Making Decisions Under High Stakes: Trustworthy and Expressive Bayesian Deep Learning

Making Decisions under High Stakes: Trustworthy and Expressive Bayesian Deep Learning

A THESIS PRESENTED
BY
WANQIAN YANG
TO
THE DEPARTMENT OF COMPUTER SCIENCE
AND
THE DEPARTMENT OF STATISTICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE JOINT DEGREE OF
BACHELOR OF ARTS (HONORS)
IN THE SUBJECTS OF
COMPUTER SCIENCE AND STATISTICS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
MAY 2020

Making Decisions under High Stakes: Trustworthy and Expressive Bayesian Deep Learning

Abstract

Machine learning models applied to high-stakes domains must navigate the trade-off between (i) having enough representation power to learn effectively from high-dimensional, high-volume datasets, while (ii) avoiding cost-prohibitive errors due to model overconfidence and poor generalization. This need has led to growing interest in Bayesian neural networks (BNN), models that perform Bayesian inference on deep neural networks. BNNs are able to quantify predictive uncertainty over an inherently rich hypothesis space, which is crucial for high-stakes decision-making.

In this thesis, we present two contributions that tackle the shortcomings of BNNs. First, BNN priors are defined in uninterpretable parameter space, which makes it difficult for end users to express functional prior knowledge independent of training data. We formulate two novel priors that incorporate functional constraints (i.e. what values the output should hold for any given input) that can easily be specified by end users. The resulting model is amenable to black-box inference. We demonstrate its efficacy on two high-stakes domains: (i) enforcing physiologically feasible interventions on a clinical action prediction task, and (ii) enforcing racial fairness constraints on a recidivism prediction task where the training data is biased.

Next, variational approximations that are typically used for BNN posterior inference do not come with provable error guarantees, making it difficult to trust their predictive estimates. By exploiting the functional form of BNNs, we bound the predictive mean error of such approximations via maximum mean discrepancy on a reproducing kernel Hilbert space. Our bound is easily estimable and directly useful to end users as it is specified in predictive space.

Submitted by: Wanqian Yang

Thesis Advisors: Finale Doshi-Velez, Professor of Computer Science
Alexander Young, Undergraduate Advisor and Lecturer in Statistics

This thesis, submitted in the midst of the 2020 coronavirus pandemic, is dedicated to the Bayesian heroes of everyday life: to those who forge coherence and trust amidst uncertainty and noise, to those who persist in the face of constraints, and to those who refuse to be bound by failure or error.

To the Bayesian heroes of my life, my mother and my sister, who have stood by me in the most uncertain and constrained of times.

Acknowledgments

The first graduate seminar I took at Harvard was CS 282R, taught by Finale. As an undergraduate in my junior year, the material was simultaneously captivating and challenging. Variational inference? Normalizing flows? Neural networks that output the parameters of another neural network? *(Has science gone too far??)* The learning curve was steep, but I greatly enjoyed myself and learnt so much. That class was the start of 2 years of research in Finale's lab. Throughout this journey, I am immensely grateful for her tireless dedication to mentoring. The emphasis that she puts into conducting responsible and meaningful machine learning research has been formative to my own development in this field.

To Alex, I am terribly grateful for his constant support and advice as I developed this body of work, and especially for his meticulous eye to detail that has helped sharpen my writing. I am also grateful to Professor Hima Lakkaraju, who has supported me in directing my research towards the meaningful areas of interpretable machine learning and criminal justice applications.

I would additionally like to thank Moritz Graule for his ceaseless support and the many helpful discussions, Lars Lorch for his significant contributions to developing some of these ideas, and the many other contributors to this research: Nari Johnson, Anirudh Suresh, Srivatsan Srinivasan, Jiayu Yao, Melanie F. Pradier and Weiwei Pan. I am grateful to be surrounded by a supportive research community.

Finally, I view this thesis as a culmination of my academic odyssey at Harvard. I would like to acknowledge the many professors and peers whose ideas and discussions have inspired me. I am especially thankful for the mentorship and friendship of Deidre Lynch, Professor of Literature at Harvard, who has opened my eyes to the beauty of other arts even as I strive to perfect my own.

... salimmo sù, el primo e io secondo,
tanto ch'i' vidi de le cose belle
che porta 'l ciel, per un pertugio tondo.
E quindi uscimmo a riveder le stelle.

... up we climbed, he first and I second, until I saw,
the beautiful things the heavens carry, through a
round opening.
And thence we came forth to look again at the stars.

— Dante Alighieri, *Inferno*
(transl. Robert Durling)

WHAT SPECTACLE CONFRONTED THEM WHEN THEY, FIRST THE HOST, THEN THE GUEST, EMERGED SILENTLY, DOUBLY DARK, FROM OBSCURITY BY A PASSAGE FROM THE RERE OF THE HOUSE INTO THE PENUMBRA OF THE GARDEN?

The heaventree of stars hung with humid nightblue fruit.

— James Joyce, *Ulysses*

Then there was wind and violent thunder. There was a star riding through clouds one night, and I said to the star, "Consume me."

— Virginia Woolf, *The Waves*

Contents

List of Symbols

Abbreviations

BBB	Bayes by Backprop
BNN	Bayesian neural network
ELBO	evidence lower bound
HMC	Hamiltonian Monte Carlo
i.i.d.	independent and identically distributed
KL	Kullback-Leibler
MAP	maximum a posteriori (estimate or statistic)
MCMC	Markov chain Monte Carlo
MLP	multilayer perceptron
MMD	maximum mean discrepancy
NN	neural network
OOD	out-of-distribution
PDF	probability density function
RKHS	reproducing kernel Hilbert space
SVGD	Stein Variational Gradient Descent
VI	variational inference
WLOG	without loss of generality

Notation

$\mathbf{a}$	(bolded) a vector
a	(unbolded) a scalar
$\mathbf{w}_j$	the j^{th} dimension of $\mathbf{w}$
$\mathbf{w}^{(j)}$	the j^{th} sample of $\mathbf{w}$
$2^{\mathcal{X}}$	power set of $\mathcal{X}$

Alphanumeric Symbols

$\mathcal{W}$	parameter space, always taken to be $\mathbb{R}^M$
$\mathcal{X}$	input space, always taken to be $\mathbb{R}^Q$
$\mathcal{Y}$	output space, $\mathbb{R}$ for regression and $\{0, 1, \ldots, K-1\}$ for classification
$\mathbf{W}$	random variable for the BNN parameter
$\mathbf{X}$	random variable for the input
Y	random variable for the output
$\mathbf{w}$	specific instance of the BNN parameter
$\mathbf{x}$	specific instance of the input
y	specific instance of the output
K	number of output classes
M	dimensionality of parameter space
N	size of training dataset
Q	dimensionality of input space
D_{tr}	training dataset
D_{te}	test dataset
D_{KL}	KL divergence
$\mathcal{H}$	hypothesis space, or (reproducing kernel) Hilbert space
$\mathcal{N}$	(multivariate) Gaussian distribution

$\mathbf{L}_H$ the H^{th} hidden layer of an MLP

$\mathbf{Q}$ variational family

$\mathbb{I}$ indicator function

$\mathbb{N}$ set of natural numbers, defined to start from 0

$\mathbb{R}$ set of real numbers

$L^p(\mathcal{X};\mu)$ the L^p space over $\mathcal{X}$ with measure μ

ℓ^p the ℓ^p space

$C(\mathcal{X})$ the space of continuous functions over $\mathcal{X}$

$C_b(\mathcal{X})$ the space of continuous and bounded functions over $\mathcal{X}$

$\Phi_{\mathbf{w}}(\mathbf{x})$ the output function (over $\mathcal{X}$) of a MLP with parameter $\mathbf{w}$

$\boldsymbol{\theta}$ variational parameter

0

A Summary for the Layperson

Author's note: *During my time at Harvard, I have been asked by several friends and acquaintances (many without STEM backgrounds) about my research. I have always believed in the importance of communication across academic disciplines, and explaining one's work at all levels of abstraction. This section represents my efforts at writing a completely non-technical abstract of the thesis that anyone can understand.*

Tom Stoppard's absurdist play *Rosencrantz and Guildenstern Are Dead* begins with the two eponymous characters flipping a coin that repeatedly lands: `Heads`, `Heads`, `Heads`... After about 76 such repetitions, the characters begin pondering the nature of their existence in a world where the laws of probability do not quite seem to hold true. It takes them another 16 `Heads` before their suspicions are fully aroused, allowing the rest of the play to proceed in earnest.

Statistical Inference

What Rosencrantz and Guildenstern engaged in is the process of *inference from data*. Presented with evidence (the results of 92 coin tosses), they deduced certain likelihoods about the world around us. For example, 92 consecutive `Heads` might imply that one is in an alternate universe where the laws of physics are different. The same data is also far stronger evidence for the (narrower) conclusion that the coin in question is biased. In our day-to-day lives, our brains are constantly engaged in inference. When we communicate, we infer intent from speech and gestures, which

might be less conclusive if the words used are ambiguous. On the road, we infer the future actions of other drivers based on their current actions, which might be less conclusive if one is driving in Massachusetts.

Inference from data is deeply intertwined with the manipulation of probabilities. Performing and understanding this process is the key concern of statistics. We do so by specifying models that describe the mathematical relationship (or what we think is the correct relationship) between the input we have and the output we want. For example, in the clinical setting, we want to build a model that takes as input a patient's symptoms and vitals, and generates as output the correct medical action to be taken.

Machine Learning

Specifying the correct model isn't quite the end of the story. This is where the data comes in. We will use our observed data to solve or tune the model, so that it matches input to output correctly (i.e. infers the correct conclusion). For example, a simple clinical model might specify that likelihood of hypertension is a function of blood pressure. Using the data we have, we will tune this model so that it knows the *precise value* of blood pressure that results in hypertension. The more complicated the model we specify, and the more data we have, the harder it is to tune the model. It will take us ages to do this by hand. In most cases, we develop algorithms, to be run on computers, to do this task for us. The use of algorithms and models to infer relationships from data is precisely the field of machine learning, which lies at the intersection of computer science and statistics.

In recent years, one particular class of models, known as deep neural networks, have gained particular interest. These models are capable of tackling a wide variety of problems ranging from image detection to language translation, so long as they can be tuned correctly. Even though neural networks were invented in the 1980s, it is the recent improvements in computational hardware and algorithms that have made tuning them a practical possibility, catapulting their widespread use.

High-Stakes Machine Learning

Today, machine learning models are used in a vast number of domains, from spam filtering to self-driving cars. However, not all domains carry equal risk. Wrongly classifying a spam email as authentic does negligible harm. Wrongly accelerating

at a traffic junction can be fatal. This thesis considers the application of machine learning for high-stakes decision-making – domains where an erroneous prediction can be disastrous.

To build models that perform well in high-stakes settings, we need our models to *reason under uncertainty*. In other words, not only must the model be able to generate predictions from data, it must quantitatively express how confident it is about these predictions. This allows us to choose whether to trust the model's output or return control to a human expert to make the final decision. As it turns out, a statistical framework known as *Bayesian inference* allows us to do just that. Applying Bayesian inference to neural networks result in (the succinctly named) *Bayesian neural networks* (BNNs), versatile models that can both (i) reason under uncertainty and (ii) be applied towards a vast set of domains.

While the concept of a BNN was first proposed in the mid-1990s, interest in these models have resurfaced in the past few years, due to (again) the general improvement in computational hardware (that has made inference practical and scalable), as well as the growing need for models that function well over high-risk domains. **This thesis contributes to the growing body of BNN research literature by tackling two core problems.**

PROBLEM 1: ENFORCING DOMAIN CONSTRAINTS

In many high-stakes scenarios, domain experts are privy to prior knowledge that is not found in the data. For example, in the healthcare setting, a doctor might know that certain classes of drugs like statins are contraindicated in pregnant women. As models infer solely from data, the only way for a model to learn these rules is to hope that our data contains plenty of examples of, say, pregnant women that were *not* prescribed statins. This is clearly not an effective approach. We thus propose methods for explicitly incorporating these prior expert knowledge (which we call *domain constraints*) into the inference process.

Why is this important? Many datasets that machine learning models are trained on are incomplete or biased. The ability to explicitly enforce constraints allow us to counteract problematic datasets. As a prominent example, in some areas of the United States, machine-learned models aid judges in reaching sentence and bail decisions based on likelihood of recidivism. The datasets that these models are trained on often reflect the demographic biases rampant in history. To prevent the model

from learning these biases, one could explictly enforce a fairness constraint across race or gender. In general, enforcing constraints are a way of imposing desiderata like safety and fairness on models.

Problem 2: Diagnosing inference algorithms

The versatility of BNNs come with a trade-off, which is that it is almost impossible to "tune" these models perfectly to the theoretically correct degree. In practice, most BNN users employ a clever method that approximates the true tuned model, known as *variational inference*. While empirical experiments have shown that variational inference generally works quite well in practice, a rigorous characterization of inference solutions coming from variational inference has not been well-studied. Furthermore, recent research has also shown that variational inference can be quite problematic in many situations.

This is an important problem, because in order to trust BNNs to perform well in high-stakes settings, we need to have proof that the tuning algorithms we use return reliable and correct solutions. As such, we tackle this problem by mathematically analyzing variational inference, and coming up with methods to diagnose how well variational inference performs during the BNN tuning process.

The theme of this thesis is reliable and safe machine learning for high-stakes decision-making. Consequentially, both of the problems above are broadly related in the sense that the goal is to increase our trust in BNNs as models that not only operate in risky domains, but do so in a reliable and safe manner. The rest, as they say, are details.

1

Introduction

A junior doctor makes her morning rounds. For each patient, she consults her notes, which summarize the symptoms and initial diagnosis given to that patient, along with the details of the hospital stay: medications given, interventions taken. If the patient is conscious, she may ask them some questions about their current condition. With all of this information, she will make an initial decision on the patient's prognosis and subsequent medical actions to be taken. Each patient she treats increases her confidence and knowledge about the correct medical decisions to be made.

Learning functions, which are maps from a set of inputs (e.g. symptoms, diagnosis, past actions) to outputs (e.g. prognosis, future actions), is an task ubiquitous in our daily lives. In machine learning, this is formalized as the problem of *supervised learning*. Given access to a finite and possibly noise-corrupted set of input-output pairs (the "data" or "evidence"), we specify statistical models that formalize our hypothesis about the input-output relationship, and we perform statistical inference to learn various statistics, e.g. model parameters if the model is parametric. We call the supervised task *regression* if the output is real and scalar, and *classification* if the output is categorical. More complicated output codomains are studied under the aegis of structured learning; this is beyond the scope of this thesis.

A class of parametric models known as *(deep) neural networks* (NN) has garnered immense interest and attention over the past decade. The simplest neural networks, known as multilayer perceptrons, describe alternating compositions of

linear and non-linear transformations applied to the initial input vector. The resulting model is an universal approximator [34] with substantial model capacity. More complicated NN architectures incorporate compositions such as convolutions, which further allows us to exploit the structures of complex input data like images or text. As a result, NNs have outperformed traditional statistical models in many important applications such as computer vision and natural language processing [30, 45, 54, 55, 72]. However, these models have their limitations: (i) they are computationally costly to train (often requiring vast amounts of data and dedicated hardware), (ii) their theoretical properties are poorly understood and still remain a field of active research, and (iii) they do not generalize well and are therefore prone to issues such as adversarial attacks [24].

The overconfidence of classical NNs has, in part, spurred recent interest in a different paradigm of training them – namely, via Bayesian inference. In the Bayesian setting, one learns a conditional distribution over the network parameters with respect to the data (the posterior), instead of the usual local optimum of the loss landscape. The posterior can then be marginalized in the predictive setting. The resulting model, known as a *Bayesian neural network* (BNN), acts using many sets of parameters (weighed by their posterior probabilities) instead of a point estimate [52, 56]. This naturally accounts for model (epistemic) uncertainty and is crucial in settings where (i) training data is sparse, and therefore multiple hypotheses explain the data equally well, and (ii) high-stake applications where overconfident model predictions are unacceptable. In the latter, BNNs have seen success in areas such as autonomous driving [38, 39] and medical diagnostics [76].

Bayesian inference is a well-understood framework in statistical literature. However, its application to NNs is not seamless and several core challenges remain. This thesis tackles two such challenges. First, the expressivity and accuracy of BNNs depend on the ability to specify prior distributions that (i) are interpretable and (ii) can encode versatile functional forms. In practice, priors used for BNN inference are vague and uninterpretable, in part due to the high-dimensional parameter space in which these models operate. We introduce a novel prior formulation that allows the user to encode functional constraints. The ability to specify constraints, ranging from safety to fairness, will greatly improve the applicability of BNNs. We show concrete applications to healthcare and criminal justice. Second, as exact posterior inference is intractable for BNNs, practitioners typically use a technique known as

variational inference to learn an approximate distribution. These approximations, however, do not come with provable guarantees on their performance. Trusting inference quality is a crucial prerequisite for the safe and reliable deployment of BNNs, which motivates bounds on the predictive error of these approximations.

THESIS STRUCTURE The remainder of this chapter concisely builds a self-contained description of BNNs from first principles and motivates the specific problems that the thesis investigates. Chapter 2 describes **Output-Constrained BNNs**, which enable functional constraints to be specified as prior knowledge on BNNs. Chapter 3 describes **KEBO-VPME**, a kernelized bound on the variational predictive mean error. Lastly, Chapter 4 offers a concluding discussion and outlines future work.

1.1 SUPERVISED LEARNING

Let $\mathbf{X} \in \mathcal{X}$ be a predictor (input) variable. In this thesis, we consider the typical setting $\mathcal{X} = \mathbb{R}^Q$, where the dimensions of $\mathbb{R}^Q$ are the Q individual predictors. We will assume that all necessary feature transformations, e.g. dummy encoding of categorical features or standardization of continuous features, have been carried out. Let $Y \in \mathcal{Y}$ be the response (output) variable. For regression tasks, $\mathcal{Y} = \mathbb{R}$. For K-classification tasks, $\mathcal{Y}$ is any set with a bijection to $\{0, 1, \ldots, K-1\}$. In supervised learning, we suppose that some true (or ideal) mapping $f^* : \mathcal{X} \to \mathcal{Y}$ exists which we seek. We also specify that $\mathbf{X} \times Y \sim \mathcal{D}$ such that $p(Y = f^*(\mathbf{X})|\mathbf{X})$ is high. In other words, $\mathcal{D}$ is an accurate data-generating distribution. For example, $Y|\mathbf{X} \sim \mathcal{N}(f^*(\mathbf{X}), \sigma_\epsilon^2)$ represents data corrupted by Gaussian noise. While we do not know either f^* or $\mathcal{D}$, we have access to observed data $D_{tr} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, which are N independent samples from $\mathcal{D}$. The response values in D_{tr} are also called *labels*.

The goal is to inductively learn f^* within the function space $\mathcal{Y}^\mathcal{X} = \{f \,|\, f : \mathcal{X} \to \mathcal{Y}\}$ using D_{tr}. Without further statistical assumptions, such induction is impossible. Instead, we specify a *model* that encodes our beliefs on $Y|\mathbf{X}$, such as continuity or linearity. This restricts our search to a hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^\mathcal{X}$, which represents the subset of functions with the specific relationship between $\mathbf{X}$ and Y that the model describes. Here we consider only *parametric models*, where $\mathcal{H} = \{f_\theta \,|\, \theta \in \mathbb{R}^M\}$ and $f_\theta(\mathbf{x})$ is fully determined from $\mathbf{x} \in \mathcal{X}$ and a real-valued vector θ (the model's parameter). Next, we need to quantify the discrepancy between the model's output and the true target value for all $\mathbf{x} \in \mathcal{X}$. This is done via a *loss function* $\mathcal{L}$

where $\mathcal{L}(f_\theta(\mathbf{x}), y)$ is the error for $\mathbf{x} \in D_{tr}$. Then the optimal parameter θ^* is the one that minimizes the expected loss (or *risk*) of the model:

$$\theta^* = \arg\min_\theta \mathbb{E}_{\mathbf{X}\times Y\sim\mathcal{D}}\big[\mathcal{L}(f_\theta(\mathbf{X}), Y)\big] \tag{1.1}$$

Without access to $\mathcal{D}$, and given that our data is i.i.d., we estimate (1.1) as:

$$\hat{\theta} = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), y_i) \tag{1.2}$$

This is known as empirical risk minimization [71]. For most cases, there are commonly used loss functions with important statistical interpretations.

The choice of model is crucial, as we have to make statistical assumptions that (i) are expressive enough to capture the ground-truth (so that $\mathcal{H}$ contains f^* itself, or some close approximation f' where $f' \approx f^*$ [1]), (ii) make induction possible (so that $\hat{\theta}$ is close to θ^*) and (iii) are tractable (so that we can devise efficient algorithms to learn $\hat{\theta}$). The common caveat is that models with higher capacity (richer $\mathcal{H}$) are more likely to induce poorly by overfitting the data: this is the so-called bias-variance tradeoff. The reader may refer to standard texts such as [17] or [3] for a more thorough treatment of supervised learning and common statistical models.

1.2 Deep Neural Networks

The particular parametric model that we consider in this thesis is known as the *(artificial) neural network*. While historically inspired by efforts to mimic the neural circuitry found in the brain [53], NNs should simply be viewed as mathematical constructs capable of modeling a diverse set of relationships. We will introduce the simplest type of NN, the *multilayer perceptron* (MLP), here in two equivalent ways: graphically and functionally.

The Graphical Interpretation A MLP is a directed graph $(\mathbf{V}, \mathbf{W})$. The nodes $\mathbf{V} = (\mathbf{L}_0, \mathbf{L}_1, \ldots, \mathbf{L}_H, \mathbf{L}_{H+1})$ are organized into *layers*, where the input layer $\mathbf{L}_0$ consist of the individual input predictors, the H intermediate layers $\mathbf{L}_{h:h\notin\{0,H+1\}}$ are called hidden layers, and $\mathbf{L}_{H+1}$ is the final output layer. For regression, $\mathbf{L}_{H+1}$

[1] So that even if it is not possible to learn f^* exactly, we can learn a "similar" function f'.

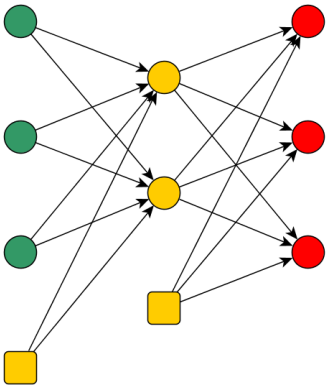


Figure 1.2.1: A typical diagram depicting the NN as a graph. This NN has an input layer (4 nodes including the bias term, i.e. $\mathcal{X} = \mathbb{R}^3$), a single hidden layer (3 nodes including the bias term) and an output layer (3 nodes, i.e. $\mathcal{Y} = \{0, 1\}$). The bias nodes are squares. Taken from: `https://stats.stackexchange.com`.

contains a single node representing the final output; for K-classification, $\mathbf{L}_{H+1}$ contains K nodes, where each node denotes the output probability for that class, i.e. $\sum_{u \in \mathbf{L}_{H+1}} u = 1$. We add nodes with a nominal value of 1 to every layer except $\mathbf{L}_{H+1}$; these nodes represent the bias or intercept term. There is a weighted edge from every $u \in \mathbf{L}_{h-1}$ to every $v \in \mathbf{L}_h$, with the exception that bias nodes have no incoming edges. $\mathbf{W}$ is therefore the set of all edges, represented as a vector of weights (the MLP's parameters). Each node can be seen as a unit of computation, receiving as input the weighted sum of all nodes in the previous layer, and applying a non-linear transformation to this weighted sum. This value is then passed on to the nodes in the next layer, and so on. In essence, the MLP can be considered as a model defined by hierarchical layers of computation, proceeding from the input vector all the way to the output vector, consisting of successive compositions of linear basis function regression models. Figure 1.2.1 shows an example of a MLP with a single hidden layer. MLPs with more than one hidden layer are called *deep*.

Since the MLP is an acyclic graph, it belongs to the class of NNs known as *feedforward neural networks*, where computation proceeds in a single direction from input layer to output layer. Many other variants of NNs exist, where the graphs have more complicated connectivity; this is referred to as the architecture of the NN. For the purpose of this thesis, we will consider only MLPs since the problems and applications herein will not require more complex architecture. Hereafter, we will use the term NN to describe only MLPs.

THE FUNCTIONAL INTERPRETATION A 1-layer MLP [2] can be represented with the parametric equation:

$$\Phi_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}_2 \left[\boldsymbol{\sigma}\Big(\mathbf{W}_1 \left[\mathbf{x}, \quad 1 \right]^\top \Big), \quad 1 \right]^\top \tag{1.3}$$

where $\mathbf{W}_1$ is the $|\mathbf{L}_1| \times |\mathbf{L}_0|$ matrix (representing all input-to-hidden weights) and $\mathbf{W}_2$ is the $|\mathbf{L}_2| \times |\mathbf{L}_1|$ matrix (representing all hidden-to-output weights) [3]. $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2)$ is the set of all network weights. Eq. (1.3) makes explicit what we earlier described in the graphical view: a non-linear transformation, represented by σ, is applied element-wise to the nodes of the hidden layer:

$$\boldsymbol{\sigma}\Big(\left[\mathbf{x}, \quad 1 \right]^\top \Big)_j = \sigma\Big(\left[\mathbf{x}, \quad 1 \right]_j \Big) \quad \forall j \in \{0, 1, \ldots, |\mathbf{L}_1| - 1\} \tag{1.4}$$

Why are these non-linear transformations (also called *activation functions*) necessary? Note that without a non-linear component, MLPs would simply reduce to linear models since the set of linear functions is closed under composition. Historically, σ were chosen to be threshold functions on the node's value [4]. For example, a common activation function is the rectified linear unit (ReLU):

$$\sigma_{RELU}(x) = \max(0, x) \tag{1.5}$$

More importantly, under certain (mild) assumptions on the choice of σ, a 1-layer MLP is able to approximate every continuous function defined on compact sets of the input space $\mathcal{X} = \mathbb{R}^Q$, i.e. $\mathcal{H}$ generated by 1-layer MLPs is dense in $C(\mathbb{R}^Q)$ [11]. This is the so-called Universal Approximation Theorem that gives neural networks their rich model capacity.

Generalizing (1.3) to the multilayered setting is straightforward:

$$\Phi_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}_{H+1} \left[\boldsymbol{\sigma}\Big(\mathbf{W}_H \left[\boldsymbol{\sigma}\Big(\ldots \boldsymbol{\sigma}\Big(\mathbf{W}_1 \left[\mathbf{x}, \quad 1 \right]^\top \Big) \ldots \Big), \quad 1 \right]^\top \Big), \quad 1 \right]^\top \tag{1.6}$$

We simply compose further linear combinations and nonlinear activations on each successive MLP layer. Note that we do not apply any nonlinear component to the

[2] Note the convention here that we refer to a MLP only by the number of hidden layers, H.

[3] In other literature, the term "weight" strictly refer to non-bias connections. Here, we use "weights" and "parameters" interchangeably since the distinction is unimportant from an inference standpoint.

[4] Hence the name activation function, a reference to the action potential in biological neuron firing.

final output layer. The exception to this is the K-classification setting, where we apply the *softmax function* to the $|\mathbf{L}_{H+1}| = K > 1$ nodes of the final layer:

$$\text{softmax}(u_j) = \frac{\exp\{u_j\}}{\sum_{l\in\{0,1,\ldots,K-1\}} \exp\{u_l\}} \quad \forall\, u_j \in \mathbf{L}_{H+1} \tag{1.7}$$

to ensure that the values of the output layer sum to 1, corresponding to classification probabilities. When the actual class label is desired, we typically choose the class corresponding to the node with the largest probability.

The parameters of the MLP are $\mathbf{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_{H+1})$, which hereafter we simply represent as a flattened vector $\mathbf{W} \in \mathbb{R}^M$. The hypothesis space generated by MLPs is $\mathcal{H}_{\text{NN}} = \{\Phi_{\mathbf{W}} \,|\, \mathbf{W} \in \mathbb{R}^M\}$. Similar to other parametric models, we learn the globally optimal parameter $\widehat{\mathbf{W}}$ by minimization of (1.2). Because the loss landscape of MLPs is highly nonconvex, provably finding $\widehat{\mathbf{W}}$ is **NP**-complete [4]. Instead, a local optimum $\widetilde{\mathbf{W}}$ is typically learned using an iterative optimization algorithm such as gradient descent; the most common method is **stochastic gradient descent** (SGD) [6] since the size of D_{tr} can reach up to the order of millions. Gradients of $\mathbf{W}$ are computed using the *backpropagation algorithm* [31], which utilizes the chain rule to differentiate across layers of the NN. Backpropagation can be easily implemented via automatic differentiation in most modern programming languages (e.g. the `PyTorch` framework in Python [59]), allowing for efficient learning of NNs.

Because the Universal Approximation Theorem generalizes to the multilayered setting [34], we are assured that MLPs are powerful function approximators and $\mathcal{H}_{\text{NN}}$ is large enough to learn most functions that we care about practically. In some sense, if our goal is to learn an efficient model, by which we mean a model that is polynomial in both sample and runtime complexity, then we can do no better than MLPs, which have polynomial sample complexity and can approximate any other efficient model over Q predictors [66, Chapter 20]. Unfortunately, the drawbacks are also significant. As we noted earlier, learning $\widehat{\mathbf{W}}$ is difficult. Algorithms like SGD tend to find reasonable solutions in many situations. However, they come with few provable guarantees and often generalize poorly outside the data distribution $\mathcal{D}$. The choice of MLP depth H and size of each hidden layer $\mathbf{L}_h$ are also "hyperparameters" that the practitioner must specify, and it is not always clear what the reasonable choices are for the application at hand. The reader is invited to consult [46] or Chapter 20 of [66] for a *deep*er discussion of neural networks.

1.3 ONWARDS TO THE BAYESIAN PARADIGM

Up to this point, we have described supervised learning using the tools of frequentist inference. In the frequentist setting, estimands of interest associated with a statistical model have fixed, ground-truth values that are unknown. We construct estimators that are correct with high probability (c.f. confidence intervals) based on the finite set of data that we observe, and we test these estimators with respect to some baseline or hypothesized value (c.f. hypothesis testing) [5].

In the context of supervised learning with neural networks, we assume that the ground-truth function $f^* : \mathcal{X} \to \mathcal{Y}$ exists and corresponds to a NN with the parameter $\mathbf{W}^*$. As $\mathbf{W}^*$ is unknown, we estimate it as $\widehat{\mathbf{W}}$ via empirical risk minimization (1.2). In fact, as noted above, $\widehat{\mathbf{W}}$ itself is difficult to obtain and we instead solve for the local optimum $\widetilde{\mathbf{W}}$ with the gradient descent algorithm. Compared to traditional statistical models, there are some differences when performing frequentist inference for NNs. In particular, hypothesis testing and construction of confidence intervals are typically not carried out in NN literature, due to both the complexity of NNs as well as minimal statistical assumptions applied to $\mathcal{D}$. Instead, $\widetilde{\mathbf{W}}$ is evaluated on a held-out test set $D_{te} \sim \mathcal{D}$, which is an independent set of observations not used to learn $\widetilde{\mathbf{W}}$ earlier.

This last approach raises concerns for training NNs in the classical manner. With sufficient free parameters (i.e. hidden layers and nodes), NNs are expressive enough to model the noise in D_{tr} and can do so under empirical risk minimization. The resulting learned function may be overfitted on D_{tr}, generalizing poorly to regions of $\mathcal{X}$ not represented in D_{tr}. As $\widetilde{\mathbf{W}}$ is a point estimate with few theoretical guarantees besides empirical success on D_{te}, one can only hope that generalizing well to D_{te} is an indication of generalization everywhere on $\mathcal{X}$. In particular, if we happen to choose some D_{te} *out-of-distribution* (i.e. $D_{te} \not\sim \mathcal{D}$), it is quite possible that test error will be significantly higher than the training error.

