procedure are not novel in their own right, employing them to understand multiplicative noise in NNs is a novel contribution. Moreover, the analysis is not restricted by the network's depth or activation functions, as previous attempts at understanding dropout have been.

As for its practical implications, my analysis suggests a new criterion for principled model compression. The closed-form regularization penalty I isolate naturally suggests a new weight pruning strategy. Interestingly, my new rule is in stark disagreement with the commonly used *signal-to-noise ratio* (SNR) [Graves, 2011, Blundell et al., 2015]. The SNR is quick to prune weights with large variances, deeming them noisy, but my approach finds large variances to be an essential characteristic of robust weights that are likely to generalize. Experimental results on well-known predictive modeling tasks show that my weight pruning mechanism is not only superior to the SNR criterion by a wide margin, but also competitive to retraining with soft-targets produced by the full network [Hinton et al., 2014, Ba and Caruana, 2014]. In each experiment my method was able to prune at least 20% more of the model's parameters than SNR before seeing a vertical

[1]The work in this chapter is from Nalisnick et al. [2015]

asymptote in test error. Furthermore, in two of these experiments, the performance of models pruned with my method reduced or matched the error rate of the retrained networks until reaching 50% reduction.

4.1 Background

Training with multiplicative noise (MN) is a regularization procedure implemented through slight modification of the base NN definition (Equation 2.4). It causes the intermediate representation $\mathbf{h}_{i,l-1}$ to become stochastically corrupted by introducing random variables to the inner product $\mathbf{h}_{i,l-1}\mathbf{W}_l$. Rewriting Equation 2.4 with MN (and again dropping the bias terms to reduce clutter), we have

$$\mathbf{h}_{i,l} = f_l(\mathbf{h}_{i,l-1}\mathbf{\Lambda}_l\mathbf{W}_l) \tag{4.1}$$

where $\mathbf{\Lambda}_l$ is a diagonal $d_{l-1} \times d_{l-1}$-dimensional matrix of random variables $\lambda_{j,j}$ drawn independently from some noise distribution $p(\lambda)$. Dropout corresponds to a Bernoulli distribution on λ [Hinton et al., 2012, Srivastava et al., 2014].

Training under MN is done by sampling a new $\mathbf{\Lambda}_l$ matrix for every forward propagation through the network. Backpropagation is done as usual using the corrupted values. We can view the sampling as Monte Carlo integration over the noise distribution, and therefore, the MN loss function can be written as

$$\begin{aligned}\mathcal{L}_{\text{MN}}(\mathbf{y}, \mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}) &= \mathbb{E}_{p(\lambda)}[-\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})] \\ &\approx \frac{1}{S}\sum_{s=1}^{S} -\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\hat{\mathbf{\Lambda}}_{l,s}\}_{l=1}^{L})\end{aligned} \tag{4.2}$$

where the expectation is taken with respect to the noise distribution $p(\lambda)$ and $\hat{\mathbf{\Lambda}}_{l,s}$ denotes the

sth set of samples for the lth layer. At test time, the bias introduced by the noise is corrected; for instance, the weights would be multiplied by $(1-p)$ when training with Bernoulli(p) noise.

Direct analysis of Equation 4.2 for NNs is difficult since the non-linear activation functions sit between the expectation and the noise variables. Nevertheless, analysis of dropout has received a significant amount of attention in the recent literature, and progress has been made by considering second order approximations [Wager et al., 2013, Baldi and Sadowski, 2013], asymptotic assumptions [Wang and Manning, 2013], linear networks [Baldi and Sadowski, 2013, Warde-Farley et al., 2013], generative models of the data [Wager et al., 2014], and convex proxy loss functions [Helmbold and Long, 2015].

4.2 Multiplicative Noise as Gaussian Scale Mixtures

In this section, I show that analysis of multiplicative noise (MN) regularization can be made tractable by adopting a Bayesian perspective. The key observation is that if we assume the weights to be Gaussian random variables, the product λw, where λ is the noise and w is a weight, defines a *Gaussian scale mixture* (GSM). GSMs can be represented hierarchically with the scale mixing variable—in this case the noise λ—becoming a hyperprior. The reparametrization works even for deep NNs (DNNs) regardless of their size or activation functions.

4.2.1 Gaussian Scale Mixtures

A random variable θ is a Gaussian scale mixture (GSM) if and only if it can be expressed as the product of a Gaussian random variable–call it u with zero mean and some variance σ_0^2–and an independent scalar random variable z [Andrews and Mallows, 1974, Beale and Mallows, 1959]:

$$\theta \stackrel{d}{=} zu, \quad u \sim \mathsf{N}(0, \sigma_0^2), \;\; z \sim p(z) \tag{4.3}$$

where $\stackrel{d}{=}$ denotes equality in distribution. The RHS is known as the GSM's *expanded parametrization* [Kuo and Mallick, 1998]. While it may not be obvious from Equation 4.3 that θ is a *scale* mixture, the result follows from the Gaussian's closure under linear transformations, resulting in the following marginal density of θ:

$$p(\theta) = \int z p(u; 0, \sigma_0^2) p(z) dz = \int N(0, \sigma_0^2 z^2) p(z) dz \tag{4.4}$$

where $p(z)$ is now clearly the mixing distribution. As I discuss at length in Section 3.1.2, super-Gaussian distributions, such as the student-t, Laplace, and horseshoe, can be represented as GSMs, and this hierarchical formulation is often used when employing these distributions as robust priors [Steel et al., 2000].

4.2.2 Noise Induced ARD Priors

Now that I have defined GSMs, I demonstrate their relationship to training under MN noise. I do so by assuming the Bayesian framework and then working to derive the Monte Carlo objective in Equation 4.2. Assume we have a L-hidden layer Bayesian NN with the following hierarchical specification:

$$y_i \sim p(y|\mathbf{x}_i, \{\mathbf{M}_l\}_{l=1}^{L+1}), \;\; m_{l,j,k} \sim \mathsf{N}(0, \sigma_0^2 \xi_{l,j,\cdot}^2), \;\; \xi_{l,j,\cdot} \sim p(\xi) \tag{4.5}$$

where $m_{l,j,k}$ denotes the NN weights and $\xi_{l,j,\cdot}$ is the Gaussian prior's scale and thus takes the role of z in the GSM definition above. Notice that I have given $\xi_{l,j,\cdot}$ a layer index (l) and row index (j) but not a column index (k). Thus, all weights in the same row share the same scale—this is the ARD framework discussed in Section 3.1.2 (Equation 3.3). As the prior on the weights is a GSM, we can reparametrize the model into the GSM's equivalent expanded form:

$$y_i \sim p(y|\mathbf{x}_i, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Xi}_l\}_{l=1}^{L}), \;\; w_{l,j,k} \sim \mathsf{N}(0, \sigma_0^2), \;\; \xi_{l,k,\cdot} \sim p(\xi) \tag{4.6}$$

where now the weights are denoted $w_{l,j,k}$ and drawn from a Gaussian with a fixed variance. The reparametrization changes the NN's hidden layers to:

$$\mathbf{h}_{i,l} = f_l(\mathbf{h}_{i,l-1}\mathbf{M}_l) \quad \overset{\text{reparametrization}}{\longrightarrow} \quad f_l(\mathbf{h}_{i,l-1}\boldsymbol{\Xi}_l\mathbf{W}_l) \tag{4.7}$$

where $\boldsymbol{\Xi}_l$ is a diagonal $d_{l-1} \times d_{l-1}$-dimensional matrix of scale values $\xi_{l,j,\cdot}$. The kth hidden unit is now computed as $h_{i,l,k} = f_l\left(\sum_{j=1}^{d_{l-1}} h_{i,l-1,j}\xi_{l,j,\cdot}w_{l,j,k}\right)$.

Lastly, now consider marginalizing out the scale variables in the expanded parametrization:

$$\begin{aligned} p(\{\mathbf{W}_l\}_{l=1}^{L+1}|\mathbf{y},\mathbf{X}) &\propto \int_{\xi} p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\boldsymbol{\Xi}_l\}_{l=1}^{L})p(\xi)\prod_{l=1}^{L+1} p(\mathbf{W}_l;0,\sigma_0^2)\,d\xi \\ &= \mathbb{E}_{p(\xi)}\left[p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\boldsymbol{\Xi}_l\}_{l=1}^{L})\right]\prod_{l=1}^{L+1} p(\mathbf{W}_l;0,\sigma_0^2). \end{aligned} \tag{4.8}$$

Doing so makes the objective for the marginal MAP estimate of the weights:

$$\begin{aligned} \mathcal{L}_{\text{MAP}}(\mathbf{y},\mathbf{X},&\{\mathbf{W}_l\}_{l=1}^{L+1}) = -\log p(\{\mathbf{W}_l\}_{l=1}^{L+1}|\mathbf{y},\mathbf{X}) \\ &\propto -\log \mathbb{E}_{p(\xi)}\left[p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\boldsymbol{\Xi}_l\}_{l=1}^{L})\right] + -\log\prod_{l=1}^{L+1} p(\mathbf{W}_l;0,\sigma_0^2) \\ &\propto -\log \mathbb{E}_{p(\xi)}\left[p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\boldsymbol{\Xi}_l\}_{l=1}^{L})\right] + \frac{1}{2\sigma_0^2}\sum_{l=1}^{L+1}\sum_{j=1}^{d_{l-1}}\sum_{k=1}^{d_l} w_{l,j,k}^2. \end{aligned} \tag{4.9}$$

I now perform the final two steps in the derivation: *(i)* I use Jensen's inequality to upper-bound

the first term, and *(ii)* I assume the variance of the prior on the weights goes to infinity.

$$
\begin{aligned}
&\mathcal{L}_{\text{MAP}}(\mathbf{y}, \mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}) \\
&\qquad \leq \mathbb{E}_{p(\xi)}\left[-\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Xi}_l\}_{l=1}^{L})\right] + \frac{1}{2\sigma_0^2}\sum_{l=1}^{L+1}\sum_{j=1}^{d_{l-1}}\sum_{k=1}^{d_l} w_{l,j,k}^2 \\
&\qquad \to \mathbb{E}_{p(\xi)}\left[-\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Xi}_l\}_{l=1}^{L})\right] \;\; (\text{assuming } \sigma_0 \to \infty) \\
&\qquad \approx \frac{1}{S}\sum_{s=1}^{S} -\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\hat{\mathbf{\Xi}}_{l,s}\}_{l=1}^{L}) \;\; (\text{Monte Carlo approximation}) \\
&\qquad = \mathcal{L}_{\text{MN}}(\mathbf{y}, \mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}) \;\; (\text{Equation 4.2 with } p(\xi) = p(\lambda)).
\end{aligned}
\tag{4.10}
$$

In the last step, I reach equality with Equation 4.2, the Monte Carlo MN optimization objective, by setting the scale distribution $p(\xi)$ to be the noise model $p(\lambda)$.

To summarize the derivation, I started with a Bayesian NN given an ARD GSM prior. I then reparametrized the model into its expanded form, moving the random Gaussian scale from the first-level prior into the likelihood function. To arrive at the objective, I considered the model's marginal map estimate of the NN weights and upper-bounded the objective via Jensen's, effectively moving the noise distribution outside the logarithm (first line in Equation 4.10). I then assumed the Gaussian prior's variance is sufficiently large so that the ridge penalty can be ignored. Equation 4.2 is then attained by assuming the expectation is computed with a Monte Carlo approximation. The assumption that the Gaussian's variance goes to infinity (second line of Equation 4.10) can be removed if using both weight decay [Hinton, 1986] and MN regularization, which is done by Srivastava et al. [2014] (see their Table 9). The variance should simply be set so that the term $1/2\sigma_0^2$ matches the weight decay strength parameter. The ARD assumption can be removed if we assume all weights have independent scales, which changes the hidden layer computation to:

$$\mathbf{h}_{i,l} = f_l(\mathbf{h}_{i,l-1}(\mathbf{\Xi}_l \odot \mathbf{W}_l))$$

where $\odot$ denotes the Hadamard product (element-wise) and $\mathbf{\Xi}_l$ is now a dense matrix. Following

the same derivation from this point reveals an equivalence to *dropconnect* regularization [Wan et al., 2013], which uses MN on each weight instead of each hidden unit. This disconnection from ARD may explain why dropconnect is not as widely used as dropout.

4.2.3 Corresponding Priors

Having shown the equivalence between GSM priors and MN, I now discuss some specific noise distributions and their corresponding priors. Starting with dropout, the noise distribution is $\lambda \sim \text{Bernoulli}(p)$, and this implies the prior on the Gaussian's variance is also Bernoulli, i.e. $\lambda^2 \sim \text{Bernoulli}(p)$, since the square of a Bernoulli random variable is still a Bernoulli of the same distribution. The marginal prior on the NN weights is then

$$p(w) = \sum_{\lambda \in \{0,1\}} \lambda \, \mathsf{N}(w; 0, \sigma_0^2) \, p(\lambda) = p \, \mathsf{N}(w; 0, \sigma_0^2) + (1 - p) \, \delta[0] \tag{4.11}$$

where $\delta[0]$ denotes the delta function located at zero. This is the *spike-and-slab* prior commonly used for Bayesian variable selection [Mitchell and Beauchamp, 1988, George and McCulloch, 1993, Kuo and Mallick, 1998]. Interestingly, the expanded parametrization was used for linear regression by Kuo and Mallick [1998], and thus their work should be considered a precursor to dropout. However, Kuo and Mallick [1998] were interested in obtaining the marginal posteriors $p(\lambda = 1|\mathbf{y}, \mathbf{X})$ rather than deriving a regularization mechanism to improve predictive performance. When dropout is performed without weight decay, dropout's prior becomes

$$p_{\text{DROP}}(w) \propto p \, \mathbb{1} + (1 - p) \, \delta[0], \tag{4.12}$$

where the improper uniform distribution $\mathbb{1}$ is derived by taking $\sigma_0 \to \infty$. Thus, we see that dropout interpolates between no regularization ($\mathbb{1}$) and absolute shrinkage ($\delta[0]$). Recalling the regularization penalties discussed in Section 3.1.2, we see that the penalties derived by Kingma

Noise Model $p(\lambda)$	Variance Prior $p(\lambda^2)$	Marginal Prior $p(w)$
Bernoulli	Bernoulli	Spike-and-Slab
Gaussian	χ^2	Unnamed
Rayleigh	Exponential	Laplace
Inverse Nagakami	Γ^{-1}	Student-t
Half-Cauchy	Unnamed	Horseshoe

Table 4.1: *Noise Models and their Corresponding Gaussian Scale Mixture Prior.*

et al. [2015] and Helmbold and Long [2015] do exactly this: they aggressively pull the weights toward zero but then relax the penalty into a plateau once sufficiently far from the origin. The region near zero corresponds to the $\delta[0]$ term and the plateau to corresponds to the $\mathbb{1}$ term.

In Table 4.1, I list several more noise models, their corresponding priors on the Gaussian variance, and their marginal distribution on the weights. Gaussian MN, which Srivastava et al. [2014] showed to work as well as or better than Bernoulli noise, corresponds to a χ^2-distribution on the variance. I am unaware of a name for the corresponding marginal distribution, however. Other notable cases are Rayleigh noise, which corresponds to a Laplace marginal, inverse Nagakami noise [Nakagami, 1960], which corresponds to a student-t, and half-Cauchy noise, which corresponds to the HSP [Carvalho et al., 2009] discussed in Section 3.1.2.

4.3 A Variational Derivation with Applications to Pruning

Having established the link between MN and GSMs, I next wish to isolate the mechanics of MN regularization. Writing $\mathbf{\Lambda}_l$ as a function of $\mathbf{W}_l$ may reveal the interplay between the noise and parameters. Such a derivation for the expanded parametrization is still difficult, as the noise is buried within the activation functions, but working with the hierarchical form has the potential to simplify the problem. Leaving the 'noise' variable as the Gaussian scale removes it from the likelihood term and thus outside of the deep NN.

Consider the following variational EM derivation for the Bayesian NN with a GSM prior in its hierarchical formulation (originally defined in Equation 4.5):

$$y_i \sim p(y|\mathbf{x}_i, \{\mathbf{M}_l\}_{l=1}^{L+1}), \;\; m_{l,j,k} \sim \mathsf{N}(0, \sigma_0^2 \lambda_{l,j,\cdot}^2), \;\; \lambda_{l,j,k} \sim p(\lambda)$$

where λ is now used for the scale / noise variable. I assume the posterior is approximated by

$$p(\{\mathbf{M}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L} | \mathbf{y}, \mathbf{X}) \approx \prod_{l=1}^{L+1} \prod_{j=1}^{d_{l-1}} \prod_{k=1}^{d_l} q(m_{l,j,k}; \mu_{i,j,k}, \sigma_{l,j,k}) \delta[\lambda_{l,j,k}]$$

where the posterior on the weights is assumed to be Gaussian with parameters $\mu_{i,j,k}$ and $\sigma_{l,j,k}$. Writing the ELBO for this model and approximation we have:

$$\begin{aligned}
\log p(\mathbf{y}|\mathbf{X}) &\geq \mathbb{E}_{q(\mathbf{M})}\left[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{M}_l\}_{l=1}^{L+1})\right] \\
&\quad + \sum_{l=1}^{L+1} -\mathbb{E}_{\delta[\lambda_l]}\left[\mathsf{KLD}[q(\mathbf{M}_l)||p(\mathbf{M}_l|\boldsymbol{\lambda}_l)]\right] + \mathbb{H}_{\delta[\lambda]}[\boldsymbol{\lambda}_l] - \mathbb{E}_{\delta[\lambda]}[\log p(\boldsymbol{\lambda}_l)] \\
&= \mathbb{E}_{q(\mathbf{M})}\left[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{M}_l\}_{l=1}^{L+1})\right] \\
&\quad + \sum_{l=1}^{L+1} -\mathsf{KLD}[q(\mathbf{M}_l)||p(\mathbf{M}_l|\hat{\boldsymbol{\lambda}}_l)] - \log p(\hat{\boldsymbol{\lambda}}_l) + \mathcal{C} \\
&= \mathbb{E}_{q(\mathbf{M})}\left[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{M}_l\}_{l=1}^{L+1})\right] \\
&\quad + \sum_{l=1}^{L+1} \sum_{j=1}^{d_{l-1}} \sum_{k=1}^{d_l} -\log \frac{\hat{\lambda}_{l,j,k}}{\sigma_{l,j,k}} - \frac{\sigma_{l,j,k}^2 + \mu_{l,j,k}^2}{2\hat{\lambda}_{l,j,k}^2} - \log p(\hat{\lambda}_{l,j,k}) + \mathcal{C}
\end{aligned} \tag{4.13}$$

where $\mathcal{C}$ is a constant. If we assume the noise model has continuous support, we can then update the noise variable by taking the derivative and setting it to zero:

$$\begin{aligned}
0 &= \frac{\partial}{\partial \hat{\lambda}_{l,j,k}} \left[\mathbb{E}_{q(\mathbf{M})}\left[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{M}_l\}_{l=1}^{L+1})\right] - \log \frac{\hat{\lambda}_{l,j,k}}{\sigma_{l,j,k}} - \frac{\sigma_{l,j,k}^2 + \mu_{l,j,k}^2}{2\hat{\lambda}_{l,j,k}^2} - \log p(\hat{\lambda}_{l,j,k}) \right] \\
&= \frac{-1}{\hat{\lambda}_{l,j,k}} + \frac{\sigma_{l,j,k}^2 + \mu_{l,j,k}^2}{\hat{\lambda}_{l,j,k}^3} - \frac{\partial}{\partial \hat{\lambda}_{l,j,k}} \log p(\hat{\lambda}_{l,j,k}) \\
&= -\hat{\lambda}_{l,j,k}^2 + \sigma_{l,j,k}^2 + \mu_{l,j,k}^2 - \hat{\lambda}_{l,j,k}^3 \frac{\partial}{\partial \hat{\lambda}_{l,j,k}} \log p(\hat{\lambda}_{l,j,k}).
\end{aligned} \tag{4.14}$$

To get the final update, I assume that $\hat{\lambda}_{l,j,k}$ nearly maximizes $\log p(\hat{\lambda}_{l,j,k})$ so that I can consider its derivative weak and therefore negligible. The approximate update for $\hat{\lambda}_{l,j,k}$ is then:

$$\hat{\lambda}_{l,j,k}^{t+1} \approx \sigma_{l,j,k,t}^2 + \mu_{l,j,k,t}^2. \tag{4.15}$$

In words, we should set the noise / variance at time $t+1$ to the sum of the weight's posterior variance and mean. The connection to the shrinkage dynamics can then be seen by plugging the update back into the Gaussian prior and viewing it as a ridge penalty:

$$\mathcal{R}(m_t) = \frac{1}{2\sigma_0^2} \sum_{l=1}^{L+1} \sum_{j=1}^{d_{l-1}} \sum_{k=1}^{d_l} \frac{m_{l,j,k,t}^2}{\hat{\lambda}_{l,j,k}^t} \approx \frac{1}{2\sigma_0^2} \sum_{l=1}^{L+1} \sum_{j=1}^{d_{l-1}} \sum_{k=1}^{d_l} \frac{m_{l,j,k,t}^2}{\sigma_{l,j,k,t-1}^2 + \mu_{l,j,k,t-1}^2}. \tag{4.16}$$

The update results in the ridge penalty being scaled inversely proportional to $\sigma_{l,j,k,t-1}^2 + \mu_{l,j,k,t-1}^2$, meaning that the larger the posterior moments, the weaker the regularization. Conversely, as $\sigma_{l,j,k,t-1}^2 + \mu_{l,j,k,t-1}^2 \to 0$, the penalty explodes, quenching the weight to true zero. This behavior roughly parallels that of the heavy-tailed penalties (Section 3.1.2).

While the above analysis provides some view into the mechanics behind MN, it does not immediately suggest an improvement upon how MN and GSMs are used in practice. However, I did find immediate and practical benefits in the context of *model compression* [Ba and Caruana, 2014, Hinton et al., 2014]. My variational EM derivation of MN regularization conspicuously conflicts with the *signal-to-noise ratio* (SNR) commonly used for NN weight pruning [Graves, 2011, Blundell et al., 2015]. The SNR heuristic is defined by the following inequality:

$$\frac{|\mu_{l,j,k}|}{\sigma_{l,j,k}} < \tau \tag{4.17}$$

where $|\mu_{l,j,k}|$ is the absolute value of the posterior mean of weight $m_{l,j,k}$, $\sigma_{l,j,k}$ is the posterior standard deviation of the same weight, and τ is some positive constant. Pruning is carried out by setting to zero all weights for which the inequality holds (i.e. $|\mu|/\sigma$ is below the threshold τ).

Blundell et al. [2015] conducted experiments using the SNR and stated it "is in fact related to test performance."

Now I propose an alternative rule based on my EM analysis. Recall that the ridge penalty in Equation 4.16 is weighted by the inverse of the posterior mean and variance. Since large posterior moments turns off the shrinkage penalty, that means that we should retain the weights with large means *and* large variances. This conclusion conflicts with the SNR since using $|\mu|/\sigma$ prunes weights with large variances first. Thus, I propose the following competing heuristic I call *signal-plus-robustness* (SPR):

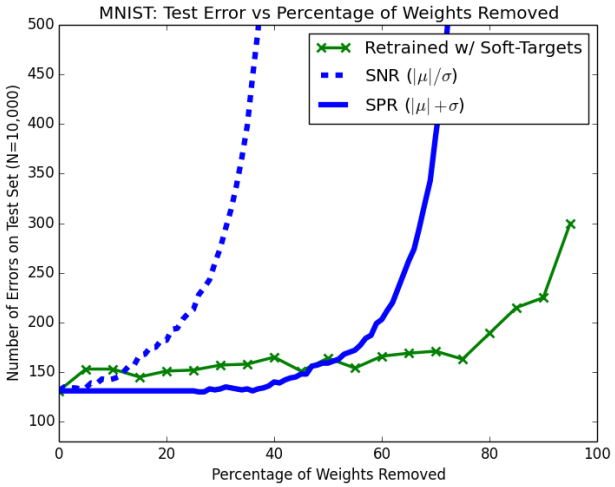
$$|\mu_{l,j,k}| + \sigma_{l,j,k} < \tau \tag{4.18}$$

where the terms are defined the same as above.