Overfitting on NNs is a well-known problem and many approaches exist that improve robustness. For example, NNs can be trained via structural risk minimization,

[5] Technically, frequentism describes both a philosophical approach to probability as well as a specific methodology for carrying out inference. As the former, frequentists treat probability as the (limit to infinity of the) frequency of an outcome, if the experiment is run infinitely many times. However, having a frequentist interpretation of probability does not imply that one has to use frequentist inference (and similarly so for Bayesian interpretation vs inference, as we will soon describe). It is the *mode of inference* that we are primarily concerned with.

whereby regularization terms are added to (1.2). This penalizes excessively large parameter values and deters learning overly expressive functions [71]. Regularization is widely studied in simpler statistical models. Another commonly used technique (unique to NNs) is dropout [68], where NN parameters are randomly zeroed during each iteration of gradient descent. However, most of these techniques do not come with theoretical guarantees such as confidence intervals, and ultimately, we still rely on evaluation of D_{te} as an empirical measure of generalization.

Unfortunately, as $\mathcal{H}_{\text{NN}}$ is such an expressive hypothesis space, enforcing strong generalization in NNs is still an open problem [6]. While empirical success on D_{te} is sufficient for applications where threshold levels of error are tolerable, in the high-stakes applications motivated at the beginning of this chapter, we cannot afford overconfident but erroneous NN predictions resulting from poor generalization. To compound the issue, these settings typically involve high-dimensional $\mathcal{X} = \mathbb{R}^Q$. Under such a large input space, it is quite possible that the real-life data that the NN is deployed on comes from a distribution different from $\mathcal{D}$ — this is the so-called curse of dimensionality. Therefore, we need the system to possess the ability to reason about its own confidence — the model should be able to recognize a test point out-of-distribution and return prediction control to a human domain expert.

We can push this argument further. In some cases, generalization is not only difficult but inherently *meaningless*. For example, what if a NN trained on classifying various functional magnetic resonance imaging (fMRI) scans is erroneously shown an electroencephalography (EEG) scan instead? [7] Given any input image, the NN will, of course, output a label — after all, we're simply running an input through (1.6). It is clear, however, that any prediction resulting from an EEG scan as input makes no sense for a NN trained on fMRI scans.

The futility of pursuing perfect generalization also challenges our earlier methodology. Thus far, we have assumed that some ground-truth function f^* exists, which corresponds (one-to-one) to an ideal parameter $\mathbf{W}^*$ that we seek to learn from data. It is quite possible, however, that multiple sets of parameters $\mathbf{W}$ explain the data equally well. In fact, since NNs represent such a rich hypothesis space, it is likely

[6] For example, the 2019 *(most recent as of the publication of this thesis)* iteration of the International Conference on Machine Learning (ICML) features a workshop dedicated to research in NN generalization; see `https://sites.google.com/view/icml2019-generalization/home`.

[7] Here, the input space is the space of all images (of a fixed size), typically represented as a $W \times H \times 3$ real tensor of pixel values.

that these models are *underspecified* by the data on hand, implying that multiple parametrizations exist that will perform well [74]. Without further data, it may be impossible to distinguish between these competing hypotheses, and there is no reason to rely solely on one such solution ($\widetilde{\mathbf{W}}$). We are faced with an irreducible source of uncertainty, known as **epistemic uncertainty**, resulting from equally compelling sets of parameters [8]. Furthermore, even if a ground-truth parameter $\mathbf{W}^*$ exists and we have enough data to eliminate other hypotheses, learning $\mathbf{W}^*$ exactly is still an impossible task under finite, noise-corrupted data. The noise represents another inherent source of randomness that cannot be accounted for — this is **aleatoric uncertainty**, i.e. uncertainty due to measurement imprecision.

Being able to reason coherently with these sources of uncertainty is useful for statistical modeling in general but especially crucial for NNs. The trade-off for the ability to model expressive functions that mirror the complexity of real-life domains is the difficulty or impossibility of finding an ideal $\mathbf{W}^*$. This warrants a pivot away from the frequentist setting, particularly in high-stakes situations where such failure to generalize can be prohibitively costly. Instead, the solution is to construct a system that is capable of quantitatively reasoning about its confidence (both with respect to the parameter $\mathbf{W}$ as well as the predicted output Y for any $\mathbf{x}$), so as to make predictions in line with its uncertainty. Formally, we consider models that can learn *probability distributions* for both $\mathbf{W}$ and $Y|\mathbf{x}$.

Bayesian inference is one such coherent system. Instead of treating quantities of interest as having fixed but unknown values that we can estimate, the Bayesian framework casts them as *random variables* with probability distributions that we can infer. [9] Consider the joint probability space of the evidence D_{tr} and the variable of interest $\mathbf{W}$. We seek the conditional distribution $\mathbf{W}|D_{tr}$, which can be computed using **Bayes' Rule**:

$$p(\mathbf{w}|D_{tr}) = \frac{p(\mathbf{w})p(D_{tr}|\mathbf{w})}{p(D_{tr})} \tag{1.8}$$

We call $p(\mathbf{w}|D_{tr})$ the **posterior** (distribution), $p(\mathbf{w})$ the **prior** (distribution) and $p(D_{tr}|\mathbf{w})$ the **likelihood** (distribution). Since D_{tr} are i.i.d. by assumption, we can

[8] Technically, epistemic (model) uncertainty comprises both parameter uncertainty and structural uncertainty (in the case of NNs, variables like number of nodes and network depth). We consider only parameter uncertainty and treat NN architecture as a hyperparameter. More complex modelling choices such as a hierarchical Bayesian model can also incorporate structural uncertainty rigorously.

[9] Again, Bayesian inference is related to, but distinct from, the Bayesian interpretation of probability, where probabilities quantify our degree of belief in some outcome based on observable evidence.

decompose the likelihood into individual observations:

$$p(D_{tr}|\mathbf{w}) = \prod_{i=1}^{N} p(Y_i|\mathbf{x}_i, \mathbf{w}) \tag{1.9}$$

The likelihood $p(Y_i = y_i|\mathbf{x}_i, \mathbf{w})$ is closely related to the loss function $\mathcal{L}(\Phi_{\mathbf{w}}(\mathbf{x}_i), y_i)$ and both measure how well $\mathbf{w}$ explains the observation $(\mathbf{x}_i, y_i)$. The prior $p(\mathbf{w})$ models our initial belief of how $\mathbf{W}$ is distributed prior to any observation of data. The evidence probability $p(D_{tr})$ is the integral

$$p(D_{tr}) = \int_{\mathcal{W}} p(D_{tr}|\mathbf{w}')p(\mathbf{w}')\,\mathrm{d}\mathbf{w}' \tag{1.10}$$

Note that the prior and likelihood are both modelling choices. Having specified them, (1.8) can be easily derived from the fundamental axioms of probability.

Treating $\mathbf{W}|D_{tr}$ as a random variable allows us to quantify uncertainty, which is effectively the spread of the posterior distribution. With appropriately defined priors and likelihoods, $\mathbf{W}|D_{tr}$ accurately models both epistemic and aleatoric [10] uncertainty. The greater $Var[\mathbf{W}|D_{tr}]$ is, the more uncertain we are about what the correct value of $\mathbf{W}$ should be.

Our probabilistic setting carries directly to prediction. For a new point $\mathbf{x}'$, the predictive distribution over output Y' is:

$$p(Y'|\mathbf{x}', D_{tr}) = \int_{\mathcal{W}} p(Y'|\mathbf{x}', \mathbf{w})p(\mathbf{w}|D_{tr})\,\mathrm{d}\mathbf{w} \tag{1.11}$$

where $p(Y'|\mathbf{x}', \mathbf{w})$ is the same likelihood as in (1.9) [11]. $p(Y'|\mathbf{x}', D_{tr})$ is known as the **posterior predictive** (distribution) and allows for uncertainty quantification in $\mathcal{Y}$ directly. Greater $Var[Y'|\mathbf{x}', D_{tr}]$ indicates greater uncertainty about what the output for $\mathbf{x}'$ should be. Having access to $Var[Y'|\mathbf{x}', D_{tr}]$ along with an actual label estimate (say, $\mathbb{E}[Y'|\mathbf{x}', D_{tr}]$) allows us to model predictions *and* their corresponding confidences, which can aid us in deciding whether the prediction itself should be trusted. Note that large $Var[\mathbf{W}|D_{tr}]$ does not necessarily imply large $Var[Y'|\mathbf{x}', D_{tr}]$ for all points. We will expect $Var[Y'|\mathbf{x}', D_{tr}]$ to be smaller for $\mathbf{x}'$ near observations in D_{tr}, and larger when further away from the training distribution.

[10] Note that the noise is actively modelled by defining $p(Y_i|\mathbf{x}_i, \mathbf{w})$.

[11] Except that we now replace the true label y_i with the NN output $\Phi_{\mathbf{w}}(\mathbf{x})$.

We have described a coherent system that not only models sources of uncertainty in the learning process but is quantitatively able to generate confidence measures for individual predictions. While Bayesian inference can be applied to any model (and readers are referred to a standard text like [21] for a comprehensive treatment), we stress again that it is particularly useful for working with NNs. From an application standpoint, NNs are used to model complex, high-stakes scenarios, where functions that generalize well across high-dimensional input spaces are hard to find and so confidence measures on predictions are necessary. From a modelling standpoint, NNs represent a rich hypothesis space where there are likely multiple competing hypotheses $\mathbf{W}$ that explains the data equally well, i.e. $\mathbf{W}|D_{tr}$ is diffuse. These various $\mathbf{W}$ may lead to different predictive distributions $Y'|\mathbf{x}', \mathbf{W}$ for a new test point $\mathbf{x}'$, which means using the full distribution $\mathbf{W}|D_{tr}$ will generate better predictions than the point estimate $\widetilde{\mathbf{W}}$.

1.4 Bayesian Neural Networks

We name the resulting model a **Bayesian neural network** (BNN) when we carry out Bayesian inference on NNs via (1.8). The complete process for supervised learning with BNNs is as follows:

1. We specify the prior and likelihood.
2. We infer the posterior. An analytical evaluation of (1.8) is not possible, due to both the integral in (1.10) as well as the complicated prior and likelihood distributions. We will instead rely on algorithms for approximate inference.
3. For prediction of a new test point $\mathbf{x}'$, we compute the posterior predictive distribution using (1.11). As (1.11) is also intractable due to the integral, we typically sample a finite set of parameters from $\mathbf{W}|D_{tr}$ as an approximation.

Notice that we can choose to use point estimates. Moving from Step 2 to Step 3, we can use the **posterior mean** $\mathbb{E}[\mathbf{W}|D_{tr}]$ or the **maximum a posterior (MAP)** estimate $mode[\mathbf{W}|D_{tr}]$ to approximate (1.11) instead of the full posterior. For decision-making, we can also use the **posterior predictive mean** $\mathbb{E}[Y'|\mathbf{x}', D_{tr}]$ or the **predictive MAP** $mode[Y'|\mathbf{x}', D_{tr}]$ to approximate the final output distribution. Choosing to take point estimates offsets the advantage of using Bayesian inference, but will be computationally cheaper. The frequentist estimator $\widetilde{\mathbf{W}}$ can be viewed as a point estimate as well. We will now describe each of the three steps above in detail.

1.4.1 LIKELIHOODS FOR BNNS

The likelihood is purely a function of the model prediction $\Phi_{\mathbf{W}}(\mathbf{x})$ and the correct target y and does not depend on $\mathbf{W}$ directly. As such, BNN likelihood distributions follow the standard choices used in other probabilistic models. In this thesis, we will use the likelihood distributions presented below:

REGRESSION We would expect $Y|\mathbf{x}, \mathbf{W}$ to be a Dirac delta function centered on $\Phi_{\mathbf{W}}(\mathbf{x})$. However, this is unrealistic as target labels for most applications are noisy. As such, we typically model output noise to be (zero-meaned) Gaussian-distributed: $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ where σ_ϵ is the variance of the noise, treated as a hyperparameter. The likelihood PDF is then simply a Gaussian centered on $\Phi_{\mathbf{W}}(\mathbf{x})$:

$$p(Y = y|\mathbf{x}, \mathbf{w}) = \frac{1}{\sigma_\epsilon\sqrt{2\pi}} \exp\{-\frac{1}{2}\Big(\frac{y - \Phi_{\mathbf{w}}(\mathbf{x})}{\sigma_\epsilon}\Big)^2\} \tag{1.12}$$

Another benefit of modeling noise is that the resulting posterior expression is amenable to gradient-based inference algorithms. A discontinuous PDF like the Dirac delta function, or a non-Lipschitz continuous approximation, can cause numerical computation issues.

CLASSIFICATION Unlike the regression setting, labels in D_{tr} are discrete and typically assumed to be correct. As such, we do not explictly model for "noise" (in the sense that the label is wrong with some small probability). As noted in Section 1.2, since the NN outputs class probabilities in the final layer $\mathbf{L}_{H+1}$ (instead of directly predicting classes), then

$$p(Y = k|\mathbf{x}, \mathbf{w}) = \Phi_{\mathbf{w}}(\mathbf{x})_k \tag{1.13}$$

i.e. simply the value of the node representing class k (after the application of the softmax function).

1.4.2 PRIORS FOR BNNS

In the Bayesian framework, the prior $p(\mathbf{W})$ expresses our initial belief about the value of the unknown $\mathbf{W}$. One might suggest using a non-informative prior, since it is difficult *a priori* to determine the relationship between specific values of individual NN weights and the type of resulting functions $\Phi_{\mathbf{W}}$ we believe to be true. A

reasonable choice for a non-informative prior would be a uniform distribution over $\mathcal{W} = \mathbb{R}^M$. However, this prior is improper since $\mathbb{R}^M$ is unbounded and there are no guarantees that the posterior will be proper and contain no pathologies.

Instead, we can consider weakly informative priors over $\mathbb{R}^m$. The common choice is an isotropic (multivariate) Gaussian $\mathbf{W} \sim \mathcal{N}(\mathbf{0}, \sigma_\omega^2 \mathbf{I})$, first proposed by MacKay [52]:

$$p(\mathbf{w}) = (2\pi\sigma_\omega^2)^{-\frac{M}{2}} \exp\{-\frac{1}{2\sigma_\omega^2}\mathbf{w}^\top \mathbf{w}\} \tag{1.14}$$

where σ_ω is the shared variance for all individual weights [12]. It was shown by Neal [56] that in the regression setting, the isotropic Gaussian prior for a BNN with a single hidden layer approaches a Gaussian process prior [13] as the number of hidden units tend to infinity, so long as the activation function chosen is bounded. This is an important theoretical connection since Gaussian processes are well-understood Bayesian models for regression. For all BNNs considered in this thesis, we will use the Gaussian radial basis function (RBF) activation

$$\sigma_{RBF}(x) = e^{-x^2} \tag{1.15}$$

which is bounded in $(0, 1]$. In Section 1.5 we will motivate more complicated priors for BNNs, including the constrained prior discussed in Chapter 2.

1.4.3 Posterior Inference and Prediction

As exact posterior inference via (1.8) is intractable, we instead rely on approximate inference algorithms, which can be grouped into a few classes based on their method of approximation. Different classes of algorithms also lead to different ways of doing prediction. Below, we present the two main classes of inference algorithms and explain how prediction can be carried out using the results of inference. We will also introduce the specific algorithms used in the experiments in this thesis; their exact details can be found in Appendix B. Figure 1.4.1 shows an example of the posterior predictive distribution that a BNN yields, in comparison to classical NNs.

[12] One may recall Bayesian linear regression, where the posterior MAP estimate from specifying an isotropic Gaussian prior is equivalent to L_2-regularization in the frequentist setting.

[13] A Gaussian process is a set S of random variables such that every finite subset of S follows a multivariate Gaussian distribution. Hence a Gaussian process may be considered as a distribution over real-valued functions. The Gaussian process prior is the distribution over functions prior to observation of data, i.e. prior to instantiation of any r.v. in S.

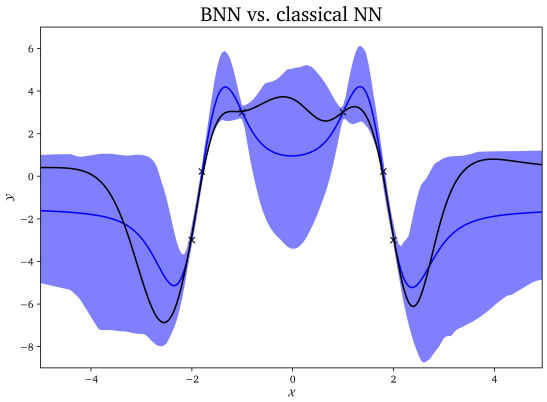


Figure 1.4.1: An example of the BNN posterior predictive distribution (blue) as well as a classical NN (black line) trained on the same dataset. We show the posterior predictive mean function (blue line) as well as the empirical at credible interval $\sigma = 3$ (blue shading) Whereas the classical NN simply fits all data points, the BNN gives us variance estimates that increase when further away from D_{tr}, representing greater predictive uncertainty. The prior used for the BNN is (1.14) and HMC was used for inference.

MARKOV CHAIN MONTE CARLO (MCMC) MCMC algorithms rely on sampling from a Markov chain whose equilibrium distribution is the posterior. In the context of BNNs, our Markov chain is a sequence of random parameters $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \ldots$ defined over $\mathcal{W}$, where the transition probability has the **Markov property**:

$$p(\mathbf{W}^{(n+1)}|\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(n)}) = p(\mathbf{W}^{(n+1)}|\mathbf{W}^{(n)}) \tag{1.16}$$

The Markov chain is (i) *positive-recurrent* if the expected amount of time to return to $\mathbf{w}$ given $\mathbf{W}^{(1)} = \mathbf{w}$ is finite for all states $\mathbf{w} \in \mathcal{W}$, (ii) *irreducible* if it is possible to get to any state from any other state in $\mathcal{W}$ and (iii) *aperiodic* if for every state $\mathbf{w}$, $\gcd\{n \,|\, p(\mathbf{W}^{(n)} = \mathbf{w}|\mathbf{W}^{(1)} = \mathbf{w}) > 0, n > 0\} = 1$. An irreducible, aperiodic and positive-recurrent chain has an *equilibrium distribution*, which is the limiting distribution of states $\mathbf{w} \in \mathcal{W}$ visited. The goal of MCMC is therefore to construct a Markov chain (by defining the transition kernel) whose equilibrium distribution is the posterior, such that a (finite) set of samples from this chain is approximately distributed according to the posterior [14]. MCMC algorithms are well-studied in statistical literature and texts such as [7] provide more details.

[14] As the number of samples tend to infinity, the empirical distribution of samples is closer to the posterior.

In this thesis, we use **Hamiltonian Monte Carlo (HMC)**, an MCMC variant that employs Hamiltonian dynamics to generate proposals on top of a Metropolis-Hastings framework [13, 57]. HMC is thus often seen as the *gold standard* for approximate BNN inference. This is because (i) MCMC algorithms sample from the true posterior (which makes them more accurate than VI approximations, discussed below) and (ii) HMC is the canonical MCMC algorithm for BNN inference [15]. However, as HMC is inefficient and cannot scale with large or high-dimensional datasets [16], it is generally reserved for low-dimensional synthetic examples.

Since MCMC algorithms return a finite set of samples $\{\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(S)}\}$, prediction using (1.11) exactly is impossible. Instead, we compute the Monte Carlo estimator:

$$p(Y'|\mathbf{x}, D_{tr}) \approx \frac{1}{S} \sum_{i=1}^{S} p(Y'|\mathbf{x}', \mathbf{w}^{(i)}) \tag{1.17}$$

We can construct credible intervals for any given $\mathbf{x}'$ using the empirical quantiles of $\{\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(S)}\}$, which allows us to quantify how confident the BNN is at $\mathbf{x}'$.

VARIATIONAL INFERENCE (VI) Variational learning for NNs [25, 33] approximates the true posterior $p(\mathbf{w}|D_{tr})$ with a **variational distribution** $q_{\boldsymbol{\theta}}(\mathbf{w})$, which has the same support $\mathcal{W}$ and is parametrized by $\boldsymbol{\theta} \in \Theta$. The variational family $\mathbf{Q} = \{q_{\boldsymbol{\theta}} | \boldsymbol{\theta} \in \Theta\}$ is typically chosen such that analytical expressions for $q_{\boldsymbol{\theta}}$ exist and so $q_{\boldsymbol{\theta}}$ is a viable proxy for sampling and prediction. [17] To find the value of $\boldsymbol{\theta}$ such that $q_{\boldsymbol{\theta}}(\mathbf{w})$ is as similar as possible to $p(\mathbf{w}|D_{tr})$, we minimize the **Kullback-Leibler (KL) divergence** [42]:

$$D_{KL}\big(p(\cdot|D_{tr}) \,||\, q_{\boldsymbol{\theta}}(\cdot)\big) = \mathbb{E}_{\mathbf{W} \sim q_{\boldsymbol{\theta}}}\Big[\log \frac{q_{\boldsymbol{\theta}}(\mathbf{W})}{p(\mathbf{W}|D_{tr})}\Big] \tag{1.18}$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \Theta} D_{KL}\big(p(\cdot|D_{tr}) \,||\, q_{\boldsymbol{\theta}}(\cdot)\big) \tag{1.19}$$

KL divergence has information-theoretic roots and can be thought of as a measure of similarity between probability distributions. In variational literature, minimizing KL divergence is equivalent to maximizing a quantity known as the **evidence lower**

[15] MCMC methods "simpler" than HMC, such as naive Metropolis-Hastings or Gibbs sampling, are intractable on high-dimensional parametric models such as BNNs. As such, HMC is in some ways the simplest algorithm used for approximate BNN inference.

[16] This is because the likelihood term cannot be batched; see Appendix B.1.

[17] In this way, MCMC can be seen as the approximation of the exact posterior whereas VI is an exact solution of an approximate posterior.

bound (ELBO):

$$\mathcal{L}_{ELBO}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[\log p(D_{tr}|\mathbf{W})\Big] + D_{KL}\big(p(\cdot)\,||\,q_{\boldsymbol{\theta}}(\cdot)\big) \tag{1.20}$$

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\Theta} \mathcal{L}_{ELBO}(\boldsymbol{\theta}) \tag{1.21}$$

We typically refer to (1.21) instead of (1.19) in VI literature. Appendix A provides a derivation of (1.21) from (1.19). Solving for $\boldsymbol{\theta}^*$ exactly is intractable due to the expectation term, and estimators for the integral in (1.20) are necessary. Once a tractable proxy is formulated, standard algorithms such as SGD can be used for the actual optimization process.

Different VI algorithms construct different estimators for optimization and consider different variational families. VI can be thought of as simplifying the process of marginalization (c.f. (1.8)) by optimization. While this seemingly brings us back to the classical framework, note that we still preserve the Bayesian advantage since we are optimizing for a *distribution* rather than a parameter.

In this thesis, we use **Bayes by Backprop (BBB)**, a VI algorithm that makes use of so-called *reparametrization trick* [40] to compute a Monte Carlo estimator of (1.20), where the variational family is Gaussian [5]. Unlike HMC, BBB is scalable and fast and therefore can be applied to high-dimensional and large datasets in real-life applications. It is the most commonly used variational approach [70]. However, as discussed in Chapter 3, VI methods like BBB can yield problematic approximations and part of this thesis deals with characterizing the quality of such approximations.

We will also use **Stein Variational Gradient Descent (SVGD)**, a VI algorithm that relies on applying successive transforms to an initial tractable distribution, in a way that incrementally minimizes the KL divergence between the transformed distributions and $p(\mathbf{w}|D_{tr})$ [47]. SVGD is also scalable to high-dimensional, large datasets. Furthermore, as the transforms that SVGD applies implicitly define a richer variational family than Gaussian approximations, SVGD tends to learn better quality posteriors than BBB.

Similar to MCMC methods, SVGD returns a set of optimized samples and prediction can be done using (1.17). In contrast, as BBB optimizes a simpler variational family, we have access to the analytical form of the final Gaussian approximation. Nevertheless, due to the nonlinearity of $\Phi_{\mathbf{W}}$, closed-form evaluation of (1.11) for prediction is still difficult (even with a Gaussian posterior). In the interest of directly

comparing all posterior inference methods, we typically sample S times from $q_{\boldsymbol{\theta}}$ and compute (1.17) when doing prediction using BBB [18].

1.5 Surveying the Current Scene

BNNs are not yet fully compelling as a widespread approach to supervised learning. Below, we summarize the major areas of shortcoming and the lines of research addressing them. At the same time, we will also motivate the research discussed in Chapters 2 and 3.

1.5.1 Richer Priors

The difficulty in specifying priors with richer (more expressive) functional forms is a challenge in BNN research. The isotropic Gaussian prior (1.14) is a dissatisfactory prior that does not fully reflect the beliefs that a human user or expert might possess about the underlying function to be learned. To be clear, (1.14) is *not* a poor choice of prior because it is vague and weakly-informative. As noted in Wilson [74], a vague prior over parameter space still leads to a structured prior over function space due to the structured functional form $\Phi_{\mathbf{W}}$ of the NN. Instead, the fundamental difficulty is translating our prior beliefs, usually expressed in function space $(\mathcal{X}, \mathcal{Y})$, into the matching distribution in parameter space $\mathcal{W}$. In other words, the distribution over functions that (1.14) induces does not correspond to the complicated distributions that human experts might express.

Much work has been done on specifying better priors by making smarter assumptions in parameter space. For example, [49] exploits the correlations between individual BNN weights (e.g. intra-layer correlations) instead of treating them as independent like (1.14) does. [23] applies a horseshoe prior that zeroes nodes that do not help explain the data D_{tr} when the BNN is over-specified.

A recent strain of research aims to specify BNN priors directly in function space. [28] introduces a Gaussian prior in function space for the purpose of detecting out-of-distribution data. One approach to specifying functional priors is to consider stochastic processes, which are functional, nonparametric analogues to the parametric BNNs. To this end, [70] uses a variational approach that defines the ELBO

[18] If $q_{\boldsymbol{\theta}}$ is Gaussian, Equation (5.172) in Bishop [3] provides a stronger analytical approximation of (1.11) than the Monte Carlo estimation in (1.17).

directly over stochastic processes. Also, [51] performs inference over a family of stochastic processes, parametrized in such a way that allows for functional priors yet enjoys the speed and flexibility of BNNs.

HOW THIS THESIS CONTRIBUTES The research discussed in Chapter 2 contributes to this latter area of specifying functional priors. We introduce **output-constrained priors**, which are specified in parameter space but can incorporate functional assumptions in the form of *output constraints* (i.e. for some $\mathbf{x}$, we specify a prior on $Y|\mathbf{x}$). This directly allows human users to encode their rich functional beliefs in an intuitive manner while still allowing for the simplicity of parameter space inference. We demonstrate real-life applications on high-stakes domains like clinical action prediction and criminal justice.

1.5.2 BETTER INFERENCE

The intractability of exact inference is the other major challenge to working with BNNs, and several strands of current research are dedicated to designing tractable but accurate approximate inference algorithms. There are two measures by which we evaluate these algorithms. The first is the *quality of approximation*, i.e. we want to learn approximate distributions as close to the true posterior as possible. The other is *scalability*, i.e. the algorithm should be time-efficient as BNNs are normally applied to high-dimensional, large datasets. MCMC approximations are typically seen as producing "higher-quality" approximations (since they are correct to the limit of sampling) but VI approaches are generally more scalable.

Some key pieces of research on scalable inference include [73] and [8], which are both MCMC approaches that are scalable since they can compute batched updates of the likelihood. [18, 35] considers a Bayesian interpretation of dropout regularization, which is widely used on classical BNNs. The advantage of this method is that dropout can be easily and cheaply implemented, even for complex NN architecture such as convolutional or recurrent NNs.

Research towards improving the quality of VI includes normalizing flows [41, 50, 62], which iteratively transform an initial, simple distribution into one that is similar to the posterior. This is a VI approach with a richer variational family implicitly defined by the transformations. SVGD, introduced earlier, is closely related to normalizing flows as both algorithms use variable transforms. [32] describes another variational approach via probabilistic backpropagation of NN weights.

A concern that has gained weight in recent years is how these various algorithms are evaluated experimentally. While scalability can be easily measured with metrics like runtime, characterizing the quality of posterior approximation is much more poorly defined. For example, should we look at over- or under-estimation of posterior predictive variance? Should we measure the accuracy of the posterior predictive mean? Why is KL divergence (and not other divergence metrics) the correct objective to minimize? Given that we do not have access to an analytic form of the posterior, is it even possible to fully characterize the quality of approximation?

This concern has been exacerbated by recent research such as [77] and [15], which have theoretically and empirically shown that variational learning can be pathological, including the miscalibration of posterior variance in interpolated input regions. While it is generally well-known that VI methods tend to underestimate the posterior variance, these results are concerning since VI methods are widely used, and it is important that we can trust the posterior approximation. This is especially pertinent for high-stakes domains where we desire more rigorous measures of uncertainty.

HOW THIS THESIS CONTRIBUTES Due to the rich representation power of neural networks and the complex, multimodal posteriors that result, there are few provable guarantees on the performance of VI approximations, especially in the predictive setting. Inspired by recent efforts such as [36] and [78], the research in Chapter 3 proposes a theoretical bound on the predictive mean error of the VI approximation that is easily estimable.

2

Output-Constrained BNN

The ability to specify informative prior distributions that encode rich functional assumptions contributes significantly to the quality of the resulting BNN posterior [74]. Such priors accurately reflect the various complicated beliefs that human experts hold about the task at hand, especially for high-dimensional real-life applications. The effects of such a targeted prior would be most keenly felt by reducing the bias and uncertainty of the posterior predictive distribution in regions of $\mathcal{X}$ sparse in training data. As we have previously motivated, predictive distributions that are accurate everywhere in $\mathcal{X}$ are vital for high-stakes domains.

The range of functional assumptions that we might wish to impose on the prior is diverse. For example, one might possess *a priori* knowledge that any correct function must be monotonic or Lipschitz continuous. In this thesis, we tackle functional assumptions in the form of *output constraints* — the set of values y is constrained to hold for any given $\mathbf{x}$. Output constraints exist for many real-life applications and are a form of knowledge about the task that domain experts can easily specify. In particular, we can use output constraints to impose additional desiderata like safety or fairness on the BNN. For example, one can specify the safety constraint that a patient with symptoms $\mathbf{x}$ should not be prescribed drug y.

Incorporating output constraints as a BNN prior presents a number of challenges. We first need to formalize a system for specifying constraints. In particular, as BNNs learn probabilistically, we must translate deterministic constraints into equivalent

probabilistic formulations. Next, BNN inference algorithms operate in parameter space $\mathcal{W}$. However, output constraints are, by definition, specified in input-output space $(\mathcal{X}, \mathcal{Y})$. In line with recent research on function-space inference [28, 48, 51], we propose a sampling-based approach that assigns probability mass to $\mathbf{w}$ based on how well $\Phi_{\mathbf{w}}$ obeys constraints on drawn samples. We describe two ways of constructing such a prior; the resulting model is an **Output-Constrained BNN** (OC-BNN). Our approach is highly interpretable since an OC-BNN user can specify any output constraint directly in its functional form. The other major benefit is that OC-BNNs are amenable to all existing black-box BNN inference algorithms as we are constructing an evaluable prior function that can be substituted into the expression $p(\mathbf{w})$. Sections 2.1 to 2.3 will introduce the core theory, and Section 2.4 compares OC-BNNs to existing literature on functional BNN priors.

Synthetic experiments in low dimensions are introduced in Section 2.5, which will provide a proof-of-concept using visualizable posterior predictive distributions and various ablation tests. In Sections 2.6 and 2.7, we demonstrate OC-BNNs on two high-stakes, high-dimensional domains: **clinical action prediction** and **recidivism prediction**. We show that the ability to incorporate constraints with OC-BNNs lead to safer and fairer machine learning.

Parts of the work introduced in this chapter has been previously published as [75]. This thesis introduces a novel mathematical framework that describes more formally the concepts in [75] and expands on the analysis of experiments carried out in [75]. The variational formulation of the output-constrained prior as well as the high-dimensional application to recidivism prediction are also novel. The code for the work in this chapter is available at `https://github.com/dtak/ocbnn-public`.