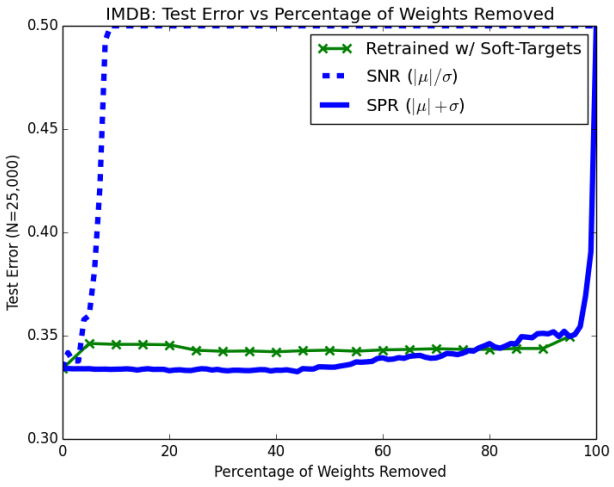
4.4 Experiments: Weight Pruning

I experimentally compared both pruning rules on three data sets, each with very different characteristics. The first is the well-known MNIST dataset ($d = 784$, $N = 50k/10k$), the second is the large IMDB movie review dataset for sentiment classification [Maas et al., 2011] ($d = 5000$, $N = 25k/25k$), and the third is a regression task using features preprocessed from the Million Song Dataset (MSD) [Lichman, 2013] ($d = 90$, $N = 460k/50k$). I trained the networks with Bernoulli MN and when convergence was reached, switched to Langevin dynamics (Equation 2.19) with no MN to collected 10,000 samples from the posterior weight distribution of each network. A polynomial decay schedule was set by validation set performance.

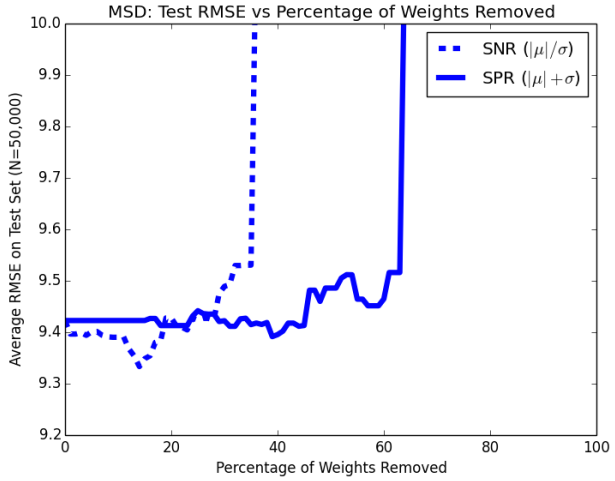
I ordered the weights of each network by SNR and SPR and then removed weights (i.e. set them to zero) in increasing order according to the two rules. Plots showing test error (number of errors, error rate, mean RMSE) vs percentage of weights removed can be seen in panels (a), (b),



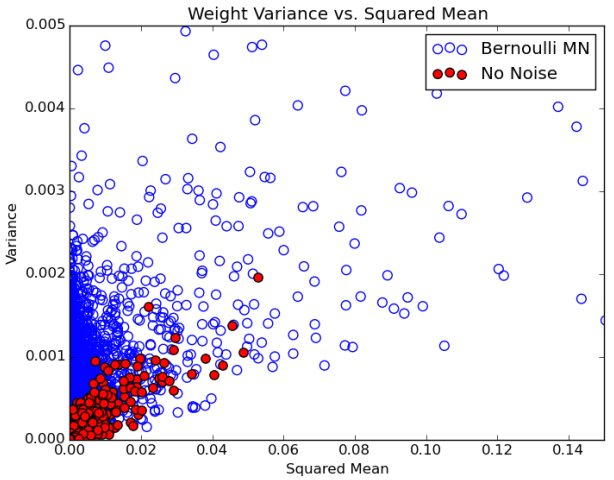
(a) MNIST, 500-300 Hidden Units



(b) IMDB, 1000 Hidden Units



(c) MSD, 120 Hidden Units



(d) Posterior Weight Moments

Figure 4.1: *Experimental Results.* Weight pruning task (a, b, c) and empirical moments (d).

and (c) of Figure 4.1. For another source of comparison, I show the performance of a network (completely) retrained on the soft-targets [Hinton et al., 2014] produced by the full network. To make comparison fair, the retrained networks had the same depth as the one on which pruning was done, splitting the parameters equally between the layers. No soft-target results are shown for (c), the MSD year prediction task, as I found training with soft-targets does not have the same benefits for regression it does for classification.

We see that my rule, SPR ($|\mu| + \sigma$), is clearly superior to SNR ($|\mu|/\sigma$). I was able to remove at least 20% more of the weights in each case before seeing a catastrophic increase in test error. The most drastic difference is seen for the IMDB dataset in (b), which I believe is due to the sparsity of the features (word counts), exaggerating SNR's preference for over-determined weights. My method, SPR, even outperformed retraining with soft-targets until at least a 50% reduction in parameters was reached. Finally, further empirical support of my findings, a scatter plot showing the first two moments of each weight for two networks–one trained with Bernoulli MN and the other without MN–can be see in panel (d) of Figure 4.1. I produce the figure to show that although my closed-form penalty technically doesn't hold for discrete noise distributions (due to the need to compute the gradient), the analysis (shrinkage vs scale robustness) most likely extends to discrete mixtures.

4.5 Conclusions

This chapter improves our understanding of how multiplicative noise regularizes the weights of deep NNs. I show that multiplicative noise can be interpreted as inducing a Gaussian scale mixture prior (under mild assumptions). This analysis holds for NNs regardless of their depth or activation function. Moreover, I derive a variational EM algorithm to isolate the mechanics of MN, writing the noise variable in terms of the posterior weight moments. Lastly, I demonstrated the utility of my findings via a new weight pruning rule that naturally extends from my analysis

of MN. Pruning via signal-plus-robustness is significantly more effective than the previously proposed signal-to-noise ratio and is even competitive to retraining with soft-targets.

Chapter 5

Approximating Objective Priors

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

Robert Frost

The Road Not Taken

In many practical situations, there are no available means for obtaining useful prior information. For example, in high-dimensional problems the parameter space is often inherently unintuitive. The usual way to proceed is to pick a noninformative prior that is *flat* and/or *objective*. By *flat prior* I mean a distribution that does not have any substantial concentration of its mass; maximum entropy priors [Jaynes, 1957] often exhibit this characteristic. An *objective* prior is one that has some formal invariance property. The two best known examples of objective priors are *Jeffreys* [Jeffreys, 1946] and *reference* [Bernardo, 1979] priors, which are both invariant to model reparametrization. Some priors are both objective and flat: the Jeffreys prior for the Gaussian mean is the (improper) uniform distribution. However, just because a prior is relatively flat does

not mean it is objective. For example, the Bernoulli's Jeffreys prior is the arcsine distribution, which, having vertical asymptotes at 0 and 1, is conspicuously not flat.

Since there are no guarantees that what looks to be a flat prior might not harbor hidden subjectivity, objective priors seem to be the better 'default' choices. However, the mathematical rigor that makes objective priors attractive also makes their use problematic: their derivation is difficult for all but the simplest models. To be specific, solving the calculus of variations problem for a reference prior requires, among other properties, an analytical form for the posterior distribution, which is rarely available.

In this chapter, I broaden the potential use of objective priors by describing a method for learning high-fidelity reference prior approximations[1]. The proposed method is akin to black-box (posterior) variational inference [Ranganath et al., 2014]: I posit a parametric family of distributions and perform derivation-free optimization to find the member of the family closest to the true reference prior. Doing so would be useful, for example, if one wishes to have an objective prior that preserves model conjugacy[2]. The modeler could employ the techniques proposed below to find the conjugate prior's parameter setting that makes it closest to objective. Moreover, these methods learn a reference prior for a given model independently of any data source[3], which means that obtaining a reference prior for a particular model needs to be done only once.

In my experimental results I demonstrate that the proposed framework recovers the Jeffreys prior better than existing numerical methods. I also analyze the optimization objective, providing intuition behind a number of hyper-parameter choices. And lastly, I learn a reference prior for a *variational autoencoder* [Kingma and Welling, 2014b] (see Section 2.3.2.2). In an interesting case study, we see that the variational autoencoder's reference prior differs markedly from the standard Normal distribution that is commonly used as the prior on the latent space.

[1] The work in this chapter is presented in Nalisnick and Smyth [2017b].

[2] Reference priors are often improper distributions, let alone conjugate.

[3] Except when the model is for a conditional distribution, i.e. $p(y|x)$. In this case, samples of x are necessary to learn the approximate reference prior.

5.1 Background and Related Work

I begin by defining reference priors, highlighting their connection to the Jeffreys prior, and summarizing the related work on computing intractable reference priors. I use the following notation throughout the chapter. Define the likelihood to be $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ are the model parameters and $\mathcal{D}$ is the data set, which is comprised of N i.i.d. observations $\mathbf{x}_i \in \mathcal{X}$. $p(\boldsymbol{\theta})$ denotes the prior, $p(\boldsymbol{\theta}|\mathcal{D})$ the posterior, and $p(\mathcal{D}) = \int_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$ the marginal likelihood (or model evidence). When I refer to the 'likelihood function' I mean the functional form of the data model, $p(\mathbf{x}|\boldsymbol{\theta})$. I write expectations with respect to the data set likelihood, but because of $\mathcal{D}$'s i.i.d. assumption, these can be written equivalently in terms of each data instance; for example:

$$\mathbb{H}_{\mathcal{D}|\boldsymbol{\theta}}[\mathcal{D}] = -\int_{\mathcal{D}} p(\mathcal{D}|\boldsymbol{\theta}) \log p(\mathcal{D}|\boldsymbol{\theta}) d\mathcal{D} = -\int_{\mathbf{x}} \prod_i p(\mathbf{x}_i|\boldsymbol{\theta}) \log \prod_i p(\mathbf{x}_i|\boldsymbol{\theta}) d\mathbf{x} = N\mathbb{H}_{\mathbf{x}|\boldsymbol{\theta}}[\mathbf{x}].$$

5.1.1 Reference Priors

Reference priors [Berger et al., 2009, Bernardo, 2005] (RPs) are objective Bayesian prior distributions derived for a given likelihood function by finding the prior that maximizes the data's influence on the posterior distribution. Equivalently, the prior's influence on the posterior is minimized, which is precisely the behavior we desire if we wish to represent a state of ignorance about the model parameters. The RP's data-driven nature yields 'frequentist-esque' posteriors: for large sample sizes, the $1-\alpha$ credible interval approximates a confidence interval with significance level α [Irony and Singpurwalla, 1997]. Thus, RPs give results that are the nearest Bayesian equivalent to maximum likelihood estimation, and this behavior is how they derive their name: RPs serve as a *reference* against which to test subjective priors.

Definition. I now state the RP definition formally. A RP $p^*(\boldsymbol{\theta})$ is the distribution that maximizes the mutual information between the parameters $\boldsymbol{\theta}$ and the data $\mathcal{D}$ [Berger et al., 2009, Bernardo,

1979]:

$$p^*(\boldsymbol{\theta}) = \underset{p(\boldsymbol{\theta})}{\operatorname{argmax}} \; I(\boldsymbol{\theta}, \mathcal{D}) = \underset{p(\boldsymbol{\theta})}{\operatorname{argmax}} \; \underbrace{\mathbb{H}[\boldsymbol{\theta}]}_{\substack{\text{maximize prior} \\ \text{uncertainty}}} - \underbrace{\mathbb{H}[\boldsymbol{\theta}|\mathcal{D}]}_{\substack{\text{minimize posterior} \\ \text{uncertainty}}}. \tag{5.1}$$

Here $I(\cdot, \cdot)$ denotes mutual information. In the second line, $I(\cdot, \cdot)$ is (by definition) decomposed into separate marginal and conditional entropy terms, showing that maximizing $I(\boldsymbol{\theta}, \mathcal{D})$ in turn maximizes the prior's uncertainty while minimizing the posterior's uncertainty. The second term reflects the RP's data-driven nature as it encourages the posterior to contract quickly (as N increases). Another way to see how the RP accentuates the data's influence is by writing the mutual information in terms of a Kullback-Leibler divergence (KLD): $I(\boldsymbol{\theta}, \mathcal{D}) = \int_{\mathcal{D}} p(\mathcal{D}) \; \text{KLD}[p(\boldsymbol{\theta}|\mathcal{D}) \; || \; p(\boldsymbol{\theta})] \; d\mathcal{D}$. This form shows that increasing $I(\boldsymbol{\theta}, \mathcal{D})$ decreases the similarity between the posterior and prior. Itti and Baldi [2006] call the KLD between prior and posterior beliefs the *Bayesian surprise*, and from this point of view, we can interpret RPs as the priors that result in the largest expected surprise.

Solution. Solving Equation 5.1 for p^* is a calculus of variations problem whose solution can be expressed by re-writing the mutual information as

$$I(\boldsymbol{\theta}, \mathcal{D}) = -\int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}) \; \log \frac{p(\boldsymbol{\theta})}{f(\boldsymbol{\theta})} \; d\boldsymbol{\theta} \text{ where } \; f(\boldsymbol{\theta}) = \exp\left\{ \int_{\mathcal{D}} p(\mathcal{D}|\boldsymbol{\theta}) \; \log p(\boldsymbol{\theta}|\mathcal{D}) \; d\mathcal{D} \right\}. \tag{5.2}$$

Clearly, the mutual information is maximized when $p(\boldsymbol{\theta}) \; \propto \; f(\boldsymbol{\theta})$. See Bernardo (1979) for a complete discussion of the derivation. Equation 5.2 also makes clear the analytical obstacles that need to be overcome to solve the optimization problem for a given model: the functional f requires that the log posterior—which is usually intractable to compute—be integrated over the likelihood function. Note that the solution is commonly not a proper distribution (that integrates to 1).

Relation to Jeffreys Priors. Jeffreys priors are defined as $\pi(\boldsymbol{\theta}) \; \propto \; \sqrt{\det \mathcal{F}[\boldsymbol{\theta}]}$ where $\mathcal{F}$ de-

notes the Fisher information matrix, and RPs are equal to the Jeffreys in one dimension but not in general. The equivalence is obtained by invoking the *Bernstein-von Mises* theorem: setting $p(\boldsymbol{\theta}|\mathcal{D}) \approx \mathsf{N}(\boldsymbol{\theta}_{\mathsf{MLE}}, \mathcal{F}^{-1}(\boldsymbol{\theta}))$ where $\boldsymbol{\theta}_{\mathsf{MLE}}$ is the maximum likelihood estimate and $\mathcal{F}$ is the Fisher information matrix. RPs, also like the Jeffreys, are invariant to model reparametrization, which follows from the fact that the mutual information is itself invariant to a change in parametrization [Berger et al., 2009].

5.1.2 Related Work

Next I review existing techniques for approximating intractable RPs. These methods have a notable lack of scalability, requiring numerical integration over the parameter space. Nonetheless, since they share some fundamental similarities with my proposed method, I reproduce their main components so that I can later discuss how my method handles the same analytical difficulties.

Numerical Algorithm. Berger et al. [2009] proposed a numerical method for computing a RP's value at any given point $\boldsymbol{\theta}_0$. Their method is, simply, to calculate $f(\boldsymbol{\theta})$ numerically via Monte Carlo approximations:

$$p^*(\boldsymbol{\theta}_0) \approx \exp\left\{ \frac{1}{J} \sum_{j=1}^{J} \log \frac{p(\hat{\mathcal{D}}_j|\boldsymbol{\theta}_0)\mathbb{1}_{\boldsymbol{\Theta}}}{\sum_{s=1}^{S} p(\hat{\mathcal{D}}_j|\hat{\boldsymbol{\theta}}_s)} \right\} \tag{5.3}$$

where $\mathbb{1}_{\boldsymbol{\Theta}}$ is an improper uniform prior over the parameter space. The method proceeds by sampling J data sets from the likelihood function, i.e. $\hat{\mathcal{D}}_j = \{\hat{\mathbf{x}}_{j,i}|\hat{\mathbf{x}} \sim p(\mathbf{x}|\boldsymbol{\theta}_0)\}$, and S parameter values from the prior, i.e. $\hat{\boldsymbol{\theta}}_s \sim \mathbb{1}_{\boldsymbol{\Theta}}$. The posterior is then approximated as $p(\boldsymbol{\theta}|\mathcal{D}) \approx p(\hat{\mathcal{D}}_j|\boldsymbol{\theta}_0)\mathbb{1}_{\boldsymbol{\Theta}} / \sum_{s=1}^{S} p(\hat{\mathcal{D}}_j|\hat{\boldsymbol{\theta}}_s)$. This numerical approximation has two significant downsides. The first is that the user must specify the points at which to compute the prior, and the second is that numerically integrating over the parameter space is computationally expensive in even low dimensions.

MCMC. Lafferty and Wasserman [2001] proposed a Markov chain Monte Carlo (MCMC) method for sampling from a RP. Their approach involves running the Metropolis-Hastings algorithm on the following ratio [Lafferty and Wasserman, 2001]:

$$\log \frac{p^{t+1}(\boldsymbol{\theta})}{p^{t+1}(\boldsymbol{\theta'})} = (t+1)(\mathbb{H}_{\mathbf{x}|\boldsymbol{\theta'}}[\mathbf{x}] - \mathbb{H}_{\mathbf{x}|\boldsymbol{\theta}}[\mathbf{x}]) + \sum_{\mathbf{x}\in\mathcal{X}} W^t(\mathbf{x})[p(\mathbf{x}|\boldsymbol{\theta'}) - p(\mathbf{x}|\boldsymbol{\theta})] \tag{5.4}$$

where t is the iteration index, $\mathbb{H}_{\mathbf{x}|\boldsymbol{\theta}}[\mathbf{x}]$ is the entropy of the likelihood function, and $W^t(\mathbf{x}) = W^{t-1}(\mathbf{x}) + \log \frac{1}{S_t} \sum_{s=1}^{S_t} p(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_s^t)$ where $\hat{\boldsymbol{\theta}}_s^t$ are the parameter samples collected during the previous iteration.

While this MCMC approach may look dissimilar to Berger et al. [2009]'s method at first glance, the two methods are in fact related. We can see the connection by examining just one of the distributions in the ratio (when $t = 0$):

$$\log p^1(\boldsymbol{\theta}) = -\mathbb{H}_{\mathbf{x}|\boldsymbol{\theta}}[\mathbf{x}] + \sum_{\mathbf{x}\in\mathcal{X}} -W^0(\mathbf{x}) p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}|\boldsymbol{\theta}) \left[\log p(\mathbf{x}|\boldsymbol{\theta}) - \log \frac{1}{S_0} \sum_{s=1}^{S_0} p(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_s^0) \right].$$

The line above becomes equivalent to Equation 5.3 if we use a Monte Carlo approximation of the expectation over $p(\mathbf{x}|\boldsymbol{\theta})$ and then exponentiate both sides. Despite this close connection between the two methods, Lafferty and Wasserman [2001]'s approach is superior to that of Berger et al. [2009]'s since it draws samples from the prior instead of merely computing its value at points the user must select. Yet the same costly discrete approximations of the integrals will be required.

Reference Distance Method. The third approach, and the only other that I am aware of for finding approximate RPs, is the *Reference Distance Method* (RDM) proposed by Berger et al. [2015]. This method focuses on finding a joint RP by minimizing the divergence between a parametric family and the marginal RPs [Berger et al., 2015]. Since we are concerned with models for which even the marginal RPs are intractable, the RDM is not a relevant point for comparison.

5.2 Learning Reference Prior Approximations

I now turn to the primary contribution of this chapter: approximating RPs by *learning* the parameters of the approximation. My proposed approach contrasts with Berger et al. [2009]'s and Lafferty and Wasserman [2001]'s in that their methods are not model-based. In other words, their procedures produce no parametric artifact for the prior unless a post-hoc step of model fitting is carried out. My black-box optimization framework subsumes the utility of the numerical and MCMC methods as it can directly learn either a parametric approximation to evaluate the prior's density or a functional sampler that can generate new samples from the prior at any later time.

Inspired by recent advances in posterior variational inference (VI), I use similar ideas to optimize an approximate *prior*—call it $p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})$ with parameters $\boldsymbol{\lambda}$—so that it is the distribution in the family closest to the true RP $p^*(\boldsymbol{\theta})$. The mutual information still serves as the natural optimization objective; the difference is that I take the argmax over $\boldsymbol{\lambda}$, instead of the density p itself, such that $p^*(\boldsymbol{\theta}) \approx p_{\boldsymbol{\lambda}^*}(\boldsymbol{\theta})$:

$$
\begin{aligned}
\boldsymbol{\lambda}^* &= \operatorname*{argmax}_{\boldsymbol{\lambda}} \; I(\boldsymbol{\theta}, \mathcal{D}) \\
&= \operatorname*{argmax}_{\boldsymbol{\lambda}} \int_{\boldsymbol{\theta}} p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \int_{\mathcal{D}} p(\mathcal{D}|\boldsymbol{\theta}) \log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) p(\mathcal{D})} d\mathcal{D} d\boldsymbol{\theta} \\
&= \operatorname*{argmax}_{\boldsymbol{\lambda}} \int_{\boldsymbol{\theta}} p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \int_{\mathcal{D}} p(\mathcal{D}|\boldsymbol{\theta}) \log \frac{p(\mathcal{D}|\boldsymbol{\theta})}{p(\mathcal{D})} d\mathcal{D} d\boldsymbol{\theta} \\
&= \operatorname*{argmax}_{\boldsymbol{\lambda}} \; \mathbb{E}_{\boldsymbol{\theta}_{\boldsymbol{\lambda}}} \left[-\mathbb{H}_{\mathcal{D}|\boldsymbol{\theta}}[\mathcal{D}] - \mathbb{E}_{\mathcal{D}|\boldsymbol{\theta}}[\log p(\mathcal{D})] \right].
\end{aligned}
\tag{5.5}
$$

In the final line above, I wrote the mutual information as the difference between the negative likelihood entropy and the expected log marginal likelihood because this is $I(\boldsymbol{\theta}, \mathcal{D})$'s most tractable form: it contains only $p(\mathcal{D})$ instead of $p(\mathcal{D})$ *and* $p(\boldsymbol{\theta}|\mathcal{D})$. I use the notional $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$ to emphasize that $\boldsymbol{\theta}$'s distribution is a function of $\boldsymbol{\lambda}$.

Bounding log p($\mathcal{D}$). The marginal likelihood term in Equation 5.5 is still problematic, and thus,

just as in posterior VI, we need some tractable bound to optimize instead. Since we need to bound $I(\boldsymbol{\theta}, \mathcal{D})$ from below, $\log p(\mathcal{D})$ must be bounded from *above*. Hence, unfortunately, we cannot use the *evidence lower bound* (ELBO) common in posterior VI. As an alternative I use the *variational Rényi bound* [Li and Turner, 2016] (VR), which is defined as:

$$\log p(\mathcal{D}) \leq \frac{1}{1-\alpha} \log \mathbb{E}_{\boldsymbol{\theta}} \left[p(\mathcal{D}|\boldsymbol{\theta})^{1-\alpha} \right] \quad \text{for} \;\; \alpha \leq 0.$$

Plugging the VR bound into Equation 5.5 yields a general lower bound on the mutual information:

$$I(\boldsymbol{\theta}, \mathcal{D}) \geq \mathbb{E}_{\boldsymbol{\theta}_\lambda} \left[-\mathbb{H}_{\mathcal{D}|\boldsymbol{\theta}}[\mathcal{D}] - \mathbb{E}_{\mathcal{D}|\boldsymbol{\theta}} \left[\frac{1}{1-\alpha} \log \mathbb{E}_{\boldsymbol{\theta}} \left[p(\mathcal{D}|\boldsymbol{\theta})^{1-\alpha} \right] \right] \right]. \tag{5.6}$$

In theory, setting $\alpha = 0$ provides the tightest bound, and decreasing α loosens the bound. However, as I discuss next, practical implementation requires a negative value for α.

Optimization Objective. The expectation within the VR bound usually will not be analytically solvable, requiring the use of a Monte Carlo approximation (which I refer to as MC-VR). Yet, introducing sampling into the VR bound can give rise to numerical challenges. The MC-VR estimator is an exponentiated form of the *harmonic mean estimator* [Raftery et al., 2007], which is notorious for its high variance. Furthermore, approximating the expectation with samples, since they reside inside the logarithm, biases the bound downward. Li & Turner (2016) propose the following *VR-max* estimator, corresponding to $\alpha \to -\infty$, to cope with these issues: $\max_s \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s)$ where s indexes samples $\hat{\theta}_s \sim p(\boldsymbol{\theta})$. I find that the VR-max estimator generally preserves the bound and needs to be checked only in high dimensions ($100+$), which is a regime not well suited for reference priors anyway (due to overfitting).

Introducing the VR-max estimator into Equation 5.5 yields a tractable lower bound on the mutual

information:

$$I(\boldsymbol{\theta}, \mathcal{D}) \geq \mathcal{J}_{\text{RP}}(\boldsymbol{\lambda}) = \mathbb{E}_{\boldsymbol{\theta}_{\boldsymbol{\lambda}}} \left[-\mathbb{H}_{\mathcal{D}|\boldsymbol{\theta}}[\mathcal{D}] - \mathbb{E}_{\mathcal{D}|\boldsymbol{\theta}}[\max_s \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s)] \right]. \tag{5.7}$$

Maximizing $\mathcal{J}_{\text{RP}}(\boldsymbol{\lambda})$ with respect to the prior's parameters $\boldsymbol{\lambda}$ results in $p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \approx p^*(\boldsymbol{\theta})$ as long as $p_{\boldsymbol{\lambda}}$ is sufficiently expressive. $\mathcal{J}_{\text{RP}}(\boldsymbol{\lambda})$ can be interpreted as follows. The first term is the entropy of the likelihood function, and thus maximizing its negation encourages certainty in the data model. The second term, the expected value of the VR-max estimator under the likelihood, encourages diversity in $p_{\boldsymbol{\lambda}}$ by forcing a data set $\mathcal{D}_0 \sim p(\mathcal{D}|\boldsymbol{\theta}_0)$ to have low probability under other parameter settings $\hat{\boldsymbol{\theta}}_s$.

Connection to Previous Work. Further understanding of $\mathcal{J}_{\text{RP}}(\boldsymbol{\lambda})$ can be gained by re-writing it to see its relationship to Berger et al. [2009]'s and Lafferty and Wasserman [2001]'s methods. Pulling out the expectation over the likelihood, we have the equivalent form:

$$\mathcal{J}_{\text{RP}}(\boldsymbol{\lambda}) = \mathbb{E}_{\boldsymbol{\theta}_{\boldsymbol{\lambda}}} \mathbb{E}_{\mathcal{D}|\boldsymbol{\theta}} \left[\log p(\mathcal{D}|\boldsymbol{\theta}) - \max_s \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s) \right],$$

which is the difference between the data's log-likelihood under the model (i.e. parameter setting) that generated this data and the data log-likelihood under several samples from the prior. We see that optimization forces the prior to place most of its mass on parameters that generate identifiable data sets—or in other words, data sets that have high probability under only their true generative model. Turning back to Berger et al. [2009]'s Equation 5.3 and recalling its connection to the MCMC method, we see each method is approximately computing $\log[p(\mathcal{D}|\boldsymbol{\theta})/p(\mathcal{D})]$ with the critical difference being that Berger et al. [2009] and Lafferty and Wasserman [2001] approximate $\log p(\mathcal{D})$ with $\log \frac{1}{S} \sum_s p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s)$ whereas I use $\max_s \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s)$ in order to ensure a proper lower bound.

5.2.0.1 Black-Box, Gradient-Based Optimization

I now address how to compute and optimize $\mathcal{J}_{\mathsf{RP}}(\boldsymbol{\lambda})$ (Equation 5.7) efficiently using differentiable Monte Carlo approximations.

Computing the Expectations. Consider first the three expectations in Equation 5.7. Starting with the $\mathbb{H}_{\mathcal{D}|\boldsymbol{\theta}}[\mathcal{D}]$ term, for many predictive models, $p(\mathbf{x}|\boldsymbol{\theta})$ is either Gaussian, as in regression, or Bernoulli, as in binary classification, meaning $\mathbb{H}_{\mathcal{D}|\boldsymbol{\theta}}[\mathcal{D}]$ can be computed analytically[4]. The second term, $\mathbb{E}_{\mathcal{D}|\boldsymbol{\theta}}[\max_s \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s)]$, is simply the cross-entropy between $p(\mathcal{D}|\boldsymbol{\theta})$ and $p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{\mathsf{max}})$ where $\hat{\boldsymbol{\theta}}_{\mathsf{max}}$ is the sample that maximizes the likelihood. This term also can usually be calculated analytically for regression and classification models. The only component that will typically be intractable is the expectation over $p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})$, as the parameters are often buried under nonlinear functions and nested hierarchies. To address this I compute the outer expectation with samples $\hat{\boldsymbol{\theta}} \sim p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})$:

$$\begin{aligned}\tilde{\mathcal{J}}_{\mathsf{RP}}(\boldsymbol{\lambda}) &= \frac{1}{S}\sum_{s=1}^{S} \mathbb{H}[p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s)||p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{\mathsf{max}})] - \mathbb{H}_{\mathcal{D}|\hat{\boldsymbol{\theta}}_s}[\mathcal{D}] \\ &= \frac{1}{S}\sum_{s=1}^{S} \mathsf{KLD}[p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s) \;||\; p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{\mathsf{max}})]\end{aligned} \tag{5.8}$$

for S samples from the RP approximation and where $\mathbb{H}[p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s)||p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{\mathsf{max}})]$ denotes the cross-entropy term mentioned above. If both entropy terms can be computed analytically, we can write the expression as a KLD, which I do in the second line by using the identity $\mathsf{KLD}[q||p] = \mathbb{H}[q||p] - \mathbb{H}[q]$. If the entropy terms are not analytically tractable, they will need to be estimated by sampling from the likelihood function.

Differentiable Sampling: We can take derivatives through each $\hat{\boldsymbol{\theta}}_s$, thereby allowing for fully gradient-based optimization, by drawing the samples via a differentiable non-centered parametriza-

[4]To keep the notation simple, in my discussion of conditional models the dependence on the features is implicit. Writing the entropy with $\mathbf{X}'$ as the feature matrix and $\mathbf{x}$ as the vector of labels, we have: $\mathbb{H}_{\mathbf{x}|\mathbf{X}',\boldsymbol{\theta}}[\mathbf{x}]$.

tion (DNCP)—the so-called 'reparametrization trick' [Kingma and Welling, 2014b], i.e.

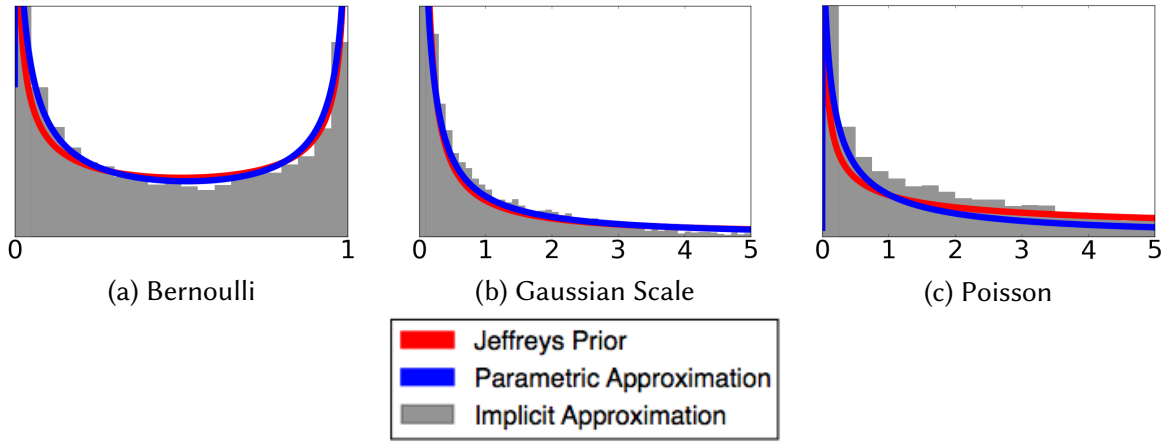
$$\frac{\partial}{\partial \boldsymbol{\lambda}} \left[\mathsf{KLD}[p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s) \mid\mid p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{\mathsf{max}})] \right] = \frac{\partial}{\partial \hat{\boldsymbol{\theta}}} \left[\mathsf{KLD}[p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s) \mid\mid p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{\mathsf{max}})] \right] \frac{\partial \hat{\boldsymbol{\theta}}}{\partial \boldsymbol{\lambda}}$$

where $\frac{\partial \hat{\boldsymbol{\theta}}}{\partial \boldsymbol{\lambda}}$ is the derivative that needs a DNCP in order to be evaluated. Requiring that p_λ has a DNCP does not significantly limit the approximating family. For instance, most mixture densities have a DNCP. When dealing with discrete data or parameters, we can use the *Concrete distribution* [Maddison et al., 2017, Jang et al., 2017], a differentiable relaxation of the discrete distribution, to still have fully gradient-based learning.

5.2.0.2 Implicit Priors

A crucial detail to note about Equation 5.8 is that it does not require evaluation of the prior's density. Rather, we need only to draw samples from it. This allows us to use black-box functional samplers as the variational family [Ranganath et al., 2016], i.e. $\hat{\boldsymbol{\theta}} = g(\boldsymbol{\lambda}, \hat{\boldsymbol{\epsilon}})$ where $\boldsymbol{\epsilon} \sim p_0$, g is some arbitrary differentiable function (such as a NN), and p_0 is a fixed noise distribution. I call $p_{\boldsymbol{\lambda}}$ an *implicit prior* in this setting since its density function is unknown.

Thus, the proposed information bound provides a 'built-in' sampling technique in lieu of Lafferty and Wasserman [2001]'s MCMC algorithm. Although we cannot guarantee the same asymptotically unbiased approximation as MCMC, the lack of restrictions on $g(\boldsymbol{\lambda}, \hat{\boldsymbol{\epsilon}})$ should allow for a sufficiently expressive sampler. Furthermore, we can persist the sampler just by saving the values of $\boldsymbol{\lambda}$; there's no need to save the samples themselves. And since learning a RP for a generative model is data set independent, $\boldsymbol{\lambda}$ could be shared easily via an online repository and users desiring a RP for the same model could download $\boldsymbol{\lambda}$ to generate an unbounded number of their own samples. The same can be done when $p_{\boldsymbol{\lambda}}$ is a proper distribution.



(a) Bernoulli (b) Gaussian Scale (c) Poisson

Figure 5.1: *Approximation via Lower Bound Optimization.*

5.2.0.3 Example: Gaussian Mean

To provide some intuition and to sanity check the proposed approach, consider learning an approximate RP for the mean parameter $\boldsymbol{\mu}$ of a Gaussian density. The RP on $\boldsymbol{\mu}$ is the improper uniform distribution, which can be approximated as a Gaussian with infinite variance: $p^*(\boldsymbol{\mu}) \propto \mathbb{1} \approx \mathsf{N}(\cdot, \infty)$. The analytical solution to the KLD term in Equation 5.8 in this case is:

$$\mathsf{KLD}[p(\mathcal{D}|\hat{\boldsymbol{\theta}}_s) \mid\mid p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{\mathsf{max}}) = \frac{1}{2} \mid\mid \hat{\boldsymbol{\mu}}_s - \hat{\boldsymbol{\mu}}_{\mathsf{max}} \mid\mid_2^2,$$

which is the squared distance between two samples from $p_{\boldsymbol{\lambda}}$. Maximizing Equation 5.8 therefore maximizes the average distance between samples from the RP approximation. If we set $p_{\boldsymbol{\lambda}} = \mathsf{N}(\boldsymbol{\mu}_{\boldsymbol{\lambda}}, \sigma_{\boldsymbol{\lambda}}^2 \mathbb{I})$ and transform to the Normal's DNCP $\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathsf{N}(\mathbf{0}, \mathbb{I})$, then the optimization objective becomes

$$\mid\mid \hat{\boldsymbol{\mu}}_s - \hat{\boldsymbol{\mu}}_{\mathsf{max}} \mid\mid_2^2 \; = \; \mid\mid \boldsymbol{\sigma}_{\boldsymbol{\lambda}} \odot (\hat{\boldsymbol{\epsilon}}_s - \hat{\boldsymbol{\epsilon}}_{\mathsf{max}}) \mid\mid_2^2,$$

and optimization would increase $\sigma_{\boldsymbol{\lambda}}^2$ without bound, agreeing with the infinite-variance Normal approximation.

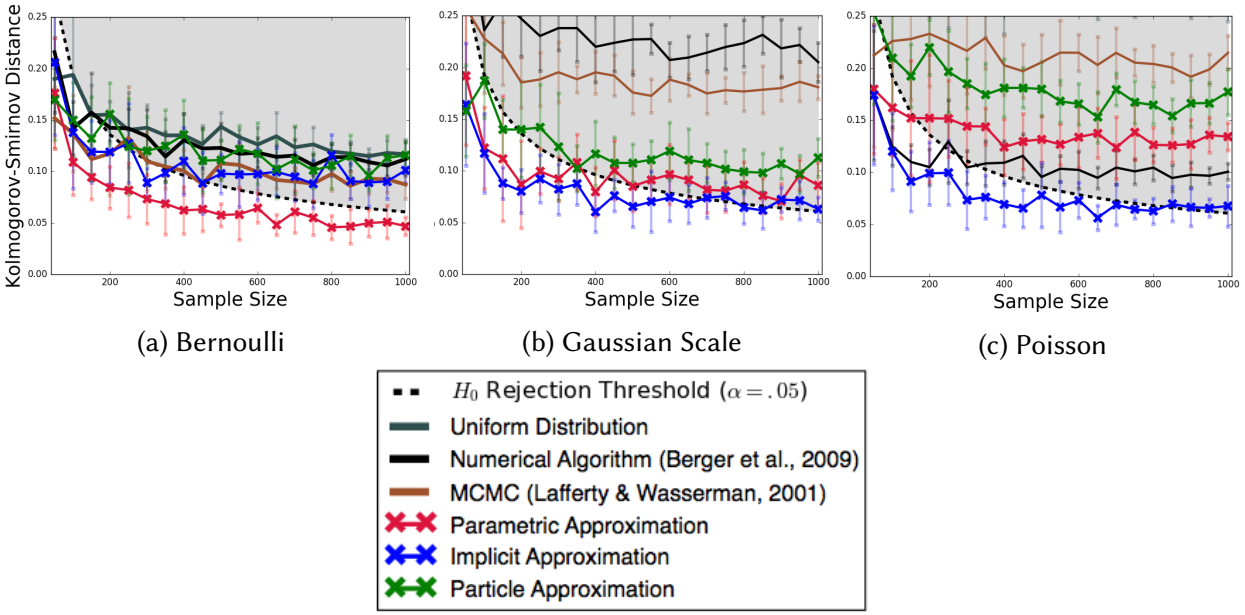
5.3 Empirical Results

Below I describe several empirical analyses of the proposed methods. Formulating experiments is somewhat difficult due to the fact that RPs do not necessarily improve a model's ability to generalize to out-of-sample data. In fact, using an RP when a model requires regularization will likely degrade performance. Thus, my main analysis is a qualitative case study of the *variational autoencoder* [Kingma and Welling, 2014b, Rezende et al., 2014]. But before analyzing the variational autoencoder's RP, I check that my methods do indeed recover known RPs for exponential family models.

For all experiments, I used the *AdaM* optimization algorithm [Kingma and Ba, 2014] with settings $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Training parameters such as the learning rate, latent dimensionality of the functional sampler $g(\boldsymbol{\lambda}, \hat{\boldsymbol{\epsilon}})$, and number of samples were chosen based on which combination gave the highest average value of the information lower bound over the last 50 updates. The number of training iterations was set at 250 and the batch size was set to 100 in all cases.

5.3.1 Recovering Jeffreys Priors

I begin experimental evaluation by attempting to recover the true RP for three one-dimensional models: the Bernoulli mean parameter, $p^*(p) \propto \text{Beta}(.5, .5)$, the Gaussian scale parameter, $p^*(\sigma) \propto 1/\sigma$, and the Poisson rate parameter, $p^*(\lambda) \propto 1/\sqrt{\lambda}$. These are also the Jeffreys priors for the respective models (since we are in the univariate case). The chosen learning rate was $.001$ for the implicit priors and $.0001$ for the parametric approximations, the number of samples drawn was 50 for all models, and the functional sampler g was a linear model with a latent dimensionality of 5, i.e. $\boldsymbol{\epsilon} \sim \text{N}(\mathbf{0}, \mathbb{I}_{5\times5})$. I used a logit-normal distribution for the Bernoulli RP's parametric approximation and a log-normal for the Gaussian scale's and Poisson's RP approximation. Both the logit- and log-normal have DNCPs.



(a) Bernoulli (b) Gaussian Scale (c) Poisson

Figure 5.2: *Quantifying the Approximation Quality.* The Kolmogorov-Smirnov distance (supremum of distance between empirical CDFs) between the Jeffreys/true reference prior and the various approximation techniques. The gray region denotes where the test's null hypothesis is rejected, meaning there is a statistical difference between the distributions.

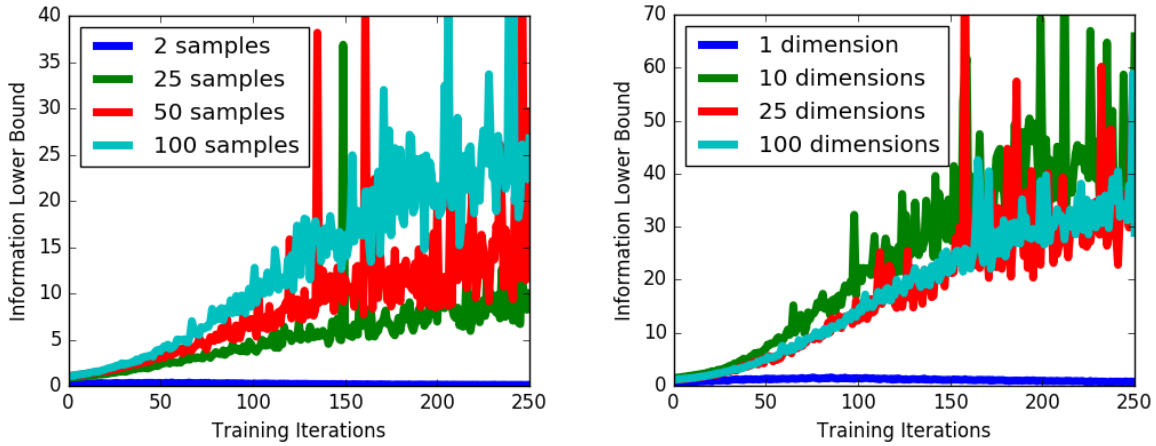
Qualitative Evaluation. Plots of the density functions learned by the lower bound method (Section 5.2) are shown in Figure 5.1. The red line shows the Jeffreys prior (the gold-standard RP), the blue line shows the parametric approximation, and the gray histogram represents $10,000$ samples from the implicit prior. Both approximation types have negligible qualitative difference to the red line.

Quantitative Evaluation. Next I quantitatively compare my methods via a two-sample test against three baselines: Berger et al. [2009]'s numerical method, Lafferty and Wasserman [2001]'s MCMC algorithm, and a uniform prior, which serves as a naive flat prior. For Berger et al. [2009]'s method, I use the same number of parameter samples (S) as my method, set the J parameter to 100, and sample data sets containing 500 points. To generate samples from Berger et al. [2009]'s method, I calculate the prior at 1000 evenly spaced grid points across the domain and then treat them as a discrete approximation with each point having probability $p(\boldsymbol{\theta}_i) / \sum_j^{1000} p(\boldsymbol{\theta}_j)$. I then

sample from this discrete distribution 1000 times. For the MCMC method, I replicate Lafferty and Wasserman [2001]'s simulations by using a uniform proposal distribution and running for $10,000$ iterations. I kept the last 1000 samples drawn (no need to account for auto-correlation due to the uniform proposal). For the Gaussian and Poisson cases, I approximated $\mathcal{X}$ using 1000 points. For all settings, I made sure my approximation methods ran no longer than the baselines, but this was never an issue: my methods converged in a fraction of the time the numerical algorithms needed to run.

I quantify the gap in the approximations via a Kolmogorov-Smirnov two-sample test (KST) under the null hypothesis $H_0 : p = q$ where p is the true RP and q is an approximation. I draw samples from the true RP, when it is improper, via the same discrete approximation used for Berger et al. [2009]'s method. The KST computes the distance (KSD) between the distributions as $\text{KSD}(p, q) = \sup_x \mid \hat{F}_p(x) - \hat{F}_q(x) \mid$ where $\hat{F}_p(x)$ is the empirical CDF.

Figure 5.2 shows the KSD between samples from the Jeffreys prior and the various approximation techniques, as the sample size increases. The black dotted line in conjunction with the gray shaded area denotes the threshold at which the null hypothesis (that the distributions are equal) is rejected. The uniform distribution is denoted by the dark gray line, the numerical algorithm by the black, MCMC by the brown, the parametric approximation by the red, the implicit prior by the blue, and the particle approximation described in Nalisnick and Smyth [2017b] by the green. We see that the latter three approximations (mine) have a lower KSD—and thus are closer to the true RP—in almost every experiment. The exceptions are that MCMC is superior to A-SVGD for the Bernoulli (and competitive with the implicit), and the Berger et al. [2009] technique bests A-SVGD and the parametric approximation for the Poisson. The parametric approximation for the Bernoulli and the implicit prior for the Gaussian scale and Poisson are the only methods that achieve conspicuous indistinguishability.

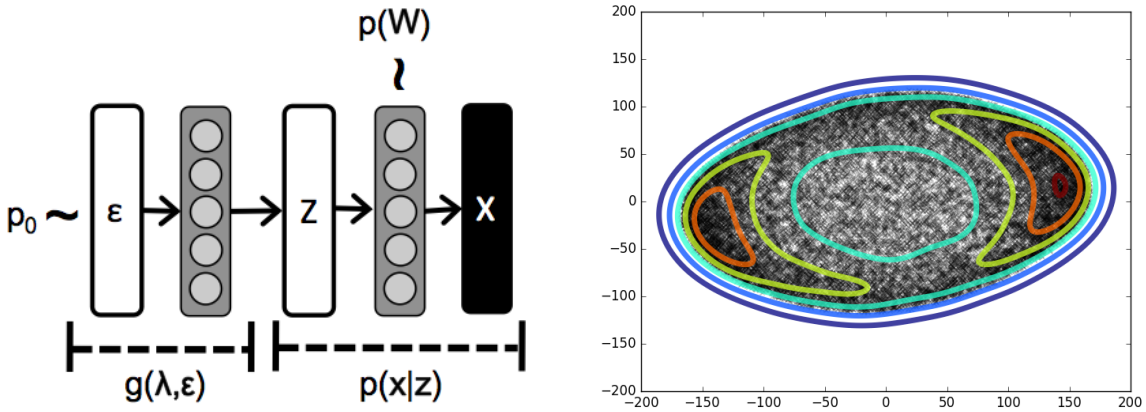


(a) Samples (b) Dimensionality

Figure 5.3: *Optimization Stability.* I train an implicit prior for a multivariate Gaussian and vary (a) the number of samples used in the VR-max estimator, and (b) the Gaussian's dimensionality.