2.1 Output Constraints

Constraints are primarily studied under constrained optimization, where the goal is to optimize a function f over $\mathbf{x} \in \mathcal{X}$, subject to a set of constraints $g_1, \ldots, g_m$. These can be classified into one of two types:

- An **equality constraint** is a relationship $g(\mathbf{x}) = 0$ that must hold true for the optimization solution.
- An **inequality constraint** is a relationship $g(\mathbf{x}) \geq 0$ that must hold true for the optimization solution.

In the setting of supervised learning with BNNs, we are primarily interested in constraining the output y for any set of inputs $\mathbf{x}$. Constraints $g(\mathbf{x}, y)$ are therefore defined jointly over the input-output space $(\mathcal{X}, \mathcal{Y})$. In the discriminative setting where we learn the conditional $Y|\mathbf{x}$ (and $\mathbf{x}$ is typically an independent variable), we are seldom interested in modeling the constraints *between* individual input variables. As such, $g(\mathbf{x}, y)$ can be expressed equivalently as the relationship between y and some $g(\mathbf{x})$. We will term this form of constraint as an *output constraint*.

Next, inequality constraints are not well-defined for classification as defining a total order does not make sense for most discrete $\mathcal{Y}$. To generalize the notions of "equality" and "inequality" for supervised learning, we instead consider *positive* constraints, which specify the set of values that y can take, and *negative* constraints, which specify what values y cannot take.

Lastly, an important distinction from the optimization setting is that, here, $\mathbf{x} \in \mathcal{X}$ itself is not being constrained. Rather, it shapes what the constraint on y is via the function g. In fact, $\mathbf{x}$ also serves another purpose in determining where in $\mathcal{X}$ the constraint should be obeyed. Putting these considerations together, we formally introduce:

Definition 2.1.1. *A **deterministic output constraint** $\mathcal{C}$ on y is a tuple $(\mathcal{C}_\mathbf{x}, \mathcal{C}_y, \circ)$ where $\mathcal{C}_\mathbf{x} \subseteq \mathcal{X}$, $\mathcal{C}_y : \mathcal{C}_\mathbf{x} \to 2^\mathcal{Y}$ and $\circ \in \{\in, \notin\}$. $\mathcal{C}$ is **satisfied** iff $\forall\, \mathbf{x} \in \mathcal{C}_\mathbf{x}$, $y \circ \mathcal{C}_y(\mathbf{x})$* [1].

*$\mathcal{C}^+ := \mathcal{C}$ is a **positive constraint** if $\circ$ is $\in$. $\mathcal{C}^- := \mathcal{C}$ is a **negative constraint** if $\circ$ is $\notin$. $\mathcal{C}$ is a **global constraint** if $\mathcal{C}_\mathbf{x} = \mathcal{X}$. $\mathcal{C}$ is a **local constraint** if $\mathcal{C}_\mathbf{x} \subset \mathcal{X}$.*

Positive and negative constraints can be loosely viewed as analogues to equality and inequality constraints respectively. For example, the inequality $y \geq 1$ is equivalent to $y \notin (-\infty, 1)$. Even though $y \in \mathcal{C}_y(\mathbf{x}) \Leftrightarrow y \notin \mathcal{Y} - \mathcal{C}_y(\mathbf{x})$, the distinction between positive and negative constraints is not trivial because the user typically has access to only one of the two forms of knowledge. Positive constraints also tend to be more informative than negative constraints. For example, in regression, prior knowledge that $y = k$ for a particular value k (or some small interval around k) is more likely than the knowledge that $y \neq k$.

As our goal is not optimization over $y \in \mathcal{Y}$, Definition 2.1.1 is not sufficient and we must define what it means for a BNN to satisfy a constraint $\mathcal{C}$. Since BNNs express

[1] Note the explicit dependence of $\mathcal{C}_y$ on $\mathbf{x}$, i.e. $\mathcal{C}_y$ replaces the constraint function g we have been using up until now.

predictive distributions over $Y|\mathbf{x}$, a natural approach is to determine the probability mass of the prior predictive that satisfies a constraint. The **prior predictive** is simply the analogue of (1.11) that replaces $\mathbf{W}|D_{tr}$ with $\mathbf{W}$:

$$p(Y'|\mathbf{x}') = \int_{\mathcal{W}} p(Y'|\mathbf{x}', \mathbf{w}) p(\mathbf{w}) \, \mathrm{d}\mathbf{w} \tag{2.1}$$

Definition 2.1.2. *A deterministic output constraint $\mathcal{C} = (\mathcal{C}_{\mathbf{x}}, \mathcal{C}_y, \circ)$ is ϵ-**satisfied** by a BNN with prior $p(\mathbf{w})$ if $\forall \, \mathbf{x} \in \mathcal{C}_{\mathbf{x}}$ and for some $\epsilon \in [0, 1]$:*

$$p(Y \circ \mathcal{C}_y(\mathbf{x})|\mathbf{x}) = \int_{\mathcal{W}} p(Y = y|\mathbf{x}, \mathbf{w}) \, \mathbb{I}[y \circ \mathcal{C}_y(\mathbf{x})] \, p(\mathbf{w}) \, d\mathbf{w} \quad \leq \epsilon \tag{2.2}$$

The strictness parameter ϵ can be related to *hard* and *soft* constraints in optimization literature; the constraint is hard iff $\epsilon = 0$. Note that while ϵ-satisfaction is useful for formalizing the goal of constrained BNN inference, ϵ is not an parameter of our prior. Instead, we use (2.2) largely as an empirical measure of success on D_{te}.

Since BNNs operate in the probabilistic setting and learn a distribution Y instead of a value y, we can in fact generalize Definition 2.1.1 further. Instead of a subset $\mathcal{C}_y(\mathbf{x}) \subseteq \mathcal{Y}$ indicating (non)-membership of y, we can specify a constraint directly as some distribution over Y:

Definition 2.1.3. *A **probabilistic output constraint** $\mathcal{C}$ on y is a tuple $(\mathcal{C}_{\mathbf{x}}, \mathcal{D}_y, \circ)$ where $\mathcal{C}_{\mathbf{x}} \subseteq \mathcal{X}$, $\mathcal{D}_y(\mathbf{x})$ is a distribution over $\mathcal{Y}$ (dependent on $\mathbf{x} \in \mathcal{C}_{\mathbf{x}}$) and $\circ \in \{\in, \notin\}$. $\mathcal{C}$ is ϵ-**satisfied** by a BNN with prior $p(\mathbf{w})$ if $\forall \, \mathbf{x} \in \mathcal{C}_{\mathbf{x}}$ and for some $\epsilon \in [0, 1]$:*

$$D_{DIV}\Big(p(Y|\mathbf{x}) \, , \, \mathcal{D}_y(\mathbf{x})\Big) \quad \leq \epsilon \tag{2.3}$$

where D_{DIV} is any valid measure of divergence between two distributions over $\mathcal{Y}$. [2]

The type of constraint we choose to specify is application-specific. For example, for classification tasks where we might have a prior belief over the individual likelihoods of each class, probabilistic constraints (e.g. a categorical distribution) may be more useful than their deterministic counterparts. Together, Definitions 2.1.1 and 2.1.3 represent a general formulation that allows for the incorporation of versatile and interpretable functional beliefs on Y.

[2] The same definitions of positive, negative, global and local constraints on Definition 2.1.1 apply here as well.

We now seek to construct a prior on $\mathcal{W}$ such that the BNN ϵ-satisfies, for some small ϵ, a set of deterministic or probabilistic constraints $\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(m)}$ that we specify. Note that satisfying some $\mathcal{C}$ on the prior predictive does not necessarily guarantee that $\mathcal{C}$ is still satisfied on the *posterior* predictive, which infers from both $\mathcal{C}$ and D_{tr}. Most of the time, we might assume that D_{tr} is consistent with $\mathcal{C}$, which implies ϵ-satisfaction on the posterior predictive distribution. However, there are certain scenarios where we might to specify $\mathcal{C}$ to correct a possibly undesirable or biased D_{tr}. Ablation experiments in Section 2.4 and the application to criminal justice in Section 2.7 are examples.

Since the desired prior is to be expressed in parameter space, a natural way to connect it to some $\mathcal{C}$ (defined in input-output space) is to evaluate how well the implicit distribution of $\Phi_{\mathbf{W}}$, induced by the distribution of $\mathbf{W}$, satisfies $\mathcal{C}$. We will present two prior formulations over $\mathcal{W}$ that do so — (i) by conditioning the prior on $\mathcal{C}$ and explicitly factoring in the likelihood $p(\mathcal{C}|\mathbf{W})$ of constraint satisfaction, and (ii) by learning a variational approximation $q(\mathbf{W})$ via optimization of objective (2.2) directly.

2.2 Conditional Output-Constrained Prior

A fully Bayesian approach of conditioning requires us to construct a probability distribution that concretely describes the extent to which $\mathbf{W}$ (and $\Phi_{\mathbf{W}}$) satisfies $\mathcal{C}$. To do so, we must develop a framework that describes $\mathcal{C}$ in probabilistic terms, i.e. to define terms like $p(\mathcal{C})$ and $p(\mathcal{C}|\mathbf{w})$. We will do so by defining $\mathcal{C}$ as a stochastic process indexed on $\mathcal{C}_{\mathbf{x}}$. We begin by establishing the measure-theoretic formalization of the BNN setup introduced in Chapter 1.

Let $\mathbf{x} \in \mathcal{C}_x$ be any fixed input variable. A random variable $Z : \Omega \to \mathcal{Y}$ is defined on the probability space $(\Omega, \mathcal{F}, \mathbf{P})$ for the measurable space $(\mathcal{Y}, \Sigma_{\mathcal{Y}})$. Ω is the sample space such that for any $Z(\omega) = z$, ω is interpreted as the event where the output for $\mathbf{x}$ is z. $\mathcal{F}$ is the typical σ-algebra we consider, i.e. the Borel σ-algebra on $\mathbb{R}$ for regression or the power set $2^{\mathcal{Y}}$ for classification. $\mathbf{P}$ is any valid probability measure. For regression where $\mathcal{Y} = \mathbb{R}$, $\Sigma_{\mathcal{Y}}$ is simply the Lebesgue measure on $\mathbb{R}$. For classification, $\Sigma_{\mathcal{Y}} = 2^{\mathcal{Y}}$. [3]

[3] While not defined earlier, $\Omega, \mathcal{B}_{\Omega}, \Sigma_{\mathcal{Y}}$ are identical to the setting for supervised learning with target variable Y that we set up in Chapter 1. We simply describe the measure-theoretic assumptions explicitly here.

Let $S_{\mathbf{x}} \subseteq \mathcal{X}$ be any set of inputs. Let $(\mathbf{\Omega}, \mathbf{F}, \mathbf{P}_g)$ where $\mathbf{\Omega} = \prod_{\mathbf{x} \in S_{\mathbf{x}}} \Omega$ is the Cartesian product of the sample space Ω (defined above) on every $\mathbf{x} \in S_{\mathbf{x}}$, $\mathbf{F}$ is the product σ-algebra and $\mathbf{P}_g = \prod_{\mathbf{x} \in S_{\mathbf{x}}} \mathbf{P}_{g,\mathbf{x}}$ is the product measure such that each $\mathbf{P}_{g,\mathbf{x}}$ corresponds to some probability distribution $p_g(\cdot|\mathbf{x})$ over $\mathcal{Y}$ (a pushforward measure from $\mathbf{P}_{g,\mathbf{x}}$). Then

$$\mathcal{CP} : \mathbf{\Omega} \to \mathcal{Y}^{S_{\mathbf{x}}} \tag{2.4}$$

is the stochastic process indexed by $S_{\mathbf{x}}$, where $\mathcal{Y}^{S_{\mathbf{x}}}$ denotes the set of all measurable functions from $S_{\mathbf{x}}$ into $\mathcal{Y}$. The law of the process $\mathcal{CP}$ is

$$p(S) = \mathbf{P}_g \circ \mathcal{CP}^{-1}(S) \quad \text{for any } S \in \mathcal{Y}^{S_{\mathbf{x}}} \tag{2.5}$$

such that for any finite subset $\{Z_1, \ldots, Z_T\} \subseteq \mathcal{CP}$ indexed by $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}\} \subseteq S_{\mathbf{x}}$, we have

$$p(Z_1, \ldots, Z_T) = \prod_{t=1}^{T} p_g(Z_t = z_t|\mathbf{x}_t) \tag{2.6}$$

Note that $\mathcal{CP}$ is a valid stochastic process as it satisfies both (finite) exchangeability and consistency, which are sufficient conditions via the Kolmogorov Extension Theorem [58]. As the BNN output $\Phi_{\mathbf{w}}(\mathbf{x})$ can be evaluated for all $\mathbf{x} \in \mathcal{C}_{\mathbf{x}}$, the stochastic process indexed by choosing $S_{\mathbf{x}} = \mathcal{C}_{\mathbf{x}}$ allows us to formally describe how much $\Phi_{\mathbf{W}}$ satisfies $\mathcal{C}$ — by determining the measure of $\Phi_{\mathbf{w}}^{\mathcal{C}_{\mathbf{x}}} \in \mathcal{Y}^{\mathcal{C}_{\mathbf{x}}}$. Since $\mathcal{C}_{\mathbf{x}}$ can be uncountable, for practical computation we will choose a finite subset of $\mathcal{C}_{\mathbf{x}}$ and compute (2.6) as an approximation. We are now ready to define the output-constrained prior:

Definition 2.2.1. *Let $\mathcal{C}$ be any (deterministic or probabilistic) constraint and $\mathcal{CP}$ the the stochastic process defined on $\mathcal{Y}^{\mathcal{C}_{\mathbf{x}}}$ according to (2.4) for some probability measure $\mathbf{P}_g$. The **conditional output-constrained prior (COCP)** on $\mathbf{W}$ is defined as*

$$p_{\mathcal{C}}(\mathbf{w}) = p_f(\mathbf{w}) p(\Phi_{\mathbf{w}}^{\mathcal{C}_{\mathbf{x}}}) \tag{2.7}$$

where $p_f(\mathbf{W})$ is any distribution on $\mathbf{W}$ that is independent of $\mathcal{C}$, and $\Phi_{\mathbf{w}}^{\mathcal{C}_{\mathbf{x}}} \in \mathcal{Y}^{\mathcal{C}_{\mathbf{x}}}$ is the realization of $\mathcal{CP}$ where $\{\mathcal{CP}(\mathbf{x}) = \Phi_{\mathbf{w}}(\mathbf{x}) : \mathbf{x} \in \mathcal{C}_{\mathbf{x}}\}$.

(2.7) can loosely be viewed as an application of Bayes' Rule, where the realization $\Phi_{\mathbf{w}}^{\mathcal{C}_{\mathbf{x}}}$ of the stochastic process $\mathcal{CP}$ is the evidence of $\mathcal{C}$ being satisfied by $\mathbf{w}$. Hence $p(\Phi_{\mathbf{w}}^{\mathcal{C}_{\mathbf{x}}})$ can be interpreted as a likelihood term, $p_f(\mathbf{w})$ is the "true" prior on $\mathbf{W}$ and

$p_{\mathcal{C}}(\mathbf{w})$ is a "posterior" distribution $\mathbf{W}|\mathcal{CP}$ [4]. This implies that the posterior that we are inferring is, instead, a distribution over $\mathbf{W}|D_{tr}, \mathcal{CP}$:

$$\begin{aligned} p_{\mathcal{C}}(\mathbf{w}|D_{tr}) := p(\mathbf{w}|D_{tr}, \mathcal{CP}) &= \frac{p(\mathbf{w}|\mathcal{CP})p(D_{tr}|\mathbf{w}, \mathcal{CP})}{p(D_{tr}|\mathcal{CP})} \\ &= \frac{p(\mathbf{w}|\mathcal{CP})p(D_{tr}|\mathbf{w})}{p(D_{tr})} \end{aligned} \tag{2.8}$$

Note that $p(D_{tr}|\mathbf{w}) = p(D_{tr}|\mathbf{w}, \mathcal{CP})$ and $p(D_{tr}|\mathcal{CP}) = p(D_{tr})$ since D_{tr} is independent of $\mathcal{C}$. To ease the burden on notation, we will drop the explicit conditioning on $\mathcal{CP}$ and treat $p_{\mathcal{C}}(\mathbf{w})$ as the unconditional prior over $\mathbf{W}$, consistent with our setup in Chapter 1. We also denote the constrained posterior as $p_{\mathcal{C}}(\mathbf{w}|D_{tr})$.

Definition 2.2.1 generalizes to multiple constraints $\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(m)}$, by conditioning on all $\mathcal{C}^{(i)}$ and evaluating $\Phi_{\mathbf{w}}$ for each constraint. Assuming the constraints to be mutually independent, (2.7) simply generalizes to:

$$p_{\mathcal{C}^{(1)},\ldots,\mathcal{C}^{(m)}}(\mathbf{w}) = p_f(\mathbf{w}) \prod_{i=1}^{m} p(\Phi_{\mathbf{w}}^{\mathcal{C}_{\mathbf{x}}^{(i)}}) \tag{2.9}$$

It remains to describe what measure $\mathbf{P}_g$ (or, equivalently, what distribution p_g over $\mathcal{CP}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{C}_x$) we should choose such that $p(\Phi_{\mathbf{w}}^{\mathcal{C}_{\mathbf{x}}})$ is a likelihood distribution that faithfully evaluates the extent to which $\mathcal{C}$ is satisfied. For probabilistic output constraints, we simply set $p_g(\cdot|\mathbf{x})$ to be $\mathcal{D}_y(\mathbf{x})$. For deterministic output constraints, we propose a number of distributions over $\mathcal{Y}$ that corresponds well to $\mathcal{C}_y$, the set of permitted (or excluded) output values.

POSITIVE GAUSSIAN CONSTRAINT A positive constraint $\mathcal{C}^+$ for regression, where $\mathcal{C}_y(\mathbf{x}) = \{y^*\}$ is a singleton for all $\mathbf{x} \in \mathcal{C}_{\mathbf{x}}$, corresponds to the situation where we know the ground-truth values over the constrained input region $\mathcal{C}_{\mathbf{x}}$. A natural choice is therefore a Gaussian distribution centered on these value:

$$\mathcal{CP}(\mathbf{x}) \sim \mathcal{N}(y^*, \sigma_{\mathcal{C}}^2) \tag{2.10}$$

where $\sigma_{\mathcal{C}}$ is a hyperparameter for the standard deviation of the Gaussian, corresponding to how strict we want $\mathcal{C}$ to be enforced.

[4] Technically, $p_{\mathcal{C}}(\mathbf{w})$ is the joint distribution. We have omitted the term $p(\Phi^{\mathcal{C}_{\mathbf{x}}}) = \int_{\mathcal{W}} p(\Phi_{\mathbf{w}}^{\mathcal{C}_{\mathbf{x}}}) p_f(\mathbf{w}) \mathrm{d}\mathbf{w}$, which, like (1.10), is unimportant (and intractable) since it is a constant for a fixed constraint $\mathcal{C}$.

We can also generalize this for the case where $\mathcal{C}_y = \{y_1, \ldots, y_K\}$ contains multiple values; the equivalent distribution is the Gaussian mixture model:

$$\mathcal{CP}(\mathbf{x}) \sim \sum_{k=1}^{K} \omega_k \mathcal{N}(y_k, \sigma_{\mathcal{C}}^2) \tag{2.11}$$

for some set of mixing weights ω_k such that $\sum_{k=1}^{K} \omega_k = 1$.

NEGATIVE EXPONENTIAL CONSTRAINT A negative constraint $\mathcal{C}^-$ for regression takes $\mathcal{C}_y(\mathbf{x})$ to be the set of values that cannot be the output of $\mathbf{x}$. Informally, we desire a distribution for $\mathcal{CP}(\mathbf{x})$ with a PDF that is 0 for every $y' \in \mathcal{C}_y(\mathbf{x})$ and some constant value otherwise. Unlike $\mathcal{C}^+$, there is no natural distribution corresponding to such a specification, which would be both improper (if $\mathcal{Y} - \mathcal{C}_y$ is unbounded) and discontinuous. For the case where $\mathcal{C}_y$ is determined from a set of inequalities of the form $\{g_1(\mathbf{x}, y) \leq 0, \ldots, g_l(\mathbf{x}, y) \leq 0\}$ [5], where *obeying* these inequalities implies that $y \in \mathcal{C}_y(\mathbf{x})$ and hence $\mathcal{C}^-$ is ***not satisfied***, we can artificially construct an exponential distribution that penalizes y based on how each inequality is violated:

$$p_g(\mathcal{CP}(\mathbf{x}) = y'|\mathbf{x}) = \exp\Big\{ -\gamma \cdot \prod_{i=1}^{l} \sigma_{\tau_0,\tau_1}(g_i(\mathbf{x}, y')) \Big\} \tag{2.12}$$

where $\sigma_{\tau_0,\tau_1}(\cdot)$ is the *constraint classifier* defined as:

$$\sigma_{\tau_0,\tau_1}(z) = \frac{1}{4}(\tanh(-\tau_0 z) + 1)(\tanh(-\tau_1 z) + 1) \tag{2.13}$$

and γ is a scale hyperparameter. The sigmoidal function $\sigma_{\tau_0,\tau_1}(g_i(\mathbf{x}, y'))$ is close to 1 if $g_i(\mathbf{x}, y') \leq 0$, resulting in $p_g(\mathcal{CP}(\mathbf{x}) = y'|\mathbf{x})$ having a small value if every $g_i(\mathbf{x}, y') \leq 0$ (i.e. all inequalities obeyed and $\mathcal{C}^-$ is violated). γ controls the rate of decay. Here $\sigma_{\tau_0,\tau_1}(g_i(\mathbf{x}, y'))$ is a soft indicator of whether $g_i \leq 0$ is satisfied, and τ_0, τ_1 are both hyperparameters controlling the smoothness of σ_{τ_0,τ_1}. [6] Figure 2.2.1 shows $\sigma_{\tau_0,\tau_1}(z)$ for $\tau_0 = 15, \tau_1 = 2$, which are the values used in this thesis.

POSITIVE DIRICHLET CONSTRAINT For K-classification, the natural distribution to consider is the Dirichlet distribution, whose support is the K-tuple of prior proba-

[5] For example, $\mathcal{C}_y(\mathbf{x}) = (-\infty, 3) \cup (5, \infty)$ can be represented as $\{3 - y \leq 0, y - 5 \leq 0\}$.

[6] In general, softening constraints in this manner allows for more tractable posterior sampling than having sharp constraints. See, for example, [12], that proposes a soft exponential kernel on parameter-space constraints for Bayesian inference.

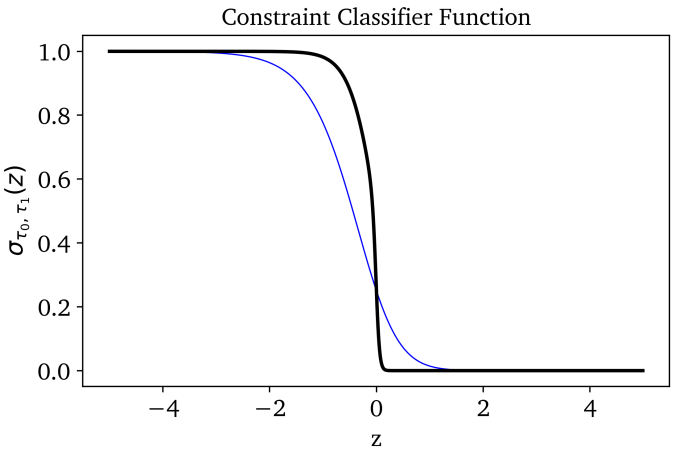


Figure 2.2.1: The graph of the constraint classifier function $\sigma_{\tau_0,\tau_1}(z)$ for $z = [-5, 5]$. The black line shows the setting where $\tau_0 = 15, \tau_1 = 2$ and the blue line shows the baseline setting where $\tau_0 = 1, \tau_1 = 1$. Our choice of hyperparameters is stricter than the naive $\tanh(-z)$ function and allows for a more abrupt transition between the inequality being obeyed ($z \leq 0$) vs violated ($z > 0$).

bilities on all classes [7]. For a positive constraint $\mathcal{C}^+$, we specify:

$$\mathcal{CP}(\mathbf{x}) \sim \mathcal{D}ir(\boldsymbol{\alpha}) \tag{2.14}$$

where

$$\boldsymbol{\alpha}_i = \begin{cases} c & \text{if } i \in \mathcal{C}_y(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \qquad \text{where } c > 0 \tag{2.15}$$

is the concentration parameter, a vector of length K. Using the Dirichlet distribution blurs the distinction between a positive and negative constraint, because we are specifying both $\mathcal{C}_y$ and $\mathcal{Y} - \mathcal{C}_y$ at the same time.

2.2.1 Inference with COCPs

A complete specification of (2.7) is sufficient for inference using COCPs, as the expression $p_{\mathcal{C}}(\mathbf{w})$ can simply be substituted for that of $p(\mathbf{w})$ in all black-box BNN inference algorithms. Since (2.7) cannot be computed exactly due to the intractability of $p(\Phi_{\mathbf{w}}^{\mathcal{C}_\mathbf{x}})$ for uncountable $\mathcal{C}_\mathbf{x}$, we will draw a finite sample $\{\mathbf{x}^{(t)}\}_{t=1}^{T}$ of $\mathbf{x} \in \mathcal{C}_\mathbf{x}$ (e.g. uniformly across $\mathcal{C}_\mathbf{x}$ or Brownian sampling if $\mathcal{C}_\mathbf{x}$ is unbounded) at the start of the

[7] Recall that for classification, $\Phi_\mathbf{w}$ is a vector of length K representing the K BNN output nodes.

inference process and compute (2.6) as an estimator of $p(\Phi_{\mathbf{w}}^{\mathcal{C}_{\mathbf{x}}})$ instead:

$$\tilde{p}_{\mathcal{C}}(\mathbf{w}) = p_f(\mathbf{w}) \prod_{t=1}^{T} p_g(\Phi_{\mathbf{w}}(\mathbf{x}^{(t)})|\mathbf{x}^{(t)}) \tag{2.16}$$

(2.16) will be computed every iteration of an MCMC algorithm, or every epoch of VI optimization. [8] The trade-off of using COCPs is that (2.16) is computationally more expensive than a naive prior like (1.14), and its computational runtime increases with the input dimensionality Q and the size of $\mathcal{C}_{\mathbf{x}}$. However, we note that many sampling techniques from statistical literature can be applied to COCPs in lieu of naive uniform sampling. We will also need to specify the $\mathcal{C}$-independent prior term $p_f(\mathbf{w})$. In this thesis, we will simply use the naive Gaussian prior (1.14).

Note that BNN inference algorithms can be applied for *prior inference* as well as posterior inference. In the former, we are inferring $p_{\mathcal{C}}(\mathbf{w})$ solely instead of $p_{\mathcal{C}}(\mathbf{w}|D_{tr})$, which we can use to visualize the prior predictive distribution. For the experiments in Section 2.5, we will demonstrate both the prior predictive and the posterior predictive distribution learnt using $p_{\mathcal{C}}(\mathbf{w})$.

2.3 Variational Output-Constrained Prior

In contrast to COCPs, a different approach for incorporating $\mathcal{C}$ into the prior can be motivated from VI. Instead of constructing a PDF over $\mathbf{w}$ with terms explicitly dependent on $\Phi_{\mathbf{w}}$ and $\mathcal{C}_y$, we can use a variational approximation $q_{\boldsymbol{\lambda}}(\mathbf{w})$ where we optimize $\boldsymbol{\lambda}$ with respect to our objectives, (2.2) and (2.3), for deterministic and probabilistic constraints respectively.

Both definitions of ϵ-satisfaction, (2.2) and (2.3), compute the prior predictive (2.1), which contains an expectation over $\mathbf{W}$. As such an integral is intractable, we will need to find a closed-form approximation for the prior predictive. For the regression and ***binary*** classification settings, there are reasonable and well-known approximations which we present below. A complete derivation, taken from [3], can be found in Appendix D.

[8] An interesting extension may be to consider stochastic subsampling of $\mathcal{C}_{\mathbf{x}}$ every iteration of (2.16), however, there may no longer be thereotical guarantees for inference, e.g. existence of MCMC equilibrium distribution.

APPROXIMATION PRIOR PREDICTIVE FOR REGRESSION Let $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ be the variational parameters for the mean and standard deviation of each $\mathbf{w}_i$. Then a Gaussian approximation of the prior predictive is:

$$\hat{Y}|\mathbf{x} \sim \mathcal{N}(\Phi_{\boldsymbol{\mu}}(\mathbf{x}), \sigma_\epsilon^2 + \mathbf{g}^\top(\boldsymbol{\sigma}^2 \cdot \mathbf{g})) \tag{2.17}$$

where

$$\mathbf{g} = \left[\nabla_{\mathbf{w}} \Phi_{\mathbf{w}}(\mathbf{x})\Big|_{\mathbf{w}=\boldsymbol{\mu}}\right] \tag{2.18}$$

and recall that σ_ϵ is the standard deviation of the output noise that we model.

APPROXIMATION PRIOR PREDICTIVE FOR BINARY CLASSIFICATION The case for binary classification is more involved. Before we present the approximation, we will need to make a slight modification to the BNN setup. Recall that a BNN for K-classification has K output nodes, over which a softmax function (1.7) is applied such that the output values sum to 1 (representing predicted class probabilities). Here, instead of using a BNN with 2 output nodes, we will use a BNN with a single output node, over which we apply the logistic sigmoid function [9]:

$$\sigma_L(x) = \frac{e^x}{e^x + 1} \tag{2.19}$$

The resulting value is then interpreted as the probability that the predicted output is 1 (the "positive" class):

$$\Phi_{\mathbf{w}}(\mathbf{x}) = p(Y = 1|\mathbf{x}) = \sigma_L(\phi_{\mathbf{w}}(\mathbf{x})) \tag{2.20}$$

where $\phi_{\mathbf{w}}(\mathbf{x})$ represents the output node's value *before* applying the sigmoid function. Note that $p(Y = 0|\mathbf{x}) = 1 - p(Y = 1|\mathbf{x})$. With this, the approximation for the prior predictive is given as:

$$p(\hat{Y} = 1|\mathbf{x}) = \sigma_L\left(\left(1 + \frac{\pi(\mathbf{g}^\top(\boldsymbol{\sigma}^2 \cdot \mathbf{g}))}{8}\right)^{-1/2} \mathbf{g}^\top \boldsymbol{\mu}\right) \tag{2.21}$$

where

$$\mathbf{g} = \left[\nabla_{\mathbf{w}} \phi_{\mathbf{w}}(\mathbf{x})\Big|_{\mathbf{w}=\boldsymbol{\mu}}\right] \tag{2.22}$$

Note that the first-order derivative $\mathbf{g}$ here is taken w.r.t. $\phi(\mathbf{x})$, not $\Phi(\mathbf{x})$.

[9] The logistic sigmoid function derives its name from logistic regression, a simpler statistical model used for classification. The softmax function can be seen as a generalization of the logistic sigmoid function for $K > 2$.

Optimization of $\boldsymbol{\lambda}$ can now be carried out by maximizing the probability mass of (2.17) or (2.21) that satisfies some constraint $\mathcal{C}$. The respective corresponding objectives are:

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}\in\Lambda} \int_{\mathcal{Y}} \mathbb{I}[y \circ \mathcal{C}_y] \cdot p(\hat{Y} = y|\mathbf{x})\, \mathrm{d}y \tag{2.23}$$

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}\in\Lambda} D_{DIV}\Big(p(\hat{Y}|\mathbf{x})\,,\, \mathcal{D}_y(\mathbf{x})\Big) \tag{2.24}$$

Note that (2.23) and (2.24) are defined for a specific $\mathbf{x} \in \mathcal{C}_{\mathbf{x}}$, hence we need to stochastically optimize w.r.t. all $\mathbf{x} \in \mathcal{C}_{\mathbf{x}}$. While (2.23) is an integral over $\mathcal{Y}$, it is tractable since we only need to compute the CDF corresponding to the boundary elements of $\mathcal{C}_y$. The resulting learnt distribution $q_{\boldsymbol{\lambda}^*}(\mathbf{w})$ is known as the **variational output-constrained prior (VOCP)**.

2.3.1 Inference with VOCPs

Unlike COCPs, where $p_{\mathcal{C}}(\mathbf{w})$ is directly evaluated during posterior inference, the variational learning process for VOCPs can be done prior to inference and we can apply the optimal parameter $\boldsymbol{\lambda}^*$ independently to any number of training datasets D_{tr}. Algorithm 1 presents the full learning procedure [10]. Again, since the constraints are conditional on $\mathbf{x}$, it will still be necessary to sample from an arbitrary distribution over $\mathcal{C}_{\mathbf{x}}$ at each iteration of the optimization process.

Algorithm 1: Variational Output-Constrained Prior

Input: (2.17) or (2.21), $q_{\boldsymbol{\lambda}}(\mathbf{w})$, $\mathcal{C}$, N_{iter} (no. of epochs), α (learning rate)
$\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\rho}) \leftarrow$ random initialization from any reasonable distribution;
for $n \leftarrow 0$ **to** $N_{iter} - 1$ **do**
  $\mathbf{x} \leftarrow$ random sample from an arbitrary distribution over $\mathcal{C}_{\mathbf{x}}$;
  $f =$ objective in (2.23) or (2.24);
  $\nabla_{\boldsymbol{\mu}} \leftarrow \frac{\partial f}{\partial \boldsymbol{\mu}}$;
  $\nabla_{\boldsymbol{\rho}} \leftarrow \frac{\partial f}{\partial \boldsymbol{\rho}}$;
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \alpha \cdot \nabla_{\boldsymbol{\mu}}$;
  $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \alpha \cdot \nabla_{\boldsymbol{\rho}}$;
end
return $\boldsymbol{\lambda}$;

[10] Note that similar to the BBB algorithm presented in Appendix B.2, we use a different parametrization $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\rho})$ where $\boldsymbol{\sigma} = \log(1 + e^{\boldsymbol{\rho}})$ element-wise. This is because $\boldsymbol{\sigma}$ must be non-negative, which can present computational issues for practical optimization algorithms.

2.4 Analysis and Related Work

COCPs and VOCPs are two variants of **output-constrained priors (OCP)**, and the resulting model is known as an **output-constrained Bayesian neural network (OC-BNN)**. While COCPs and VOCPs both aim to learn a prior over $\mathcal{W}$ that obeys a constraint $\mathcal{C}$, there are some important differences between the two approaches.

Unlike COCPs, which evaluates $p_{\mathcal{C}}(\mathbf{w})$ for every instance of inference (with or without D_{tr}), learning VOCPs is a one-time cost as the optimized variational parameter $\boldsymbol{\lambda}^*$ can be saved for future use. In particular, one can sequentially optimize $\boldsymbol{\lambda}$ for a new constraint $\mathcal{C}^{(t+1)}$ by initializing the parameter of the $(t+1)^{\text{th}}$ run of Algorithm 1 to $\boldsymbol{\lambda}^{(t)}$, the optimized parameter from the previous t constraints. VOCPs are also faster than COCPs as (i) the objectives in (2.17) and (2.21) can be quickly computed, and (ii) modern automatic differentiation and optimization packages are efficient.

However, VOCPs are less robust than COCPs as variational inference is sensitive to initialization of $\boldsymbol{\lambda}$ and multiple runs may be necessary [11]. The use of various approximations, such as the Gaussian variational family as well as approximate closed-form expressions of the prior predictive, also limits the ability of VOCPs to learn complicated constraints. Finally, VOCPs do not generalize to K-classification for $K > 2$ due to the difficulty of finding a good approximation for the prior predictive.

An important similarity tying together both approaches is the necessity of sampling from $\mathcal{C}_{\mathbf{x}}$. As Definitions 2.1.1 and 2.1.3 do not impose any structure over $\mathcal{C}_y(\mathbf{x})$ with respect to the topology on $\mathcal{X}$, we are not assured that any two distributions ϵ-satisfying the points $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C}_{\mathbf{x}}$ respectively are bounded in divergence. However, such an assumption is implicit in both COCPs and VOCPs, which maximize the expected ϵ-satisfaction with respect to an arbitrarily chosen (typically uniform) distribution over $\mathcal{C}_x$. This is generally a reasonable assumption to make for most constraints of practical interest, and for $\mathcal{C}_{\mathbf{x}}$ sufficiently bounded in size. For example, a regression constraint $\mathcal{C}^+$ in the form of a continuous ground-truth function implies that any $\mathbf{w}$ that ϵ-satisfies $\mathbf{x}_1$ will also ϵ-satisfy a neighbouring $\mathbf{x}_2$, since $\Phi_{\mathbf{w}}$ is continuous. Exploiting additional structures on $\mathcal{C}_y$ can lead to more sample-efficient ways (w.r.t. $\mathbf{x} \in \mathcal{C}_{\mathbf{x}}$) of computing COCPs and VOCPs. Table 2.4.1 summarizes both approaches.

[11] This is a problem that also plagues VI algorithms such as BBB.

Output-Constrained Priors		
	COCP	**VOCP**
Sampling from $\mathcal{C}_{\mathbf{x}}$	Necessary	Necessary
Constraint Type	Valid on all output constraints	Valid on all output constraints
Applicable for multiple constraints $\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(m)}$	Yes	Yes
Supervised Task	Valid on all tasks	Valid on regression and **binary** classification only
Runtime	Every instance of posterior inference	One-time optimization *before* posterior inference

Table 2.4.1: A summary of the similarities and differences between COCPs and VOCPs.

Our work can also be situated in relation to existing literature on BNN functional priors and inference. Below, we present some key research and discuss them within the context of OC-BNNs.

NOISE CONTRASTIVE PRIORS Motivated by the similar need for BNNs to make reasonable predictions out-of-distribution, Hafner et al. [28] proposes a generative "data prior" in function space, where both $\mathbf{X} \sim DP_{\mathbf{X}}$ and $Y|\mathbf{X} \sim DP_{Y|\mathbf{X}}$ are modeled as noisy Gaussians and $\mathbb{E}_{DP}[Y|\mathbf{x}] = 0$ if $\mathbf{x}$ is OOD. Hence the data prior directly specifies a predictive distribution that is robust to noise but not overconfident OOD. VI is used for inference, where the learning objective to maximize is:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[\log p(D_{tr}|\mathbf{W})\Big] + \mathbb{E}_{\mathbf{X}\sim DP_{\mathbf{X}}}\Big[D_{KL}\big(q_{\boldsymbol{\theta}}(Y|\mathbf{X})\,||\,p_{DP}(Y|\mathbf{X})\big)\Big] \quad (2.25)$$

Hence the goal is to maximize the training likelihood, while learning a prior predictive distribution that is close to the data prior. Since the KL divergence term above is taken w.r.t. predictive distributions on Y, and VI is conducted over $\mathcal{W}$, Hafner et al. [28] faces the similar challenge of expressing a function-space prior in $\mathcal{W}$. However, as the data priors are Gaussians, (2.25) can be expressed in terms of KL divergence of distributions on $\boldsymbol{\theta}$.

NCPs are similar to OC-BNNs as both methods involve placing a prior on function space but performing inference ultimately in parameter space. However, OC-BNNs model output constraints, which encode a richer class of functional beliefs than the simpler zero-mean Gaussian assumptions encoded by NCPs.

EQUALITY/INEQUALITY CONSTRAINTS FOR DEEP PROBABILISTIC MODELS Most similar to OC-BNNs is the work by Lorenzi and Filippone [48], which considers equality and inequality constraints for (multivariate) regression for various deep probabilistic models. Constraints of their form are specified as h^{th}-order differential equations:

$$\mathcal{C} = \left\{ \mathbf{f}(t) \middle| \frac{d^h f_i(\mathbf{x})}{dt^h} \circ \mathcal{C}_f \right\}, \quad \circ \in \{=, >\} \tag{2.26}$$

where t indexes times at which the observation $\mathbf{y}$ is collected from some input $\mathbf{x}$, and the value $\mathcal{C}_f$ is a function of $t, \mathbf{f}$ and q^{th}-order derivatives. Similar to COCPs, Lorenzi and Filippone [48] enforces the constraints by defining a joint probability distribution $p(Y, \mathcal{C}|\mathbf{x})$, where $p(\mathcal{C}|\mathbf{x})$ is modeled as Gaussian and Student-t distributions for equality constraints and logistic distributions for inequality constraints. However, $p(\mathcal{C}|\mathbf{x})$ is interpreted as a likelihood rather than a prior.

The main differences between Lorenzi and Filippone [48] and OC-BNNs are that: (i) their methods are applied to deep Gaussian processes (GPs), which are non-parametric models where inference is carried out directly in function space [12], (ii) their experiments focus on low-dimensional systems where the constraints originate from simulated ordinary differential equations, (iii) their method do not extend to the classification setting. In contrast, OC-BNNs present a generalized framework for incorporating constraints, and are meant to be tractable for real-life applications with high input dimensionality and large datasets.

NEURAL PROCESSES A recent class of models known as neural processes (NP), first introduced in Garnelo et al. [20] and Garnelo et al. [19], aims to combine the computational tractability of BNNs with the ability of deep GPs to carry out function-space inference. NPs are trained on sequences of input-output tuples $\{(\mathbf{x}, y)_i\}_{i=1}^m$ constructed from the dataset and therefore learn a distribution over functions. However, they are computationally efficient since the underlying model is a NN. Building on NPs, Louizos et al. [51] introduces the functional neural process, which improves on vanilla NPs by modeling the correlation structure of $\mathcal{X}$ as dependency graphs, resulting in the ability to specify richer functional assumptions.

Compared to OC-BNNs, NPs represent a largely distinct direction of work, since inference is carried out directly in function space. The set of functional assumptions induced by NPs is different from OC-BNNs; in particular, NPs do not have the ability to encode output constraints of the form that OC-BNNs consider.

[12] Since deep GPs are closely related to BNNs, an extension of their approach for BNNs is viable.

2.5 Low-Dimensional Simulations

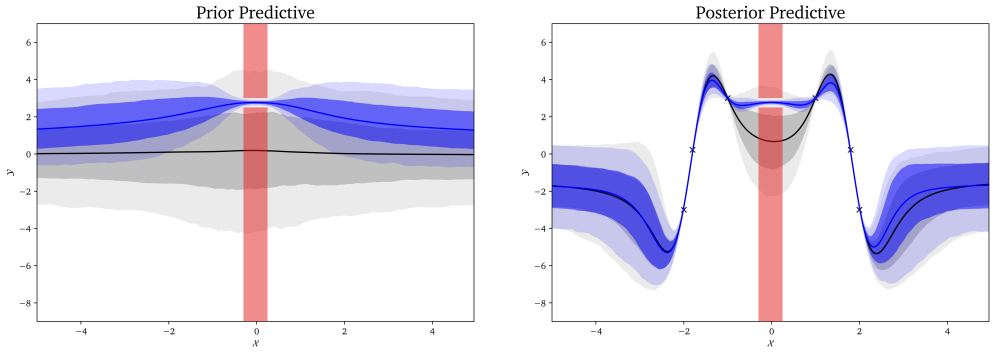
As a proof-of-concept for OC-BNNs, we simulate synthetic constraints and datasets for small input dimension and visualize the prior/posterior predictive distributions.

General Experimental Setup All regression experiments are 1D ($\mathcal{X} = \mathbb{R}$) and all classification experiments are 2D ($\mathcal{X} = \mathbb{R}^2$), with either $K = 2$ (binary) or $K = 3$ classes. For each experiment, we will simulate either $\mathcal{C}^+$ or $\mathcal{C}^-$, and use either COCPs or VOCPs. A baseline BNN using the naive Gaussian prior (1.14) is also shown for comparison. As all experiments are low-dimensional, we run HMC for inference. The only exception is the experiment in Figure 2.5.3b, where we use SVGD to capture the multiple posterior modes. The empirical predictive distribution is computed according to (1.17). Refer to Appendix E for hyperparameters and setup details.

Plot Legend For all regression plots, the x-axis represents $\mathcal{X} = \mathbb{R}$, a one-dimensional input space. The y-axis represents $\mathcal{Y} = \mathbb{R}$. For each predictive distribution, we plot the mean function (bold line) as well as the credible intervals for $\sigma = 1$ (darker shading) and $\sigma = 2$ (lighter shading). The predictive distribution using OC-BNNs is plotted in blue and the baseline distribution is plotted in grayscale. Negative constrained regions, defined by $\mathcal{C}^-_{\mathbf{x}}$ and $\mathcal{C}^-_y(\mathbf{x})$, are shaded in red. Positive constrained regions, defined by $\mathcal{C}^+_{\mathbf{x}}$ and $\mathcal{C}^+_y(\mathbf{x})$, are shaded in green.

For classification plots, the x- and y-axes represent $\mathcal{X} = \mathbb{R}^2$, the two-dimensional input space. If $K = 3$, the output classes are color-coded red, green and blue. If $K = 2$, the output classes are color-coded blue (0) and orange (1). Each point on the graph is shaded the corresponding color for a class k if: (i) ≥ 0.9 of the predictive mass is assigned to k (darker shading), or if (ii) ≥ 0.65 of the predictive mass is assigned to k (lighter shading). WLOG we consider only positive constraints, where $\mathcal{C}^+_{\mathbf{x}}$ is delineated by a black border shaded by the desired class color(s). Since each plot is able to represent only a *single* predictive distribution, the baseline distribution is included as an inset.

All observed data points in D_{tr} are marked as crosses. If a ground-truth function exists (from which D_{tr} is generated), it is denoted as a thick green line.



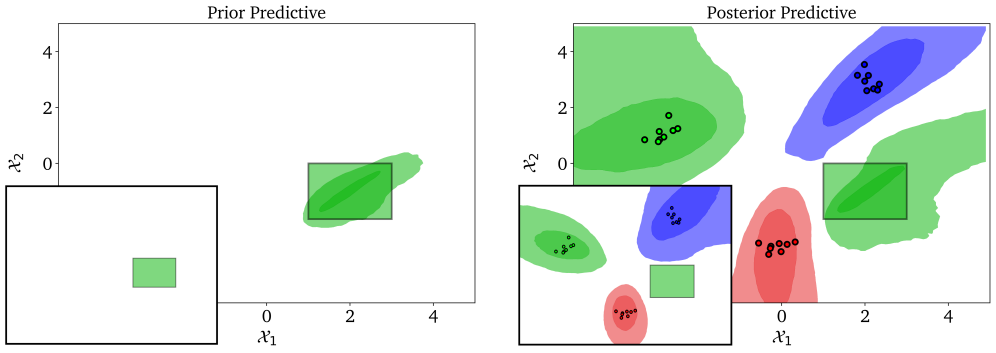
(a) Prior predictive plots. **(b)** Posterior predictive plots.

Figure 2.5.1: 1D regression with the negative constraint: $\mathcal{C}_{\mathbf{x}}^{-} = [-0.3, 0.3]$ and $\mathcal{C}_{y}^{-} = (-\infty, 2.5] \cup [3, \infty)$. The negative exponential COCP (2.12) is used. Even with such a restrictive constraint, the predictive uncertainty of the OC-BNN (blue) drops sharply to fit within the permitted region of $\mathcal{Y}$ compared to the baseline (gray). In particular, note that the OC-BNN posterior uncertainty matches the baseline everywhere except near $\mathcal{C}_{\mathbf{x}}$.

2.5.1 Key Results

OC-BNNs model uncertainty in a manner that respects constrained regions *and* explains training data, without making overconfident predictions outside $\mathcal{C}_{\mathbf{x}}$. Figure 2.5.1 shows the prior and posterior predictive plots for a negative output constraint in regression. A large $\mathcal{C}_y$ (i.e. the permitted region $\mathcal{Y} - \mathcal{C}_y$ is narrow) was intentionally chosen to demonstrate the ability of OC-BNNs to fit highly restrictive constraints. Unlike the naive baseline BNN, the prior predictive satisfies the specified constraint, with the predictive variance smoothly narrowing as $\mathbf{x}$ approaches $\mathcal{C}_{\mathbf{x}}$ such that the entire probability mass of the predictive distribution is fully confined within $\mathcal{Y} - \mathcal{C}_y$. Outside $\mathcal{C}_{\mathbf{x}}$, the predictive variance remains wide to reflect uncertainty everywhere else in $\mathcal{X}$ prior to observing data. After learning from D_{tr}, the posterior predictive distribution adjusts to fit all data points closely. Near or within $\mathcal{C}_{\mathbf{x}}$, the predictive variance remains narrows to keep probability mass constrained inside $\mathcal{Y} - \mathcal{C}_y$, reflecting our increased confidence in the constrained region. In regions of $\mathcal{X}$ far from $\mathcal{C}_{\mathbf{x}}$, the OC-BNN does not become overconfident, maintaining a similarly wide variance to the baseline naive BNN.

Figure 2.5.2 shows the predictive plots for a positive output constraint in classification. The results are analogous to regression and show the OC-BNN prior pre-



(a) Prior predictive plots. **(b)** Posterior predictive plots.

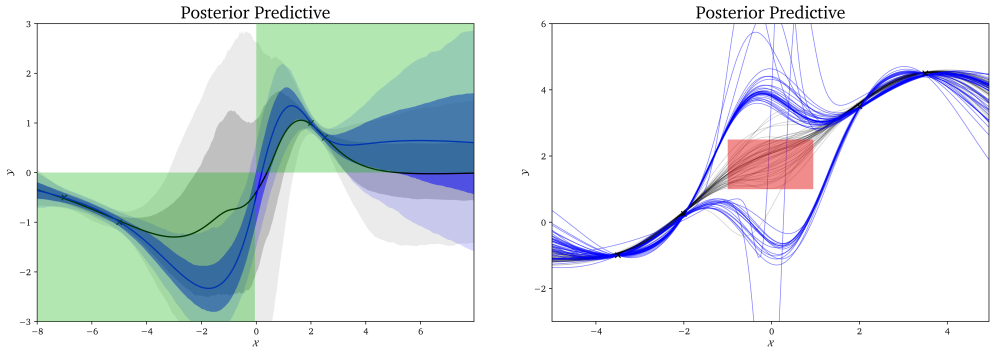
Figure 2.5.2: 2D 3-classification with the positive constraint: $\mathcal{C}_{\mathbf{x}}^{+} = (1,3) \times (-2,0)$ and $\mathcal{C}_{y}^{+} = \{\text{green}\}$. The positive Dirichlet COCP (2.14) is used. D_{tr} contains 8 data points in each class, generated from the Gaussian means $(3,1)$,$(0,3)$ and $(2,3)$. In both the prior and posterior, the constrained region (green rectangle) enforces the prediction of the green class.

dictive respecting the positive constraint. Importantly, outside $\mathcal{C}_{\mathbf{x}}$, the OC-BNN does not show a strong preference for any class ($\mathbb{E}_{\mathbf{w}\sim\mathbf{W}}[\Phi_{\mathbf{w}}(\mathbf{x}) = k] < 0.65$ for all k). Just like a naive BNN would, the OC-BNN avoids making overconfident predictions in areas of $\mathcal{X}$ it has little knowledge about. The posterior predictive distribution behaves similarly, except that all points in D_{tr} are also well-fitted.

Both of these examples demonstrate how OC-BNNs perform strictly better than naive BNNs, in that they adequately enforce constraints without (i) sacrificing predictive accuracy under the training distribution or (ii) making overconfident predictions outside $\mathcal{C}_{\mathbf{x}}$ (that a normal BNN would not have made).

OC-BNNs can capture global relationships between $\mathcal{X}$ and $\mathcal{Y}$, subject to sampling efficacy. Instead of the local constraints presented in the two examples above, OC-BNNs are also able to model global structural constraints between input and output. Figure 2.5.3a shows an example where we enforce the constraint that $xy \geq 0$. Even though the training data themselves adhere to this constraint, learning from D_{tr} alone is insufficient. The OC-BNN posterior predictive distribution significantly narrows (compared to the baseline) to fit the constraint, particularly near $x = 0$. The caveat is that OC-BNNs can only learn as well as sampling from $\mathcal{C}_{\mathbf{x}}$ permits.

OC-BNNs can capture posterior multimodality. As NNs are highly expressive, BNN posteriors can contain multiple modes of significance. For OC-BNNs to make



(a) Posterior predictive plots. **(b)** Posterior predictive plots.

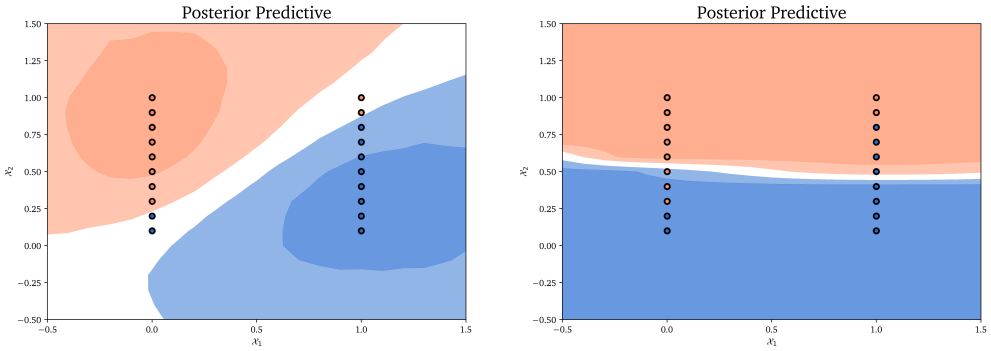
Figure 2.5.3: (a) 1D regression with the positive constraint: $\mathcal{C}_{\mathbf{x}}^{+} = \mathbb{R}$ and $\mathcal{C}_{y}^{+}(\mathbf{x}) = \{y \,|\, \mathbf{x}{\cdot}y \geq 0\}$. The VOCP (2.23) is used. Unlike the baseline, the OC-BNN has most of its probability mass constrained within the permitted (green) region. **(b)** 1D regression with the negative constraint: $\mathcal{C}_{\mathbf{x}}^{-} = [-0.3, 0.3]$ and $\mathcal{C}_{y}^{-} = (-\infty, 2.5] \cup [3, \infty)$. The negative exponential COCP (2.12) is used. Instead of HMC, SVGD is used for inference here to allow efficient mode exploration. The 50 SVGD particles represent functions that pass above and below the constrained region, corresponding to two distinctly significant predictive modes. ϵ-satisfaction is not perfectly achieved as SVGD particles tend to repel each other; see Appendix B.3 for details.

accurate predictions, they must be able to capture most or all such predictive modes. Figure 2.5.3b demonstrates such an example. The negative constraint is specified in such a way as to allow for functions that fit D_{tr} passing above or below the constrained region. Indeed, the resulting OC-BNN posterior predictive distribution contains significant probability mass on either side of $\mathcal{C}_y$, affirming the presence of both significant posterior modes. Importantly, note that the negative exponential COCP does not *explicitly* indicate the presence of multiple modes [13]. The fact that OC-BNNs encourage mode exploration without needing them to be explicitly specified in the prior is promising for high-dimensional applications.

OC-BNNs can model interpretable desiderata represented as output constraints. OC-BNNs can be used to enforce important desiderata or qualities that the system should possess, so long as these considerations can be expressed as output constraints. Figure 2.5.4 demonstrates a fairness constraint known as **demographic parity**:

$$p(Y = 1 | \mathbf{x}_A = 1) = p(Y = 1 | \mathbf{x}_A = 0) \tag{2.27}$$

[13] This is in contrast to the positive multimodal Gaussian COCP, where each $y' \in \mathcal{C}_y$ represents a mode.



(a) Baseline posterior predictive plot. **(b)** OC-BNN posterior predictive plot.

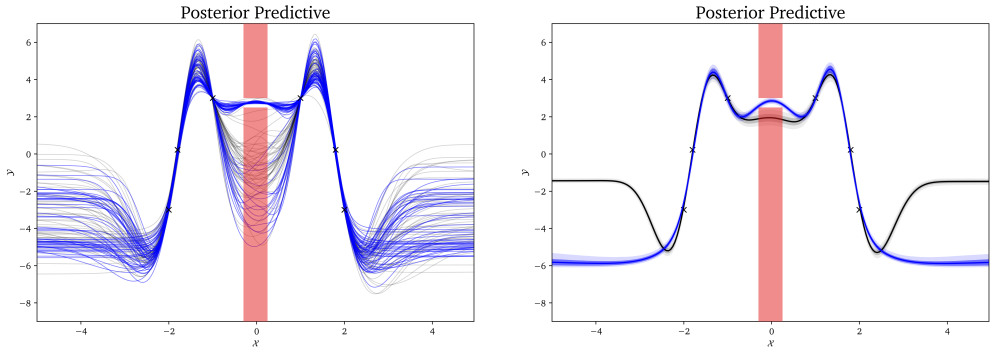
Figure 2.5.4: 2D binary classification. Suppose a hiring task where $\mathcal{X}$ represents features of potential candidates. $\mathcal{X}_1$ is a binary feature indicating the membership of a protected trait, e.g. gender or race. $\mathcal{X}_2$ denotes skill level. Hence a positive (orange) classification should be correlated with higher values of $\mathcal{X}_2$. The dataset D_{tr} displays historic bias, where members of the protected class ($\mathbf{x}_1 = 1$) are discriminated against ($Y = 1$ iff $\mathbf{x}_1 = 1, \mathbf{x}_2 \geq 0.8$, but $Y = 1$ iff $\mathbf{x}_1 = 0, \mathbf{x}_2 \geq 0.2$). A naive BNN **(a)** would learn an unfair linear separator from D_{tr}. However, learning the probabilistic constraint: $\mathcal{D}_y(\mathbf{x})$ as the distribution where $p(\Phi(\mathbf{x}) = 1) = \mathbf{x}_2$ with a VOCP (2.24) allows the OC-BNN **(b)** to learn a fair separator, **despite the presence of** a biased dataset.

where $\mathbf{x}_A$ is a protected feature such as race or gender. (2.27) is expressed as a probabilistic output constraint in Figure 2.5.4. We can see that the OC-BNN not only learns to respect this constraint, it does so in the presence of **conflicting training data** (D_{tr} is an unfair dataset). Hence OC-BNNs may be used to model a diverse range of output desiderata that the human user may wish to impose on the learning system.

2.5.2 Ablation Experiments

We briefly present the results of a number of "ablation" experiments below to highlight some insights about how OC-BNNs operate.

Choice of Inference Algorithm Figure 2.5.5 shows the same negative constraint presented in Figure 2.5.1b, but using BBB and SVGD for inference. The three plots show that there is little difference in the three posterior predictive plots (besides the individual quirks of each inference method, see figure caption for details), and that all inference methods are well-suited for use with OC-BNNs.

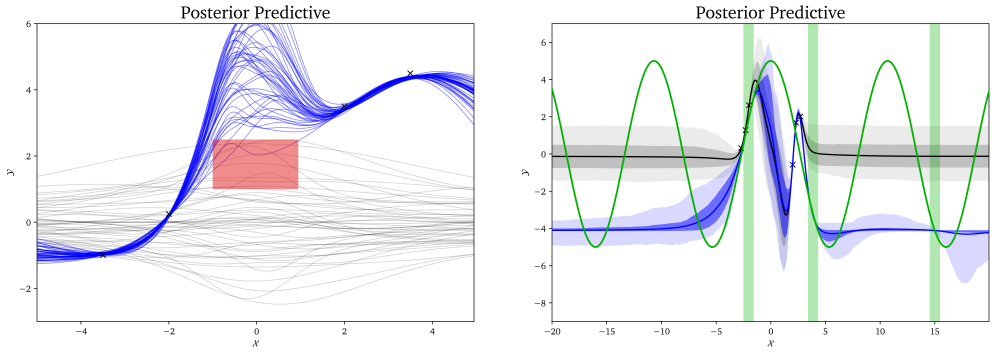


(a) Posterior predictive plot. **(b)** Posterior predictive plot.

Figure 2.5.5: Same experimental setup as in Figure 2.5.1, except that **(a)** uses SVGD for posterior inference, and **(b)** uses BBB. Like HMC, both posterior predictive plots obey the constraints and fit D_{tr}. However, SVGD posterior violates the constraint more than HMC or BBB, as SVGD particles are optimized to be as far from each other (in $\mathcal{W}$) as possible. The BBB posterior has smaller variance than HMC or SVGD everywhere in $\mathcal{X}$, which is an empirically well-documented problem for VI.

CHOICE OF OCP In general, COCPs are more versatile than VOCPs because the latter can only represent a diverse range of output constraints using Gaussian approximations. For example, Figure 2.5.6a shows the same negative constraint presented in Figure 2.5.3b, but learnt using a VOCP instead of a COCP. As can be seen, the VOCP fails to capture both posterior predictive modes. Despite these limitations, VOCPs work well on simpler constraints (for which the Gaussian approximation is a reasonable one) and present significant speedups compared to COCPs. Improving VOCPs by using advanced variational techniques that avoid Gaussian approximations (e.g. normalizing flows) is also a promising direction for future work.

CHOICE OF DATASET In real-life applications, we might expect specified constraints to be consistent with D_{tr}. In the case that $\mathcal{C}$ and D_{tr} are incompatible (e.g. adversarial or noisy data), the resulting posterior depends on (i) model capacity (whether it is possible for the BNN to fit both D_{tr} and $\mathcal{C}$) as well as (ii) factors such as volume of data or COCP/VOCP hyperparameters, which affects the prior and likelihood distributions. For example, Figure 2.5.6b shows a BNN with enough capacity to fit both the noisy data as well as ground-truth constraints, resulting in overfitting of the posterior predictive distribution. However, the earlier example in Figure 2.5.4 shows the OC-BNN ignoring D_{tr} labels to satisfy the fairness constraint, as the small num-



(a) Posterior predictive plot. **(b)** Posterior predictive plot.

Figure 2.5.6: (a) Same experimental setup as in Figure 2.5.3b. Here, SVGD only captures one of two possible modes. The other mode can be captured by adjusting the initialization of VOCP parameter $\boldsymbol{\lambda}$ (not depicted here), but not both modes at the same time. **(b)** The ground-truth function is $y^* = 5\cos(x/1.7)$. D_{tr} contains 6 points perturbed by large Gaussian noise around ground-truth values. Three positive constraints (vertical green bands) are also placed using positive Gaussian COCPs (2.10) around the corresponding ground-truth values. The resulting OC-BNN posterior predictive distribution fits both the noisy data and all three constraints.

ber of data points constitutes weaker evidence than the constrained prior. Tuning hyperparameters such as COCP variance/smoothness, or the number of optimization epochs for VOCPs, can give us control over the interplay of factors.

2.6 Application 1: Clinical Action Prediction

To demonstrate the efficacy of OC-BNNs, we simulate meaningful and interpretable output constraints on real-life datasets. In this section, we consider clinical action prediction from physiological features of intensive care unit (ICU) patients.

Data Source The MIMIC-III database [37] is a freely accessible, benchmark database for healthcare research, developed by the MIT Lab for Computational Physiology. It consists of de-identified health data associated with 53,423 distinct admissions to critical care units at the Beth Israel Deaconess Medical Center in Boston, Massachusetts, between 2001 and 2012. Multiple classes of data are available, such as (i) time-stamped, nurse-verified bedside physiological measurements, (ii) time-stamped interventions taken, (iii) laboratory results, and more. Further data processing can be done to construct datasets for specific machine learning tasks.

		filtered		unfiltered	
		BNN	**OC-BNN**	**BNN**	**OC-BNN**
Train	Accuracy	0.745	0.741	0.881	0.878
	F1 Score	0.805	0.801	0.882	0.880
	Violation Fraction	0.151	0.149	N/A	N/A
Test	Accuracy	0.660	0.665	0.647	0.649
	F1 Score	0.746	0.748	0.725	0.736
	Violation Fraction	0.132	0.126	**0.117**	**0.039**

Table 2.6.1: Experimental results with and without filtering out points in D_{tr} incompatible with $\mathcal{C}$. In all cases, accuracy and F1 score remain unchanged when using OC-BNNs. For the experiment with filtration, the violation fraction decreases by a factor of 3 when using OC-BNNs instead of the naive BNN.