5.3.2 Optimization Stability

As discussed in Section 5.2, the VR-max estimator used in Equation 5.7 has intrinsically high-variance. While I have just shown in the previous section that my approximations better recover the true RP in one dimension, scaling to higher dimensions is a concern (as is also the case for existing techniques). Here I examine optimization progress of an RP approximation for the scale parameters of a multivariate Gaussian with a diagonal covariance matrix. I produce two plots: one showing the information lower bound's progress (for a linear model implicit prior) when using a different number of samples over which to take the maximum (Figure 5.3a) and another showing progress as the Gaussian's dimensionality increases (Figure 5.3b). For the former, using a five dimensional Gaussian, we see there is a trade-off between lower bound maximization and the number of samples used: using more samples increases the rate of progress but also the objective's variance. I find that in less than ten dimensions, using around 50 samples (red line) results in a good variance vs progress balance. In Figure 5.3b, in which I vary the dimensionality of the Gaussian while keeping the number of samples fixed at 100, we see that the objective's variance decreases with dimensionality. While this may seem non-intuitive at first, recall that the VR-max estimator acts as a diversity term, finding points in space that give the data high



(a) Training Configuration

(b) Approximation

Figure 5.4: *Learning the Variational Autoencoder's Reference Prior.* (a) computational pipeline from the implicit prior through the VAE decoder; (b) RP approximation (contours are generated via kernel density estimation on $10,000$ samples).

probability even though it was generated with different parameters. As dimensionality inflates, it becomes harder and harder for a finite number of samples to capture these points and thus the $-\mathbb{H}$ term in Equation 5.8 becomes prone to mode seeking.

5.3.3 VAE Case Study

Lastly, I study learning an RP approximation for an intractable, neural-network-based model: a *variational autoencoder* [Kingma and Welling, 2014b] (VAE). The standard Normal distribution is often chosen as the prior on the VAE's latent space [Kingma and Welling, 2014b, Rezende et al., 2014, Burda et al., 2016], and this choice is made more for analytical simplicity rather than convictions based on prior information[5]. Thus, I learn an RP for the VAE to investigate the qualities of its objective prior, which was previously intractable.

I trained an implicit prior (IP) for a VAE with 784 output dimensions (MNIST's size), 100 encoder hidden units with hyperbolic tangent activations, and a two-dimensional latent space for

[5]"I chose the simple Gaussian prior N(0,I) because it's simple to demonstrate but also because it results in a relatively friendly objective function." — D. Kingma, comment taken from *r/MachineLearning*, 4/12/16.

purposes of visualization. The IP $g(\boldsymbol{\lambda}, \hat{\boldsymbol{\epsilon}})$ is also a one-hidden layer NN[6]. The computational pipeline is depicted in Figure 5.4a, where $p(\mathbf{x}|\mathbf{z})$ denotes the VAE likelihood function (decoder) and $\mathbf{z} = g(\boldsymbol{\lambda}, \hat{\epsilon})$ denotes the functional sampler. Note that the VAE has two sets of parameters: $\mathbf{z}$, the latent variable on which I place the prior, and the weights of the decoder, denoted as $\mathbf{W}$. The weights must have some value during RP training and thus I place a standard normal prior on $\mathbf{W}$ and sample from this prior during optimization of g.

Figure 5.4b shows samples from the VAE's RP. We see that the learned IP is drastically different than the standard Normal that is typically used: the IP is multimodal and has a much larger variance. Yet, the difference is intuitive: placing most prior mass at opposite sides of the latent space encourages the VAE to space it's latent representations with as much distance as possible, ensuring they are as identifiable w.r.t. the model likelihood, the VAE decoder, as possible. Interestingly, recent work by Hoffman and Johnson [2016] suggests that VAEs can be improved by multimodal priors: "[T]he [VAE]'s individual encoding distributions $q(\mathbf{z}_i|\mathbf{x}_i)$ do not have significant overlap...then perhaps we should investigate multimodal priors that can meet $q(\mathbf{z})$ halfway". This suggests using multimodal, dispersed priors encourages flexibility and objectivity in the posterior distribution.

We can also see analytically that the distribution in Figure 5.4b allows the VAE 'to follow the data' as a good RP should. For simplicity, consider using a bivariate Gaussian as the RP approximation, and assuming it captures the same distribution as in Figure 5.4 (b), its parameters would be approximately $\{\boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = 200\ \mathbb{I}_{2\times 2}\}$. Next recall the VAE's optimization objective (the ELBO): $\mathcal{L}_{\text{VAE}} = \mathbb{E}_q[-\log p(\mathbf{x}|\mathbf{z})] + \text{KLD}[q(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \ ||\ p(\mathbf{0}, 200\ \mathbb{I}_{2\times 2})]$. The first term optimizes the model w.r.t. the data and the second acts as regularization, ensuring the variational posterior q is close to the prior. Assuming q's covariance matrix is also diagonal, we can write $\text{KLD}[q(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \ ||\ p(\mathbf{0}, 200\ \mathbb{I}_{2\times 2})] = \text{KLD}[q(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \ ||\ p(\mathbf{0}, \mathbb{I}_{2\times 2})] - \frac{1}{2}\log 200$. This means that using the standard Normal up-weights the regularization (towards the prior) by about a factor of

[6]Architecture / training paramters: 2000 latent noise dimensions, 1000 hidden dimensions, ReLU activations, $.0003$ learning rate, 50 samples for VR-max term.

$\sqrt{200}$.

5.4 Conclusions

I have introduced a flexible, widely applicable, and derivation-free method for approximating reference priors. It optimizes a new lower bound on the reference prior objective and allows for parametric or non-parametric approximations to be employed, depending on whether the user prefers to easily evaluate the prior or to have a maximally expressive approximation. I demonstrated quantitatively and qualitatively that these methods can recover the true reference priors for univariate distributions as well as generalize to more exotic models such as variational autoencoders.

Looking forward, I believe using similar techniques for constructing priors that optimize objectives other than mutual information presents a promising next step. For example, Liu et al. [2014] showed that priors that maximize divergence measures other than KLD, such as Hellinger distance, between the prior and posterior have desirable properties. Extending the proposed approximation techniques to these other families of objectives may enable new classes of Bayesian prior distributions.

Chapter 6

Learning Priors for Invariance

> I have had my results for a long time: but I do not yet know how I am to arrive at them.
>
> Carl Friedrich Gauss

The modeler often knows some prior information that is essential for obtaining good performance, and it is common to incorporate this knowledge via 'engineering tricks' or methods less principled than Bayesian inference. For example, achieving state-of-the-art performance on image classification frequently requires data set augmentation [He et al., 2016]: creating new training instances by flipping, scaling, rotating, etc. the original images [Baird, 1992]. Another example is using feature dropout on bag-of-words representations to simulate the effect of varying a document's length [Wager et al., 2016]. Training a model under these stochastic augmentation or perturbation strategies is in effect inducing a prior, one that encourages robustness with respect to these known invariances.

While methods for including prior information via means other than the likelihood or prior are undoubtedly highly effective in practice, they are difficult to reconcile with principled probabilis-

tic inference. One problem is that the resulting Bayesian posteriors are conditioned on an artificial training set, not on what is truly observed. Another issue is that, as is the case when training with dropout, it is unclear how to interpret the regularization mechanism: are the masking variables model parameters and if so should we be computing their posterior? These and related questions motivate recent work on formulating dropout as (approximate) Bayesian inference [Gal and Ghahramani, 2016a,b, Kingma et al., 2015].

In this chapter, I propose a method for transferring a modeler's knowledge about invariances into a corresponding Bayesian prior[1]. Doing so allows data set augmentation, dropout, and other effective regularization strategies to be incorporated into the model as a proper Bayesian prior. Once this is done, Bayesian inference can proceed as usual without complication of or the need to re-interpret the inference strategy (whatever it may be: Markov chain Monte Carlo, variational inference, maximum a posteriori estimation, etc.).

My proposed approach is to formulate a variational problem [Blei et al., 2017]: given a parameteric family, find the member of the family that, when used as a prior, makes the model as near to invariant as possible. To do this, I first derive a lower bound that quantifies the model's invariance under some specific perturbation process. I then maximize this bound with respect to the parameters of the parameteric family. An important detail to note is that I am *not* performing empirical Bayesian inference. Rather, I learn the prior from the data *model*, similarly to how objective priors are specified [Jeffreys, 1946, Bernardo, 2005, Nalisnick and Smyth, 2017]. For supervised models, this means that only the features are needed, making my method well suited for semi-supervised settings, as the experiments demonstrate.

[1]The work in this chapter was originally presented in Nalisnick and Smyth [2018]

6.1 Preliminaries

Before describing the proposed methods, I begin by defining perturbation processes and invariant statistical models. I use the following notation throughout the chapter. As our primary focus is on supervised learning, I denote input features as $\mathbf{x}_i \in \mathbb{R}^d$ and labels (indicating class membership or a real-valued response) as y_i, where i indexes the observed data. Define the data model (likelihood function) to be $p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ are the model parameters. Thus, in the Bayesian setting, $p(\boldsymbol{\theta})$ denotes the prior and $p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X})$ the posterior. I write all expectations, entropies, and divergences in their continuous form (i.e. with integrals), but sums should be used when the support is discrete.

6.1.1 Perturbation Processes

Many of the recent successes in supervised machine learning have come from data augmentation and corruption processes that perturb observations and parameters. These processes have the effect of regularizing the classifier to which they are applied by implicitly encoding user knowledge. I define them formally and generally as follows. Call a generative process that takes in a random variable $\mathbf{z} \in \mathcal{Z}$ and samples a random transformation $\tilde{\boldsymbol{z}} \in \tilde{\mathcal{Z}}(\boldsymbol{z})$ a *perturbation process* (PP):

$$\tilde{\mathbf{z}} \sim q(\tilde{\mathbf{z}}; \mathbf{z}, \boldsymbol{\zeta}) \tag{6.1}$$

where $\mathbf{z}$ denotes the random variable pre-transformation, $\tilde{\mathbf{z}}$ denotes the same variable post-transformation, and $\boldsymbol{\zeta}$ are the parameters of q. Below I describe dropout and rotation as a PP acting on the features (i.e. $\mathbf{z} = \mathbf{x}$).

Dropout. Dropout corruption—where elements of the data or model parameters are set to zero at random—has been observed to consistently improve the held-out performance of logistic re-

gression [Wager et al., 2013, Maaten et al., 2013] and deep NNs [Srivastava et al., 2014]. In this chapter I focus on feature dropout, which can be written as a PP as follows:

$$\tilde{\mathbf{x}} = \boldsymbol{b} \odot \mathbf{x} \quad \text{where} \quad \boldsymbol{b} \sim \text{Bernoulli}(1-\zeta) \tag{6.2}$$

where $\odot$ denotes an element-wise product and $\zeta \in (0,1)$ the dropout probability. The random variable $\boldsymbol{b}$ acts, simply, as a element-wise mask on the feature vector $\boldsymbol{x}$.

Rotation. As mentioned in the Introduction, many image data sets exhibit rotations, and classifier performance can be improved by augmenting the data set with rotated version of the true observations. As a PP, 2D rotation can be written as

$$\tilde{\mathbf{x}} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{6.3}$$

where $\phi \sim \text{Uniform}(\zeta \in [0, 2\pi])$. Usually padding or some other standardization is used to keep the image size consistent.

In this chapter, I focus on dropout and rotation transformations, illustrating the proposed technique for point-wise and affine transformations, respectively. Applying the techniques to other operations in these classes would proceed in a similar manner. The techniques I propose can also be applied to just about any functional transform as long as the parameterized prior is sufficiently expressive.

6.1.2 Invariant Models

Invariant statistical models have been well studied, both in theory [Eaton, 1989] and in practice [Cohen and Welling, 2016, Schmidt and Roth, 2012]. The classic formulation is group theoretic, as in Eaton (1989) [Eaton, 1989]. I use a similar definition except that I require invariance with

respect to all members of a PP's support, which may not form a proper algebraic group.

Definition 6.1. *Let $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}(\mathbf{x})$ be a realization from a perturbation process $q(\cdot; \boldsymbol{\zeta})$ acting on $\mathbf{x} \in \mathcal{X}$, and let $\mathcal{P}_{y|x}$ be a family of models indexed by their parameters. A statistical model $p \in \mathcal{P}_{y|x}$ is $q_{\boldsymbol{\zeta}}$-invariant if*

$$p(y|\mathbf{x}) \;\; = \;\; p(y|\tilde{\mathbf{x}}) \quad \forall\, \tilde{\mathbf{x}} \in \tilde{\mathcal{X}}(\mathbf{x}).$$

Intuitively, this invariance property can be thought of as robustness: a dropout-invariant classifier, for instance, should produce the same output distribution no matter how the input features are corrupted. In the case of the usual Bernoulli(0.5) noise, however, it is unlikely a classifier could be meaningfully dropout-invariant since the probability that all features will be masked is non-zero.

6.2 Learning Invariant Priors

Having introduced PPs and defined model invariance, I next detail the proposed methodology. I begin by proposing a quantity representing a 'nearness' to invariance and then discuss how to minimize such a quantity with respect to the model's prior.

6.2.1 Quantifying Approximate Invariance

Recall that our goal is to learn a prior that prefers invariance, and thus we need some continuous quantity that represents how near to invariant a model is. Definition 2.1 is not appropriate as is, because it would require the equality be checked for all $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}(\mathbf{x})$. Instead, I consider the expectation of the model under q, which is also invariant:

If $p \in \mathcal{P}$ is q_ς-invariant, then $\mathbb{E}_{q_\varsigma}[p(y|\tilde{\mathbf{x}})]$ is q_ς-invariant:

$$\mathbb{E}_{q_\varsigma}[p(y|\tilde{\mathbf{x}})] = \int_{\tilde{\mathcal{X}}} p(y|\tilde{\mathbf{x}})\, q(\tilde{\mathbf{x}};\mathbf{x})\, d\tilde{\mathbf{x}} = p(y|\mathbf{x}) \int_{\tilde{\mathcal{X}}} q(\tilde{\mathbf{x}};\mathbf{x})\, d\tilde{\mathbf{x}} = p(y|\mathbf{x}). \tag{6.4}$$

This fact is useful for quantifying nearness to invariance because it weights $p(y|\tilde{\mathbf{x}})$ over $\tilde{\mathcal{X}}$, meaning that a lack of invariance for a particular $\tilde{\mathbf{x}}$ can be excused or neglected if $q(\tilde{\mathbf{x}};\mathbf{x})$ is near zero. Thus, quantifying the degree of invariance of a model reduces to computing some divergence between $p(y|\mathbf{x})$ and $\mathbb{E}_{q_\varsigma}[p(y|\tilde{\mathbf{x}})]$. I use the Kullback-Leibler divergence—$\mathsf{KLD}[\, p(y|\mathbf{x}) \,||\, \mathbb{E}_{q_\varsigma}[p(y|\tilde{\mathbf{x}})]\,]$—which is zero if and only if $p(y|\mathbf{x}) = \mathbb{E}_{q_\varsigma}[p(y|\tilde{\mathbf{x}})]$ almost everywhere and is positive otherwise. Since $\mathbb{E}_{q_\varsigma}[p(y|\tilde{\mathbf{x}})]$ will be intractable for most models of interest, I use the following upper bound on the divergence so that we can obtain an unbiased Monte Carlo approximation of the expectation, obtaining an upper bound via Jensen's inequality:

$$\begin{aligned}
\mathsf{KLD}[\, p(y|\mathbf{x}) \,||\, \mathbb{E}_{q_\varsigma}[p(y|\tilde{\mathbf{x}})]\,] &= \mathbb{E}_{p(y|\mathbf{x})}[\log p(y|\mathbf{x})] - \mathbb{E}_{p(y|\mathbf{x})}[\log \mathbb{E}_{q_\varsigma}[p(y|\tilde{\mathbf{x}})]] \\
&\leq \mathbb{E}_{p(y|\mathbf{x})}[\log p(y|\mathbf{x})] - \mathbb{E}_{p(y|\mathbf{x})}\mathbb{E}_{q_\varsigma}[\log p(y|\tilde{\mathbf{x}})] \\
&= \mathbb{E}_{q_\varsigma}\mathsf{KLD}[\, p(y|\mathbf{x}) \,||\, p(y|\tilde{\mathbf{x}})\,].
\end{aligned} \tag{6.5}$$

6.2.2 Exposing the Prior

I now discuss how to introduce Bayesian thinking into my formulations of invariance. Consider the aforementioned models as marginal likelihoods:

$$p(y|\mathbf{x}) = \int_{\boldsymbol{\Theta}} p(y|\mathbf{x},\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} = \mathbb{E}_{p(\boldsymbol{\theta})}[p(y|\mathbf{x},\boldsymbol{\theta})].$$

Looking ahead, our ultimate goal is to optimize Equation 6.5 with respect to $p(\boldsymbol{\theta})$. Ideally we would do this in its current marginalized form, but computing the marginal likelihood is notoriously difficult, even for relatively simple models. Hence, I again upper bound the divergence,

which in turn makes the quantity amenable to an unbiased Monte Carlo approximation:

$$\begin{aligned}\mathsf{KLD}[\, p(y|\mathbf{x}) \,||\, p(y|\tilde{\mathbf{x}}) \,] &= \mathsf{KLD}[\, \mathbb{E}_{p(\boldsymbol{\theta})}[p(y|\mathbf{x},\boldsymbol{\theta})] \,||\, \mathbb{E}_{p(\boldsymbol{\theta})}[p(y|\tilde{\mathbf{x}},\boldsymbol{\theta})] \,] \\ &\leq \mathbb{E}_{p(\boldsymbol{\theta})}\mathsf{KLD}[\, p(y|\mathbf{x},\boldsymbol{\theta}) \,||\, p(y|\tilde{\mathbf{x}},\boldsymbol{\theta}) \,].\end{aligned} \tag{6.6}$$

The bound follows directly from the fact that KLD is a convex function over the domain of probability distributions. With this upper bound, I expose $p(\boldsymbol{\theta})$ and make it accessible for optimization.

6.2.3 Optimization Objective

Let $\boldsymbol{\lambda}$ denote the parameters of the prior $p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})$. I propose optimizing $\boldsymbol{\lambda}$ by the following objective, which is formed by combining Equations 6.5 and 6.6 with an entropy term:

$$\begin{aligned}\mathcal{L}^*(\boldsymbol{\lambda};\mathbf{x}) &= \mathbb{H}_{\boldsymbol{\lambda}}[\boldsymbol{\theta}] - \mathsf{KLD}[\, p(y|\mathbf{x}) \,||\, \mathbb{E}_{q_{\boldsymbol{\zeta}}}[p(y|\tilde{\mathbf{x}})] \,] \\ &\geq \mathbb{H}_{\boldsymbol{\lambda}}[\boldsymbol{\theta}] - \mathbb{E}_{q_{\boldsymbol{\zeta}}}\mathsf{KLD}[\, p(y|\mathbf{x}) \,||\, p(y|\tilde{\mathbf{x}}) \,] \\ &\geq \mathbb{H}_{\boldsymbol{\lambda}}[\boldsymbol{\theta}] - \mathbb{E}_{p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}\mathbb{E}_{q_{\boldsymbol{\zeta}}}\mathsf{KLD}[\, p(y|\mathbf{x},\boldsymbol{\theta}) \,||\, p(y|\tilde{\mathbf{x}},\boldsymbol{\theta}) \,] = \mathcal{J}(\boldsymbol{\lambda};\mathbf{x})\end{aligned} \tag{6.7}$$

where $\mathbb{H}_{\boldsymbol{\lambda}}[\boldsymbol{\theta}] = -\int_{\boldsymbol{\theta}} p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \, \log p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \, d\boldsymbol{\theta}$. I assume the objective is optimized under the empirical distribution of feature observations, i.e. $\mathbb{E}_{p(\hat{\mathbf{x}})}[\mathcal{J}(\boldsymbol{\lambda};\mathbf{x})] = \frac{1}{N}\sum_i \mathcal{J}(\boldsymbol{\lambda};\mathbf{x}_i)$. Maximizing $\mathcal{J}$ w.r.t. $\boldsymbol{\lambda}$ means that we are finding the distribution that minimizes the expected divergence between the unperturbed and perturbed model—or in other words, the invariance—under the prior. This objective *does not* depend on the observed y's, only the model output distribution over y. Because of this fact we can use unlabeled feature observations during learning of the prior.

The inclusion of the entropy term in Equation 6.7 is motivated by the *principle of maximum entropy*, i.e., that the appropriate distribution for representing prior beliefs is one that obeys known constraints and has maximum entropy otherwise [Jaynes, 1957, Uffink, 1996]. This behavior is precisely what Equation 6.7 encourages; the first term encourages maximum entropy and the second imposes the invariance constraints. In practice, the entropy term encourages the prior to

avoid spurious solutions. For example, a NN could become dropout-invariant by learning as the prior a delta function at zero. I will show this phenomenon analytically for linear regression in Section 6.3.

Equation 6.7 is amenable to a wide range of parametric forms for the prior. For example, it supports mixture densities $p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p_{\boldsymbol{\lambda}_k}(\boldsymbol{\theta})$ where $p_{\boldsymbol{\lambda}_k}$ is the kth component with parameters $\boldsymbol{\lambda}_k$ and π_k is the corresponding mixture weight. When using a mixture for the prior, the divergence component of the objective can be written as

$$\mathbb{E}_{p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}\mathbb{E}_{q_\zeta(\boldsymbol{\zeta})}\text{KLD}[p_{\boldsymbol{\theta}}||p_{\boldsymbol{\theta}}(\boldsymbol{\zeta})] = \sum_k \pi_k \mathbb{E}_{p_{\boldsymbol{\lambda}_k}(\boldsymbol{\theta})}\mathbb{E}_{q_\zeta(\tilde{\mathbf{x}};\boldsymbol{\zeta})}\text{KLD}[p_{\boldsymbol{\theta}}(y_i|\mathbf{x}_i,\boldsymbol{\theta})||p_{\boldsymbol{\theta}}(y_i|\tilde{\mathbf{x}}_i,\boldsymbol{\theta})],$$

where the objective is evaluated under each component distribution and a weighted average taken according to the mixture weights.

6.3 Analytical Solution for Linear Regression

To build intuition and to further examine the proposed objective (Equation 6.7), I next show an analytical solution for linear regression under dropout noise and its connection to the popular *g-prior* [Goel and Zellner, 1986]. I use the unbiased form of dropout, meaning $\mathbb{E}[\tilde{\mathbf{x}}] = \mathbf{x}$ and $\text{Var}[\tilde{\mathbf{x}}] = \frac{1}{1-\zeta}\mathbf{x}^2$ [Maaten et al., 2013], and I set the prior to be a multivariate normal $p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) = \text{N}(\boldsymbol{\mu}_{\boldsymbol{\lambda}}, \text{diag}(\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}))$ with diagonal covariance matrix. Define the data model to be a standard linear model with Gaussian error: $y = \mathbf{x}^T\boldsymbol{\theta} + \epsilon_0, \epsilon_0 \sim \text{N}(0, \sigma_0^2)$. The divergence portion of the objective

simplifies to:

$$\begin{aligned}\mathbb{E}_{p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}\mathbb{E}_{q_{\boldsymbol{\zeta}}}\mathsf{KLD}[\, p(y|\mathbf{x},\boldsymbol{\theta}) \,||\, p(y|\tilde{\mathbf{x}},\boldsymbol{\theta}) \,] &= \mathbb{E}_{p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}\mathbb{E}_{q_{\boldsymbol{\zeta}}}\left[\frac{(\mathbf{x}^T\boldsymbol{\theta} - \tilde{\mathbf{x}}^T\boldsymbol{\theta})^2}{2\sigma_0^2}\right] \\ &= \mathbb{E}_{p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}\left[\frac{(\mathbf{x}^T\boldsymbol{\theta})^2}{2\sigma_0^2(1-\zeta)}\right] \\ &= \frac{(\mathbf{x}^T\boldsymbol{\mu}_{\boldsymbol{\lambda}})^2 + \mathbf{x}^T\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}\mathbf{x}}{2\sigma_0^2(1-\zeta)}\end{aligned} \tag{6.8}$$

If the proposed objective consisted of only the divergence (invariance) term, minimizing the equation above would clearly lead to both $\boldsymbol{\mu}_{\boldsymbol{\lambda}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}$ being set to zero. In other words, the optimal prior would be $p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) = \boldsymbol{\delta}_0$, the delta function placed at zero.