PROBLEM FORMULATION We consider a binary classification task of whether clinical interventions for hypotension management — namely, vasopressors or IV fluids — should be taken for an ICU patient. From the MIMIC-III database, we construct a time-independent dataset with 298K points and 9 physiological features (consisting of measured vital signs and laboratory results). 10% of the dataset is held out as the test set.

We specify the **physiologically feasible**, positive (deterministic) constraint as follows: if the mean arterial pressure is less than 65 units ($\mathcal{C}_{\mathbf{x}} = \{\mathbf{x} \,|\, \mathbf{x}_{\texttt{map}} \leq 65\}$), then some hypotension intervention should be taken ($\mathcal{C}_y = \{1\}$). Refer to Appendix E for details on data processing steps, dataset features as well as experimental setup.

METHODOLOGY We conduct posterior inference using both the baseline BNN (using the naive Gaussian prior) as well as OC-BNNs. The positive Dirichlet COCP is used. In D_{tr}, some data points are inconsistent with the specified constraint. We train our model both with and without filtering out these incompatible points.

EVALUATION As $Q = 9$ is large, the posterior predictive distribution cannot be directly visualized. Instead, we use the standard metrics of **accuracy** and **F1 score** on D_{te} to evaluate predictive accuracy. We also measure **violation fraction** [14], which is the fraction of predictions on D_{te} that do not satisfy $\mathcal{C}$, to evaluate how well the BNN posterior obeys the constraint.

[14]For binary classification, measuring the violation fraction is more useful than measuring ϵ-satisfaction over $\mathcal{C}_{\mathbf{x}}$ since it is the actual output labels and not the posterior predictive mass on each class that we care about.

RESULTS Table 2.6.1 summarizes the experimental results. The main takeaway is that **OC-BNNs maintain classification accuracy while reducing physiologically infeasible constraint violations**. The results show that OC-BNNs match standard BNNs on all predictive accuracy metrics, while satisfying the constraint to a far greater extent. The caveat is that the improvement in constraint satisfaction is significant only for the case where points originally in the constrained region are filtered out. However, we note that the OC-BNN performance is still significant because quite a significant portion of D_{tr} is incompatible with the specified constraint.

This experiment affirms the low-dimensional simulations that we carried out in the previous section, which shows that OC-BNNs are able to learn constraints well *without sacrificing* predictive power. As we chose a constraint that is physiologically feasible and consistent with the decision that a human expert (clinician) will make, the results also demonstrate how OC-BNNs can be used to enforce real-world constraints efficiently in the Bayesian setting.

2.7 APPLICATION 2: RECIDIVISM PREDICTION

In this section, we consider the prediction of recidivism risk of criminal defendants, imposing a fairness constraint similar to the example in Figure 2.5.4 that prevents the exploitation of protected traits (in our case, race) by a biased or unfair dataset. Recidivism prediction, and more broadly, machine learning for criminal justice, are applications of growing interest to ML researchers (particularly the fairness and interpretable ML communities), because of the high stakes involved and because such models are increasingly deployed in real life [1, 43, 44, 67].

BACKGROUND COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is a proprietary model developed by Equivant [15] and used widely by the United States criminal justice system to aid with parole and bail decisions. Amongst other goals, COMPAS scores criminal defendants on their risk of recidivism, which they define as any "finger-printable arrest involving a charge and a filing for any uniform crime reporting (UCR) code." A study by ProPublica in 2016 created a linear explanation model for COMPAS and found the original model to be racially biased towards African American defendants [1, 44].

[15] Originally known as Northpointe.

DATA SOURCE The team behind the 2016 ProPublica study created a dataset containing information about 6172 defendants from Broward Couty, Florida. Along with the COMPAS scores for these defendants, the dataset contains features related to their criminal history and their demographic attributes. Note that the defendant's race and actual recidivism history (whether they recidivated within two years) are both explicit features of the dataset. We will use the ProPublica dataset and follow identical data processing steps as the original team. Our final dataset D_{tr} contains 6172 data points with 9 features.

PROBLEM FORMULATION We consider the same binary classification task in Slack et al. [67] (a later work that also uses the ProPublica dataset), which is predicting if a defendant is profiled by COMPAS as having a high risk of recidivism.

We specify the **fairness** constraint that the probability of predicting high-risk recidivism should not depend on the defendant's race. Formally, we have a probabilistic, global output constraint as follows: for all data points ($\mathcal{C}_{\mathbf{x}} = \mathbb{R}^9$), the probability of high-risk should be identical to the defendant's actual recidivism history ($\mathcal{D}_y$ is such that $p(y = 1) = \texttt{two\_year\_recid}$). Refer to Appendix E for details on data processing steps, dataset features as well as experimental setup.

METHODOLOGY We conduct posterior inference using both the baseline BNN (using the naive Gaussian prior) as well as OC-BNNs using a VOCP. We train on two versions of D_{tr} — with and without the inclusion of race as an explicit feature. Like the example in Figure 2.5.4 and the MIMIC-III application, D_{tr} is incompatible with this constraint (since the COMPAS scores themselves demonstrate racial bias). However, it is not possible to remove inconsistent data points since we are specifying a probabilistic constraint. Furthermore, like in Figure 2.5.4, our goal here is to enforce the fairness constraint *despite* an inconsistent dataset, in simulating the real-life scenario that unfair or biased datasets are used for training. As the dataset is small and imbalanced, we do not create a test set. Evaluation is performed solely on the training dataset.

EVALUATION We will use **accuracy** and the **F1 score** as metrics for predictive accuracy. To measure constraint satisfaction, we report the **fraction of the sensitive group** (African American defendants vs. non-African American defendants) predicted as high-risk recidivists.

		with race feature		without race feature	
		BNN	**OC-BNN**	**BNN**	**OC-BNN**
Train	Accuracy	0.818	0.568	0.806	0.525
	F1 Score	0.556	0.239	0.557	0.320
	African American High-Risk Fraction	**0.365**	**0.395**	0.350	0.544
	Non-African American High-Risk Fraction	**0.077**	**0.400**	0.148	0.480

Table 2.7.1: Experimental results with and without training with race as an explicit dataset feature. Using OC-BNNs leads to the two sensitive groups having almost equal rates of high-risk recidivism prediction, compared to standard BNNs where is a 5-factor difference. Accuracy and F1 score decrease when using OC-BNNs instead of BNNs, which is *expected* since the dataset itself is incompatible with the specified constraint.

RESULTS Table 2.7.1 summarizes the experimental results. We see that by constraining the recidivism prediction to the defendant's actual criminal history, **OC-BNNs strictly enforce a fairness constraint**. For both the scenarios where race is/is not used as an explicit feature, the fraction of African Americans and non-African Americans being predicted as high-risk recidivists equalized after imposing the constraint. (For the naive BNN results, the difference in fraction between these two groups is more stark when race is used as an explicit feature, likely because the model learns the positive correlation between race and the output label in D_{tr}.)

Unlike the previous clinical action application, OC-BNNs have lower predictive accuracy on D_{tr} than standard BNNs. This is to be expected since the training dataset is biased, and therefore enforcing racial fairness comes at the expense of correctly predicting biased labels. In general, **the two high-dimensional applications represent two possible objectives for learning constraints with OC-BNNs**. The MIMIC-III experiment simulates the case where $\mathcal{C}$ and D_{tr} are complementary sources of ground-truth knowledge, whereas the COMPAS experiment simulates the case where we wish to strictly enforce $\mathcal{C}$ regardless of (or despite) the training data D_{tr} we observe. Our results show that applying the correct OCP can allow us to achieve either objective.

3

VI Predictive Error Bound for BNNs

MCMC methods have long been the tool of choice for approximate Bayesian inference, and they carry theoretical convergence guarantees along with well-understood diagnostic metrics [10]. However, the domains that deep learning is applied to often come with high-dimensional and massive datasets. As MCMC techniques do not scale well to data quantity or dimensionality, they are unsuitable for BNN posterior inference in most applications of practical interest.

In recent years, VI has emerged as a widely-used, scalable alternative. This is due to the development of "black box" VI algorithms that can be easily applied to a wide range of models without requiring model-specific derivations of the variational updates [5, 60], as well as massive speedups in computational packages for automatic differentiation and optimization [59]. While VI methods like BBB are practically efficient, the quality of approximation is empirically poorer than MCMC sampling. VI approximations are restricted not only by (i) the choice of variational family, which is often far simpler than the true BNN posterior, but also (ii) the limitations of optimization techniques like SGD, such as the sensitivity to initial parameters.

In settings where BNNs are deployed, particularly high-stakes applications, it is crucial that we have accuracy guarantees for the posterior approximation or its point estimates. This necessity is heightened by the poor quality of VI approximations, particularly in regions of $\mathcal{X}$ sparsely represented by training data, which can result in significant error in the posterior predictive distribution [77].

There are two general approaches that we can follow to obtain such guarantees. The first is to produce provable error bounds on statistics of interest [1]. However, producing tight error bounds of this nature is difficult, due to the complexity of $\Phi_{\mathbf{W}}$ as well as the resulting multimodal posterior over the high-dimensional space $\mathcal{W}$. The second, related approach is to design diagnostics (which may or may not be empirical) that can distinguish more accurate VI approximations from poorer ones. Unfortunately, benchmark diagnostics across the BNN research community do not exist either. This stems not only from similar difficulties of evaluating complex BNN posteriors, but also as it is not entirely clear what the correct desiderata are when comparing two probability distributions. [2]

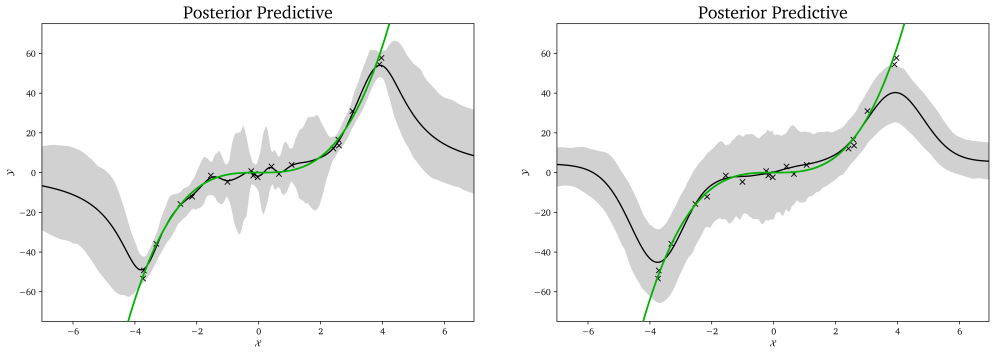
As a result, in most BNN research literature, we resort to low-dimensional simulations for evaluation, where direct visualization of the posterior predictive distribution is possible. HMC is typically used as the proxy for the true BNN posterior for comparison [78]. Figure 3.0.1 shows a typical visualization of the problems emblematic of VI approximations. For real-world, high-dimensional applications, visualization is neither possible nor desirable.

In this chapter, we follow the first approach and develop an error bound on VI approximations. In doing so, we have two specific motivations. First, our quantity of interest is the posterior predictive mean $\mathbb{E}[Y|\mathbf{X}, D_{tr}]$. As $\mathcal{W}$ is largely an uninterpretable space, bounds on posterior statistics (over $W|D_{tr}$) are not meaningful. Instead, it is the predictive setting $Y|\mathbf{X}, D_{tr}$ that is of ultimate interest and impact to end users. Second, we desire these error bounds to be computed cheaply and efficiently, so that they can easily be applied to high-dimensional VI runs in practice.

To this end, we introduce the **Kernelized Bound on Variational Predictive Mean Error (KEBO-VPME)**, an efficiently estimable value based on a RKHS approximation to a maximum mean discrepancy bound on $\mathbb{E}[Y|\mathbf{X}, D_{tr}]$. We will first discuss related work in Section 3.1, and introduce the core theory behind KEBO-VPME in Section 3.2. Sections 3.3 evaluates the efficacy of KEBO-VPME on low-dimensional simulations.

[1] By error, we mean the difference of some variable of interest when computed under the true posterior versus the variational approximation.

[2] For example, simply comparing the numerical value of the ELBO is insufficient. Not only does it lies on an uninterpretable scale, it ignores the normalizing constant (1.10) that changes with reparametrization of $\mathbf{W}$ (e.g. as used in the reparametrization trick), making comparisons between approximations meaningless [78].



(a) HMC posterior predictive plot. **(b)** BBB posterior predictive plot.

Figure 3.0.1: We simulate the same 1D regression toy example as in Section 5.3 of [32], a commonly used example in BNN literature. The ground-truth function (green) is $y^* = x^3$, and we uniformly sample 20 inputs $x \in [-4, 4]$ for D_{tr}, each independently perturbed by Gaussian noise $\mathcal{N}(0, 3^2)$. See Appendix F for further details. The posterior predictive distribution plot is shaded at the credible interval $\sigma = 3$. We conduct inference with HMC and BBB. HMC shows wider predictive variances than BBB, especially at regions of $\mathcal{X}$ further away from D_{tr}. Further away from D_{tr}, the predictive mean for BBB also becomes a poorer fit to data.

NOTATION Everywhere in this chapter, where appropriate, we will use p and $q_{\boldsymbol{\theta}}$ to denote the posterior $p(\mathbf{w}|D_{tr})$ and variational approximation $q_{\boldsymbol{\theta}}(\mathbf{w})$ respectively. We will also use $\bar{p}$ to denote the *unnormalized* posterior, $p(\mathbf{w}, D_{tr})$, where the distinction is necessary.

3.1 RELATED WORK

PSIS AND VSBC Yao et al. [78] introduces two diagnostics for VI evaluation. The first is Pareto smoothed importance sampling (PSIS). The importance-sampled Monte Carlo estimate of $\mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}[h(\mathbf{W})]$ for some function h is:

$$\mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}[h(\mathbf{W})] \approx \frac{\sum_{i=1}^{S} \gamma_i \cdot h(\mathbf{w}^{(i)})}{\sum_{i=1}^{S} \gamma_i} \qquad \text{where } \gamma_i = \frac{\bar{p}}{q_{\boldsymbol{\theta}}} \text{ for } \mathbf{w}^{(i)} \tag{3.1}$$

The $\{\gamma_i\}_{i=1}^{S}$ are known as importance ratios. PSIS fits a Pareto distribution to the largest m importance ratios of S samples drawn from $q_{\boldsymbol{\theta}}$, where $m < S$. As it turns out, the fitted shape parameter $\hat{k}$ of this Pareto distribution can be related to the

Rényi (or α) divergence between p and $q_{\boldsymbol{\theta}}$:

$$\hat{k} \approx \inf \left\{ k' > 0 : D_{1/k'}\big(p(\cdot) \,||\, q_{\boldsymbol{\theta}}(\cdot)\big) < \infty \right\} \tag{3.2}$$

By sampling from $q_{\boldsymbol{\theta}}$ and computing (3.2), the value of $\hat{k}$ can be used as an empirical guide for whether the VI approximation is reliable.

The second diagnostic is the variational simulation-based calibration (VSBC). While PSIS evaluates the accuracy of the full VI posterior, VSBC assesses the average performance of point estimates from the posterior. It relies on the result in statistical validation from [9] that if $\mathbf{w}^{(0)} \sim p(\mathbf{W})$ and $D_{tr} \sim p(D_{tr}|\mathbf{w}^{(0)})$, then

$$p_{\mathbf{w}^{(0)}, D_{tr}}\Big(p(\mathbf{W} < \mathbf{w}^{(0)}|D_{tr}) \leq \cdot\Big) = p(U \leq \cdot) \quad \text{where } U \sim \text{Unif}(0,1) \tag{3.3}$$

Based on (3.3), Yao et al. [78] designs a procedure where S parameters $\{\mathbf{w}^{(i)}\}_{i=1}^{S}$ are sampled from the prior and S corresponding datasets $\{D_{tr}^{(i)}\}_{i=1}^{S}$ sampled from the likelihood distribution of each parameter. Variational approximations $\{q_{\boldsymbol{\theta}}^{(i)}\}_{i=1}^{S}$ are constructed for each dataset and the marginal calibration probabilities $p_{ij} = p(\mathbf{w}_j^{(i)} < \mathbf{w}_j | \mathbf{w} \sim q_{\boldsymbol{\theta}}^{(i)})$ are computed. Asymmetry of $\{p_{ij}\}_{i=1}^{S}$ implies bias of the corresponding VI approximation in the corresponding dimension of $\mathbf{W}$.

VALIDATING VI USING POSTERIOR ERROR BOUNDS Huggins et al. [36] introduces efficiently-computable theoretical bounds on variational posterior mean and variance errors. These theoretical bounds are obtained via three sequential bounds: (i) by bounding posterior mean and variance errors via Wasserstein distance between p and $q_{\boldsymbol{\theta}}$, (ii) by bounding Wasserstein distance via Rényi divergences between p and $q_{\boldsymbol{\theta}}$ using moment conditions on $q_{\boldsymbol{\theta}}$, and finally (iii) by bounding Rényi divergences using ELBO and the χ upper bound (CUBO). As ELBO and CUBO are easily computable objectives already used in the VI process, the overall bounds can be cheaply evaluated. To this end, Huggins et al. [36] also designs a practical workflow for evaluating the quality of VI approximation by computing and analyzing these bounds.

The work in this thesis is closer to [36] than [78], as we construct easily computable, theoretical error bounds rather than empirical diagnostics. However, unlike [36], our focus is specifically the posterior predictive mean error [3]. We also make use of a different metric (maximum mean discrepancy) to construct the bound.

[3] Though, we note that Proposition 3.6 in [36] also extends their bounds to the predictive setting.

3.2 Kernelized Bound on Variational Predictive Mean Error

Let us denote $\mathbf{Q} = \{q_{\boldsymbol{\theta}} \mid \boldsymbol{\theta} \in \Theta\}$ as the variational family, and assume that $\mathcal{Y} = \mathbb{R}$ (the regression setting) [4]. Let $\mathbb{E}_p[\Phi]$ and $\mathbb{E}_{q_{\boldsymbol{\theta}}}[\Phi]$ denote, respectively, the (true) posterior predictive mean and variational predictive mean. To be clear, $\mathbb{E}_p[\Phi] = \mathbb{E}_{\mathbf{W}\sim p}[\Phi_{\mathbf{W}}(\cdot)] : \mathcal{X} \to \mathcal{Y}$, where the expectation is taken w.r.t. $\mathbf{W}$, is a *function* from input space to output space (the same applies for $q_{\boldsymbol{\theta}}$).

Let $\mathcal{V} \subseteq \mathcal{X}$ be a set of points of interest in the input space. We can define the maximum error of the variational predictive mean in $\mathcal{V}$:

Definition 3.2.1. *The **variational predictive mean error** (VPME) of the approximation $q_{\boldsymbol{\theta}}$ is:*

$$\mathit{VPME}(p, q_{\boldsymbol{\theta}}) = \Big(\mathbb{E}_p[\Phi] - \mathbb{E}_{q_{\boldsymbol{\theta}}}[\Phi]\Big)^2 \tag{3.4}$$

Then VPME is upper-bounded in $\mathcal{V} \subseteq \mathcal{X}$ by the supremum of its image in $\mathcal{V}$:

$$\sup_{\mathcal{V}} \mathit{VPME}(p, q_{\boldsymbol{\theta}}) = \sup_{\mathbf{x}\in\mathcal{V}} \Big(\mathbb{E}_p[\Phi(\mathbf{x})] - \mathbb{E}_{q_{\boldsymbol{\theta}}}[\Phi(\mathbf{x})]\Big)^2 \tag{3.5}$$

Since the posterior predictive mean is a point estimate often used for actual prediction, bounding $\sup_{\mathcal{V}} \mathrm{VPME}(p, q_{\boldsymbol{\theta}})$ over some well-chosen $\mathcal{V}$ of interest (e.g. the convex hull of D_{tr}) gives us a reliable estimate of the worst-case performance of the variational approximation.

3.2.1 Maximum Mean Discrepancy

(3.5) is, in fact, related to a class of statistics known as the *maximum mean discrepancy* (MMD), first introduced in [26] and based off earlier ideas in [16]:

$$\mathrm{MMD}(p, q, \mathcal{F}) = \sup_{f\in\mathcal{F}} \Big(\mathbb{E}_p[f] - \mathbb{E}_q[f]\Big) \tag{3.6}$$

where $\mathcal{F}$ is a set of functions $f : \mathcal{W} \to \mathbb{R}$. With the right choice of $\mathcal{F}$, MMD can be used to quantify the distance between two arbitrary distributions p and q. For example, Lemma 1 in [26] states that $\mathrm{MMD}(p, q, C_b(\mathcal{W}))$, where $C_b(\mathcal{W})$ is the space of continuous bounded functions over $\mathcal{W}$, is 0 iff $p = q$. It follows that

[4] The results in this section can be extended to the classification setting.

$\text{MMD}^2(p, q, C_b(\mathcal{W}))$ is a proper metric [5] between probability distributions, where:

$$\text{MMD}^2(p, q, \mathcal{F}) = \sup_{f \in \mathcal{F}} \Big(\mathbb{E}_p[f] - \mathbb{E}_q[f] \Big)^2 \tag{3.7}$$

By viewing $\Phi_{\mathbf{w}}(\mathbf{x})$ as a function over $\mathcal{W}$ instead of $\mathcal{X}$, it is clear that $\sup_{\mathcal{V}} \text{VPME}(p, q_{\boldsymbol{\theta}}) = \text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{V}_{\mathcal{F}})$ where $\mathcal{V}_{\mathcal{F}} = \{\Phi(\mathbf{x}) \,|\, \mathbf{x} \in \mathcal{V}\}$.

Note that it is no longer important that $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{V}_{\mathcal{F}})$ is a proper (statistical) metric or divergence, since our goal is merely to evaluate it for a given choice of $\mathcal{V}_{\mathcal{F}}$. Unfortunately, it is not clear that for any arbitrary NN architecture Φ or choice of $\mathcal{V}$, the resulting MMD can be practically evaluated or approximated. Instead, following [26], we will choose to approximate $\mathcal{V}_{\mathcal{F}}$ with a reproducing kernel Hilbert space (RKHS) instead.

3.2.2 Kernelized MMD Bound for VPME

Let $\mathcal{H}$ be a RKHS of functions $f : \mathcal{W} \to \mathbb{R}$ with the reproducing kernel $k : \mathcal{W} \times \mathcal{W} \to \mathbb{R}$. Let $||\cdot||$ and $\langle \cdot, \cdot \rangle$ denote the corresponding norm and inner product of $\mathcal{H}$ [6]. Let $\mathcal{H}_r$ denote the r-ball of $\mathcal{H}$, i.e. $\mathcal{H}_r = \{f : f \in \mathcal{H}, ||f|| \leq r\}$ where $r > 0$. Readers may refer to Appendix C for a general introduction to RKHS as well as specific results used for the proofs of this section.

If $\mathcal{V}_{\mathcal{F}} \subset \mathcal{H}_r$ (for some choice of r to be determined), then $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r)$ is an upper bound for $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{V}_{\mathcal{F}})$ and consequently an upper bound for $\text{VPME}(p, q_{\boldsymbol{\theta}})$ over $\mathcal{V}$. While this is not generally true for arbitrary $\mathcal{V}_{\mathcal{F}}$ or $\mathcal{H}$, we can prove the weaker condition that $\Phi(\mathbf{x}) \in \mathcal{V}_{\mathcal{F}}$ is close to some $f \in \mathcal{H}_r$, in the sense that the difference of their expectations over p or $\boldsymbol{\theta}$ is bounded. We will choose $\mathcal{H}$ to be the RKHS with the Gaussian kernel (C.5).

Theorem 3.2.2. *Fix any $\mathbf{x} \in \mathcal{X}$. Let $\Phi(\mathbf{x}) : \mathcal{W} \to \mathbb{R}$ represent the MLP with $H \geq 1$ hidden layers of arbitrary depth and Gaussian RBF activations. Let $\mathcal{H}$ be a RKHS of functions $f : \mathcal{W} \to \mathbb{R}$ with the Gaussian kernel $k(\mathbf{w}, \mathbf{w}') = \exp\Big\{ -\frac{||\mathbf{w}-\mathbf{w}'||_2^2}{2\sigma_k^2} \Big\}$ with characteristic length σ_k. Then for any measure p over $\mathcal{W}$, there exists some $\bar{f} \in \mathcal{H}_r$ such that*

$$\Big| \mathbb{E}_p[\bar{f}] - \mathbb{E}_p[\Phi(\mathbf{x})] \Big| \leq \mathbb{E}_p\Big[\sum_l \big| \mathbf{W}_l \big| \Big] \tag{3.8}$$

[5] A statistical distance d is a *metric* if: (i) $d(p, q) = d(q, p)$, (ii) $d(p, q) = 0$ iff $p = q$, (iii) $d(p, q) \geq 0$ and (iv) $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality). A *divergence* need not satisfy (iii) or (iv).

[6] We will explicitly use subscripts to denote other norms or inner products, e.g. $|| \cdot ||_2$ for the Euclidean norm.

where $\mathbf{W}_l$ are the parameters from the last hidden layer $\mathbf{L}_H$ to the output node,

$$r = \sqrt{\frac{|\mathbf{L}_H|}{2B_k^{1+M/2}}} \tag{3.9}$$

and B_k is a function over σ_k, defined below in (3.14).

Theorem 3.2.2 implies that $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{V}_{\mathcal{F}})$ is bounded by $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r)$ plus factors of $\mathbb{E}_p\Big[\sum_l \big|\mathbf{W}_l\big|\Big]$ and $\mathbb{E}_{q_{\boldsymbol{\theta}}}\Big[\sum_l \big|\mathbf{W}_l\big|\Big]$, which will not be too vast if the last layer is not too wide. Note that Theorem 3.2.2 holds true for all $\mathbf{x} \in \mathcal{X}$, and so the choice of r is independent of $\mathcal{V}_{\mathcal{F}}$. We will prove Theorem 3.2.2 using the spectral representation of $\mathcal{H}$. To do so, let us first introduce the following lemma.

Lemma 3.2.3. *Let μ be the standard isotropic Gaussian measure $\mathcal{N}(\mathbf{0}, \mathbf{I})$ over $\mathcal{W}$. The Gaussian kernel $k(\mathbf{w}, \mathbf{w}') = \exp\Big\{-\frac{||\mathbf{w}-\mathbf{w}'||_2^2}{2\sigma_k^2}\Big\}$ has the eigenexpansion:*

$$k(\mathbf{w}, \mathbf{w}') = \sum_{j_1 \in \mathbb{N}, \ldots, j_M \in \mathbb{N}} \lambda_{(j_1,\ldots,j_M)} \mathbf{e}_{(j_1,\ldots,j_M)}(\mathbf{w}) \mathbf{e}_{(j_1,\ldots,j_M)}(\mathbf{w}') \tag{3.10}$$

where

$$\lambda_{(j_1,\ldots,j_M)} = B_k^{M/2} (2B_k)^{j_1+\cdots+j_M} \tag{3.11}$$

$$\mathbf{e}_{(j_1,\ldots,j_M)}(\mathbf{w}) = \exp\{-C_k||\mathbf{w}||_2^2\} \prod_{m=1}^{M} H_{j_m}\Big(\sqrt{2C_k + \frac{1}{2}}\mathbf{w}_m\Big) \tag{3.12}$$

is an orthonormal basis of $L^2(\mathcal{W}, \mu)$, and

$$H_j(w) = (-1)^j \exp\{w^2\} \frac{d^j}{dw^j} \exp\{-w^2\} \tag{3.13}$$

are the Hermite polynomials [7] of order j, and

$$B_k = \frac{2\sigma_k^2}{\sigma_k^2 + 2 + \sigma_k\sqrt{4 + \sigma_k^2}} \tag{3.14}$$

$$C_k = \frac{\sqrt{4 + \sigma_k^2} - \sigma_k}{4\sigma_k} \tag{3.15}$$

are functions of σ_k. We define $\mathbb{N} = \{0, 1, \ldots\}$ to start at 0.

[7] Note that the Hermite polynomials have one-dimensional inputs.

Proof. See Appendix G. □

Proof of Theorem 3.2.2. From Theorem C.0.8, the spectral representation of $\mathcal{H}$ is

$$\mathcal{H} = \left\{ f = \sum_{j_1 \in \mathbb{N}, \ldots, j_M \in \mathbb{N}} a_{(j_1,\ldots,j_M)} \mathbf{e}_{(j_1,\ldots,j_M)}(\mathbf{w}) : \left\{ \frac{a_{(j_1,\ldots,j_M)}}{\sqrt{\lambda_{(j_1,\ldots,j_M)}}} \right\} \in \ell^2 \right\} \tag{3.16}$$

with the eigenbasis as given by Lemma 3.2.3.

Fix any $\mathbf{x} \in \mathcal{X}$. (Note that any Φ below is implicitly a function of $\mathbf{x}$ too.) Let l index all the nodes in the last hidden layer of BNN. Denote $u_l \in \mathbf{L}_H$ and $\mathbf{w}_l$ to be the hidden-to-output parameter corresponding to each u_l. Then:

$$\Phi(\mathbf{w}) = \sum_l \mathbf{w}_l \cdot \exp\{-u_l(\mathbf{w}, \mathbf{x})^2\} \tag{3.17}$$

where we write u_l as a function of $\mathbf{w}$ and $\mathbf{x}$ to make it clear that the value of these nodes depend on the input and the parameters of the previous layers. Now, consider the eigenfunction where $j_l = 1$ and $j_m = 0$ for all $m \neq l$. Then:

$$\mathbf{e}_l(\mathbf{w}) = \mathbf{w}_l \cdot \exp\{-C_k ||\mathbf{w}||_2^2\} \tag{3.18}$$

where we abuse notation and use l to denote the M-tuple where $j_i = \delta_{il}$ (the Kronecker delta) [8]. Consider the function $\bar{f}(\mathbf{w}) = \sum_l \mathbf{e}_l(\mathbf{w})$, i.e. the coefficients a for all other eigenfunctions not of the form (3.18) are set to 0. $\bar{f} \in \mathcal{H}$ since there are only $|\mathbf{L}_H|$ coefficients $a_l = 1$ and so the convergence condition in (3.16) is satisfied.

Let p be an arbitrary distribution over $\mathcal{W}$. We have:

$$\begin{aligned}
\Big|\mathbb{E}_p[\bar{f}] - \mathbb{E}_p[\Phi]\Big| &= \Big|\mathbb{E}_p[\bar{f} - \Phi]\Big| \\
&\leq \mathbb{E}_p\big[|\bar{f} - \Phi|\big] \\
&= \int_{\mathcal{W}} \Big| \sum_l \mathbf{w}_l \Big(\exp\{-C_k ||\mathbf{w}||_2^2\} - \exp\{-u_l(\mathbf{w}, \mathbf{x})^2\} \Big) \Big| \, p(\mathbf{w}) \mathrm{d}\mathbf{w} \\
&\leq \int_{\mathcal{W}} \sum_l \Big| \mathbf{w}_l \Big(\exp\{-C_k ||\mathbf{w}||_2^2\} - \exp\{-u_l(\mathbf{w}, \mathbf{x})^2\} \Big) \Big| \, p(\mathbf{w}) \mathrm{d}\mathbf{w} \\
&\leq \int_{\mathcal{W}} \sum_l \big| \mathbf{w}_l \big| \, p(\mathbf{w}) \mathrm{d}\mathbf{w} \\
&= \mathbb{E}_p\Big[\sum_l \big| \mathbf{W}_l \big| \Big]
\end{aligned}$$

[8] (3.18) follows as $H_0(w) = 1$ and $H_1(w) = w$.