The solution becomes much more interesting when the entropy term is included. The full objective can be written as:

$$\mathcal{J}_{\mathsf{LR}}(\boldsymbol{\lambda};\mathbf{x}) = \log\det(\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}) - \frac{(\mathbf{x}^T\boldsymbol{\mu}_{\boldsymbol{\lambda}})^2 + \mathbf{x}^T\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}\mathbf{x}}{2\sigma_0^2(1-\zeta)}. \tag{6.9}$$

Since the entropy term does not include the prior's mean, the optimal solution for this parameter is still $\boldsymbol{\mu}_{\boldsymbol{\lambda}} = \mathbf{0}$. Differentiating $\mathcal{J}_{\mathsf{LR}}$ with respect to $\sigma_{\boldsymbol{\lambda}}$, the optimal covariance matrix is $\sigma_0^2(1-\zeta)\mathsf{diag}(\mathbf{x}^T\mathbf{x})^{-1}$. Putting these together I obtain the final solution for the prior: $p^*_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) = \mathsf{N}(\mathbf{0}, \sigma_0^2(1-\zeta)\mathsf{diag}(\mathbf{x}^T\mathbf{x})^{-1})$.

Interestingly, the solution is equivalent to a diagonalized version of the well-known *g-prior* [Goel and Zellner, 1986]—$\mathsf{N}(\mathbf{0}, g(\mathbf{x}^T\mathbf{x})^{-1})$—with g set by the dropout level. The g-prior has the nice property that the posterior mean is a linear combination of the prior mean and maximum likelihood estimator: $\boldsymbol{\theta}_{\mathsf{post}} = \frac{g}{1+g}\boldsymbol{\theta}_{\mathsf{MLE}} + (1 - \frac{g}{1+g})\boldsymbol{\mu}_{\lambda}$. Thus, in the case of the prior learned by my proposed method, we see the dropout rate plays the role of multiplicative shrinkage of the ML solution.

6.4 Black-Box Learning for Intractable Models

For most problems of interest we will not be able to analytically solve the objective's required integrals, as in the previous section. Hence, in this section I describe how to make learning derivation-free and 'black-box' using recently developed techniques from posterior variational inference [Ranganath et al., 2014, Kingma and Welling, 2014a]. Specifically, I use Monte Carlo approximations combined with differentiable non-centered parameterizations [Kingma and Welling, 2014a] to make learning fully gradient-based no matter how complicated the likelihood function is. I also discuss how to use what I call an 'implicit prior'—a highly expressive functional sampler.

Monte Carlo Expectations. For most modern, large-scale models, computing the expectations w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\zeta}$ will not be feasible analytically. Thus we turn to a nested Monte Carlo (MC) approximation:

$$\mathbb{E}_{p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}\mathbb{E}_{q_{\boldsymbol{\zeta}}}\mathsf{KLD}[p(y|\mathbf{x},\boldsymbol{\theta})||p(y|\tilde{\mathbf{x}},\boldsymbol{\theta})] \approx \frac{1}{SM}\sum_{s=1}^{S}\sum_{m=1}^{M}\mathsf{KLD}[p(y|\mathbf{x},\hat{\boldsymbol{\theta}}_s)||p(y|\hat{\tilde{\mathbf{x}}}_m,\hat{\boldsymbol{\theta}}_s)] \tag{6.10}$$

such that M samples are drawn from the perturbation process $\hat{\tilde{\mathbf{x}}}_m \sim q(\tilde{\mathbf{x}};\boldsymbol{\zeta})$ and S samples are drawn from the prior we wish to learn $\hat{\boldsymbol{\theta}}_s \sim p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})$.

Differentiable Sampling. Using MC approximations makes computing derivatives w.r.t. the prior's parameters $\boldsymbol{\lambda}$ difficult, as they need to be computed through the samples $\hat{\boldsymbol{\theta}}_s$:

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\lambda}}\sum_{s=1}^{S}\sum_{m=1}^{M}&\mathsf{KLD}[p(y|\mathbf{x},\hat{\boldsymbol{\theta}}_s)||p(y|\hat{\tilde{\mathbf{x}}}_m,\hat{\boldsymbol{\theta}}_s)] \\ &= \sum_{s=1}^{S}\sum_{m=1}^{M}\frac{\partial}{\partial\hat{\boldsymbol{\theta}}_s}\mathsf{KLD}[p(y|\mathbf{x},\hat{\boldsymbol{\theta}}_s)||p(y|\hat{\tilde{\mathbf{x}}}_m,\hat{\boldsymbol{\theta}}_s)]\frac{\partial\hat{\boldsymbol{\theta}}_s}{\partial\boldsymbol{\lambda}}.\end{aligned} \tag{6.11}$$

One way we can ensure $\frac{\partial\hat{\boldsymbol{\theta}}_s}{\partial\boldsymbol{\lambda}}$ is computable is by sampling $\boldsymbol{\theta}$ by way of a *differentiable non-centered parameterization* [Kingma and Welling, 2014a] (DNCP), which has the general form $\hat{\boldsymbol{\theta}} = g(\boldsymbol{\lambda},\hat{\boldsymbol{\epsilon}})$ where $\hat{\boldsymbol{\epsilon}} \sim p(\boldsymbol{\epsilon})$. $\boldsymbol{\epsilon}$ is an auxiliary variable drawn from some fixed distribution and

g is a differentiable function. A well-known example of a DNCP is the Gaussian's location-scale form $\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \hat{\boldsymbol{\epsilon}}$ where $\boldsymbol{\epsilon}$ is drawn from a standard Normal distribution.

Implicit Priors. Notice that when using MC approximations of the integrals (Equation 6.10), the only term in Equation 6.7 that requires the prior's density be evaluated is the entropy term. Thus, using a nonparametric estimate for $\mathbb{H}[\boldsymbol{\theta}]$ [Beirlant et al., 1997] can completely remove the need to evaluate the prior. Doing so allows us to use what I call an *implicit* prior: a prior from which we can draw samples but which we cannot evaluate as a density function, i.e. $\hat{\boldsymbol{\theta}} = f(\boldsymbol{\lambda}, \hat{\boldsymbol{\epsilon}})$ where $\hat{\boldsymbol{\epsilon}}$ is a sample drawn from some fixed distributions and $f(\cdot)$ is some differentiable, sufficiently flexible function such as a NN. Treating the prior as a simulator in this way is similar to the ideas behind *Generative Adversarial Networks* [Goodfellow et al., 2014] and *Variational Programs* [Ranganath et al., 2016]. The benefit of using an implicit prior is that we can have an extremely flexible distribution over $\boldsymbol{\theta}$; the downside is that we will eventually need to evaluate the implicit prior's density—possibly having to turn to nonparametric density estimation.

6.5 Related Work

The closest work to what I propose is prior work on the definition and specification of objective priors [Jeffreys, 1946, Bernardo, 2005, Nalisnick and Smyth, 2017]. I say that not because my method learns noninformative priors—quite the opposite—but because the method I propose here learns a prior based on the data model, just as objective priors do. For conditional models, the feature variables must be included to define the model, and thus, the prior is dependent on the observed data. This fact links my method (and objective priors) to empirical Bayesian inference [Casella, 1985]. However, a significant difference between my method and empirical Bayesian methods is that the variable being modeled (the classification label or regression response) is not considered in my prior's specification, as it typically is for most empirical Bayesian methods [Casella, 1985].

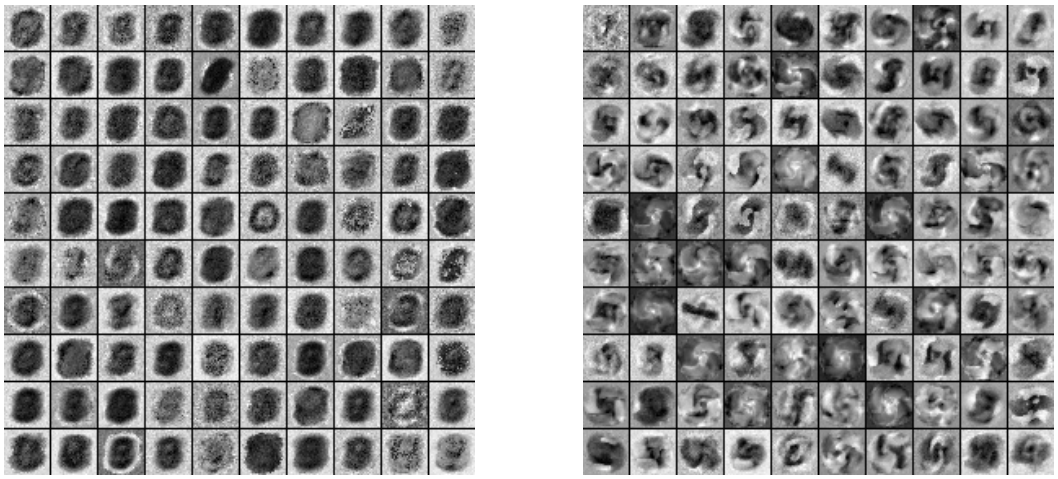
As for the specifics of the proposed optimization objective, its form is motivated by the principle of maximum entropy, which has a long history dating back to statistical physics [Jaynes, 1957, Guiasu and Shenitzer, 1985]. There has been some work on learning invariant maximum entropy distributions [Nieves et al., 2010] and approximations to such distributions [Li and Risteski, 2016], but this previous work is tailored to specific settings (soil analysis and pairwise moment mean parameters, respectively). In contrast, my approach requires only samples from the perturbation and the distribution to be estimated (the prior).

More closely related is the work of Bachman et al. [2014] on *pseudo-ensemble agreement* regularization. They propose a regularization penalty of the form:

$$\mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{x \sim p_x} \mathbb{E}_{\xi \sim p_\xi} \mathcal{V}[f_{\boldsymbol{\theta}}(x), f_{\boldsymbol{\theta}}(x; \xi)],$$

where the first expectation is with respect to the empirical distribution of the features, the second expectation is with respect to a noise process (such as dropout corruption [Srivastava et al., 2014]), and $\mathcal{V}[\cdot, \cdot]$ is some way to measure the discrepancy between the unperturbed and pertrubed model f_θ. The divergence term I propose in Equation 6.7 is a special case of Bachman et al.'s penalty: Equation 6.7 can be obtained by setting $\mathcal{V}[\cdot, \cdot]$ to be KLD (as Bachman et al. [2014] do in some experiments) and adding an expectation over the model parameters. The key difference between Bachman et al.'s work and what I propose is that they use their regularization term within a penalized likelihood framework. There is no concept of learning a Bayesian prior nor one of transferring the stochastic regularization into a probability distribution.

Lastly, this work has been inspired by recent efforts to analyze dropout both from the perspectives of penalized likelihood [Bachman et al., 2014, Wager et al., 2013, Wang and Manning, 2013] and approximate Bayesian inference [Kingma et al., 2015, Gal and Ghahramani, 2016a,b]. In the former category, Bachman et al. [2014], Wager et al. [2013], and Wang and Manning [2013] carry out analyses of linear regression that are similar to that in Section 6.3. However, their analyses



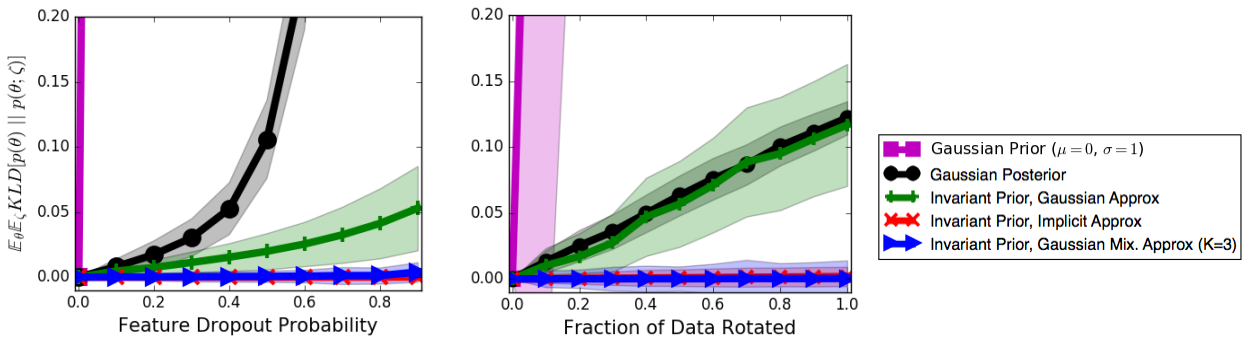
(a) Dropout Invariant (b) Rotation Invariant

Figure 6.1: *Weight Visualization.* Above I show filter visualizations for 100 weight matrices sampled from two learned implicit priors, one invariant to dropout and one invariant to rotation. Both were trained on MNIST. The dropout invariant prior can be seen to down-weight features found around the center of the image, which is where the active features usually are found. The rotation invariant prior learns spiral feature transformations roughly similar to some of the features learned by Toroidal Subgroup Analysis (see Figure 3 in Cohen and Welling [2014]).

are motivated by seeking a closed-form regularization penalty that mimics the effect of dropout. There are no notions of Bayesian priors, and my development of the connection to the g-prior is new. In the latter category, Kingma et al. [2015] and Gal and Ghahramani [2016a,b] show that dropout can be interpreted as approximate Bayesian inference under certain variational posteriors. This work has similar motivations—that is, to link dropout and Bayesian methodology—but I do so via the Bayesian prior. I formulate the prior that corresponds to dropout thus allowing inference to proceed with no constraints and by way of either MCMC or variational methods.

6.6 Experiments

In this section I report empirical analyses of the proposed methods, focusing on dropout and rotation transformations. First, I discuss some qualitative properties of the learned priors by visualizing them as weight filters. Second, I quantitatively analyze the degree of invariance of



(a) Feature Dropout (b) Rotation

Figure 6.2: *Invariance vs Perturbation Magnitude.* The plots above shows the robustness of several distributions (y-axis shows $\mathbb{E}_{p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}\mathbb{E}_{q_\zeta}KLD[\ p(y|\mathbf{x},\boldsymbol{\theta})\ ||\ p(y|\tilde{\mathbf{x}},\boldsymbol{\theta})\ ]$) to dropout and rotation perturbations of increasing magnitude (x-axis). I compare the proposed invariant priors—three approximations: implicit (red), factorized Gaussian (green), Gaussian mixture (blue)—to a standard Normal prior (pink) and the posterior (black) after training on perturbed data. We see the learned invariant priors exhibit invariance across all perturbation magnitudes, especially when using implicit or mixture approximations.

several distributions with respect to dropout and rotation perturbations. And lastly, I describe classification tasks to demonstrate that using the proposed invariant priors results in accuracy on par with that of non-Bayesian methods. The multi-class classification experiments use NN likelihoods of the form $y_i \sim \text{Multinoulli}(\mathbf{p} = \gamma_{\boldsymbol{\theta}}(\mathbf{x}_i))$ where $\gamma_{\boldsymbol{\theta}}(\mathbf{x}_i) = \text{softmax}(\mathbf{h}_L\boldsymbol{\theta}_{L+1})$, the softmax output of one or more NN layers with the form $\mathbf{h}_{l+1} = \text{ReLU}(\mathbf{h}_l\boldsymbol{\theta}_{l+1})$, where $\mathbf{h}_0 = \mathbf{x}_i$. The binary classification (sentiment analysis) experiment uses a logistic regression likelihood of the form $y_i \sim \text{Bernoulli}(p = \text{logistic}(\mathbf{x}_i\boldsymbol{\theta}))$.

Regarding hyperparameter selection, Adam [Kingma and Ba, 2014] was used for all experiments with a learning rate chosen from $\{.001, .0005, .0001, .00005\}$ via a validation set (other parameters kept at Tensorflow defaults). For the Monte Carlo approximations used to learn the invariant priors, 50 samples were used for both the parameters and perturbation process. The best priors were selected based on those which obtained the highest value of Equation 6.7 upon convergence. All posteriors were obtained via *Stochastic Gradient Variational Bayes* [Kingma and Welling, 2014b], and the posterior mean was used to calculate test performance in all cases.

Qualitative Analysis. I begin by performing visual inspection of the invariant priors. I do this by

	Test Error (%)
SVM [Larochelle et al., 2007]	10.38
Bayesian Neural Net w/ N$(0, .0025)$ prior	10.08
Neural Net w/ Dropout	8.85
CNN [Cohen and Welling, 2016]	5.03
Bayesian Neural Net w/ Invariant Prior (Factorized Gaussian)	9.41
Bayesian Neural Net w/ Invariant Prior (Mixture of Three Gaussians)	8.29
Rotation-Invariant RBM [Sohn and Lee, 2012]	4.20
Rotation-Aware ConvRBM [Schmidt and Roth, 2012]	3.98
Group Equivariant CNN [Cohen and Welling, 2016]	2.28
Harmonic Networks [Worrall et al., 2017]	**1.69**

Table 6.1: *Rotated MNIST.* Test classification error on a data set of rotated hand-written digits [Larochelle et al., 2007]. The first four models (from the top) have no notion of rotation, the next two have rotation invariant priors (ours), and the last two have rotations explicitly parameterized in the model and represent the current state-of-the-art.

learning an implicit prior (one-hidden-layer NN, 1000 hidden units) for the two weight matrices of a one-hidden-layer NN with 500 hidden units. I trained the prior on the MNIST data set under dropout and rotation perturbations (separately).

Samples from the prior on the first layer weights are shown in Figure 6.1. Subfigure (a) shows filter samples from the prior learned under dropout noise. The weights near the center of the image are conspicuously lower (i.e. darker) than those on the edges. This is expected, as placing low-weights on frequently active features reduces the effect of dropping out those features. Wager et al. [2013] come to a similar conclusion: dropout penalizes the weights of rare features less harshly than it does those of common features. Subfigure (b) shows the filter samples learned under rotation perturbations. We see they exhibit spiral transformations, which is expected since being rotation invariant would require that features similar distances from the image center receive near equal weight.

Quantitative Analysis. Next I quantitatively analyze the invariance properties of the priors. I quantify invariance based on the the proposed objective's KLD term, i.e. $\mathbb{E}_{p_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}\mathbb{E}_{q_{\boldsymbol{\zeta}}}\text{KLD}[\, p(y|\mathbf{x}, \boldsymbol{\theta}) \,||\, p(y|\tilde{\mathbf{x}}, \boldsymbol{\theta}) \,]$. I calculate this quantity by drawing a sample from the prior, drawing a sample perturbation, and

	Test Error (%)
Logistic Regression w/ L2	14.22^*
Bayesian Logistic Regression w/ $\mathrm{N}(0, .25)$ prior	14.19^*
Transductive SVM	13.98
Logistic Regression w/ MC Dropout	12.83^*
Logistic Regression w/ CF Dropout	11.90
Bayesian Logistic Regression w/ Invariant Prior (Factorized Gaussian)	11.93
Bayesian Logistic Regression w/ Invariant Prior (Mixture of Three Gaussians)	**11.81**

Table 6.2: *IMDB Sentiment Analysis.* Test classification error on the (large) IMDB sentiment analysis data set [Maas et al., 2011]. $*$ indicates a method was trained without the unlabeled examples. *MC*: Monte Carlo, *CF*: Closed-Form.

computing the KLD between the unperturbed and perturbed models with the sampled parameters and perturbation. I repeat the process 500 times and average the runs to obtain the final result. Again, the model I used for the experiment was a one-hidden-layer NN (500 hidden units) and the data set was MNIST undergoing dropout and rotation perturbations.

I trained three forms of invariant priors—a factorized Gaussian (green), a three-component mixture of factorized Gaussians (blue), and an implicit prior (red) parameterized by a one-hidden-layer NN (1000 hidden units)—and compare them to a standard Normal prior (pink) and a factorized Gaussian posterior (black) in Figure 6.2. The Gaussian posterior was obtained by training the network on MNIST with stochastic perturbations sampled for each forward pass. We see that, in the case of dropout (Subfigure a), all learned priors are markedly more robust to dropout noise than the two Gaussian baselines. The Gaussian mixture and implicit priors remain invariant at even a high noise level (> 0.8), showing only a slight upward trend. In the case of rotation (Subfigure b), the factorized Gaussian posterior and invariant prior have nearly identical invariance, but again the implicit and mixture invariant priors are notably robust across all perturbation levels.

Fully-Supervised Classification. Next I report results on (fully) supervised classification experiments on the rotated MNIST data set [Larochelle et al., 2007], which consists of 12,000 train-

ing images and 50,000 test images. 2,000 images were used as a validation set and recombined into the training set to obtain the final test performance, following [Cohen and Welling, 2016, Larochelle et al., 2007]. I trained three Bayesian NNs (NNs)—one with a standard Normal prior (variance chosen by validation set), one with a factorized Gaussian rotation invariant prior, and one with a mixture of Gaussians ($K = 3$) rotation invariant prior—and a NN with Bernoulli$(.5)$ dropout. All networks had two hidden layers with 2,750 units each.

Test set classification error is shown in Table 6.1. The table is divided into three sections: the first has no concept of rotation, the second has a rotation invariant prior, and the third has rotation-invariance built into the data model. We see that the invariant priors allow the Bayesian NNs to perform comparably to (factortized Gaussian) or better than (mixture of Gaussians) all of the models with no built-in concept of rotation except the Convolutional NN. However, the performance gap between the models with invariant priors and models with rotations explicitly parameterized (bottom four) is still considerable. I conjecture that the gap is due to the prior learning coarse rotational invariance. To elaborate, the filters preferred by the prior (Figure 1 (b)) do not exhibit the fine, digit-specific rotated edge detectors learned by the parameterized models, as seen in Schmidt and Roth [2012]. The ability to learn these refined rotations likely boosts performance considerably. Moreover, these models have been extensively hand-crafted to be rotationally invariant while my method is general and requires no additional effort from the modeler.

Semi-Supervised Classification. Lastly I report results on semi-supervised classification experiments on the large IMDB data set [Maas et al., 2011], which consists of 50,000 unlabeled examples and 25,000 for training and testing each. 5,000 of the training examples were used as a validation set. I trained several Bayesian and non-Bayesian logistic regression models, including one with the closed-form (CF) dropout penalty proposed by Wager et al. [2013][2]. I used the unlabeled data to train dropout invariant priors as well as the CF dropout penalty.

[2]The error is higher than what is reported by Wager et al. [2013] due to using unigrams and a smaller vocabulary (20,000 words).

Test set classification error is shown in Table 6.2. The Bayesian logistic regression model with a Gaussian mixture invariant prior achieves the lowest error rate, even besting the closed-form dropout penalty, which has the ability to learn the regularization and data model jointly. I conjecture that invariant priors were able to achieve better comparative performance in this setting because dropout is a simpler perturbation.

6.7 Conclusions

I have proposed an optimization objective (Equation 6.7) for learning priors that represent known invariance constraints. When the objective has an analytical solution (Section 6.3), we see that the resulting distribution is sensible and that both the objective's components are necessary. Experimentally, I demonstrated use of the prior results in better performance than when the invariance is not accounted for. Only models extensively hand-crafted for the invariance setting outperformed use of my proposed prior. This work, I believe, represents an important first step in allowing subjective priors to be specified for modern, large-scale Bayesian models.

Chapter 7

Nonparametric Priors for Density Networks

> There is a concept which corrupts and upsets all others. I refer not to Evil, whose limited realm is that of ethics; I refer to the infinite.
>
> Jorge Luis Borges
>
> Avatars of the Tortoise

Deep generative models trained via *stochastic gradient variational Bayes* (SGVB) [Kingma and Welling, 2014b, Rezende et al., 2014] efficiently couple the expressiveness of deep neural networks with the robustness to uncertainty of probabilistic latent variables. This combination has lead to their success in tasks ranging from image generation [Gregor et al., 2015, Rezende et al., 2016] to semi-supervised learning [Kingma et al., 2014, Maaløe et al., 2016] to language modeling [Bowman et al., 2016]. Various extensions to SGVB have been proposed [Burda et al., 2016, Maaløe et al., 2016, Salimans et al., 2015], but one conspicuous absence is an extension to Bayesian nonparametric processes. Using SGVB to perform inference for nonparametric distri-

butions is quite attractive. For instance, SGVB allows for a broad class of non-conjugate approximate posteriors and thus has the potential to expand Bayesian nonparametric models beyond the exponential family distributions to which they are usually confined. Moreover, coupling nonparametric processes with neural network inference models equips the networks with automatic model selection properties such as a self-determined width, which I explore in this chapter.

I make progress on this problem by first describing how to use SGVB for posterior inference for the weights of stick-breaking processes[1] [Ishwaran and James, 2001]. This is not a straightforward task as the beta distribution, the natural choice for an approximate posterior, does not have the differentiable non-centered parametrization that SGVB requires. I bypass this obstacle by using the little-known *Kumaraswamy* distribution [Kumaraswamy, 1980].

Using the Kumaraswamy as an approximate posterior, I then reformulate two popular deep generative models—the *variational autoencoder* [Kingma and Welling, 2014b] and its semi-supervised variant (model M2 proposed by Kingma et al. [2014])—into their nonparametric analogs. These models perform automatic model selection via an infinite capacity hidden layer that employs as many stick latent variables as the data requires. I experimentally show that, for data sets of natural images, stick-breaking priors improve upon previously proposed deep generative models by having a latent representation that better preserves class boundaries and provides beneficial regularization for semi-supervised learning.

7.1 Stick-Breaking Processes

First I define *stick-breaking processes* with the ultimate goal of using their weights for the VAE's prior $p(\mathbf{z})$. A random measure is referred to as a *stick-breaking prior* (SBP) [Ishwaran and James, 2001] if it is of the form $G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\zeta_k}$ where δ_{ζ_k} is a discrete measure concentrated at $\zeta_k \sim G_0$, a draw from the base distribution G_0 [Ishwaran and James, 2001]. The π_ks are random

[1]The work in this chapter is a combination of Nalisnick and Smyth [2017c] and Nalisnick et al. [2016]

weights independent of G_0, chosen such that $0 \leq \pi_k \leq 1$, and $\sum_k \pi_k = 1$ almost surely. SBPs have been termed as such because of their constructive definition known as the *stick-breaking process* [Sethuraman, 1994]. Mathematically, this definition implies that the weights can be drawn according to the following iterative procedure:

$$\pi_k = \begin{cases} v_1 \text{ if } k = 1 \\ v_k \prod_{j<k}(1 - v_j) \text{ for } k > 1 \end{cases} \tag{7.1}$$

where $v_k \sim \text{Beta}(\alpha, \beta)$. When $v_k \sim \text{Beta}(1, \alpha_0)$, then we have the stick-breaking construction for the *Dirichlet Process* [Ferguson, 1973]. In this case, the name for the joint distribution over the infinite sequence of stick-breaking weights is the Griffiths, Engen and McCloskey distribution with concentration parameter α_0 [Pitman, 2002]: $(\pi_1, \pi_2, \ldots) \sim \text{GEM}(\alpha_0)$.

7.2 SGVB for GEM Random Variables

Having covered the relevant background material, I now discuss the first contribution of this chapter, using Stochastic Gradient Variational Bayes for the weights of a stick-breaking process. I focus on performing inference for just the series of stick-breaking weights, which I refer to as GEM random variables after their joint distribution.

7.2.1 Composition of Gamma Random Variables

In the original SGVB paper, Kingma and Welling [2014b] suggest representing the Beta distribution as a composition of Gamma random variables by using the fact $v \sim \text{Beta}(\alpha, \beta)$ can be sampled by drawing Gamma variables $x \sim \text{Gamma}(\alpha, 1)$, $y \sim \text{Gamma}(\beta, 1)$ and composing them as $v = x/(x+y)$. However, this representation still does not admit a DNCP as the Gamma distribution does not have one with respect to its shape parameter. Knowles [2015] suggests that

when the shape parameter is near zero, the following asymptotic approximation of the inverse CDF is a suitable DNCP:

$$F^{-1}(\hat{u}) \approx \frac{(\hat{u}a\Gamma(a))^{\frac{1}{a}}}{b} \tag{7.2}$$

for $\hat{u} \sim \text{Uniform}(0, 1)$, shape parameter a, and scale parameter b. This approximation becomes poor as a increases, however, and Knowles recommends a finite difference approximation of the inverse CDF when $a \geq 1$.

7.2.2 The Kumaraswamy Distribution

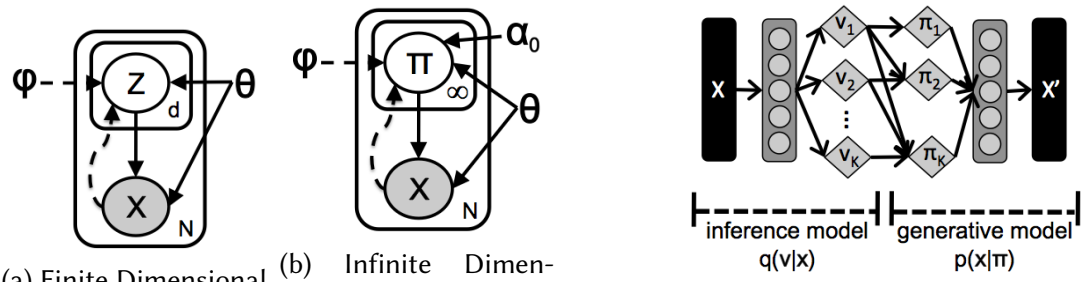
Another candidate posterior is the little-known *Kumaraswamy* distribution [Kumaraswamy, 1980]. It is a two-parameter continuous distribution also on the unit interval with a density function defined as

$$\text{Kumaraswamy}(x; a, b) = abx^{a-1}(1 - x^a)^{b-1} \tag{7.3}$$

for $x \in (0, 1)$ and $a, b > 0$. In fact, if $a = 1$ or $b = 1$ or both, the Kumaraswamy and Beta are equivalent, and for equivalent parameter settings, the Kumaraswamy resembles the Beta albeit with higher entropy. The DNCP we desire is the Kumaraswamy's closed-form inverse CDF. Samples can be drawn via the inverse transform:

$$x \sim (1 - u^{\frac{1}{b}})^{\frac{1}{a}} \text{ where } u \sim \text{Uniform}(0, 1). \tag{7.4}$$

Not only does the Kumaraswamy make sampling easy, its KL-divergence from the Beta can be closely approximated in closed-form (for ELBO computation).



(a) Finite Dimensional (b) Infinite Dimensional (c) The Stick-Breaking Variational Autoencoder.

Figure 7.1: Subfigures (a) and (b) show the plate diagrams for the relevant latent variable models. Solid lines denote the generative process and dashed lines the inference model. Subfigure (a) shows the finite dimensional case considered in [Kingma and Welling, 2014b], and (b) shows the infinite dimensional case of our concern. Subfigure (c) shows the feedforward architecture of the *Stick-Breaking Autoencoder*, which is a neural-network-based parametrization of the graphical model in (b).

7.2.2.1 Gauss-Logit Parametrization

Another promising parametrization is inspired by the *Probit Stick-Breaking Process* [Rodriguez and Dunson, 2011]. In a two-step process, we can draw a Gaussian and then use a squashing function to map it on $(0, 1)$:

$$\hat{v}_k = g(\mu_k + \sigma_k \epsilon) \tag{7.5}$$

where $\epsilon \sim \mathrm{N}(0, 1)$. In the Probit SBP, $g(\cdot)$ is taken to be the Gaussian CDF, and it is chosen as such for posterior sampling considerations. This choice is impractical for our purposes, however, since the Gaussian CDF does not have a closed form. Instead, I use the logistic function $g(x) = 1/(1 + e^{-x})$.

7.3 Stick-Breaking Variational Autoencoders

Given the discussion above, I now propose the following novel modification to the VAE. Instead of drawing the latent variables from a Gaussian distribution, I draw them from the GEM distri-

bution, making the hidden representation an infinite sequence of stick-breaking weights. I term this model a *Stick-Breaking Variational Autoencoder* (SB-VAE) and below detail the generative and inference processes implemented in the decoding and encoding models respectively.

7.3.1 Generative Process

The generative process is nearly identical to previous VAE formulations. The crucial difference is that we draw the latent variable from a stochastic process, the GEM distribution. Mathematically, the hierarchical formulation is written as

$$\boldsymbol{\pi}_i \sim \text{GEM}(\alpha_0)\ ,\ \mathbf{x}_i \sim p_{\boldsymbol{\theta}}(\mathbf{x}_i|\boldsymbol{\pi}_i) \tag{7.6}$$

where $\boldsymbol{\pi}_i$ is the vector of stick-breaking weights and α_0 is the concentration parameter of the GEM distribution. The likelihood model $p_{\boldsymbol{\theta}}(\mathbf{x}_i|\boldsymbol{\pi}_i)$ is a density network as described in Section 2.2.2.

7.3.2 Inference

The inference process—how to draw $\boldsymbol{\pi}_i \sim q_{\boldsymbol{\phi}}(\boldsymbol{\pi}_i|\mathbf{z}_i)$—requires modification from the standard VAE's in order to sample from the GEM's stick-breaking construction. Firstly, an inference network computes the parameters of the K fraction distributions and samples values $v_{i,k}$ according to one of the parametrizations in Section 7.2. Next, a linear-time operation composes the stick segments from the sampled fractions:

$$\boldsymbol{\pi}_i = (\pi_{i,1}, \pi_{i,2}, \ldots, \pi_{i,K}) = \left(v_{i,1}, v_{i,2}(1 - v_{i,1}), \ldots, \prod_{j=1}^{K-1}(1 - v_{i,j}) \right). \tag{7.7}$$

The computation path is summarized in Figure 7.1 (c) with arrows denoting the direction of feedforward computation. The gray blocks represent any deterministic function that can be trained with gradient descent—i.e. one or more neural network layers. Optimization of the SB-VAE is done just as for the VAE, by optimizing Equation 2.31 w.r.t. $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. The KL divergence term can be computed (or closely approximated) in closed-form for all three parametrizations under consideration.

An important detail is that the Kth fraction $v_{i,K}$ is always set to one to ensure the stick segments sum to one. This truncation of the variational posterior does *not* imply that we are using a finite dimensional prior. As explained by Blei and Jordan [2006], the truncation level is a variational parameter and not part of the prior model specification. Truncation-free posteriors have been proposed, but these methods use split-and-merge steps [Hughes et al., 2015] or collapsed Gibbs sampling, both of which are not applicable to the models I consider. Nonetheless, because SGVB imposes few limitations on the inference model, it is possible to have an untruncated posterior. I conducted exploratory experiments using a truncation-free posterior by adding extra variational parameters in an on-line fashion, initializing new weights if more than 1% of the stick remained unbroken. However, I found this made optimization slower without any increase in performance.

7.3.3 Semi-Supervised Model

I also propose an analogous approach for the semi-supervised relative of the VAE, the M2 model described by Kingma et al. [2014]. A second latent variable y_i is introduced that represents a class label. Its distribution is the categorical one: $q_{\boldsymbol{\phi}}(y_i|\mathbf{x}_i) = \text{Cat}(y|g_y(\mathbf{x}_i))$ where g_y is a non-linear function of the inference network. Although y's distribution is written as independent of $\mathbf{z}$, the two share parameters within the inference network and thus act to regularize one another. I assume the same factorization of the posterior and use the same objectives as in the finite dimensional version [Kingma et al., 2014]. Since y_i is present for some but not all observations,

semi-supervised DGMs need to be trained with different objectives depending on whether the label is present or not. If the label is present, following Kingma et al. [2014] we optimize

$$\tilde{\mathcal{J}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i, y_i) = \frac{1}{S} \sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i | \boldsymbol{\pi}_{i,s}, y_i) - KL(q_{\boldsymbol{\phi}}(\boldsymbol{\pi}_i | \mathbf{x}_i) || p(\boldsymbol{\pi}_i; \boldsymbol{\alpha}_0)) + \log q_{\boldsymbol{\phi}}(y_i | \mathbf{x}_i) \quad (7.8)$$

where $\log q_{\boldsymbol{\phi}}(y_i | \mathbf{x}_i)$ is the log-likelihood of the label. And if the label is missing, we optimize

$$\begin{aligned} \tilde{\mathcal{J}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i) = \frac{1}{S} \sum_{s=1}^{S} \sum_{y_j} q_{\boldsymbol{\phi}}(y_j | \mathbf{x}_i) \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i | \boldsymbol{\pi}_{i,s}, y_j) \right] + \mathbb{H}[q_{\boldsymbol{\phi}}(y | \mathbf{x}_i)] \\ - KL(q_{\boldsymbol{\phi}}(\boldsymbol{\pi}_i | \mathbf{x}_i) || p(\boldsymbol{\pi}_i; \boldsymbol{\alpha}_0)) \end{aligned} \quad (7.9)$$

where $\mathbb{H}[q_{\boldsymbol{\phi}}(y_i | \mathbf{x}_i)]$ is the entropy of y's variational distribution.

7.3.4 Related Work

To the best of my knowledge, neither SGVB nor any of the other recently proposed amortized VI methods [Kingma and Welling, 2014a, Rezende and Mohamed, 2015, Rezende et al., 2014, Tran et al., 2016] have been used in conjunction with BNP priors. There has been work on using nonparametric posterior approximations—in particular, the *Variational Gaussian Process* [Tran et al., 2016]—but in that work the variational distribution is nonparametric, not the generative model. Moreover, I am not aware of prior work that uses SGVB for Beta (or Beta-like) random variables.

In regard to the autoencoder implementations I describe, they are closely related to the existing work on representation learning with adaptive latent factors—i.e. where the number of latent dimensions grows as the data necessitates. The best known model of this kind is the infinite binary latent feature model defined by the Indian Buffet Process [Ghahramani and Griffiths, 2005]; but its discrete latent variables prevent this model from admitting fully differentiable inference. Recent work that is much closer in spirit is the *Infinite Restricted Boltzmann Machine*

(iRBM) [Côté and Larochelle, 2016], which has gradient-based learning, expands its capacity by adding hidden units, and induces a similar ordering on latent factors. The most significant difference between the SB-VAE and the iRBM is that the latter's nonparametric behavior arises from a particular definition of the energy function of the Gibbs distribution, not from an infinite dimensional Bayesian prior. Lastly, my training procedure bears some semblance to *Nested Dropout* [Rippel et al., 2014], which removes all hidden units at an index lower than some threshold index. The SB-VAE can be seen as performing soft nested dropout since the latent variable values decrease as their index increases.

7.3.5 Experiments

I analyze the behavior of the three parametrizations of the SB-VAE and examine how they compare to the Gaussian VAE. I do this by examining their ability to reconstruct the data (i.e. density estimation) and to preserve class structure. Following the original DGM papers [Kingma et al., 2014, Kingma and Welling, 2014b, Rezende et al., 2014], I performed unsupervised and semi-supervised tasks on the following image datasets: Frey Faces[2], MNIST, MNIST+rot, and Street View House Numbers[3] (SVHN). MNIST+rot is a dataset I created by combining MNIST and rotated MNIST[4] for the purpose of testing the latent representation under the conjecture that the rotated digits should use more latent variables than the non-rotated ones.

Complete implementation and optimization details can be found in the code repository[5]. In all experiments, to best isolate the effects of Gaussian versus stick-breaking latent variables, the same architecture and optimization hyperparameters were used for each model. The only difference was in the prior: $p(\mathbf{z}) = \mathsf{N}(\mathbf{0}, \mathbb{1})$ for Gaussian latent variables and $p(v) = \mathsf{Beta}(1, \alpha_0)$ (Dirichlet process) for stick-breaking latent variables. I cross-validated the concentration pa-

[2]Available at `http://www.cs.nyu.edu/~roweis/data.html`
[3]Available at `http://ufldl.stanford.edu/housenumbers/`
[4]Available at `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations`
[5]Theano implementations available at `https://github.com/enalisnick/stick-breaking_dgms`

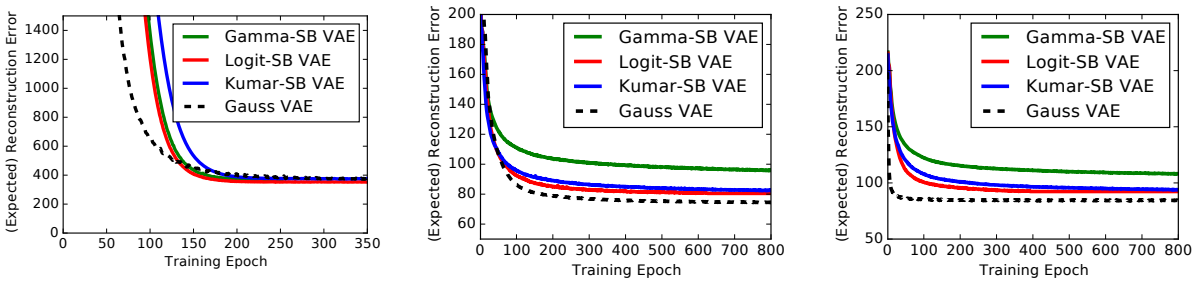
rameter over the range $\alpha_0 \in \{1, 3, 5, 8\}$. The Gaussian model's performance potentially could have been improved by cross validating its prior variance. However, the standard Normal prior is widely used as a default choice [Bowman et al., 2016, Gregor et al., 2015, Kingma et al., 2014, Kingma and Welling, 2014b, Rezende et al., 2014, Salimans et al., 2015], and my goal is to experimentally demonstrate that a stick-breaking prior is a competitive alternative.

7.3.5.1 Unsupervised

I first performed unsupervised experiments testing each model's ability to recreate the data as well as preserve class structure (without having access to labels). The inference and generative models both contained one hidden layer of 200 units for Frey Faces and 500 units for MNIST and MNIST+rot. For Frey Faces, the Gauss VAE had a 25 dimensional (factorized) distribution, and I set the truncation level of the SB-VAE also to $K = 25$, so the SB-VAE could use only as many latent variables as the Gauss VAE. For the MNIST datasets, the latent dimensionality/truncation-level was set at 50. Cross-validation chose $\alpha_0 = 1$ for Frey Faces and $\alpha_0 = 5$ for both MNISTs.

Density Estimation. In order to show each model's optimization progress, Figure 7.2 (a), (b), and (c) report test expected reconstruction error (i.e. the first term in the ELBO) vs training progress (epochs) for Frey Faces, MNIST, and MNIST+rot respectively. Optimization proceeds much the same in both models except that the SB-VAE learns at a slightly slower pace for all parametrizations. This is not too surprising since the recursive definition of the latent variables likely causes coupled gradients.

I compare the final converged models in Table 7.1, reporting the marginal likelihood of each model via the MC approximation $\log p(\mathbf{x}_i) \approx \log \frac{1}{S} \sum_s p(\mathbf{x}_i|\hat{\mathbf{z}}_{i,s}) p(\hat{\mathbf{z}}_{i,s}) / q(\hat{\mathbf{z}}_{i,s})$ using 100 samples. The Gaussian VAE has a better likelihood than all stick-breaking implementations (~ 96 vs ~ 98). Between the stick-breaking parametrizations, the Kumaraswamy outperforms both the Gamma and Gauss-Logit on both datasets, which is not surprising given the others' flaws.



(a) Frey Faces (b) MNIST (c) MNIST+rot

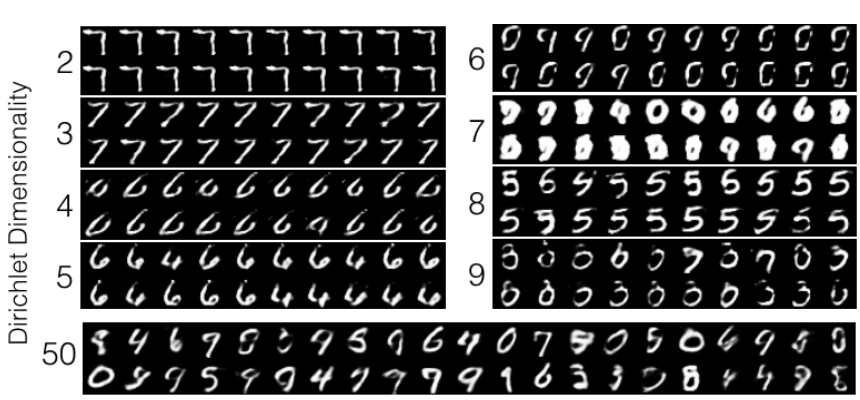
Figure 7.2: Subfigure (a) shows test (expected) reconstruction error vs training epoch for the SB-VAE and Gauss VAE on the Frey Faces dataset, subfigure (b) shows the same quantities for the same models on the MNIST dataset, and subfigure (c) shows the same quantities for the same models on the MNIST+rot dataset.

	$-\log p(\mathbf{x}_i)$	
Model	**MNIST**	**MNIST+rot**
Gauss VAE	96.80	108.40
Kumar-SB VAE	98.01	112.33
Logit-SB VAE	99.48	114.09
Gamma-SB VAE	100.74	113.22

Table 7.1: Marginal likelihood results (estimated) for Gaussian VAE and the three parametrizations of the Stick-Breaking VAE.

Given this result, I used the Kumaraswamy parametrization for all subsequently reported experiments. Note that the likelihoods reported are worse than the ones reported by Burda et al. [2016] because my training set consisted of 50k examples whereas theirs contained 60k (training and validation).

I also investigated whether the SB-VAE is using its adaptive capacity in the manner I expect, i.e., the SB-VAE should use a larger latent dimensionality for the rotated images in MNIST+rot than it does for the non-rotated ones. I examined if this is the case by tracking how many 'breaks' it took the model to deconstruct 99% of the stick. On average, the rotated images in the training set were represented by 28.7 dimensions and the non-rotated by 27.4. Furthermore, the rotated images used more latent variables in eight out of ten classes. Although the difference is not as large as I was expecting, it is statistically significant. Moreover, the difference is made smaller



(a) SB-VAE: MNIST Samples Drawn from Prior

(b) Gauss VAE: MNIST Samples Drawn from Prior

Figure 7.3: Subfigure (a) depicts samples from the SB-VAE trained on MNIST. I show the ordered, factored nature of the latent variables by sampling from Dirichlet's of increasing dimensionality. Subfigure (b) depicts samples from the Gauss VAE trained on MNIST.

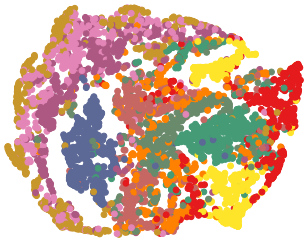
by the non-rotated one digits, which use 32 dimensions on average, the most for any class. The non-rotated average decreases to 26.3 when ones are excluded.

Figure 7.3 (a) shows MNIST digits drawn from the SB-VAE by sampling from the prior—i.e. $v_k \sim \text{Beta}(1, 5)$, and Figure 7.3 (b) shows Gauss VAE samples for comparison. SB-VAE samples using all fifty dimensions of the truncated posterior are shown in the bottom block. Samples from Dirichlets constrained to a subset of the dimensions are shown in the two columns in order to test that the latent features are concentrating onto lower-dimensional simplices. This is indeed the case: adding a latent variable results in markedly different but still coherent samples. For instance, the second and third dimensions seem to capture the 7-class, the fourth and fifth the 6-class, and the eighth the 5-class. The seventh dimension seems to model notably thick digits.

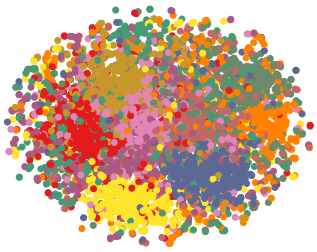
Discriminative Qualities. The discriminative qualities of the models' latent spaces are assessed by running a k-Nearest Neighbors classifier on (sampled) MNIST latent variables. Results are shown in the table in Figure 7.4 (a). The SB-VAE exhibits conspicuously better performance than the Gauss VAE at all choices of k, which suggests that although the Gauss VAE converges to a better likelihood, the SB-VAE's latent space better captures class structure. I also report results for two Gaussian *mixture* VAEs: Dilokthanakul et al. [2016]'s *Gaussian mixture Variational Au-*

	k=3	k=5	k=10
SB-VAE	9.34	8.65	8.90
DLGMM	9.14	8.38	8.42
Gauss VAE	28.4	20.96	15.33
Raw Pixels	2.95	3.12	3.35
GMVAE[6]	—	8.96	—

(a) MNIST: Test error for kNN on latent space



(b) MNIST SB-VAE



(c) MNIST Gauss VAE

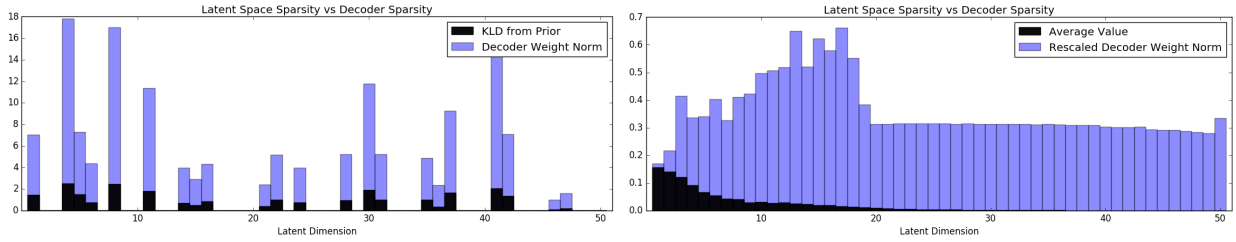
Figure 7.4: Subfigure (a) shows results of a kNN classifier trained on the latent representations produced by each model. Subfigures (b) and (c) show t-SNE projections of the latent representations learned by the SB-VAE and Gauss VAE respectively.

toencoder (GMVAE) and Nalisnick et al. [2016]'s *Deep Latent Gaussian Mixture Model* (DLGMM). The GMVAE[6] has sixteen mixture components and the DLGMM has five, and hence both have many more parameters than the SB-VAE. Despite the SB-VAE's lower capacity, we see that its performance is competitive to the mixture VAEs' (8.65 vs $8.38/8.96$).