The norm of $\bar{f}$ gives us the choice of r:

$$\begin{aligned} r = ||\bar{f}|| &= \sqrt{\langle \sum_l \mathbf{e}_l, \sum_l \mathbf{e}_l \rangle} \\ &= \sqrt{\sum_l \frac{1}{\lambda_l}} \\ &= \sqrt{\sum_l \frac{1}{2B_k^{1+M/2}}} \\ &= \sqrt{\frac{|\mathbf{L}_H|}{2B_k^{1+M/2}}} \end{aligned}$$

□

Corollary 3.2.4. *Theorem 3.2.2 can be applied to a BNN with any non-linear activation bounded in* $[0, 1]$.

Proof. We can modify the derivation of $\left|\mathbb{E}_p[\bar{f}] - \mathbb{E}_p[\Phi(\mathbf{x})]\right|$ above, except replacing $\exp\{-u_l(\mathbf{w}, \mathbf{x})^2\}$ with $\sigma\{u_l(\mathbf{w}, \mathbf{x})\}$ for some bounded activation σ. (It is straightforward to extend this by a multiplicative factor to activations bounded in $[a, b]$). □

Using Theorem 3.2.2, we can bound $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{V}_{\mathcal{F}})$ as:

$$\begin{aligned} &\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{V}_{\mathcal{F}}) \\ &= \sup_{\mathbf{x} \in \mathcal{V}} \Big(\mathbb{E}_p[\Phi(\mathbf{x})] - \mathbb{E}_{q_{\boldsymbol{\theta}}}[\Phi(\mathbf{x})]\Big)^2 \\ &\leq \left(\left|\mathbb{E}_p[\bar{f}] - \mathbb{E}_{q_{\boldsymbol{\theta}}}[\bar{f}]\right| + \mathbb{E}_p\Big[\sum_l \left|\mathbf{W}_l\right|\Big] + \mathbb{E}_{q_{\boldsymbol{\theta}}}\Big[\sum_l \left|\mathbf{W}_l\right|\Big]\right)^2 \\ &= \Big(\mathbb{E}_p[\bar{f}] - \mathbb{E}_{q_{\boldsymbol{\theta}}}[\bar{f}]\Big)^2 + 3\max\left(\Big(\mathbb{E}_p[\bar{f}] - \mathbb{E}_{q_{\boldsymbol{\theta}}}[\bar{f}]\Big)^2, \Big(\mathbb{E}_p\Big[\sum_l \left|\mathbf{W}_l\right|\Big] + \mathbb{E}_{q_{\boldsymbol{\theta}}}\Big[\sum_l \left|\mathbf{W}_l\right|\Big]\Big)^2, 1\right) \\ &\leq \text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r) + 3\max\left(\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r), \Big(\mathbb{E}_p\Big[\sum_l \left|\mathbf{W}_l\right|\Big] + \mathbb{E}_{q_{\boldsymbol{\theta}}}\Big[\sum_l \left|\mathbf{W}_l\right|\Big]\Big)^2, 1\right) \end{aligned} \tag{3.19}$$

Unfortunately, the exact difference between $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{V}_{\mathcal{F}})$ and $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r)$ is not fully computable since $\mathbb{E}_p\Big[\sum_l \left|\mathbf{W}_l\right|\Big]$ is unknown. However, it may be that $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r) >> \Big(\mathbb{E}_p[\bar{f}] - \mathbb{E}_{q_{\boldsymbol{\theta}}}[\bar{f}]\Big)^2$ and so is ultimately still a far larger term than the difference.

It remains to find a way to compute $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r)$ itself. Assuming the mild condition that the kernel has finite expectation (which is true for the Gaussian kernel), we can derive a closed-form expression for $\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r)$. Indeed, the existence of an analytical solution to the MMD is precisely the motivation for using RKHS as our choice of $\mathcal{F}$.

Proposition 3.2.5. *If $\mathbb{E}_{\mathbf{W}\sim p}[k(\mathbf{W}, \mathbf{W})] < \infty$ and $\mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}[k(\mathbf{W}, \mathbf{W})] < \infty$, then*

$$MMD^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r) = r^2\Big(\mathbb{E}_{p,p}[k(\cdot, \cdot\cdot)] - 2 \cdot \mathbb{E}_{p,q}[k(\cdot, \cdot\cdot)] + \mathbb{E}_{q,q}[k(\cdot, \cdot\cdot)]\Big) \tag{3.20}$$

*where $\mathbb{E}_{p,p}[k(\cdot, \cdot\cdot)] = \mathbb{E}_{\mathbf{W}\sim p, \mathbf{W}'\sim p}[k(\mathbf{W}, \mathbf{W}')]$ and the expectation is taken w.r.t. the product distribution of **independent** copies of p. The same applies for the other two expectation terms above.*

Proposition 3.2.5 is a straightforward adaptation of Lemmas 4 and 6 found in [27] [9] for variable RKHS radius r. In order to generalize the results in [27] to variable r, we will first require the following lemmas:

Lemma 3.2.6 (Lemma 3 of [27], adapted)**.** *If $\mathbb{E}_{\mathbf{W}\sim p}[k(\mathbf{W}, \mathbf{W})] < \infty$, then $\exists\, \mu_p \in \mathcal{H}$ such that $\mathbb{E}_p[f] = \langle f, \mu_p \rangle, \quad \forall\, f \in \mathcal{H}$. Furthermore,*

$$\mu_p(\cdot) = \mathbb{E}_{\mathbf{W}\sim p}[k(\mathbf{W}, \cdot)] \tag{3.21}$$

Proof. See Appendix G. □

Lemma 3.2.7. *Fix an arbitrary $g \in \mathcal{H}$. Then:*

$$\sup_{f \in \mathcal{H}_r} \langle f, g \rangle^2 = r^2 ||g||^2 \tag{3.22}$$

Proof. See Appendix G. □

[9] [27] contains detailed proofs of the ideas proposed in [26].

Proof of Proposition 3.2.5. We have that:

$$
\begin{aligned}
\text{MMD}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r) &= \sup_{f \in \mathcal{H}_r} \Big(\mathbb{E}_p[f] - \mathbb{E}_{q_{\boldsymbol{\theta}}}[f] \Big)^2 \\
&= \sup_{f \in \mathcal{H}_r} \Big(\langle f, \mu_p \rangle - \langle f, \mu_q \rangle \Big)^2 \quad \text{(Lemma 3.2.6)} \\
&= \sup_{f \in \mathcal{H}_r} \langle f, \mu_p - \mu_q \rangle^2 \\
&= r^2 ||\mu_p - \mu_q||^2 \quad \text{(Lemma 3.2.7)} \\
&= r^2 \cdot \Big(\langle \mu_p, \mu_p \rangle - 2\langle \mu_p, \mu_q \rangle + \langle \mu_q, \mu_q \rangle \Big) \\
&= r^2 \cdot \Big(\big\langle \mathbb{E}_{\mathbf{W} \sim p}[k(\mathbf{W}, \cdot)], \mathbb{E}_{\mathbf{W} \sim p}[k(\mathbf{W}, \cdot)] \big\rangle - 2 \big\langle \mathbb{E}_{\mathbf{W} \sim p}[k(\mathbf{W}, \cdot)], \mathbb{E}_{\mathbf{W} \sim q_{\boldsymbol{\theta}}}[k(\mathbf{W}, \cdot)] \big\rangle \\
&\quad + \big\langle \mathbb{E}_{\mathbf{W} \sim q_{\boldsymbol{\theta}}}[k(\mathbf{W}, \cdot)], \mathbb{E}_{\mathbf{W} \sim q_{\boldsymbol{\theta}}}[k(\mathbf{W}, \cdot)] \big\rangle \Big) \\
&= r^2 \cdot \Big(\mathbb{E}_{\mathbf{W} \sim p, \mathbf{W}' \sim p}[\big\langle k(\mathbf{W}, \cdot), k(\mathbf{W}', \cdot) \big\rangle] - 2\mathbb{E}_{\mathbf{W} \sim p, \mathbf{W}' \sim q_{\boldsymbol{\theta}}}[\big\langle k(\mathbf{W}, \cdot), k(\mathbf{W}', \cdot) \big\rangle] \\
&\quad + \mathbb{E}_{\mathbf{W} \sim q_{\boldsymbol{\theta}}, \mathbf{W}' \sim q_{\boldsymbol{\theta}}}[\big\langle k(\mathbf{W}, \cdot), k(\mathbf{W}', \cdot) \big\rangle] \Big) \\
&= r^2 \cdot \Big(\mathbb{E}_{p,p}[k(\cdot, \cdot)] - 2\mathbb{E}_{p,q}[k(\cdot, \cdot)] + \mathbb{E}_{q,q}[k(\cdot, \cdot)] \Big)
\end{aligned}
$$

Note that one function that achieves the supremum above is $f^* = r(\mu_p - \mu_q)$. $\square$

Remark. *The condition in Proposition 3.2.5 applies to the Gaussian kernel:*

$$
\mathbb{E}_{\mathbf{W} \sim p} \left[\exp \Big\{ - \frac{||\mathbf{W} - \mathbf{W}||_2^2}{2\sigma_k^2} \Big\} \right] = 1 \tag{3.23}
$$

for any distribution p over $\mathcal{W}$. In fact, it holds for any stationary kernel, i.e. $k(\mathbf{w}, \mathbf{w}')$ is a function of $\mathbf{w} - \mathbf{w}'$.

3.2.3 Computing the Kernelized MMD

We need to be able to evaluate (3.20) in practice. As an analytical solution is intractable, and as we cannot sample from p, we will use the importance-sampled Monte Carlo estimator with samples from $q_{\boldsymbol{\theta}}$:

$$
\begin{aligned}
&\widehat{\text{MMD}}^2(p, q_{\boldsymbol{\theta}}, \mathcal{H}_r) \\
&= r^2 \cdot \sum_{i,j} \frac{\gamma_i \gamma_j k(\mathbf{w}^{(i)}, \mathbf{w}^{(j)})}{\Big(\sum_i \gamma_i \Big) \Big(\sum_j \gamma_j \Big)} - \frac{\gamma_i k(\mathbf{w}^{(i)}, \mathbf{w}^{(j')})}{\sum_i \gamma_i} - \frac{\gamma_j k(\mathbf{w}^{(i')}, \mathbf{w}^{(j)})}{\sum_j \gamma_j} + k(\mathbf{w}^{(i')}, \mathbf{w}^{(j')})
\end{aligned} \tag{3.24}
$$

where $\{\mathbf{w}^{(i)}\}_{i=1}^S$ and $\{\mathbf{w}^{(i')}\}_{i=1}^S$ are two independent sets of S samples from $q_{\boldsymbol{\theta}}$ (hence we draw $2S$ samples independently) [10]. $\{\gamma_i\}_{i=1}^S$ are the importance ratios as defined in (3.1), applicable to the first set of samples $\{\mathbf{w}^{(i)}\}_{i=1}^S$ only. In general, any unbiased variance reduction technique can be applied in lieu of (3.24).

We will refer to our general approach, where $\mathcal{H}$ is constructed from the Gaussian kernel, as the **Kernelized Bound on Variational Predictive Mean Error (KEBO-VPME)**. We denote (3.19) as the *strict upper bound* and (3.24) (or any other viable estimator) as the *KEBO-VPME estimator*.

There are a number of ways that KEBO-VPME can be deployed practically. For example, we can compute (3.24) at the end of each run and choose $\boldsymbol{\theta}$ corresponding to the VI run with the smallest KEBO-VPME. If the user possesses domain knowledge about the maximum tolerable predictive mean error, they can reject any VI run for which KEBO-VPME exceeds this value.

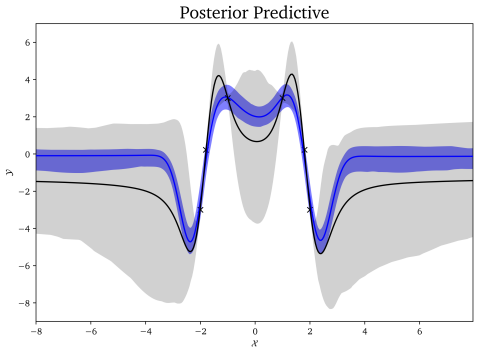
3.3 Low-Dimensional Simulations

We evaluate KEBO-VPME on a simple low-dimensional example to empirically explore various aspects of its performance.

General Experimental Setup Throughout this section, we will consider the same 1D ($\mathcal{X} = \mathbb{R}$) regression dataset in Figure 2.5.1. We use the naive Gaussian prior (1.14) throughout. We use BBB with a Gaussian variational approximation as the VI method. The empirical predictive distribution is computed according to (1.17). Refer to Appendix F for hyperparameters and setup details.

We will evaluate KEBO-VPME by using the HMC posterior as a proxy for the ground-truth posterior. We will compare both the KEBO-VPME estimator (3.24) as well as the *strict* upper bound (3.19) to the true maximum predictive mean error in $\mathbb{R}$ (3.5). To compute (3.19), we will use the samples from HMC and BBB to estimate $\mathbb{E}_p\Big[\sum_l |\mathbf{W}_l|\Big]$ and $\mathbb{E}_{q_{\boldsymbol{\theta}}}\Big[\sum_l |\mathbf{W}_l|\Big]$ respectively. Note also that we do not need to carry out importance sampling in (3.24) since we are substituting the first set of $\{\mathbf{w}^{(i)}\}_{i=1}^S$ with actual HMC samples, i.e. we set all $\gamma_i = 1$. Finally, while the definitions (3.5), (3.19) and (3.24) use the *squared* error of expectation, in this

[10] The first set of S samples is meant to simulate samples from p, whereas the second set represents samples from $q_{\boldsymbol{\theta}}$ itself.



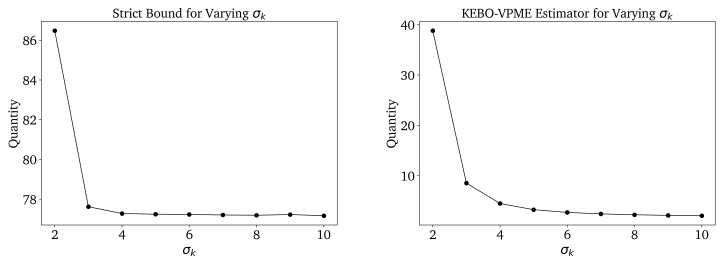
Metric	Value
$\sup_{\mathcal{X}} \text{VPME}(p, q_{\boldsymbol{\theta}})$	2.292
KEBO-VPME Est. (3.24)	3.270
Strict Bound (3.19)	77.213

Figure 3.3.1 & Table 3.3.1: (a) The posterior predictive plot for HMC (gray) and BBB (blue), showing the posterior predictive mean (thick line) and shaded at the credible interval $\sigma = 3$. The VI predictive mean function deviates from the HMC function at several points in $\mathcal{X}$. Note that the presence of a nonlinear activation squashing extreme values of $\mathbf{x}$ means that the predictive error will not diverge beyond what this graph shows. **(b)** From top to bottom, values for (i) the true VPME, i.e. maximum deviation of the predictive mean functions, (ii) the KEBO-VPME estimator (3.24) and (iii) the strict upper bound (3.19).

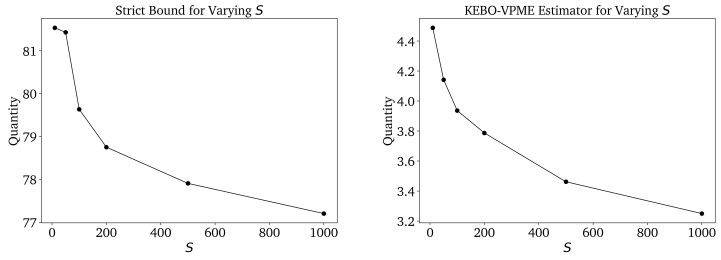
section (and the next) we apply the square root to all 3 values, so that they can be compared on the actual scale of $\mathcal{Y}$.

With access to the true posterior simulated, (3.19) indeed bounds the maximum predictive mean error. Table 3.3.1 shows that the strict upper bound, which we have proved is a true bound, is a highly conservative bound, being an order of magnitude higher than the true maximum predictive mean error. In this case, the KEBO-VPME estimator is also slightly higher than the true maximum error, even though it *may not* necessarily have been so. This is because (3.24) is not a strict bound for VPME (only (3.19) is) and there are no guarantees when comparing (3.24) and $\text{VPME}(p, q_{\boldsymbol{\theta}}, \mathbb{R})$ directly.

An appropriate large choice of σ_k (the characteristic length of the Gaussian kernel) **is sufficient for convergence of the KEBO-VPME estimator.** Figure 3.3.2a and 3.3.2b shows the plot of the KEBO-VPME estimator and the strict bound computed at varying choices of σ_k. Both quantities rapidly converge as σ_k increases. This is in line with what we might expect theoretically. Recall that σ_k factors into KEBO-VPME by determining r as well as the expectation terms over $k(\cdot, \cdot)$. As $\sigma_k \to \infty$,



(a) Plot of (3.19) against σ_k. **(b)** Plot of (3.24) against σ_k.



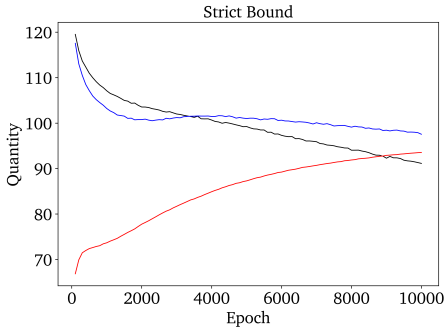
(c) Plot of (3.19) against S. **(d)** Plot of (3.24) against S.

Figure 3.3.2: (a) and **(b)**, computed at fixed $S = 1000$, show that both (3.19) and (3.24) converge as σ_k increases. **(c)** and **(d)**, computed at fixed $\sigma_k = 5$, show that (3.19) and (3.24) converge as S increases.

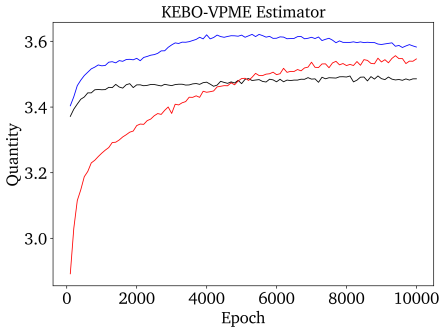
both terms converge: $r \to \sqrt{\frac{|\mathbf{L}_H|}{2B_k}}$ and $\mathbb{E}_p[k(\cdot,\cdot)] \to 1$. This also affirms that the values in Table 3.3.1, which were computed at $\sigma_k = 5$, are reliable.

KEBO-VPME is sample-efficient. Figure 3.3.2c and 3.3.2d shows the plot of the KEBO-VPME estimator and the strict bound computed at varying sample size S (as defined in (3.24)). Like the plots for σ_k, both quantities converge as S increases. KEBO-VPME is therefore computationally cheap and can be easily added the VI workflow.

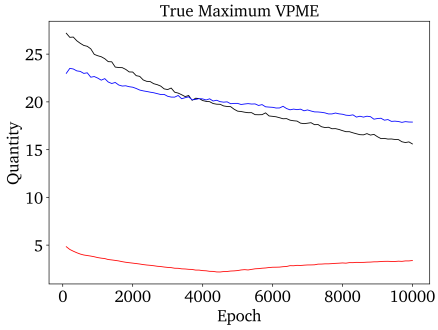
KEBO-VPME is not able to distinguish between VI runs well. Unfortunately, while KEBO-VPME serves as a theoretical bound for the predictive mean error, it is not tight enough to distinguish between VI approximations when the differences are small, nor does it serve as a good diagnostic for the progress of the variational optimization across epochs. Figure 3.3.3 shows 3 separate VI runs, where the initial parameter $\boldsymbol{\theta}$ is intentionally chosen to result in different local VI modes obtained. While the strict bound still accurately bounds the maximum predictive mean error, the plot does not serve to inform us which run is better. The KEBO-VPME estimator is also poor at distinguishing between VI runs.



(a) Plot of (3.19) per epoch.



(b) Plot of (3.24) per epoch.



(c) Plot of maximum VPME per epoch.

Figure 3.3.3: In each of the plots, the 3 lines (red, blue, black) denote 3 separate VI runs with different initial θ. The strict bound plot seems to partially mirror the true maximum VPME for two of the runs, but does not reflect the change in maximum error for the VI run in red. In constrast, the KEBO-VPME estimator has no distinguishing power, and in fact does not bound the true error for the VI runs in black and blue.

4

Conclusion

BNNs are obtained by performing Bayesian inference over deep NNs, a class of models representing a rich hypothesis space. BNNs combine powerful function approximation with the ability to reason systematically about parameter and predictive uncertainty. This makes them ideal models to deploy in high-stakes applications, where we desire to learn accurate predictions from high-dimensional, large datasets, but without the cost-prohibitive errors that classical NNs tend to make on out-of-distribution data.

The ability to learn entire distributions over NNs does not come without tradeoffs, namely, model interpretability as well as inference accuracy. In this thesis, we presented two approaches that respectively mitigate each of these problems. OC-BNNs, in Chapter 2, allow the user to express output constraints via two novel prior formulations, both of which are amenable to black-box BNN inference methods. They form a bridge between interpretable domain knowledge that end users can easily specify and the powerful but opaque BNN inference process. Output constraints can be used to enforce a wide range of desiderata of practical concerns, such as safety and fairness. We applied OC-BNNs to two high-stakes domains: (1) for healthcare, we imposed physiologically-feasible constraints (that a doctor might specify) on a clinical action prediction task, and (2) for criminal justice, we imposed racial fairness constraints on a recidivism prediction task. For the latter, the constraint was enforced despite a biased and unfair dataset.

For BNNs to be truly useful for high-stakes applications, it is crucial that we can evaluate and therefore trust the approximate posterior. In Chapter 3, we addressed the issue of verifying the predictive accuracy of VI approximations by introducing KEBO-VPME, a bound for the posterior predictive mean error using the kernelized maximum mean discrepancy. While KEBO-VPME is largely useful for as a theoretical guarantee, we note that it is efficiently estimable and opens the door to exploiting RKHS for future BNN diagnostics.

4.1 Future Directions

There is much work to be done in making BNNs expressive and trustworthy models for widespread, practical use. Below, we outline some promising directions of research, either immediate extensions of our work or more general approaches:

Better sampling techniques can be leveraged to improve the sample efficiency of OC-BNNs. OC-BNNs are less effective as the constrained input region $\mathcal{C}_{\mathbf{x}}$ grows, due to the difficulty of obtaining representative samples from $\mathcal{C}_{\mathbf{x}}$. By making use of advanced sampling or variance reduction techniques, e.g. antithetic sampling [22], the variance of $p_{\mathcal{C}}(\mathbf{w})$ can be managed, which is especially crucial for high-dimensional $\mathcal{X}$ as representative sampling becomes exponentially more difficult as dimensionality Q increases. We note that work done in Chapter 2 has shown promising directions. For example, we have observed empirically that at higher dimensions, sampling at the boundary of $\mathcal{C}_{\mathbf{x}}$ closest to D_{tr} is sometimes sufficient to learn the constraint well. This is likely due to the fact that non-Lipschitz continuous functions, i.e. those that satisfy points at boundary of $\mathcal{C}_{\mathbf{x}}$ but not the rest of $\mathcal{C}_{\mathbf{x}}$ (assuming small, compact $\mathcal{C}_y$), are penalized heavily by the prior term. As such, we hypothesize that by controlling the continuity of $\Phi_{\mathbf{W}}$ learnt (e.g. by varying the standard deviation hyperparameter σ_ω in (1.14)), sampling from the boundary may be sufficient to learn the whole $\mathcal{C}$.

Priors should be able to express more complicated functional beliefs. In general, the ability to translate between NN parameter space and function space is a fundamental challenge for BNNs. While a panacea is not immediately apparent, it is possible to chip away at this problem depending on the domain of interest. For example, while this work considers knowledge in the form of output constraints, there are desirable properties such as monotonicity that cannot be easily represented as such. Exploiting the structure $\Phi_{\mathbf{W}}$ can allow us to construct tentative relationships between $\mathcal{W}$ and function spaces of interest.

OC-BNNs can be used with human-in-the-loop processes to actively incorporate human expert knowledge in the BNN training process. Since OC-BNNs allow for human input in the form of output constraints, it is possible to envision a human-in-the-loop machine learning system where a human expert is tasked with specifying constraints for regions or points in $\mathcal{X}$ that will result in the most informative BNN posterior (e.g. w.r.t. a fixed D_{tr}). This is a direction that we are actively working on exploring. Preliminary results show that querying for output constraints at strategic points in $\mathcal{X}$ can lead to better posteriors, i.e. posteriors with (correctly) reduced variance even at regions not represented in D_{tr}.

Kernelized bounds can be extended to predictive variance error. Since VI approximations are known to underestimate the true predictive variance, bounding the variance error will be a helpful diagnostic. We might seek an equivalent of KEBO-VPME for the variance, i.e. approximating $\sup_f \Big(Var_p[f] - Var_{q_{\boldsymbol{\theta}}}[f]\Big)^2$ with an RKHS. Evaluating such a quantity will be more difficult due to the nonlinearity of $Var(\cdot)$. However, by expressing $Var_p[f] = \mathbb{E}_p[f^2] - \mathbb{E}_p[f]^2$ and making certain assumptions on f^2 or $\mathbb{E}_p[f]$, it might be possible to derive a bound. For example, we can prove that $\Phi^2_{\mathbf{W}}$ can be represented in some RKHS, or we can consider the special case where $\mathbb{E}_p[\Phi] = 0$. In general, it is worth asking if the moment errors are interesting.

Empirical diagnostics can be constructed for VI approximations. In general, tight theoretical bounds for variational predictive errors can be difficult to achieve, due to both the complexity of $\Phi_{\mathbf{W}}$ and the resulting complicated, multimodal posterior. It is worth exploring if there are empirical diagnostics that can be practically useful for gauging certain aspects of VI quality, especially in high dimensions. For example, as we expect variance to increase in regions of $\mathcal{X}$ further away from D_{tr}, we want to observe whether predictive variance changes appropriately across $\mathcal{X}$.

For machine learning systems to be useful, they must be faithful to the needs of the human user. We desire our models to be expressive enough to support a wide range of complicated human beliefs or desiderata and trustworthy enough such that using them will do no harm. It is my hope that the work presented in this thesis contributes towards these goals.

A

Derivation of ELBO

Beginning with (1.19), we have:

$$
\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}\in\Theta} D_{KL}\big(p(\cdot|D_{tr}) \,||\, q_{\boldsymbol{\theta}}(\cdot)\big) \\
&= \arg\min_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[\log \frac{q_{\boldsymbol{\theta}}(\mathbf{W})}{p(\mathbf{W}|D_{tr})}\Big] \\
&= \arg\min_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[\log q_{\boldsymbol{\theta}}(\mathbf{W}) - \log p(\mathbf{W}|D_{tr})\Big] \\
&= \arg\min_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[\log q_{\boldsymbol{\theta}}(\mathbf{W}) - \log p(\mathbf{W}) - \log p(D_{tr}|\mathbf{W}) + \log p(D_{tr})\Big] \\
&= \arg\min_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[\log q_{\boldsymbol{\theta}}(\mathbf{W}) - \log p(\mathbf{W}) - \log p(D_{tr}|\mathbf{W})\Big] \\
&\quad \text{(since } p(D_{tr}) \text{ is a constant independent of } \boldsymbol{\theta}\text{)} \\
&= \arg\max_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[-\log q_{\boldsymbol{\theta}}(\mathbf{W}) + \log p(\mathbf{W}) + \log p(D_{tr}|\mathbf{W})\Big] \\
&= \arg\max_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[\log p(D_{tr}|\mathbf{W})\Big] + \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[\log p(\mathbf{W}) - \log q_{\boldsymbol{\theta}}(\mathbf{W})\Big] \\
&= \arg\max_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{\mathbf{W}\sim q_{\boldsymbol{\theta}}}\Big[\log p(D_{tr}|\mathbf{W})\Big] + D_{KL}\big(p(\cdot) \,||\, q_{\boldsymbol{\theta}}(\cdot)\big)
\end{aligned}
$$

An alternative derivation using Jensen's inequality is also possible, beginning with (1.10):

$$
\begin{aligned}
\log p(D_{tr}) &= \log \int_{\mathcal{W}} p(\mathbf{w}) p(D_{tr}|\mathbf{w}) \, \mathrm{d}\mathbf{w} \\
&= \log \int_{\mathcal{W}} q_{\boldsymbol{\theta}}(\mathbf{w}) \cdot \frac{p(\mathbf{w}) p(D_{tr}|\mathbf{w})}{q_{\boldsymbol{\theta}}(\mathbf{w})} \, \mathrm{d}\mathbf{w} \\
&= \log \mathbb{E}_{\mathbf{W} \sim q_{\boldsymbol{\theta}}} \Big[\frac{p(\mathbf{W}) p(D_{tr}|\mathbf{W})}{q_{\boldsymbol{\theta}}(\mathbf{W})} \Big] \\
&\geq \mathbb{E}_{\mathbf{W} \sim q_{\boldsymbol{\theta}}} \Big[\log \frac{p(\mathbf{W}) p(D_{tr}|\mathbf{W})}{q_{\boldsymbol{\theta}}(\mathbf{W})} \Big] \\
&\quad \text{(Jensen's inequality)} \\
&= \mathbb{E}_{\mathbf{W} \sim q_{\boldsymbol{\theta}}} \Big[\log p(D_{tr}|\mathbf{W}) \Big] + D_{KL}\big(p(\cdot) \,||\, q_{\boldsymbol{\theta}}(\cdot)\big)
\end{aligned}
$$

Since $p(D_{tr})$ is constant, optimizing for $\boldsymbol{\theta}$ is equivalent to maximizing ELBO.

B

Inference Algorithms for BNNs

B.1 Hamiltonian Monte Carlo

We assume familiarity with the Metropolis-Hastings algorithm [29], a basic MCMC method. HMC, first introduced in [13], is an MCMC variant that augments the Metropolis-Hastings framework by using Hamiltonian dynamics for Markov chain transition proposals, which cover more of the parameter space $\mathbb{R}^M$ than naive Gaussian proposals. This is important for sample-efficient BNN inference since M is large. The treatment below is a concise summary from [57], which is a theoretical and practical review of HMC for BNN inference. Readers are referred to [57] for a more substantial discussion.

Hamiltonian Dynamics

MCMC algorithms iteratively collect samples of $\mathbf{W} \in \mathcal{W} = \mathbb{R}^M$. We can interpret this as a physical system describing $\mathbf{W}$ as a particle in motion, whereby $\mathbf{q} := \mathbf{W}$ is the position vector on $\mathcal{W}$ and each iteration of MCMC updates $\mathbf{q}$. Then Hamiltonian dynamics describes how the position $\mathbf{q}$ and the momentum $\mathbf{p}$ of the particle changes over time, under the constraint of:

$$\frac{\mathrm{d}\mathbf{q}_j}{\mathrm{d}t} = (\mathbf{M}^{-1}\mathbf{p})_j \tag{B.1}$$

$$\frac{\mathrm{d}\mathbf{p}_j}{\mathrm{d}t} = -\frac{\partial U}{\partial \mathbf{q}_j} \tag{B.2}$$

where t represents time and $j \in \{0, 1, \ldots, M-1\}$ indexes $\mathbf{q}$ and $\mathbf{p}$. $U(\mathbf{q})$ is the potential energy of the system, which we define as the negative log-posterior probability:

$$U(\mathbf{q}) = -\big(\log p(\mathbf{q}) + \log p(D_{tr}|\mathbf{q})\big) \tag{B.3}$$

$\mathbf{M}$ represents any symmetric, positive-definite matrix corresponding to particle mass. We will choose a diagonal $\mathbf{M}$ with entries $m_0, \ldots, m_{M-1}$. The kinetic energy of the system is

$$K(\mathbf{p}) = \frac{1}{2}\mathbf{p}^\top \mathbf{M}^{-1}\mathbf{p} = \sum_{d=j}^{M} \frac{\mathbf{p}_j^2}{2m_i} \tag{B.4}$$

The Hamiltonian is defined as

$$H(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p}) \tag{B.5}$$

and can be interpreted as the total energy of the system. The goal is therefore to solve (B.1) and (B.2) to obtain some operator $\mathbf{T}_\delta$ that maps the physical state at time t, $(\mathbf{q}, \mathbf{p})_t$, to the physical state at time $t + \delta$, $(\mathbf{q}, \mathbf{p})_{t+\delta}$. Note that the auxiliary momentum vector $\mathbf{p} \in \mathbb{R}^M$ is introduced artificially and so $\mathbf{T}_\delta$ operates on $\mathbb{R}^{M+M}$.