The discriminative qualities of the SB-VAE's latent space are further supported by Figures 7.4 (b) and (c). t-SNE was used to embed the Gaussian (c) and stick-breaking (b) latent MNIST representations into two dimensions. Digit classes (denoted by color) in the stick-breaking latent space are clustered with noticeably more cohesion and separation.

Combating Decoder Pruning. The 'component collapsing' behavior of the variational autoencoder has been well noted [Maaløe et al., 2016]: the model will set to zero the outgoing weights of latent variables that remain near the prior. Figure 7.5 (a) depicts this phenomenon for the Gauss VAE by plotting the KL divergence from the prior and outgoing decoder weight norm for each latent dimension. We see the weights are only nonzero in the dimensions in which there is posterior deviation. Ostensibly the model receives only sampling noise from the dimensions that remain at the prior, and setting the decoder weights to zero quells this variance. While the behavior of the Gauss VAE is not necessarily improper, all examples are restricted to pass through

[6]The GMVAE's evaluation is different from performing kNN. Rather, test images are assigned to clusters and whole clusters are given a label. Thus results are not strictly comparable but the ultimate goal of unsupervised MNIST classification is the same.



(a) Gauss VAE (b) Stick-Breaking VAE

Figure 7.5: Sparsity in the latent representation vs in the decoder network. The Gaussian VAE 'turns off' unused latent dimensions by setting the outgoing weights to zero (in order to dispel the sampled noise). The SB VAE, on the other hand, also has sparse representations but without decay of the associated decoder weights.

the same latent variables. A sparse-coded representation—one having few active components per example (like the Gauss VAE) but diversity of activations across examples–would likely be better.

I compare the activation patterns against the sparsity of the decoder for the SB-VAE in Figure 7.5 (b). Since KL-divergence doesn't directly correspond to sparsity in stick-breaking latent variables like it does for Gaussian ones, the black lines denote the average activation value per dimension. Similarly to (a), blue lines denoted the decoder weight norms, but they had to be down-scaled by a factor of 100 so they could be visualized on the same plot. The SB-VAE does not seem to have any component collapsing, which is not too surprising since the model can set latent variables to zero to deactivate decoder weights without being in the heart of the prior. I conjecture that this increased capacity is the reason stick-breaking variables demonstrate better discriminative performance in many of my experiments.

7.3.5.2 Semi-Supervised

I also performed semi-supervised classification, replicating and extending the experiments in the original semi-supervised DGMs paper [Kingma et al., 2014]. I used the MNIST, MNIST+rot, and SVHN datasets and reduced the number of labeled training examples to 10%, 5%, and 1% of the total training set size. Labels were removed completely at random and as a result, class imbalance

	MNIST (N=45,000)			**MNIST+rot** (N=70,000)			**SVHN** (N=65,000)		
	10%	5%	1%	10%	5%	1%	10%	5%	1%
SB-DGM	$4.86_{\pm.14}$	$5.29_{\pm.39}$	$\mathbf{7.34}_{\pm.47}$	$\mathbf{11.78}_{\pm.39}$	$\mathbf{14.27}_{\pm.58}$	$\mathbf{27.67}_{\pm1.39}$	$\mathbf{32.08}_{\pm4.00}$	$\mathbf{37.07}_{\pm5.22}$	$\mathbf{61.37}_{\pm3.60}$
Gauss-DGM	$\mathbf{3.95}_{\pm.15}$	$\mathbf{4.74}_{\pm.43}$	$11.55_{\pm2.28}$	$21.78_{\pm.73}$	$27.72_{\pm.69}$	$38.13_{\pm.95}$	$36.08_{\pm1.49}$	$48.75_{\pm1.47}$	$69.58_{\pm1.64}$
kNN	$6.13_{\pm.13}$	$7.66_{\pm.10}$	$15.27_{\pm.76}$	$18.41_{\pm.01}$	$23.43_{\pm.01}$	$37.98_{\pm.01}$	$64.81_{\pm.34}$	$68.94_{\pm.47}$	$76.64_{\pm.54}$

Table 7.2: Percent error on three semi-supervised classification tasks with 10%, 5%, and 1% of labels present for training. The DGM with stick-breaking latent variables (SB-DGM) is compared with a DGM with Gaussian latent variables (Gauss-DGM), and a k-Nearest Neighbors classifier (k=5).

was all but certainly introduced. Similarly to the unsupervised setting, I compared DGMs with stick-breaking (SB-DGM) and Gaussian (Gauss-DGM) latent variables against one another and a baseline k-Nearest Neighbors classifier (k=5). I used 50 for the latent variable dimensionality / truncation level. The MNIST networks use one hidden layer of 500 hidden units. The MNIST+rot and SVHN networks use four hidden layers of 500 units in each. The last three hidden layers have identity function skip-connections. Cross-validation chose $\alpha_0 = 5$ for MNISTs and $\alpha_0 = 8$ for SVHN.

Quantitative Evaluation. Table 7.2 shows percent error on a test set when training with the specified percentage of labeled examples. We see the the SB-DGM performs markedly better across almost all experiments. The Gauss-DGM achieves a superior error rate only on the easiest tasks: MNIST with 10% and 5% of the data labeled.

7.4 Latent Dirichlet Process Mixtures

I now describe a novel modification of the DLGM/VAE in which I use a Gaussian mixture model (GMM) as the prior and approximate posterior. I modify the generative process to be $\boldsymbol{\pi}_i \sim \text{Dir}(\boldsymbol{\alpha})$, $\mathbf{z}_i \sim \sum_{k=1}^{K} \pi_{i,k} N(\mathbf{z}; \boldsymbol{\theta}_k)$, $\mathbf{x}_i \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_i)$ where $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_i)$ is the density network. I assume

the posterior factorizes as

$$q(\boldsymbol{\pi}, \mathbf{z}|\mathbf{x}_i) = q(\boldsymbol{\pi}|\mathbf{x}_i)q(\mathbf{z}|\boldsymbol{\pi}_i, \mathbf{x}_i) = \prod_{K-1} \text{Kumar}(a_{i,k}, b_{i,k}) \sum_{k=1}^{K} \pi_{i,k} N_{\boldsymbol{\theta}_k}(z|\mathbf{x}_i)$$

where Kumar(a,b) denotes the Kumaraswamy distribution [Jones, 2009]. Notice that we bypass the complication of sampling valid mixture weights $\boldsymbol{\pi}_i$ by, firstly, using the Dirichlet's marginal (aka 'stick-breaking') construction and, secondly, employing the Kumaraswamy as the approximate posterior for the Dirichlet's marginal Betas. The Kumaraswamy has a closed-form inverse CDF that can serve as a valid *differentiable non-centered parametrization* (DNCP) [Nalisnick and Smyth, 2017c] whereas the Beta has no such DNCP. Having defined the prior and posterior, we now can write the SGVB evidence lowerbound (ELBO) for this model as:

$$\begin{aligned} \mathcal{L}_{\text{SGVB}} = \sum_k \mu_{\pi_k} [\frac{1}{S} \sum_s \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,k,s}) + \mathbb{E}_{q_k}[\log p(\mathbf{z}_i)]] \\ - \text{KLD}[q(\boldsymbol{\pi}_k|\mathbf{x}_i)||p(\boldsymbol{\pi}_k)] - \frac{1}{S} \sum_s \log \sum_k \hat{\pi}_{i,k,s} q(\hat{\mathbf{z}}_{i,k,s}; \boldsymbol{\phi}_k) \end{aligned} \tag{7.10}$$

where $\hat{\pi}$ and $\hat{z}$ are S samples taken via non-centered parametrizations and μ_{π_k} is the mean of the posterior weight distribution. This model has the benefit of relatively straightforward DNCPs but has the drawback of needing to run the density network ('decoder') K times, where K is the number of components, for each forward pass. This expensive marginalization is required because of the difficulty in sampling from the mixture directly, i.e. $\mathbf{z} \sim \sum_k \pi_k q_k(\mathbf{z})$[7].

The computation path of the the proposed DLGMM is summarized in Figure 7.6. The inference network computes the parameters of the K mixture components, and the density network is run for a sample from each. The mixture weight, once sampled, is used nowhere in the computation path to reconstruct the data. Rather its influence is in the ELBO, weighting each term according to the corresponding component. Equation 7.10 can be extended to multiple stochastic layers,

[7] Alex Graves' note *Stochastic Backpropagation through Mixture Density Distributions* describes a technique for calculating gradients though samples from a mixture model, but I found the method requires many samples (100+) of the latent variables and did not result in models with competitive marginal likelihoods.

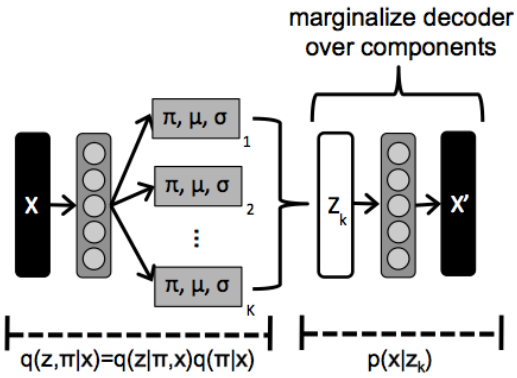


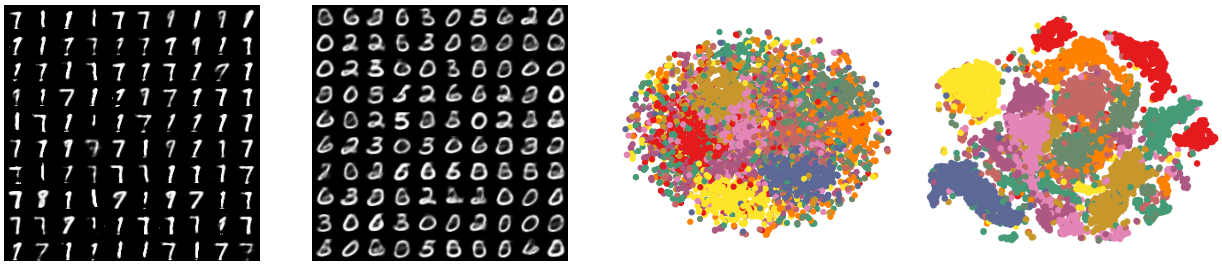
Figure 7.6: Computation graph of a *Deep Latent Gaussian Mixture Model* (DLGMM). The inference network computes the parameters of K mixture components. The decoder network receives a sample from each and computes the reconstruction. The recursive process by which the mixture weights π_k are generated is omitted.

but the density network must be run K^s times, where K is the number of components and s the number of stochastic layers, for each forward pass.

As we are already using the Dirichlet's stick-breaking construction, it is easy to extend the model to infinite mixtures defined by the Dirichlet Process (assuming posterior truncation), i.e. $G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\zeta_k}$ where δ_{ζ_k} is a discrete measure concentrated at $\zeta_k \sim G_0$ and the π_ks are, again, random weights chosen independent of G_0 such that $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$. The only significant change is the prior on the Beta marginals. For all k, we have $v_{i,k} \sim \text{Beta}(1, \alpha_0)$ where α_0 is the concentration parameter. I assume the variational posterior takes the same form as above and is truncated to T components, as is usually done when performing variational inference for DP mixtures [Blei and Jordan, 2006].

7.4.1 Experiments

I compared the proposed *deep latent Gaussian mixture model* (DLGMM) and *deep latent Dirichlet Process mixture model* (DLDPMM) to the single-Gaussian VAE/DLGM (Gauss-VAE) [Kingma and Welling, 2014b, Rezende et al., 2014] and the *stick-breaking* VAE (SB-VAE) Nalisnick and Smyth [2017c] on the binarized MNIST dataset and Omniglot [Lake et al., 2015], using the pre-defined train/valid/test splits. I optimized all models using AdaM [Kingma and Ba, 2014] with a learning rate of 0.0003 (other parameters kept at their Tensorflow defaults), batch sizes of 100, and early stopping with 30 look-ahead epochs. For the marginal likelihood results, all Gaussian priors are



(a) $\mu = -1.5$ (b) $\mu = 1.5$ (c) Single Gaussian (d) Gaussian Mix. (K=5)

Figure 7.7: Subfigures (a) and (b) show samples from the two mixture components at the extremes of the latent space. Subfigures (c) and (d) show t-SNE embeddings of the Gauss-VAE and DLGMM latent space (respectively).

standard Normals and all Dirichlets are symmetric, with $\alpha = 1$, except for the DLDPMM, which has $\alpha_0 = 1$.

Qualitative evaluation. First I compared the models qualitatively by examining samples and class distribution within the latent space. Samples from two components of a 5-component DLGMM are shown in Subfigures (a) and (b) of Figure 7.7. Normal priors were placed on the five components with means set to $\mu = \{-1.5, -.75, 0, .75, 1.5\}$ and all variances set to one. The samples are from the extremes of the prior, i.e. $\mu_1 = -1.5$ and $\mu_5 = 1.5$. We see that the DLGMM learned not only recognizable MNIST digits but also to divide their factors of variation into different parts of the latent space. Thin digits such as sevens are generated from the component with the most negative prior mean and wide digits such as zeros are generated from the component with the most positive prior mean. Also I visualized the MNIST class distribution in the latent space via t-SNE projection. The 2D embeddings are shown in Figures 7.7 (c) and (d) for the Gauss-VAE and DLGMM respectively; colors denote digit classes. The DLGMM's latent space exhibits conspicuously better clustering. I validate this observation empirically below using kNN.

Quantitative evaluation. I compared the models quantitatively using a k-Nearest Neighbors (kNN) classifier on their latent space as well as by calculating the marginal likelihood of a held-out set. Table (a) of Figure 7.8 reports MNIST test error for kNN classifiers trained on the latent

	k=3	k=5	k=10
DLGMM	**9.14**	**8.38**	**8.42**
SB-VAE	9.34	8.65	8.90
Gauss-VAE	28.4	20.96	15.33

(a) MNIST test error for kNN on latent space

	$-\log p_{\boldsymbol{\theta}}(\mathbf{x}_i)$	
	MNIST	OMNIGLOT
DLGMM (500d-3x25s)	**96.50**	123.50
DLDPMM (500d-17tx25s)	96.91	123.76
Gauss-VAE (500d-25s)	96.80	**119.18**
SB-VAE (500d-25t)	98.01	–

(b) Estimated Marginal Likelihood

Figure 7.8: Subtable (a) shows MNIST test error for kNN classifiers trained on samples from the latent distributions. Results for 3, 5, and 10 (k) neighbors are given. Each model was trained with no label supervision. Subfigure (b) reports the (Monte Carlo) estimated marginal likelihood on the test set.

space of the Gauss-VAE, a SB-VAE, and the proposed DLGMM. Note that none of these models had access to labels during training. We see from the table that the DLGMM performs markedly better than the Gauss-VAE—supporting our visual analysis above of the t-SNE projections—and slightly better than the SB-VAE. Moreover, the DLGMM's superior performance holds across all number of neighbors tested ($k = \{3, 5, 10\}$).

Lastly, in Figure 7.8 (b) I report the (Monte Carlo) estimated marginal likelihood for the various models on MNIST and Omniglot. The network architectures are given in parentheses: *d* denotes a deterministic layer, *s* a stochastic layer, $K\times$ the number of mixture components, and *t* the truncation level for the DP and SB models. I find that using a mixture latent space improves the likelihood modestly for MNIST but not at all for Omniglot.

7.5 Conclusions

I have described how to employ the Kumaraswamy distribution to extend stochastic gradient variational Bayes to the weights of stick-breaking Bayesian nonparametric priors. Using this development I then defined deep generative models with infinite dimensional latent variables and showed that their latent representations are more discriminative than those of the popular Gaus-

sian variant. Moreover, the only extra computational cost is in assembling the stick segments, a linear operation on the order of the truncation size. Not only are the ideas herein immediately useful as presented, they are an important first-step to integrating black box variational inference and Bayesian nonparametrics, resulting in scalable models that have differentiable control of their capacity. Furthermore, differentiable stick-breaking has the potential to increase the dynamism and adaptivity of neural networks, a subject of recent interest [Graves, 2016], in a probabilistically principled way.

Chapter 8

Open Problems and Conclusions

That was it. That was really it. She knew that she had told herself that that was it only seconds earlier, but this was now the final real ultimate it.

Douglas Adams

Dirk Gently's Holistic Detective Agency

This dissertation has presented the fundamentals of Bayesian neural networks (Chapter 2), a survey of the various priors used for the model (Chapter 3), and original work on Gaussian scale mixture (Chapter 4), objective (Chapter 5), invariant (Chapter 6), and stick-breaking priors (Chapter 7). I now part with some discussion of open problems and concluding thoughts.

8.1 Open Problems

While advancements in technology continually lighten the computational burden of training and deploying neural networks, there exist some core challenges that cannot be readily solved

by throwing more computation at the problem. I list some of them below, focusing in particular on the interplay between optimization-based inference and latent structures without derivatives.

1. **Inference Bottlenecks**: As mentioned previously, gradient-based methods currently are the only viable optimization procedure for deep neural networks. This constraint then necessarily follows to posterior inference, forcing variational methods to also be gradient-based. This is not a problem for priors with continuous support, but many interesting distributions—discrete, nonparameteric, structure-based—do not readily admit differentiation and so considerable work needs to be done on gradient approximations. As automatic differentiation and advances in probabilistic programming have allowed modelers to bypass deriving bespoke inference algorithms, crafting gradient approximations has become the new-found bottleneck in the model building pipeline. *Variational optimization* strategies [Salimans et al., 2017] are perhaps the most intriguing work residing at the research frontier. These methods can approximate a function's gradient via perturbation, and while a single perturbation is only marginally informative, the method can be parallelized to get reliable estimates. Whether or not these methods are viable for a large class of non-differentiable functions has yet to be determined. Fortunately, with the increased interest in reinforcement learning, many people are now thinking hard about optimizing without analytic derivatives.

2. **Bayesian Reasoning Over Structures**: Experiments have shown NNs to be sensitive to the network architecture, and peak performance on tasks such as ImageNet has been achieved only by careful specification of the filter sizes, residual blocks, pooling types, etc. The Bayesian priors described can perform some reasoning over structure; for instance, an automatic relevance determination (ARD) prior essentially prunes nodes. Moreover, stick-breaking priors allow for network width to be adaptive. However, these priors over structure are relatively limited, and it would be much more valuable if the Bayesian crank could be applied to LSTM unit configurations, for instance. Work is ongoing in this direc-

tion [Fortunato et al., 2017] but the gradient bottleneck discussed above still presents a strong headwind against progress.

3. **Hybrid Models:** Given the obstacles in non-differentiable inference and reasoning over structures, researchers have shifted to hybrid models that support mixed inference strategies—that is, composing neural networks with more traditional graphical models. Johnson et al. [2016] made perhaps the first notable contribution to this research direction, but their work is limited to conjugate updates for the latent graphical model. Subsequent work by Lin et al. [2018] has already extended the method to non-conjugate updates. A related trend is composing models with high-capacity invertible transformations [Dinh et al., 2017]. Switching out non-invertible neural networks for the invertible transformations introduced by Dinh et al. [2017] can (with some thought) make inference exact on the latent graphical structure. Gradient updates can then be run on the marginal likelihood to update the transformations. Korshunova et al. [2018] use this idea to define an expressive recurrent model for exchangeable data, with the exchangeability being guaranteed by a student-t process (with closed-form inference).

4. **Neural Nonparametrics**: As researchers with Bayesian leanings have resorted to hybrid models and mixed inference to incorporate structured stochastic variables, deep learning researchers have formulated structured but still differentiable neural components, many having nonparametric qualities. The foremost example is differentiable memory architectures [Graves et al., 2016, Santoro et al., 2016, Sukhbaatar et al., 2015] that use a soft (i.e. differentiable) read/write head to update and extract vectors from an external memory unit. Architectures of this variety have found success in tasks with limited labeled data [Santoro et al., 2016, Vinyals et al., 2016, Bartunov and Vetrov, 2018]. Similar ideas were used by Graves [2016] to define an RNN that can self-determine its number of recursions per time step, essentially giving it an unbounded depth. Linking these 'neural nonparametric' models with traditional Bayesian nonparametrics is an interesting research direction

that might suggest *(i)* principled training algorithms for the neural models, *(ii)* neural-network-based inference algorithms for traditional Bayesian nonparametric priors, and *(iii)* novel neural network architectures based on more exotic nonparametric priors, such as the *Beta-GOS process* [Airoldi et al., 2014].

8.2 Conclusions

Deep neural networks have triggered an influx of hope and hype to machine learning, and it is important to keep grounded in the *George-Box-ian* view of the model building process [Box and Tiao, 1973, Blei, 2014] (a.k.a. *Box's loop*), I believe. For Bayesian models, the first step in that loop is to specify knowledge via the prior. Neural networks make this specification hard, as I have discussed, but that is no reason to give up on doing so. Work analyzing the relationship between optimization, priors, and approximate posteriors (e.g. [Hoffman and Johnson, 2016, Chen et al., 2017, Tomczak and Welling, 2018]) has undoubtedly led us to a better understanding of deep generative models and improved their empirical performance, but I fear that deriving priors solely by reverse engineering the ELBO can lead us away from principled model building. However, I recognize the link between priors and inference is fundamental, and the most efficacious contributions will lie at this intersection, as I discuss in the open problems above. If I must emphasize one point from this dissertation, it is: use the recent advances in variational inference and optimization to help specify priors. When working with Bayesian neural networks, what really matters is the induced distribution on the model's output, and hence we should optimize this quantity directly (if only approximately). My work on reference (Chapter 5) and invariant (Chapter 6) priors serves as humble but illustrative examples of doing just that.

Bibliography

Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring. In *Proceedings of the 29th International Conference on International Conference on Machine Learning (ICML)*, pages 1771–1778, 2012.

Edoardo M Airoldi, Thiago Costa, Federico Bassetti, Fabrizio Leisen, and Michele Guindani. Generalized Species Sampling Priors with Latent Beta Reinforcements. *Journal of the American Statistical Association*, 109(508):1466–1480, 2014.

Yasuo Amemiya. On Nonlinear Factor Analysis. In *Proceedings of the ASA Social Statistics Section*, pages 290–294, 1993.

David F Andrews and Colin L Mallows. Scale Mixtures of Normal Distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 99–102, 1974.

Jimmy Ba and Rich Caruana. Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems (NIPS)*, pages 2654–2662, 2014.

Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with Pseudo-Ensembles. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3365–3373, 2014.

Henry S Baird. Document Image Defect Models. In *Structured Document Image Analysis*, pages 546–556. Springer, 1992.

Carlo Baldassi, Alfredo Braunstein, Nicolas Brunel, and Riccardo Zecchina. Efficient Supervised Learning in Networks with Binary Synapses. *BMC Neuroscience*, 8(2):S13, 2007.

Pierre Baldi and Kurt Hornik. Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima. *Neural Networks*, 2(1):53–58, 1989.

Pierre Baldi and Peter J Sadowski. Understanding Dropout. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2814–2822, 2013.