The Hamiltonian is related to the posterior, our distribution of interest, by means of the potential energy $U(\mathbf{q})$ [1]. In statistical mechanics, the energy function (B.5) results in a *canonical distribution* over $(\mathbf{q}, \mathbf{p})$:

$$p(\mathbf{q}, \mathbf{p}) \propto \exp\Big\{ -\frac{U(\mathbf{q})}{T} \Big\} \exp\Big\{ -\frac{K(\mathbf{p})}{T} \Big\} \tag{B.6}$$

where T is the temperature of the system (which we typically set to 1). $\mathbf{q}$ and $\mathbf{p}$ are independent, and therefore the canonical distribution for $\mathbf{q}$, our desired variable, is precisely the posterior. Note that our choice of K above implies an isotropic Gaussian distribution for the momentum $\mathbf{p}$ where m_j are the variances of each component of $\mathbf{p}$.

Hamiltonian dynamics is attractive for constructing MCMC updates for a number of reasons: (i) As $\mathbf{T}_\delta$ is invertible, Hamiltonian dynamics are **reversible** and therefore the Markov chain transitions that arise from using Hamiltonian dynamics for proposals is also reversible, leaving the stationary distribution invariant. (ii) The Hamiltonian $H(\mathbf{q}, \mathbf{p})$ is **conserved**, and so the acceptance probability of the transition is 1 [2]. (iii) $\mathbf{T}_\delta$ is a **volume-preserving** operator (c.f. Liouville's theorem), which simplifies the computation of the acceptance probability of the transition since any change in volume must be accounted for [3].

[1] Note that technically, the potential energy is proportional to the log-posterior since we cannot compute $p(D_{tr})$. This is not an issue since $p(D_{tr})$ is a constant.

[2] Since we are finding an approximate solution, this will not be true. However, as we will see, an acceptance probability that leaves the stationary distribution invariant can still be derived.

[3] In fact, Hamiltonian dynamics is symplectic, a stronger statement which implies volume conservation.

Discrete-Time Approximation

We require a solution $\mathbf{T}_\delta$ to (B.1) and (B.2), which must be discrete-time in order to be computable. [57] proposes a discrete approximation known as the **leapfrog method**:

$$(\mathbf{p}_j)_{t+\epsilon/2} = (\mathbf{p}_j)_t - \frac{\epsilon}{2} \cdot \frac{\partial U}{\partial \mathbf{q}_j}\Big|_{\mathbf{q}_t} \tag{B.7}$$

$$(\mathbf{q}_j)_{t+\epsilon} = (\mathbf{q}_j)_t + \epsilon \cdot \frac{(\mathbf{p}_j)_{t+\epsilon/2}}{m_j} \tag{B.8}$$

$$(\mathbf{p}_j)_{t+\epsilon} = (\mathbf{p}_j)_{t+\epsilon/2} - \frac{\epsilon}{2} \cdot \frac{\partial U}{\partial \mathbf{q}_j}\Big|_{\mathbf{q}_{t+\epsilon}} \tag{B.9}$$

where $\epsilon, 2\epsilon, \ldots$ indexes the time steps (i.e. discretized δ). It should be noted that even though (B.7) to (B.9) are approximate solutions, they still follow the invariances outlined above.

Constructing the HMC Algorithm

From (B.7) to (B.9), an MCMC algorithm can be constructed as follows: every iteration t, we sample $\mathbf{p}_t$ from its Gaussian distribution (independent of the current value of $\mathbf{q}$). Beginning with $\mathbf{p}_t$ and $\mathbf{q}_t$ (the most recent sample of $\mathbf{q}$), we then simulate Hamiltonian dynamics by performing L steps of the leapfrog update (B.7) to (B.9) with stepsize ϵ. The Metropolis proposal is then $\mathbf{q}_{t+L\epsilon}$. We then accept this proposal with probability

$$p(\text{accept}) = \min\Big(1, \exp\{-U(\mathbf{q}_{t+L\epsilon}) + U(\mathbf{q}_t) - K(\mathbf{p}_{t+L\epsilon}) + K(\mathbf{p}_t)\}\Big) \tag{B.10}$$

As per the usual Metropolis-Hastings framework, if we accept $\mathbf{q}_{t+L\epsilon}$ then it becomes the next state, otherwise, $\mathbf{q}_t$ remains as the next state. It can be shown that the Metropolis update with the acceptance probability above is reversible and therefore leaves the canonical distribution of $\mathbf{q}$ invariant. We can discard $\mathbf{p}_t$ and $\mathbf{p}_{t+L\epsilon}$ after computing the acceptance probability. At the end of the algorithm, we will have a set of particles $\{\mathbf{W}_1, \ldots, \mathbf{W}_S\}$ drawn from the posterior.

Practical Details Both ϵ and L are hyperparameters to be tuned. For the BNNs used in this thesis, we set all $m_j = 1$. In practice, it is typical to (i) discard some initial samples of $\mathbf{q}$ (this is known as "burn-in"), and after the burn-in period, to (ii) only collect samples every few iterations (instead of every iteration). Hence the number of burn-in iterations and the frequency of collection are both hyperparameters. [57] explains how to tune these values, and in general, how to diagnose HMC.

The full algorithm, taken from [57], is shown in Algorithm 2.

Algorithm 2: Hamiltonian Monte Carlo

Input: $p(\cdot|D_{tr})$, L, ϵ, N_{burnin}, N_{actual}, ω (sampling frequency)
$U(\mathbf{q}) := p(\mathbf{q}|D_{tr})$;
$K(\mathbf{p}) := \sum_{d=j}^{M} \frac{\mathbf{p}_j^2}{2}$;
S $\leftarrow$ [];
$\mathbf{q}_0 \leftarrow$ random sample (e.g. from prior distribution);
push $\mathbf{q}_0$ into S;
for $t \leftarrow 0$ **to** $N_{burnin} - 1$ **do**
    $\mathbf{q}_{old} \leftarrow S[-1]$;
    $\mathbf{p}_{old} \leftarrow$ random sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$;
    for $l \leftarrow 0$ **to** $L - 1$ **do**
        $\mathbf{q}_{new}, \mathbf{p}_{new} \leftarrow$ (B.7) to (B.9);
    end
    $a \leftarrow$ random sample from Unif$(0, 1)$;
    if $a <$ *(B.10)* **then**
        push $\mathbf{q}_{new}$ into S;
    else
        push $\mathbf{q}_{old}$ into S;
    end
end
S $\leftarrow$ [S[-1]];
for $t \leftarrow 0$ **to** $N_{actual} - 1$ **do**
    $\mathbf{q}_{old} \leftarrow S[-1]$;
    $\mathbf{p}_{old} \leftarrow$ random sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$;
    for $l \leftarrow 0$ **to** $L - 1$ **do**
        $\mathbf{q}_{new}, \mathbf{p}_{new} \leftarrow$ (B.7) to (B.9);
    end
    $a \leftarrow$ random sample from Unif$(0, 1)$;
    if $a <$ *(B.10)* **then**
        push $\mathbf{q}_{new}$ into S;
    else
        push $\mathbf{q}_{old}$ into S;
    end
end
return $S[:: \omega]$(every ω^{th} sample in S) ;

B.2 BAYES BY BACKPROP

This section is a condensed treatment of BBB, introduced in [5], and readers are referred to the original text for proofs and detailed analysis. Suppose that we use a diagonal (multivariate) Gaussian variational family, where the variational parameter $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ is the means and standard deviations for each (independent) parameter:

$$q_{\boldsymbol{\theta}}(\mathbf{w}) = (2\pi)^{-\frac{M}{2}}(\sigma_1 \dots \sigma_M)^{-1} \exp\{-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\} \tag{B.11}$$

where $\boldsymbol{\Sigma}$ is the diagonal matrix with entries $\sigma_1^2, \dots, \sigma_M^2$. Hence $\boldsymbol{\theta}$ is a vector of size $2M$.

Recall that our goal is to minimize the ELBO (c.f. (1.20)). However, it cannot be evaluated analytically due to the presence of an expectation over $\mathbf{W} \sim q_{\boldsymbol{\theta}}$ (an integral). Instead, BBB optimizes a Monte Carlo estimator of (1.20). However, in order to make this estimator differentiable, BBB relies on a technique known in variational literature as the reparametrization trick.

REPARAMETRIZATION TRICK

It can be shown that given certain mild conditions, we can swap the expectation and differential operator:

Proposition B.2.1. *Let $\boldsymbol{\epsilon}$ be a random variable with PDF $q(\boldsymbol{\epsilon})$ and let $\mathbf{w} = t(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ be a deterministic transformation t of $(\boldsymbol{\theta}, \boldsymbol{\epsilon})$. Suppose also that the marginal PDF of $\mathbf{w}$, $q(\mathbf{w}|\boldsymbol{\theta})$, is such that $q(\boldsymbol{\epsilon})d\boldsymbol{\epsilon} = q(\mathbf{w}|\boldsymbol{\theta})d\mathbf{w}$. Then for a function f with derivatives in $\mathbf{w}$:*

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathbb{E}_{\mathbf{w}\sim q(\mathbf{w}|\boldsymbol{\theta})}\Big[f(\mathbf{w}, \boldsymbol{\theta})\Big] = \mathbb{E}_{\boldsymbol{\epsilon}\sim q(\boldsymbol{\epsilon})}\Big[\frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \mathbf{w}} \cdot \frac{\partial \mathbf{w}}{\partial \boldsymbol{\theta}} + \frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big] \tag{B.12}$$

If we refer to (1.20), we can see that by choosing $f(\mathbf{w}, \boldsymbol{\theta}) = \log p(D_{tr}|\mathbf{w}) + \log p(\mathbf{w}) - \log q_{\boldsymbol{\theta}}(\mathbf{w})$, the L.H.S. of Proposition B.2.1) is precisely the derivative of the ELBO, our optimization objective. The use of Proposition B.2.1 is crucial here because taking the differential of $\log p(D_{tr}|\mathbf{w})$ requires backpropagation through the NN, which cannot be done if $\mathbf{w}$ is random. As such, only the R.H.S. can be evaluated. Proposition B.2.1 is therefore useful because it allows us to differentiate by reparametrizing the source of randomness from $\mathbf{w}$ to $\boldsymbol{\epsilon}$.

CONSTRUCTING THE BBB ALGORITHM

The reparametrization trick can be easily applied to the Gaussian variational family by means of the deterministic transformation:

$$\mathbf{w} = t(\boldsymbol{\theta}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon} \tag{B.13}$$

where $\circ$ is the component-wise multiplication operator and $\boldsymbol{\epsilon}$ is a M-sized random vector with the unit Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$ distribution. Since $\boldsymbol{\sigma}$ must be non-negative, which can be tricky for practical optimization algorithms, we choose instead a different parametrization $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\rho})$ where $\boldsymbol{\sigma} = \log(1 + e^{\boldsymbol{\rho}})$ element-wise. Then our transformation becomes:

$$\mathbf{w} = t(\boldsymbol{\theta}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \log(1 + e^{\boldsymbol{\rho}}) \circ \boldsymbol{\epsilon} \tag{B.14}$$

and now both $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ have support in $\mathbb{R}^M$. With the reparametrization trick, the Monte Carlo estimator of the R.H.S. of (B.12) is precisely the gradient that we take a step in each iteration of gradient descent, which is computable using automatic differentiation. Typically, only 1 sample of $\boldsymbol{\epsilon}$ is enough to compute the estimator. The full algorithm, taken from [5], is shown in Algorithm 3.

Algorithm 3: Bayes by Backprop

Input: $p(D_{tr}|\mathbf{w})$, $p(\mathbf{w})$, $q_{\boldsymbol{\theta}}(\mathbf{w})$, N_{iter}, α (learning rate)
$\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\rho}) \leftarrow$ random initialization from any reasonable distribution;
for $n \leftarrow 0$ **to** $N_{iter} - 1$ **do**
  $\boldsymbol{\epsilon} \leftarrow$ random sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $\mathbf{w} \leftarrow$ (B.14);
  $f(\mathbf{w}, \boldsymbol{\theta}) \leftarrow \log p(D_{tr}|\mathbf{w}) + \log p(\mathbf{w}) - \log q_{\boldsymbol{\theta}}(\mathbf{w})$;
  $\nabla_{\boldsymbol{\mu}} \leftarrow \frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \boldsymbol{\mu}}$;
  $\nabla_{\boldsymbol{\rho}} \leftarrow \frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \mathbf{w}} \cdot \frac{\boldsymbol{\epsilon}}{1+e^{-\boldsymbol{\rho}}} + \frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \boldsymbol{\rho}}$;
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \alpha \cdot \nabla_{\boldsymbol{\mu}}$;
  $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \alpha \cdot \nabla_{\boldsymbol{\rho}}$;
end
return $\boldsymbol{\theta}$;

B.3 Stein Variational Gradient Descent

We present a concise summary of SVGD, introduced in [47]. Readers are referred to the original paper for more details. Like BBB, SVGD is a VI approach and learns an approximation of the true posterior. However, instead of optimizing the parameters of a fixed variational family, SVGD iteratively applies smooth transforms to an initial, tractable distribution in a way that minimizes the KL divergence of the resulting distribution. These transforms are determined via a form of functional gradient descent that relies on a general result in probability theory known as Stein's Method.

VI via Smooth Transforms

Let p denote the BNN posterior and $\bar{p}$ the unnormalized posterior (i.e. without the denominator term $p(D_{tr})$). Let q_0 be any initial, tractable distribution over $\mathcal{W} = \mathbb{R}^M$, e.g. a Gaussian distribution. $Q = \{q_{[\mathbf{T}]}\}$ is the set of distributions of all $\mathbf{w}' = \mathbf{T}(\mathbf{w})$ where $\mathbf{w}$ is distributed according to q_0 and $\mathbf{T} : \mathcal{X} \to \mathcal{X}$ is any smooth bijective transform. For any such $\mathbf{T}$, we have the resulting PDF:

$$q_{[\mathbf{T}]}(\mathbf{w}') = q_0\big(\mathbf{T}^{-1}(\mathbf{w})\big) \cdot \Big| \det\big(\nabla_{\mathbf{w}} \mathbf{T}^{-1}(\mathbf{w})\big)\Big| \tag{B.15}$$

where $\nabla_{\mathbf{w}} \mathbf{T}^{-1}$ is the Jacobian matrix of $\mathbf{T}^{-1}$. While Q is a powerful variational family (in the sense that some measurable $\mathbf{T}$ always exists between any two diffuse densities p and q), specifying a set of transforms $\mathbf{T}$ that will make the resulting VI objective (KL divergence minimization or ELBO maximization) tractable is non-trivial. Liu and Wang [47] uses Stein's method to define iterative transforms that effectively perform gradient descent on $D_{KL}\big(q_{[\mathbf{T}]} \,||\, p\big)$.

Stein's Method

If p is a smooth PDF and $\boldsymbol{\phi} : \mathcal{W} \to \mathcal{W}$ is a smooth vector-valued function (with some mild zero boundary and regularity conditions), then **Stein's identity** states that

$$\mathbb{E}_p[\mathcal{A}_p \boldsymbol{\phi}(\mathbf{w})] = \mathbf{0} \qquad \text{where } \mathcal{A}_p \boldsymbol{\phi}(\mathbf{w}) = \boldsymbol{\phi}(\mathbf{w}) \nabla_{\mathbf{w}} \log p(\mathbf{w})^\top + \nabla_{\mathbf{w}} \boldsymbol{\phi}(\mathbf{w}) \tag{B.16}$$

$\mathcal{A}_p$ is known as the **Stein operator** (under p) and $\boldsymbol{\phi}$ is in the **Stein class** of p. Supposing the expectation in (B.16) was taken under a different distribution q, then the resulting value $\mathbb{E}_q[\mathcal{A}_p \boldsymbol{\phi}(\mathbf{w})] \neq 0$. This fact allows us to characterize a discrepancy measure between two distributions, known as **Stein discrepancy**:

$$\mathbb{S}(q, p) = \max_{\boldsymbol{\phi} \in \mathcal{F}} \Big\{ \mathbb{E}_q\big[\text{trace}\big(\mathcal{A}_p \boldsymbol{\phi}(\mathbf{w})\big)\big]^2 \Big\} \tag{B.17}$$

where $\mathcal{F}$ is some set of functions over $\mathcal{W}$. The discriminative power and computational tractability of $\mathbb{S}(q, p)$ greatly depends on the choice of $\mathcal{F}$.

One choice of $\mathcal{F}$ makes use of a reproducing kernel Hilbert space (RKHS). For readers unfamiliar with RKHS, see Appendix C for an brief introduction. Let $\mathcal{H}$ be the RKHS with the kernel $k(\mathbf{w}, \mathbf{w}') : \mathcal{W} \times \mathcal{W} \to \mathbb{R}$, and denote by $\mathcal{H}^M$ the space of vector-valued functions $\mathbf{f} = [f_1, \ldots, f_M]$ where each $f_m \in \mathcal{H}$. Assuming that k is in the Stein class of p, then the **kernelized Stein discrepancy** is:

$$\mathbb{S}(q, p) = \max_{\boldsymbol{\phi} \in \mathcal{H}^M} \Big\{ \mathbb{E}_q\big[\text{trace}\big(\mathcal{A}_p \boldsymbol{\phi}(\mathbf{w})\big)\big]^2 \quad \text{where } ||\boldsymbol{\phi}||_{\mathcal{H}^M} \leq 1 \Big\} \tag{B.18}$$

where $||\cdot||_{\mathcal{H}^M}$ is the norm of the RKHS. As it turns out, we can derive a closed-form optimal solution to (B.18). Let

$$\boldsymbol{\phi}^*_{q,p}(\cdot) = \mathbb{E}_q[\mathcal{A}_p k(\mathbf{w}, \cdot)] = \mathbb{E}_q[k(\mathbf{w}, \cdot)\nabla_{\mathbf{w}} \log p(\mathbf{w}) + \nabla_{\mathbf{w}} k(\mathbf{w}, \cdot)] \tag{B.19}$$

Then the optimal solution to (B.18) is $\boldsymbol{\phi}^*_{q,p}(\cdot)/||\boldsymbol{\phi}^*_{q,p}(\cdot)||_{\mathcal{H}^M}$ and $\mathbb{S}(q.p) = ||\boldsymbol{\phi}^*_{q,p}||^2_{\mathcal{H}^M}$. The kernelized Stein discrepancy is a proper divergence in the sense that $\mathbb{S}(q, p) = 0$ iff $p = q$ and k is strictly positive definite (which is the case for many common kernels). Also note that (B.19) can be computed using the unnormalized $\bar{p}$ (instead of p) since $\log p(\mathbf{w})^\top = \log \bar{p}(\mathbf{w})^\top$.

Stein Operators for VI

Returning to the VI setting, let us consider transforms of the type $\mathbf{T}(\mathbf{w}) = \mathbf{w} + \epsilon \cdot \boldsymbol{\phi}(\mathbf{w})$, i.e. small perturbations of the identity mapping. Note that sufficiently small $|\epsilon|$ (a scalar) guarantees that $\mathbf{T}$ is bijective by the inverse function theorem. As it turns out, there is an important connection between these transforms and Stein operators:

$$\nabla_\epsilon D_{KL}\big(q_{[\mathbf{T}]} \,||\, p\big)\Big|_{\epsilon=0} = -\mathbb{E}_q\big[\text{trace}\big(\mathcal{A}_p \boldsymbol{\phi}(\mathbf{w})\big)\big] \tag{B.20}$$

This implies that $\boldsymbol{\phi}^*_{q,p}$ is the perturbation that results in the steepest descent on the KL divergence in the unit ball of $\mathcal{H}^M$, and for $\mathbf{T}^*(\mathbf{w}) = \mathbf{w} + \epsilon \boldsymbol{\phi}^*_{q,p}(\mathbf{w})$, we have $\nabla_\epsilon D_{KL}\big(q_{[\mathbf{T}^*]} \,||\, p\big)\Big|_{\epsilon=0} = -\mathbb{S}(q, p)$.

We can therefore consider an iterative process as follows: Starting with the initial distribution q_0, we apply the transform $\mathbf{T}^*_0(\mathbf{w}) = \mathbf{w} + \epsilon_0 \cdot \boldsymbol{\phi}^*_{q_0,p}(\mathbf{w})$. The resulting distribution q_1 over $\mathbf{T}^*_0(\mathbf{w})$ has KL divergence (against p) smaller than q_0 by the value $\epsilon_0 \cdot \mathbb{S}(q_0, p)$. We can then apply $\mathbf{T}^*_1(\mathbf{w}) = \mathbf{w} + \epsilon_1 \cdot \boldsymbol{\phi}^*_{q_1,p}(\mathbf{w})$, so on and so forth, until the final distribution q_T is sufficiently close to p.

In order to carry the above process tractably, we must sample a finite number of particles $\{\mathbf{w}^{(i)}\}_{i=1}^S$ from q_0 to apply the iterative transforms to. We can also estimate the expectation in (B.19) using the same set of particles. Since we do not need a closed-form expression for q_0, we can in fact initialize these particles using any arbitrary method. The complete SVGD procedure is presented in Algorithm 4.

Algorithm 4: Stein Variational Gradient Descent

Input: $\bar{p}$, q_0, k, ϵ, N_{iter}
$\{\mathbf{w}^{(i)}\}_{i=1}^{S} \leftarrow$ sampled from q_0;
for $n \leftarrow 0$ **to** $N_{iter} - 1$ **do**
  $\hat{\boldsymbol{\phi}}^*(\mathbf{w}) \leftarrow \frac{1}{S}\sum_{i=1}^{S}\left[k(\mathbf{w}^{(i)}, \mathbf{w})\nabla_{\mathbf{w}^{(i)}} \log \bar{p}(\mathbf{w}^{(i)}) + \nabla_{\mathbf{w}^{(i)}} k(\mathbf{w}^{(i)}, \mathbf{w})\right]$;
  for $i \leftarrow 1$ **to** S **do**
    $\mathbf{w}^{(i)} \leftarrow \mathbf{w}^{(i)} + \epsilon \cdot \hat{\boldsymbol{\phi}}^*(\mathbf{w}^{(i)})$;
  end
end
return $\{\mathbf{w}^{(i)}\}_{i=1}^{S}$;

For all the experiments in this thesis, we will use the RBF kernel:

$$k(\mathbf{w}, \mathbf{w}') = \exp\Big\{-\frac{1}{h}||\mathbf{w} - \mathbf{w}'||_2^2\Big\} \quad \text{where } h = \frac{\text{med}^2}{\log S} \tag{B.21}$$

and med is the median pairwise distance between any two particle in $\{\mathbf{w}^{(i)}\}_{i=1}^{S}$. This is the same kernel used in Liu and Wang [47]. Notice that the two terms of (B.19) can be interpreted as serving different purposes: the first term drives the particles to have high probability w.r.t. $p(\mathbf{w})$, whereas the second term drives the particle to be far from each other (in $\mathcal{W}$ space), thus preventing them fro collapsing into the same local mode(s) of p. As the bandwidth $h \to 0$, the second term vanishes and SVGD reduces to the MAP estimate. Choosing $S = 1$ also implies the same.

SVGD returns a finite set of particles instead of an analytical solution of the approximate posterior [4]. SVGD is an efficient algorithm and scales to high input dimensions and large datasets. The efficiency bottleneck is the computation of $\nabla_{\mathbf{w}} \log \bar{p}(\mathbf{w})$; however, like BBB, the likelihood term in $\bar{p}$ can be batched for computational speedups.

[4] In this respect, it is similar to MCMC approaches like HMC.

C

Reproducing Kernel Hilbert Spaces

This section provides a brief background into reproducing kernel Hilbert space (RKHS) theory. Readers should refer to [2], [64] or [65] for more details.

Hilbert Spaces and Linear Functionals

Let $\mathcal{H}$ be a real or complex vector space over the field $\mathcal{F}$. $\mathcal{H}$ is known as an *inner product space* if it is endowed with an inner product, which is a mapping:

$$\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \rightarrow \mathcal{F} \tag{C.1}$$

that is positive-definite, linear and (conjugate) symmetrical. For any element $f \in \mathcal{H}$, we can define the *norm* of f as $||f|| = \sqrt{\langle f, f \rangle}$. If we define the distance between any $f, g \in \mathcal{H}$ to be $d(f, g) = ||f - g||$, then d is a proper metric and so $\mathcal{H}$ is a metric space. If $\mathcal{H}$ is also a complete metric space, i.e. every Cauchy sequence in $\mathcal{H}$ converges in $\mathcal{H}$, then $\mathcal{H}$ is known as a *Hilbert space*.

We define a *linear functional* $\mathbf{L}$ as a mapping:

$$\mathbf{L} : \mathcal{H} \rightarrow \mathcal{F} \tag{C.2}$$

For any $\mathbf{L}$, the following three conditions are equivalent: (i) $\mathbf{L}$ is continuous on $\mathcal{H}$, (ii) $\mathbf{L}$ is continuous at 0 on $\mathcal{H}$, and (iii) $\mathbf{L}$ is bounded, i.e. $\exists M > 0$ s.t. $|\mathbf{L}f| \leq M||f||$ for every $f \in \mathcal{H}$. The space of continuous linear functionals on $\mathcal{H}$, which we denote by $\mathcal{H}'$, is known as the *topological dual space* of $\mathcal{H}$. If $\mathcal{H}$ is a Hilbert space, so is $\mathcal{H}'$. For any fixed $g \in \mathcal{H}$, the mapping $f \rightarrow \langle f, g \rangle$ is a continuous linear functional on $\mathcal{H}$. The reverse is also true:

Theorem C.0.1 (Riesz representation theorem (Thm 4.12, Rudin [63])). *For any continuous linear functional* $\mathbf{L}$ *on* $\mathcal{H}$*, there is a unique* $g \in \mathcal{H}$ *such that* $\mathbf{L}f = \langle f, g \rangle$ *for all* $f \in \mathcal{H}$*.*

REPRODUCING KERNELS AND RKHS

Suppose that $\mathcal{H}$ is a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Define $\mathbf{E}_x : \mathcal{H} \to \mathbb{R}$ to be the *evaluation functional* on $\mathcal{H}$, where $\mathbf{E}_x f = f(x)$ for all $x \in \mathcal{X}$. In fact, this is sufficient to define an RKHS:

Definition C.0.2. *Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. If $\mathcal{X}$ is a non-empty set and $\mathbf{E}_x$ is continuous on $\mathcal{H}$ for all $x \in \mathcal{X}$, then $\mathcal{H}$ is a **reproducing kernel Hilbert space**.*

A direct implication of continuous $\mathbf{E}_x$ is that functions on $\mathcal{H}$ that converge in norm also converge at every point, i.e. $\lim_{n\to\infty} ||f_n - f|| = 0 \Rightarrow \lim_{n\to\infty} f_n(x) = f(x)$ for all $x \in \mathcal{X}$.

Definition C.0.2 does not cover the eponymous property of an RKHS, which is the reproducing kernel. We will now define this term and show how it connects to Definition C.0.2.

Definition C.0.3. *Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. A **reproducing kernel** of $\mathcal{H}$ is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that:*

1. *$\forall x \in \mathcal{X}$, $k(\cdot, x) \in \mathcal{H}$,*
2. *$\forall x \in \mathcal{X}$, $\forall f \in \mathcal{H}$, $\langle f, k(\cdot, x)\rangle = f(x)$.*

The second property is the so-called reproducing property of the kernel.

As it turns out, $\mathcal{H}$ is an RKHS if and only if it has a reproducing kernel k. This is a consequence of the Riesz representation theorem (that allows $\mathbf{E}_x$ itself to be represented as an inner product), and thus we refer to $k(\cdot, x)$ as the representer of evaluation at $x \in \mathcal{X}$. Furthermore, such a k is also (i) unique for any given $\mathcal{H}$, and (ii) positive-definite.

CONSTRUCTING RKHS FROM KERNELS

We have seen that every RKHS has a unique reproducing kernel. The converse is also true: for every positive-definite function k, there is a corresponding unique RKHS $\mathcal{H}$ (for which k is the reproducing kernel). Constructing such a $\mathcal{H}$ is a two-fold process. First, we construct (from k) an initial space $\mathcal{H}_0$ known as a *pre-RKHS*. Any valid pre-RKHS needs to satisfy two properties:

1. $\mathbf{E}_x$ is continuous on $\mathcal{H}_0$ for all $x \in \mathcal{X}$.
2. Any Cauchy sequence in $\mathcal{H}_0$ which converges pointwise to 0 also converges in norm (of $\mathcal{H}_0$) to 0.

We then define $\mathcal{H}$ to be the set of all functions $f : \mathcal{X} \to \mathbb{R}$ such that there exists a $\mathcal{H}_0$-Cauchy sequence of functions converging pointwise to f. It can be proven that such a $\mathcal{H}$ is a valid RKHS. The Moore-Aronszajn theorem shows us how to build $\mathcal{H}_0$:

Theorem C.0.4 (Moore-Aronszajn). *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be positive-definite. There is a unique RKHS $\mathcal{H}$ with k as the reproducing kernel. The space $\mathcal{H}_0 = span\big[\{k(\cdot, x)\}_{x \in \mathcal{X}}\big]$ endowed with the inner product*

$$\langle f, g \rangle_{\mathcal{H}_0} = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, y_j) \tag{C.3}$$

where $f = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i)$ and $g = \sum_{j=1}^{m} \beta_j k(\cdot, y_i)$ is a valid pre-RKHS.

Since every reproducing kernel is positive-definite and every positive-definite function is a reproducing kernel for a unique $\mathcal{H}$, these two concepts are equivalent. Here, we can also introduce the notion of a kernel (without the reproducing qualification):

Definition C.0.5. *A **kernel** is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that there exists a real Hilbert space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ and*

$$k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \tag{C.4}$$

*for all $x, y \in \mathcal{X}$. ϕ is known as a **feature map** and $\mathcal{H}$ the corresponding **feature space**. (Note that ϕ may not be unique).*

In fact, reproducing kernels are positive-definite by virtue of the fact that they are kernels. Hence reproducing kernels, kernels and positive-definite functions are all equivalent concepts and we can establish a bijection between the set of all positive-definite functions (which we denote as $\mathbb{R}^{\mathcal{X} \times \mathcal{X}}$) and the set of all RKHS (which we denote as $Hilb(\mathbb{R}^{\mathcal{X}})$).

Kernel functions are well-studied in machine learning, and (C.4) is the basis for the so-called "kernel trick", which allows us to operate in the feature space $\mathcal{H}$ instead of the original input space $\mathcal{X}$. A widely-studied kernel is the *Gaussian kernel*:

$$k(x, y) = \exp\Big\{ - \frac{||x - y||_2^2}{2\sigma_k^2} \Big\} \tag{C.5}$$

SPECTRAL REPRESENTATION OF RKHS

Given additional structure on $\mathcal{X}$ and k, it is possible to derive a spectral representation of the resulting RKHS. From here on, let $(\mathcal{X}, d_{\mathcal{X}})$ be a compact metric space and μ be a (positive) Borel measure on $\mathcal{X}$. Let k be a continuous kernel. The associated *Hilbert-Schmidt integral operator* [1] is $\mathbf{K} : L^2(\mathcal{X}; \mu) \to L^2(\mathcal{X}; \mu)$ where

$$(\mathbf{K}f)(\cdot) = \int k(\cdot, y) f(y) \, \mathrm{d}\mu(y) \qquad \forall f \in L^2(\mathcal{X}; \mu) \tag{C.6}$$

[1] Technically, $L^2(\mathcal{X})$ is the space of equivalence classes of functions that are identical almost everywhere. We abuse notation here and treat $f \in L^2(\mathcal{X}; \mu)$ as a function in the equivalence class.