David J Bartholomew. *Latent Variable Models and Factors Analysis*. Oxford University Press, Inc., 1987.

Sergey Bartunov and Dmitry Vetrov. Few-Shot Generative Modeling with Generative Matching Networks. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84, pages 670–678, 2018.

Thomas Bayes. An Essay Towards Solving a Problem in the Doctrine of Chances. *Philosophical Transactions (1683-1775)*, pages 370–418, 1763.

EML Beale and CL Mallows. Scale Mixing of Symmetric Distributions with Zero Means. *The Annals of Mathematical Statistics*, 30(4):1145–1151, 1959.

Jan Beirlant, Edward J Dudewicz, László Györfi, and Edward C Van der Meulen. Nonparametric Entropy Estimation: An Overview. *International Journal of Mathematical and Statistical Sciences*, 6(1):17–39, 1997.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013a.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients through Stochastic Neurons for Conditional Computation. *ArXiv e-prints*, 2013b.

Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized Denoising Auto-Encoders as Generative Models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 899–907, 2013c.

James O Berger, José M Bernardo, Dongchu Sun, et al. The Formal Definition of Reference Priors. *The Annals of Statistics*, 37(2):905–938, 2009.

James O Berger, Jose M Bernardo, and Dongchu Sun. Overall Objective Priors. *Bayesian Analysis*, 10(1):189–221, 2015.

Jose M Bernardo. Reference Posterior Distributions for Bayesian Inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 113–147, 1979.

José M Bernardo. Reference Analysis. *Handbook of Statistics*, 25:17–90, 2005.

Michael Betancourt. The Fundamental Incompatibility of Scalable Hamiltonian Monte Carlo and Naive Data Subsampling. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 533–540, 2015.

David M Blei. Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models. *Annual Review of Statistics and Its Application*, 1:203–232, 2014.

David M Blei and Michael I Jordan. Variational Inference for Dirichlet Process Mixtures. *Bayesian Analysis*, pages 121–143, 2006.

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

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *Proceedings of The 32nd International Conference on Machine Learning (ICML)*, pages 1613–1622, 2015.

Guillaume Bouchard and Balaji Lakshminarayanan. Approximate Inference with the Variational Holder Bound. *ArXiv e-prints*, 2015.

Hervé Bourlard and Yves Kamp. Auto-Association by Multilayer Perceptrons and Singular Value Decomposition. *Biological Cybernetics*, 59(4-5):291–294, 1988.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *CoNLL*, 2016.

George EP Box and George C Tiao. *Bayesian Inference in Statistical Analysis*, volume 40. John Wiley & Sons, 1973.

Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

Wray L Buntine and Andreas S Weigend. Bayesian Back-Propagation. *Complex Systems*, 5(6): 603–643, 1991.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders. *International Conference on Learning Representations (ICLR)*, 2016.

Carlos M Carvalho, Nicholas G Polson, and James G Scott. Handling Sparsity via the Horseshoe. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 73–80, 2009.

G. Casella and R.L. Berger. *Statistical Inference*. Thomson Learning, 2002.

George Casella. An Introduction to Empirical Bayes Data Analysis. *The American Statistician*, 39 (2):83–87, 1985.

Ismaël Castillo, Johannes Schmidt-Hieber, Aad Van der Vaart, et al. Bayesian Linear Regression with Sparse Priors. *The Annals of Statistics*, 43(5):1986–2018, 2015.

Tian Chen, Jeffrey Streets, and Babak Shahbaba. A Geometric View of Posterior Approximation. *ArXiv e-prints*, 2015.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1683–1691, 2014.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational Lossy Autoencoder. *International Conference on Learning Representations (ICLR)*, 2017.

Taco Cohen and Max Welling. Learning the Irreducible Representations of Commutative Lie Groups. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1755–1763, 2014.

Taco Cohen and Max Welling. Group Equivariant Convolutional Networks. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pages 2990–2999, 2016.

Marc-Alexandre Côté and Hugo Larochelle. An Infinite Restricted Boltzmann Machine. *Neural Computation*, 2016.

G.W. Cottrell, P. Munro, and D Zipser. Image Compression by Back Propagation: A Demonstration of Extensional Programming. *Models of Cognition*, pages 208–240, 1989.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with Binary Weights During Propagations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3123–3131, 2015.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

Carl De Boor, Carl De Boor, Etats-Unis Mathématicien, Carl De Boor, and Carl De Boor. *A Practical Guide to Splines*, volume 27. Springer-Verlag New York, 1978.

Nando De Freitas, Pedro Højen-Sørensen, Michael I Jordan, and Stuart Russell. Variational MCMC. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 120–127, 2001.

Alexander G. de G Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian Process Behaviour in Wide Deep Neural Networks. *International Conference on Learning Representations (ICLR)*, 2018.

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 (methodological)*, pages 1–38, 1977.

Li Deng and Dong Yu. Deep learning: Methods and Applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014.

Adji B Dieng, Dustin Tran, Rajesh Ranganath, John Paisley, and David M Blei. The χ-Divergence for Approximate Inference. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Nat Dilokthanakul, Pedro Mediano, Marta Garnelo, Matthew Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *ArXiv e-prints*, 2016.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density Estimation Using Real NVP. *International Conference on Learning Representations (ICLR)*, 2017.

Morris L Eaton. Group Invariance Applications in Statistics. In *Regional Conference Series in Probability and Statistics*, pages i–133. JSTOR, 1989.

Harrison Edwards and Amos Storkey. Towards a Neural Statistician. *International Conference on Learning Representations (ICLR)*, 2017.

Thomas S Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1973.

R. A. Fisher. Review of J. M. Keynesffs Treatise on Probability. *Eugenics Review*, 14:46–50, 1922.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian Recurrent Neural Networks. *ArXiv e-prints*, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016a.

Yarin Gal and Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1019–1027, 2016b.

Edward I George and Robert E McCulloch. Variable Selection via Gibbs Sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

Samuel Gershman and Noah Goodman. Amortized Inference in Probabilistic Reasoning. In *Proceedings of the Cognitive Science Society*, volume 36, 2014.

Zoubin Ghahramani. Should All Machine Learning be Bayesian? Should All Bayesian Models be Non-Parametric?, October 2008. Bayesian Research Kitchen Workshop (BARK), Grasmere, England.

Zoubin Ghahramani and Thomas L Griffiths. Infinite Latent Feature Models and the Indian Buffet Process. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

Soumya Ghosh and Finale Doshi-Velez. Model Selection in Bayesian Neural Networks via Horseshoe Priors. *NIPS Workshop on Bayesian Deep Learning*, 2017.

WA Gibson. Nonlinear Factors in Two Dimensions. *Psychometrika*, 25(4):381–392, 1960.

P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Applications of Mathematics: Stochastic Modeling and Applied Probability. Springer, 2004.

P. Goel and A. Zellner. On Assessing Prior Distributions and Bayesian Regression Analysis with g-Prior Distributions. *Bayesian Inference and Decision Techniques*, pages 233–243, 1986.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Cyril Goutte and Lars Kai Hansen. Regularization with a Pruning Prior. *Neural Networks*, 10(6): 1053–1059, 1997.

Alex Graves. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2348–2356, 2011.

Alex Graves. Adaptive Computation Time for Recurrent Neural Networks. *ArXiv e-prints*, 2016.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, and John Agapiou. Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature*, 538(7626):471, 2016.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. DRAW: A Recurrent Neural Network For Image Generation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Silviu Guiasu and Abe Shenitzer. The Principle of Maximum Entropy. *The Mathematical Intelligencer*, 7(1):42–48, 1985.

W Keith Hastings. Monte Carlo Sampling Methods Using Markov Chains and their Applications. *Biometrika*, 57(1):97–109, 1970.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

DO Hebb. *The Organization of Behavior; a Neuropsychological Theory*. Wiley, 1949.

David P Helmbold and Philip M Long. On the Inductive Bias of Dropout. *Journal of Machine Learning Research*, 16:3403–3454, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *International Conference on Learning Representations (ICLR)*, 2017.

Geoffrey E Hinton. Learning Distributed Representations of Concepts. In *Proceedings of the 8TH Annual Conference of the Cognitive Science Society*, volume 1, page 12. Amherst, MA, 1986.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.

Geoffrey E Hinton and Drew Van Camp. Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights. In *Proceedings of the 6th Annual Conference on Computational Learning Theory*, pages 5–13. ACM, 1993.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. *ArXiv e-prints*, 2012.

Geoffrey E Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *NIPS Workshop on Deep Learning*, 2014.

Arthur E Hoerl and Robert W Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970.

Matthew Hoffman and Matthew Johnson. ELBO Surgery: Yet Another Way to Carve Up the Variational Evidence Lower Bound. *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2016.

Matthew D Hoffman. Learning Deep Latent Gaussian Models with Markov Chain Monte Carlo. In *In Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1510–1519, 2017.

Matthew D. Hoffman, Carlos Riquelme, and Matthew J. Johnson. The β-VAE's Implicit Prior. *NIPS Workshop on Bayesian Deep Learning*, 2017.

John J Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

Jiri Hron, Alexander Matthews, and Zoubin Ghahramani. Two Problems with Variational Gaussian Dropout. *NIPS Workshop on Bayesian Deep Learning*, 2017.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks. In *Advances in Neural Information Processing Systems (NIPS) 29*, pages 4107–4115, 2016.

Michael C Hughes, Dae Il Kim, and Erik B Sudderth. Reliable and Scalable Variational Inference for the Hierarchical Dirichlet Process. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

Telba Z Irony and Nozer D Singpurwalla. Non-Informative Priors Do Not Exist: A Dialogue with Jose M. Bernardo. *Journal of Statistical Planning and Inference*, 65(159):189, 1997.

Hemant Ishwaran and Lancelot F James. Gibbs Sampling Methods for Stick-Breaking Priors. *Journal of the American Statistical Association*, 2001.

Laurent Itti and Pierre Baldi. Bayesian Surprise Attracts Human Attention. In *Advances in Neural Information Processing Systems (NIPS)*, pages 547–554, 2006.

E. Jang, S. Gu, and B. Poole. Categorical Reparameterization with Gumbel-Softmax. *International Conference on Learning Representations (ICLR)*, 2017.

Edwin T Jaynes. Information Theory and Statistical Mechanics. *Physical Review*, 106(4):620, 1957.

Edwin T Jaynes. Prior Probabilities. *IEEE Transactions on Systems Science and Cybernetics*, 4(3): 227–241, 1968.

Harold Jeffreys. An Invariant Form for the Prior Probability in Estimation Problems. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 186, pages 453–461. The Royal Society, 1946.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and SanDeep R Datta. Composing Graphical Models with Neural Networks for Structured Representations and Fast Inference. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2946–2954, 2016.

MC Jones. Kumaraswamyffs Distribution: A Beta-Type Distribution with Some Tractability Advantages. *Statistical Methodology*, 6(1):70–81, 2009.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, 1999.

Ata Kabán. On Bayesian Classification with Laplace Priors. *Pattern Recognition Letters*, 28(10): 1271–1282, 2007.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2014.

Diederik Kingma and Max Welling. Efficient Gradient-Based Inference Through Transformations Between Bayes Nets and Neural Nets. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1782–1790, 2014a.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations (ICLR)*, 2014b.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-Supervised Learning with Deep Generative Models. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

David A Knowles. Stochastic Gradient Variational Bayes for Gamma Approximating Distributions. *ArXiv e-prints*, 2015.

I. Korshunova, J. Degrave, F. Huszár, Y. Gal, A. Gretton, and J. Dambre. A Generative Deep Recurrent Model for Exchangeable Data. *ArXiv e-prints*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

Ponnambalam Kumaraswamy. A Generalized Probability Density Function for Double-Bounded Random Processes. *Journal of Hydrology*, 1980.

Lynn Kuo and Bani Mallick. Variable Selection for Regression Models. *Sankhyā: The Indian Journal of Statistics, Series B*, pages 65–81, 1998.

John Lafferty and Larry Wasserman. Iterative Markov Chain Monte Carlo Computation of Reference Priors and Minimax Risk. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 293–300, 2001.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-Level Concept Learning Through Probabilistic Program Induction. *Science*, 350(6266):1332–1338, 2015.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 473–480, 2007.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Herbert KH Lee. Priors for Neural Networks. In *Classification, Clustering, and Data Mining Applications*, pages 141–150. Springer, 2004.

Herbert KH Lee. Neural Networks and Default Priors. In *Proceedings of the American Statistical Association, Section on Bayesian Statistical Science*, 2005.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep Neural Networks as Gaussian Processes. *International Conference on Learning Representations (ICLR)*, 2018.

Yingzhen Li and Richard E Turner. Variational inference with Renyi divergence. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1073–1081, 2016.

Yuanzhi Li and Andrej Risteski. Approximate Maximum Entropy Principles via Goemans-Williamson with Applications to Provable Variational Methods. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4635–4643, 2016.

M. Lichman. UCI Machine Learning Repository, 2013. URL `http://archive.ics.uci.edu/ml`.

W. Lin, N. Hubacher, and M. Emtiyaz Khan. Variational Message Passing with Structured Inference Networks. *International Conference on Learning Representations (ICLR)*, 2018.

Qiang Liu and Dilin Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2378–2386, 2016.

Ruitao Liu, Arijit Chakrabarti, Tapas Samanta, Jayanta K Ghosh, and Malay Ghosh. On Divergence Measures Leading to Jeffreys and Other Reference Priors. *Bayesian Analysis*, 9(2): 331–370, 2014.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary Deep Generative Models. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 142–150, 2011.

Laurens Maaten, Minmin Chen, Stephen Tyree, and Kilian Q Weinberger. Learning with Marginalized Corrupted Features. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 410–418, 2013.

David JC MacKay. Bayesian Interpolation. *Neural Computation*, 4(3):415–447, 1992a.

David JC MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 4(4):590–604, 1992b.

David JC MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1992c.

David JC MacKay. Bayesian Non-Linear Modeling for the Prediction Competition. In *Maximum Entropy and Bayesian Methods*, pages 221–234. Springer, 1994.

David JC MacKay and Mark N Gibbs. Density Networks. *Statistics and Neural Networks: Advances at the Interface*, pages 129–144, 1999.

C. J. Maddison, A. Mnih, and Y. Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *International Conference on Learning Representations (ICLR)*, 2017.

Alireza Makhzani, Jonathon Shlens, NavDeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders. *ICLR Workshop*, 2016.

Christopher D Manning. Computational Linguistics and Deep Learning. *Computational Linguistics*, 2016.

Eddy Mayoraz and Frédéric Aviolat. Constructive Training Methods for Feedforward Neural Networks with Binary Weights. *International Journal of Neural Systems*, 7(02):149–166, 1996.

Warren S McCulloch and Walter Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biology*, 52(1-2):99–115, 1943.

Roderick P McDonald. A General Approach to Nonlinear Factor Analysis. *Psychometrika*, 27(4): 397–415, 1962.

Thomas P Minka. Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 362–369, 2001.

Toby J Mitchell and John J Beauchamp. Bayesian Variable Selection in Linear Regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.

Andriy Mnih and Karol Gregor. Neural Variational Inference and Learning in Belief Networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1791–1799, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. Human-Level Control Through Deep Reinforcement Rearning. *Nature*, 518(7540):529–533, 2015.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational Dropout Sparsifies Deep Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2498–2507, 2017.

Peter Müller and David Rios Insua. Issues in Bayesian Analysis of Neural Network Models. *Neural Computation*, 10(3):749–770, 1998.

Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

Minoru Nakagami. The M-Distribution: A General Formula of Intensity Distribution of Rapid Fading. In *Statistical Methods in Radio Wave Propagation*, pages 3–36. Elsevier, 1960.

Eric Nalisnick and Padhraic Smyth. Learning Approximately Objective Priors. *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

Eric Nalisnick and Padhraic Smyth. The Amortized Bootstrap. *ICML Workshop on Implicit Models*, 2017a.

Eric Nalisnick and Padhraic Smyth. Learning Approximately Objective Priors. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 33, 2017b.

Eric Nalisnick and Padhraic Smyth. Stick-Breaking Variational Autoencoders. *International Conference on Learning Representations (ICLR)*, 2017c.

Eric Nalisnick and Padhraic Smyth. Learning Priors for Invariance. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 366–375, 2018.

Eric Nalisnick, Anima Anandkumar, and Padhraic Smyth. A Scale Mixture Perspective of Multiplicative Noise in Neural Networks. *ArXiv e-prints*, 2015.

Eric Nalisnick, Lars Hertel, and Padhraic Smyth. Approximate Inference for Deep Latent Gaussian Mixtures. *NIPS Workshop on Bayesian Deep Learning*, 2016.

Radford M Neal. Bayesian Learning via Stochastic Dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, pages 475–482, 1993.

Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1994.

Radford M Neal. MCMC Using Hamiltonian Dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.

Radford M Neal and Geoffrey E Hinton. A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants. In *Learning in Graphical Models*, pages 355–368. Springer, 1998.

John Ashworth Nelder and R Jacob Baker. *Generalized Linear Models*. Wiley Online Library, 1972.

Veronica Nieves, Jingfeng Wang, Rafael L Bras, and Elizabeth Wood. Maximum Entropy Distributions of Scale-Invariant Processes. *Physical Review Letters*, 105(11):118701, 2010.

John Paisley, David M Blei, and Michael I Jordan. Variational Bayesian Inference with Stochastic Search. In *Proceedings of the 29th International Conference on International Conference on Machine Learning (ICML)*, pages 1363–1370, 2012.

Carsten Peterson and James R Anderson. A Mean Field Theory Learning Algorithm for Neural Networks. *Complex Systems*, 1:995–1019, 1987.

Bill Petti. *baseballr*, 2016. R package version 0.4.

Jim Pitman. Combinatorial Stochastic Processes. *UC Berkeley Technical Report (621)*, 2002.

David C Plaut, S J Nowlan, and G E Hinton. Experiments on Learning by Back Propagation. *Technical Report*, 1986.

Arya A. Pourzanjani, Richard M. Jiang, and Linda R. Petzold. Improving the Identifiability of Neural Networks for Bayesian Inference. *NIPS Workshop on Bayesian Deep Learning*, 2017.

Adrian E Raftery, Michael A Newton, Jaya M Satagopan, and Pavel N Krivitsky. Estimating the Integrated Likelihood via Posterior Simulation Using the Harmonic Mean Identity. *Bayesian Statistics*, pages 1–45, 2007.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 814–822, 2014.

Rajesh Ranganath, Dustin Tran, Jaan Altosaar, and David Blei. Operator Variational Inference. In *Advances in Neural Information Processing Systems (NIPS)*, pages 496–504, 2016.

Carl Edward Rasmussen. Gaussian Processes in Machine Learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

Danilo Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of The 32nd International Conference on Machine Learning (ICML)*, pages 1530–1538, 2015.

Danilo Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1278–1286, 2014.

Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-Shot Generalization in Deep Generative Models. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

Oren Rippel, Michael A Gelbart, and Ryan P Adams. Learning Ordered Representations with Nested Dropout. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.

Christian Robert. *The Bayesian Choice*. Springer, 2001.

Mark Robinson. Priors for Bayesian Neural Networks. Master's thesis, University of British Columbia, 2001.

Abel Rodriguez and David B Dunson. Nonparametric Bayesian Models Through Probit Stick-Breaking Processes. *Bayesian Analysis*, 2011.

Frank Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological review*, 65(6):386, 1958.

Sheldon M Ross. *Simulation*. Academic Press, 2006.

David Saad and Emanuel Marom. Training Feed Forward Nets with Binary Weights Via a Modified CHIR Algorithm. *Complex Systems*, 4(5), 1990.

Tim Salimans, Diederik Kingma, and Max Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *ArXiv e-prints*, 2017.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-Learning with Memory-Augmented Neural Networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1842–1850, 2016.

Lawrence K Saul, Tommi Jaakkola, and Michael I Jordan. Mean Field Theory for Sigmoid Belief Networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

Leonard J Savage. *The Foundations of Statistics*. Courier Corporation, 1972.

Jürgen Schmidhuber. Deep learning in Neural Networks: An Overview. *Neural Networks*, 61: 85–117, 2015.

Uwe Schmidt and Stefan Roth. Learning Rotation-Aware Features: From Invariant Priors to Equivariant Descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2050–2057. IEEE, 2012.

Larry Schumaker. *Spline Functions: Basic Theory*. Cambridge University Press, 2007.

Jayaram Sethuraman. A Constructive Definition of Dirichlet Priors. *Statistica Sinica*, 1994.

Oran Shayar, Dan Levi, and Ethan Fetaya. Learning Discrete Weights Using the Local Reparameterization Trick. *International Conference on Learning Representations (ICLR)*, 2018.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian Optimization Using Deep Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2171–2180, 2015.

Kihyuk Sohn and Honglak Lee. Learning Invariant Representations with Local Transformations. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1311–1318, 2012.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder Variational Autoencoders. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3738–3746, 2016.

Daniel Soudry, Itay Hubara, and Ron Meir. Expectation Backpropagation: Parameter-Free Training of Multilayer Neural Networks with Continuous or Discrete Weights. In *Advances in Neural Information Processing Systems (NIPS)*, pages 963–971, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Mark FJ Steel et al. Bayesian Regression Analysis With Scale Mixtures of Normals. *Econometric Theory*, 16(01):80–101, 2000.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-End Memory Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2440–2448, 2015.

Luke Tierney, Robert E Kass, and Joseph B Kadane. Fully Exponential Laplace Approximations to Expectations and Variances of Nonpositive Functions. *Journal of the American Statistical Association*, 84(407):710–716, 1989.

Jakub M Tomczak. Prediction of Breast Cancer Recurrence Using Classification Restricted Boltzmann Machine with Dropping. *ArXiv e-prints*, 2013.

Jakub M Tomczak and Max Welling. VAE with a VampPrior. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Udo v Toussaint, Silvio Gori, and Volker Dose. Invariance Priors for Bayesian Feed-Forward Neural Networks. *Neural Networks*, 19(10):1550–1557, 2006.

Dustin Tran, Rajesh Ranganath, and David M Blei. Variational Gaussian process. *International Conference on Learning Representations (ICLR)*, 2016.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. REBAR: Low-Variance, Unbiased Gradient Estimates for Discrete Latent Variable Models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2624–2633, 2017.

Jos Uffink. The Constraint Rule of the Maximum Entropy Principle. *Studies in History and Philosophy of Modern Physics*, 27(1):47–79, 1996.

Aki Vehtari, Simo Sarkka, and Jouko Lampinen. On MCMC Sampling in Bayesian MLP Neural Networks. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, volume 1, pages 317–322. IEEE, 2000.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3630–3638, 2016.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout Training as Adaptive Regularization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 351–359, 2013.

Stefan Wager, William Fithian, Sida Wang, and Percy S Liang. Altitude Training: Strong Bounds for Single-Layer Dropout. In *Advances in Neural Information Processing Systems (NIPS)*, pages 100–108, 2014.

Stefan Wager, William Fithian, and Percy Liang. Data Augmentation via Levy Processes. *Perturbations, Optimization, and Statistics*, 2016.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of Neural Networks Using DropConnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1058–1066, 2013.

Chong Wang and David M Blei. Variational Inference in Nonconjugate Models. *Journal of Machine Learning Research*, 14(Apr):1005–1031, 2013.

Sida Wang and Christopher Manning. Fast Dropout Training. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 118–126, 2013.

David Warde-Farley, Ian J Goodfellow, Aaron Courville, and Yoshua Bengio. An Empirical Analysis of Dropout in Piecewise Linear Networks. *ArXiv e-prints*, 2013.

Max Welling and Yee W Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 681–688, 2011.

Peter M Williams. Bayesian Regularization and Pruning Using a Laplace Prior. *Neural Computation*, 7(1):117–143, 1995.

Peter M Williams. Matrix Logarithm Parametrizations for Neural Network Covariance Models. *Neural Networks*, 12(2):299–308, 1999.

Ronald J Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic Networks: Deep Translation and Rotation Equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5028–5037, 2017.

Y. Yao, A. Vehtari, D. Simpson, and A. Gelman. Yes, but Did It Work?: Evaluating Variational Inference. *ArXiv e-prints*, 2018.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning Hierarchical Features from Generative Models. In *Proceedings of The 34th International Conference on Machine Learning (ICML)*, 2017.