Symmetry of k implies that $\mathbf{K}$ is self-adjoint, i.e. $\langle f, \mathbf{K}g\rangle = \langle \mathbf{K}f, g\rangle$ for all $f, g \in L^2(\mathcal{X}, \mu)$. By the Arzela-Ascoli theorem (Thm 11.28, Rudin [63]), continuity of k implies $\mathbf{K}$ is also a compact operator. We can then invoke the spectral theorem.

Theorem C.0.6 (Spectral)**.** *Let $\mathcal{H}$ be a Hilbert space and $\mathbf{T} : \mathcal{H} \to \mathcal{H}$ be a self-adjoint, compact operator. Then there is an at most countable orthonormal set $\{\mathbf{e}_j\}_{j\in J}$ of $\mathcal{H}$ and $\{\lambda_j\}_{j\in J}$ with $|\lambda_1| \geq |\lambda_2| \geq \cdots > 0$ converging to zero, such that*

$$(\mathbf{T}f)(\cdot) = \sum_{j\in J} \lambda_j \langle f, \mathbf{e}_j\rangle \mathbf{e}_j(\cdot) \qquad \forall f \in \mathcal{H} \tag{C.7}$$

Applying the spectral theorem to $\mathbf{K}$ gives us Mercer's theorem:

Theorem C.0.7 (Mercer's)**.** *Let $\mathcal{X}$ and k possess the additional structures as defined above. Then*

$$k(x, y) = \sum_{j\in J} \lambda_j \mathbf{e}_j(x)\mathbf{e}_j(y) \qquad \forall x, y \in \mathcal{X} \tag{C.8}$$

where the eigenbasis is w.r.t. $L^2(\mathcal{X}; \mu)$ and the sum converges uniformly on $\mathcal{X} \times \mathcal{X}$ and absolutely for all $x, y \in \mathcal{X}$.

Mercer's theorem provides a feature map $\phi : \mathcal{X} \to \ell^2(J)$ for k where $\phi(x) = \left\{\sqrt{\lambda_j}\mathbf{e}_j(x)\right\}_{j\in J}$. More importantly, Mercer's theorem also provides an alternative representation of the RKHS of k (denoted as $\mathcal{H}_k$) via the eigenfunctions of $\mathbf{K}$.

Theorem C.0.8. *Let $\mathcal{X}$ and k possess the additional structures as defined above. Define the space*

$$\mathcal{H} = \left\{ f = \sum_{j\in J} a_j \mathbf{e}_j : \left\{\frac{a_j}{\sqrt{\lambda_j}}\right\} \in \ell^2(J) \right\} \tag{C.9}$$

with the inner product

$$\left\langle \sum_{j\in J} a_j \mathbf{e}_j, \sum_{j\in J} b_j \mathbf{e}_j \right\rangle_{\mathcal{H}} = \sum_{j\in J} \frac{a_j b_j}{\lambda_j} \tag{C.10}$$

Then $\mathcal{H} = \mathcal{H}_k$, i.e. they have the same space and same inner product.

Note that the condition $\left\{\frac{a_j}{\sqrt{\lambda_j}}\right\} \in \ell^2(J)$ guarantees that $\sum_{j\in J} a_j \mathbf{e}_j$ converges and so is well-defined. This convergence is connected to regularity properties of f. Even though $\mathcal{H}$ is defined by the eigenfunctions $\mathbf{e}_j$ of $\mathbf{K}$, which itself is defined w.r.t. a measure μ, the uniqueness of RKHS implies that $\mathcal{H}$ is independent of μ.

Given further assumptions that $k(x, \cdot) \in L^2(\mathcal{X}, \mu)$ for all $x \in \mathcal{X}$ and $\mathbf{K}$ is a positive, bounded operator with at most countably many positive eigenvalues, then Mercer's theorem can be extended to non-compact $(\mathcal{X}, d_{\mathcal{X}})$. In particular, the Gaussian kernel on $\mathbb{R}^M$ is Mercer [69].

D

Approximations for BNN Prior Predictive

We succinctly present the derivation of (2.17) and (2.21), taken from Chapter 5.7 of Bishop [3]. Readers are referred to the main text for a detailed exposition.

D.1 BNN Prior Predictive for Regression

In the regression setting, the likelihood (1.12) is Gaussian. Our variational prior $q_{\boldsymbol{\lambda}}(\mathbf{w})$ is also chosen to be Gaussian. Substituting these two distributions into the prior predictive (2.1), we can write $p(Y|\mathbf{x})$ as the convolution of two Gaussian distributions:

$$p(Y = y|\mathbf{x}) = \int_{\mathcal{W}} \mathcal{N}(y'; \Phi_{\mathbf{w}}(\mathbf{x}), \sigma_\epsilon^2) \cdot \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I}) \, \mathrm{d}\mathbf{w} \tag{D.1}$$

where $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$. Despite the Gaussian terms, (D.1) is still intractable due to the non-linearity of $\Phi_{\mathbf{w}}(\mathbf{x})$. In order to find a linear approximation of $\Phi_{\mathbf{w}}(\mathbf{x})$, we will assume that the $q_{\boldsymbol{\lambda}}(\mathbf{w})$ has a small spread around its mode $\mathbf{w}_{\text{MAP}} = \boldsymbol{\mu}$ [1] relative to the characteristic length-scale of $\mathbf{w}$ over which $\Phi_{\mathbf{w}}(\mathbf{x})$ is varying. This gives us the first-order Taylor approximation of $\Phi_{\mathbf{w}}(\mathbf{x})$ as:

$$\hat{\Phi}_{\mathbf{w}}(\mathbf{x}) = \Phi_{\mathbf{w}_{\text{MAP}}}(\mathbf{x}) + \mathbf{g}^\top(\mathbf{w} - \mathbf{w}_{\text{MAP}}) \tag{D.2}$$

where

$$\mathbf{g} = \left[\nabla_{\mathbf{w}} \Phi_{\mathbf{w}}(\mathbf{x})\Big|_{\mathbf{w}=\mathbf{w}_{\text{MAP}}}\right] \tag{D.3}$$

is the first-order derivative at $\mathbf{w}_{\text{MAP}}$. Note that $\mathbf{w}$ is a variable above while $\mathbf{w}_{\text{MAP}}$ is fixed.

At this point, we will introduce (without proof) a well-established identity:

[1] Recall that the MAP estimator is the mode of the distribution.

Lemma D.1.1. *If*

$$\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$$
$$\mathbf{y}|\mathbf{w} \sim \mathcal{N}(\mathbf{A}\mathbf{w} + \mathbf{b}, \mathbf{L}^{-1})$$

then the marginal distribution is

$$\mathbf{y} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\top})$$

Substituting $\hat{\Phi}_{\mathbf{w}}(\mathbf{x})$ for $\Phi_{\mathbf{w}}(\mathbf{x})$ into (D.1) and using the lemma above gives us the final approximation of $p(Y = y|\mathbf{x})$ as:

$$\hat{Y}|\mathbf{x} \sim \mathcal{N}(\Phi_{\mathbf{w}_{\text{MAP}}}(\mathbf{x}), \sigma_\epsilon^2 + \mathbf{g}^\top(\boldsymbol{\sigma}^2 \cdot \mathbf{g})) \tag{D.4}$$

In other words, the prior predictive approximation is centered at the BNN function $\Phi(\mathbf{x})$ computed at the MAP estimate. The variance is an additive contribution of both the modeled noise σ_ϵ, as well as the prior (variational) variance of $\mathbf{W}$ scaled by our uncertainty (w.r.t. $\mathbf{W}$) of $\Phi(\mathbf{x})$ near the MAP estimate $\boldsymbol{\mu}$.

D.2 BNN Prior Predictive for Binary Classification

As mentioned in Section 2.3, we use a slightly modified BNN setup for binary classification, where there is only a single output node and the logistic sigmoid function is applied to the node's value to yield the final output $\Phi_{\mathbf{w}}(\mathbf{x}) \in [0, 1]$. In this case, the likelihood is simply:

$$p(Y = y|\mathbf{w}) = y \cdot \Phi_{\mathbf{w}}(\mathbf{x}) + (1 - y) \cdot (1 - \Phi_{\mathbf{w}}(\mathbf{x})) \tag{D.5}$$

noting that $y \in \{0, 1\}$. The resulting prior predictive with the variational prior is then:

$$p(Y = y|\mathbf{x}) = \int_{\mathcal{W}} \Big(y \cdot \Phi_{\mathbf{w}}(\mathbf{x}) + (1 - y) \cdot (1 - \Phi_{\mathbf{w}}(\mathbf{x}))\Big) \cdot \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})\ \mathrm{d}\mathbf{w} \tag{D.6}$$

where $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ again. Just like the regression setting, we will need to find an approximation for $\Phi_{\mathbf{w}}(\mathbf{x})$ in order to approximate (D.6). However, unlike the regression setting, a linear approximation of $\Phi_{\mathbf{w}}(\mathbf{x})$ directly is not appropriate due to the non-linearity of the logistic sigmoid function applied to the BNN's output. Instead, we will first construct a linear approximation on $\phi_{\mathbf{w}}(\mathbf{x})$; recall that this denotes the output node's value *before* applying the sigmoid function. Similar to regression, we can use the first-order Taylor approximation:

$$\hat{\phi}_{\mathbf{w}}(\mathbf{x}) = \phi_{\mathbf{w}_{\text{MAP}}}(\mathbf{x}) + \mathbf{g}^\top(\mathbf{w} - \mathbf{w}_{\text{MAP}}) \tag{D.7}$$

where $\mathbf{g}$ is, once again, first-order derivative at $\mathbf{w}_{\text{MAP}}$ (applied to ϕ instead of Φ).

The "prior predictive" over the output node function ϕ is then:

$$p(\phi(\mathbf{x}) = a|\mathbf{x}) = \int_{\mathcal{W}} \delta[a = \hat{\phi}_{\mathbf{w}}(\mathbf{x})] \cdot \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I}) \, \mathrm{d}\mathbf{w} \tag{D.8}$$

where $\delta(\cdot)$ is the Dirac delta function. In fact, (D.8) is simply a Gaussian distribution:

$$\phi(\mathbf{x})|\mathbf{x} \sim \mathcal{N}(\phi_{\mathbf{w}_{\text{MAP}}}(\mathbf{x}), \mathbf{g}^\top(\boldsymbol{\sigma}^2 \cdot \mathbf{g})) \tag{D.9}$$

Finally, we need to transform our approximation of $\phi(\mathbf{x})$ to $\Phi(\mathbf{x})$, the predicted output label. Since we have already integrated over $\mathbf{W}$ in the approximation above, we can rewrite (D.6) in terms of an integral over ϕ instead:

$$\begin{aligned} p(\hat{Y} = 1|\mathbf{x}) &= \int_{\mathbb{R}} \sigma_L(a) p(\phi(\mathbf{x}) = a|\mathbf{x}) \, \mathrm{d}a \\ &= \int_{\mathbb{R}} \sigma_L(a) \cdot \mathcal{N}(a; \phi_{\mathbf{w}_{\text{MAP}}}(\mathbf{x}), \mathbf{g}^\top(\boldsymbol{\sigma}^2 \cdot \mathbf{g})) \, \mathrm{d}a \end{aligned} \tag{D.10}$$

Note that $p(\hat{Y} = 0|\mathbf{x}) = 1 - p(\hat{Y} = 1|\mathbf{x})$. Lastly, as the convolution of a Gaussian with a logistic sigmoid is intractable, we need to apply one final approximation:

$$p(\hat{Y} = 1|\mathbf{x}) = \sigma_L\left(\left(1 + \frac{\pi(\mathbf{g}^\top(\boldsymbol{\sigma}^2 \cdot \mathbf{g}))}{8}\right)^{-1/2} \mathbf{g}^\top \mathbf{w}_{\text{MAP}}\right) \tag{D.11}$$

E

OC-BNN: Experimental Details

E.1 Low-Dimensional Simulations

The model for all experiments is a BNN with a single 10-node RBF hidden layer ($H = 1$, $|\mathbf{L}_1| = 10$). For the naive Gaussian prior (1.14), we set $\sigma_\omega = 1$. The output noise in (1.12) for regression experiments is modeled as $\sigma_\epsilon = 0.1$.

Where HMC is used for inference, we discard 10000 samples as burn-in, before collecting 1000 samples at intervals of 10 (hence a total of 20000 Metropolis-Hastings iterations are carried out). $L = 50$ and ϵ is variably adjusted such that the overall acceptance rate is approximately 0.9. Where SVGD is used for inference, we use 50 particles and run 1000 update iterations. We use the AdaGrad optimizer [14] with an initial learning rate of 0.75.

In Figure 2.5.1, the hyperparameters for the negative exponential COCP are: $\gamma = 10000$, $\tau_0 = 15$ and $\tau_1 = 2$. In Figure 2.5.2, the hyperparameters for the positive Dirichlet COCP are: $\boldsymbol{\alpha}_i = 0.85$ if $i \in \mathcal{C}_y(\mathbf{x})$, and 0 otherwise. In both Figure 2.5.3a and Figure 2.5.4, we run 125 epochs of VOCP optimization. $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ is initialized to 0 for all means and 1 for all variances. The AdaGrad optimizer with an initial learning rate of 0.1 is used for optimization. In Figure 2.5.3b, the hyperparameters for the negative exponential COCP are: $\gamma = 10000$, $\tau_0 = 15$ and $\tau_1 = 2$.

In Figure 2.5.5b, 10000 epochs of BBB is run. $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ is initialized to 0 for all means and 1 for all variances. The AdaGrad optimizer with an initial learning rate of 0.1 is used. Each epoch, we draw 5 samples of $\boldsymbol{\epsilon}$ (instead of 1, as presented in Algorithm 3) and take the average of the 5 resulting gradients. 1000 samples are drawn for prediction. In Figure 2.5.6a, we run 50 epochs of VOCP optimization. $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ is initialized to 1 for all means and 1 for all variances. The AdaGrad optimizer with an initial learning rate of 0.1 is used. In Figure 2.5.6b, the positive Gaussian COCP is used for all 3 constraints with $\sigma_{\mathcal{C}} = 1.25$. The training data is perturbed with Gaussian noise with mean 0 and standard deviation 1.

E.2 HIGH-DIMENSIONAL APPLICATIONS

CLINICAL ACTION PREDICTION From the original MIMIC-III dataset, data cleaning and feature selection were performed. The final dataset contains 9 features and a binary target, listed below. The features correspond to measured vital signs and various laboratory results. Each data point represents an hourly timestamp, however, as this is treated as a time-independent prediction problem, the timestamps themselves were not used as features. All continuous features were standardized.

- `map`: Continuous. Mean arterial pressure.
- `urine`: Continuous. Urine output.
- `weight`: Continuous. Weight of patient.
- `creatinine`: Continuous. Level of creatinine in the blood.
- `fio2`: Continuous. Fraction of inspired oxygen the patient is breathing in. 0.21 if the patient is on room air, but can be higher if artificial ventilation is given.
- `hr`: Continuous. Heart rate.
- `lactate`: Continuous. Level of lactate in the blood.
- `fio2_ind`: Binary. 1 if `fio2` was directly measured at that timestamp. 0 if the population median was used, or if it was imputed using the most recent value.
- `lactate_ind`: Binary. 1 if `lactate` was directly measured at that timestamp. 0 if the population median was used, or if it was imputed using the most recent value.
- `action` **(target)**: Binary. 1 if the amount of either vasopressor or IV fluid given to the patient (at that particular time step) is more than 0, and 0 otherwise.

The model for all experiments is a BNN with two 200-node RBF hidden layers ($H = 2$, $|\mathbf{L}_1| = |\mathbf{L}_2| = 200$). SVGD is used for inference with 50 particles, 1500 iterations and the AdaGrad optimizer with an initial learning rate of 0.75. The dataset is also batched during inference for efficiency. The size of the full dataset is 298666; this reduces to 124706 when points incompatible with $\mathcal{C}$ are filtered out. The train test split is 9:1. The positive Dirichlet COCP is used with $\boldsymbol{\alpha}_0 = 0.01$ and $\boldsymbol{\alpha}_1 = 10$.

RECIDIVISM PREDICTION We followed the same data processing steps as [44]. The only additional step taken was standardization of all continuous features. The final dataset contains 6172 data points, each corresponding to a defendant. There are 9 features and a binary target:

- `age`: Continuous. Age of defendant.
- `two_year_recid`: Binary. 1 if the defendant recidivated within two years of the current charge.
- `priors_count`: Continuous. Number of prior charges the defendant had.
- `length_of_stay`: Continuous. The number of days the defendant stayed in jail for the current charge.
- `c_charge_degree_F`: Binary. 1 if the current charge is a felony.
- `c_charge_degree_M`: Binary. 1 if the current charge is a misdemeanor.
- `sex_Female`: Binary. 1 if the defendant is female.
- `sex_Male`: Binary. 1 if the defendant is male.
- `race`: Binary. 1 if the defendant is African American.
- `compas_high_risk` **(target)**: Binary. 1 if COMPAS predicted the defendant as having a high risk of recidivism.

The model for all experiments is a BNN with two 100-node RBF hidden layers ($H = 2$, $|\mathbf{L}_1| = |\mathbf{L}_2| = 100$). SVGD is used for inference with 50 particles, 1000 iterations and the AdaGrad optimizer (with variable initial learning rates for the different experiments). The dataset is also batched during inference for efficiency. The VOCP is used with $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ initialized to 0 for all means and 1 for all variances. 50 epochs of VOCP optimization are performed using the AdaGrad optimizer at an initial learning rate of 0.1. We draw 30 samples from the convex hull of D_{tr} each iteration to compute the approximation for (2.24).

F

KEBO-VPME: Experimental Details

F.1 Toy Simulation in Figure 3.0.1

The model used is a BNN with a single 100-node RBF hidden layer ($H = 1$, $|\mathbf{L}_1| = 100$). The prior used is isotropic Gaussian (1.14) where $\sigma_\omega = 4$. The output noise in (1.12) is modeled as $\sigma_\epsilon = 3$ (same as the true random perturbation of D_{tr}).

In Figure 3.0.1a, for HMC, we discard 10000 samples as burn-in, before collecting 1000 samples at intervals of 10 (hence a total of 20000 Metropolis-Hastings iterations are carried out). We set $L = 100$ and $\epsilon = 0.03$.

In Figure 3.0.1b, for BBB, we perform 10000 epochs of updates. $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ is initialized to 2 for all means and e for all variances. The AdaGrad optimizer with an initial learning rate of 0.1 is used. Each epoch, we draw 5 samples of $\boldsymbol{\epsilon}$ (instead of 1, as presented in Algorithm 3) and take the average of the 5 resulting gradients. 1000 samples are drawn for prediction.

F.2 Low-Dimensional Simulations

The model for all experiments is a BNN with a single 10-node RBF hidden layer ($H = 1$, $|\mathbf{L}_1| = 10$). For the naive Gaussian prior (1.14), we set $\sigma_\omega = 1$. The output noise in (1.12) for regression experiments is modeled as $\sigma_\epsilon = 0.1$.

Where HMC is used, we discard 10000 samples as burn-in, before collecting 1000 samples at intervals of 10 (hence a total of 20000 Metropolis-Hastings iterations are carried out). We set $L = 50$ and $\epsilon = 0.03$. Where BBB is used, we run 10000 epochs of optimization, using the AdaGrad optimizer with an initial learning rate of 0.1. Each epoch, we draw 5 samples of $\boldsymbol{\epsilon}$ (instead of 1, as presented in Algorithm 3) and take the average of the 5 resulting gradients.

In Figure 3.3.1, we initialize $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ to 0 for all means and 1 for all variances. For computing the KEBO-VPME estimator and the strict bound, we set $\sigma_k = 5$ and

use $S = 1000$ samples of both HMC and VI.

In Figure 3.3.2, we use the same optimized $\boldsymbol{\theta}^*$ as in Figure 3.3.1, except compute the KEBO-VPME estimator and the strict bound and different σ_k and S. When S is varied, σ_k is set to 5. When σ_k is varied, S is set to 1000.

In Figure 3.3.3, for the 3 different VI runs, we initialize:

- Red run: $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ to 0 for all means and 1 for all variances.
- Blue run: $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ to -5 for all means and 1 for all variances.
- Black run: $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ to 5 for all means and 1 for all variances.

For computing the KEBO-VPME estimator and the strict bound, we set $\sigma_k = 5$ and use $S = 1000$ samples of both HMC and VI.

G

KEBO-VPME: Additional Proofs

G.1 Proof of Lemma 3.2.3

Proof. The Gaussian kernel over $\mathcal{W} = \mathbb{R}^M$ is Mercer [69] and so the eigenexpansion (3.10) exists. To derive (3.11) and (3.12), we will generalize from the one-dimensional case.

Section 4.3 of [61] gives us the eigenexpansion for $k(w, w') = \left\{ -\frac{(w-w')^2}{2\sigma_k^2} \right\}$ with the Gaussian measure $\mathcal{N}(0, 1)$ [1] as:

$$\lambda_j = \sqrt{B_k}(2B_k)^j \tag{G.1}$$

$$\mathbf{e}_j(w) = \exp\{-C_k w^2\} H_j\left(\sqrt{2C_k + \frac{1}{2}}w\right) \tag{G.2}$$

where $j \in \mathbb{N}$.

Since the Gaussian kernel is of product form, extending (G.1) and (G.2) to the M-dimensional case is straightforward:

$$\begin{aligned}
k(\mathbf{w}, \mathbf{w}') &= \exp\left\{ -\frac{||\mathbf{w} - \mathbf{w}'||_2^2}{2\sigma_k^2} \right\} \\
&= \prod_{m=1}^{M} \exp\left\{ -\frac{(\mathbf{w}_m - \mathbf{w}'_m)^2}{2\sigma_k^2} \right\} \\
&= \prod_{m=1}^{M} \sum_{j=1}^{\infty} \lambda_j \mathbf{e}_j(\mathbf{w}_m)\mathbf{e}_j(\mathbf{w}'_m) \quad \text{by (C.8)} \\
&= \sum_{j_1=1}^{\infty} \cdots \sum_{j_M=1}^{\infty} \Big((\lambda_{j_1} \dots \lambda_{j_M}) \left(\mathbf{e}_{j_1}(\mathbf{w}_1) \dots \mathbf{e}_{j_M}(\mathbf{w}_M)\right) \left(\mathbf{e}_{j_1}(\mathbf{w}'_1) \dots \mathbf{e}_{j_M}(\mathbf{w}'_M)\right) \Big)
\end{aligned}$$

[1] This result was first introduced in [79].

We want to factor the summations above in the form of (C.8), where we let $\lambda_{(j_1,\ldots,j_M)} = \lambda_{j_1} \ldots \lambda_{j_M}$ and $\mathbf{e}_{(j_1,\ldots,j_M)}(\mathbf{w}) = \mathbf{e}_{j_1}(\mathbf{w}_1) \ldots \mathbf{e}_{j_M}(\mathbf{w}_M)$. Let us verify that the resulting eigenbasis, (3.11) and (3.12), is valid.

Since the Cartesian product of countable sets is countable, the eigenbasis for $\mathcal{H}$ can indeed be indexed by the M-tuple $(j_1, \ldots, j_M)$. Since $\lambda_0 \geq \lambda_1 \geq \cdots > 0$ converges to 0, there is a valid ordering of $\left\{\lambda_{(j_1,\ldots,j_M)}\right\}_{j_1\in\mathbb{N},\ldots,j_M\in\mathbb{N}}$ such that the sequence is positive and converges to 0. Finally, let us verify that $\left\{\mathbf{e}_{(j_1,\ldots,j_M)}\right\}_{j_1\in\mathbb{N},\ldots,j_M\in\mathbb{N}}$ is an orthonormal set. We have:

$$
\begin{aligned}
&\left\langle \mathbf{e}_{(j_1,\ldots,j_M)}, \mathbf{e}_{(j'_1,\ldots,j'_M)} \right\rangle_{L^2(\mathcal{W};\mu)} \\
&= \int_{\mathcal{W}} \mathbf{e}_{(j_1,\ldots,j_M)}(\mathbf{w}) \mathbf{e}_{(j'_1,\ldots,j'_M)}(\mathbf{w}) \, \mathrm{d}\mu \\
&= \int_{\mathcal{W}} \mathbf{e}_{(j_1,\ldots,j_M)}(\mathbf{w}) \mathbf{e}_{(j'_1,\ldots,j'_M)}(\mathbf{w}) (2\pi)^{-M/2} e^{-\frac{1}{2}\mathbf{w}^\top \mathbf{w}} \, \mathrm{d}\mathbf{w} \\
&= \int_{\mathbb{R}} \cdots \int_{\mathbb{R}} \left(\mathbf{e}_{(j_1,\ldots,j_M)}(\mathbf{w}) \mathbf{e}_{(j'_1,\ldots,j'_M)}(\mathbf{w}) (2\pi)^{-M/2} e^{-\frac{1}{2}\mathbf{w}^\top \mathbf{w}} \right) \mathrm{d}\mathbf{w}_1 \ldots \mathrm{d}\mathbf{w}_M \\
&= \int_{\mathbb{R}} \ldots \left(\int_{\mathbb{R}} \mathbf{e}_{j_1}(\mathbf{w}_1) \mathbf{e}_{j'_1}(\mathbf{w}_1) (2\pi)^{-1/2} e^{-\frac{1}{2}\mathbf{w}_1^2} \mathrm{d}\mathbf{w}_1 \right) \ldots \mathrm{d}\mathbf{w}_M \\
&= \int_{\mathbb{R}} \ldots \left(\left\langle \mathbf{e}_{j_1}, \mathbf{e}_{j'_1} \right\rangle_{L^2(\mathbb{R},\mathcal{N}(0,1))} \right) \ldots \mathrm{d}\mathbf{w}_M
\end{aligned}
$$

Since each inner product is 1 iff $j_m = j'_m$ (and 0 otherwise), the set of nested integrals is 1 iff $(j_1, \ldots, j_M) = (j'_1, \ldots, j'_M)$ (and 0 otherwise). □

G.2 Proof of Lemma 3.2.6

This proof is taken from Lemma 3 in [27], except with the modification of the condition on the kernel to be $\mathbb{E}_{\mathbf{W}\sim p}[k(\mathbf{W},\mathbf{W})] < \infty$ instead of $\mathbb{E}_{\mathbf{W}\sim p}[\sqrt{k(\mathbf{W},\mathbf{W})}] < \infty$ (and similarly for the expectation over $q_{\boldsymbol{\theta}}$).

Proof. By the Riesz representation theorem (Theorem C.0.1), to show the existence of μ_p, it suffices to show that $\mathbb{E}_p$ is a bounded linear functional over $\mathcal{H}$. It is clear that $\mathbb{E}_p$ is linear. To show boundedness, it suffices to show that $\exists M > 0$ s.t. $|\mathbb{E}_p[f]| \leq M||f||$ for every $f \in \mathcal{H}$. Fix an arbitrary $f \in \mathcal{H}$. We have that:

$$
\begin{aligned}
|\mathbb{E}_p[f]| &= \Big| \int_{\mathcal{W}} p(\mathbf{w}) f(\mathbf{w}) \, \mathrm{d}\mathbf{w} \Big| \\
&\leq \int_{\mathcal{W}} p(\mathbf{w}) |f(\mathbf{w})| \, \mathrm{d}\mathbf{w} \\
&= \int_{\mathcal{W}} p(\mathbf{w}) |\langle k(\mathbf{w},\cdot), f\rangle| \, \mathrm{d}\mathbf{w} \\
&\leq \int_{\mathcal{W}} p(\mathbf{w}) \cdot ||f|| \cdot ||k(\mathbf{w},\cdot)|| \, \mathrm{d}\mathbf{w} \quad \text{(Cauchy-Schwarz Inequality)} \\
&= ||f|| \cdot \int_{\mathcal{W}} p(\mathbf{w}) \cdot \sqrt{k(\mathbf{w},\mathbf{w})} \, \mathrm{d}\mathbf{w} \quad \text{(since } \langle k(\cdot,\mathbf{w}), k(\cdot,\mathbf{w}')\rangle = k(\mathbf{w},\mathbf{w}')\text{)} \\
&= ||f|| \cdot \mathbb{E}_{\mathbf{W}\sim p}[\sqrt{k(\mathbf{W},\mathbf{W})}] \\
&\leq ||f|| \cdot \sqrt{\mathbb{E}_{\mathbf{W}\sim p}[k(\mathbf{W},\mathbf{W})]} \quad \text{(Jensen's Inequality)}
\end{aligned}
$$

which is sufficient since $\mathbb{E}_{\mathbf{W}\sim p}[k(\mathbf{W},\mathbf{W})]$ is finite. Furthermore, if μ_p exist, then:

$$
\mu_p(\mathbf{w}) = \langle \mu_p, k(\cdot,\mathbf{w})\rangle = \mathbb{E}_{\mathbf{W}\sim p}[k(\mathbf{w},\mathbf{W})] \quad \forall \mathbf{w} \in \mathcal{W} \tag{G.3}
$$

by the reproducing property of k. □

G.3 Proof of Lemma 3.2.7

Proof. It suffices to show that:

1. $\sup_{f \in \mathcal{H}_r} \langle f, g \rangle = r||g||$
2. $\inf_{f \in \mathcal{H}_r} \langle f, g \rangle = -r||g||$

We will prove it for the supremum, the proof for the infimum is analogous.

($\geq$)

$$\begin{aligned} |\langle f, g \rangle| &\leq ||f|| \cdot ||g|| \quad \forall f \in \mathcal{H}_r \text{ (Cauchy-Schwarz Inequality)} \\ &\leq r||g|| \quad \forall f \in \mathcal{H}_r \\ \langle f, g \rangle &\leq r||g|| \quad \forall f \in \mathcal{H}_r \\ \sup_{f \in \mathcal{H}_r} \langle f, g \rangle &\leq r||g|| \end{aligned}$$

($\leq$)

$$\begin{aligned} r||g||^2 &= r\langle g, g \rangle \\ &= ||g|| \Big\langle g, r\frac{g}{||g||} \Big\rangle \\ &\leq ||g|| \cdot \sup_{f \in \mathcal{H}_r} \langle f, g \rangle \\ &\quad \Big(\text{since } \Big|\Big| r\frac{g}{||g||} \Big|\Big| = r \Rightarrow r\frac{g}{||g||} \in \mathcal{H}_r\Big) \\ r||g|| &\leq \sup_{f \in \mathcal{H}_r} \langle f, g \rangle \end{aligned}$$

□

References

[1] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. propublica. *See https://www. propublica. org/article/machine-bias-risk-assessments-in-criminal-sentencing*, 2016.

[2] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[4] Avrim L Blum and Ronald L Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992.

[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[6] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[7] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

[8] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691, 2014.

[9] Samantha R Cook, Andrew Gelman, and Donald B Rubin. Validation of software for bayesian models using posterior quantiles. *Journal of Computational and Graphical Statistics*, 15(3):675–692, 2006.

[10] Mary Kathryn Cowles and Bradley P Carlin. Markov chain monte carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

[11] Balázs Csanád Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7, 2001.

[12] Leo L Duan, Alexander L Young, Akihiko Nishimura, and David B Dunson. Bayesian constraint relaxation. *Biometrika*, 107(1):191–204, 2020.

[13] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[14] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[15] Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 'in-between'uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.

[16] Robert Fortet and Edith Mourier. Convergence de la répartition empirique vers la répartition théorique. In *Annales scientifiques de l'École Normale Supérieure*, volume 70, pages 267–285, 1953.

[17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[18] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015.

[19] Marta Garnelo, Dan Rosenbaum, Chris J Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J Rezende, and SM Eslami. Conditional neural processes. *arXiv preprint arXiv:1807.01613*, 2018.

[20] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[21] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[22] John Geweke. Antithetic acceleration of monte carlo integration in bayesian inference. *Journal of Econometrics*, 38(1-2):73–89, 1988.

[23] Soumya Ghosh and Finale Doshi-Velez. Model selection in bayesian neural networks via horseshoe priors. *arXiv preprint arXiv:1705.10388*, 2017.

[24] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[25] Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

[26] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.

[27] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[28] Danijar Hafner, Dustin Tran, Timothy Lillicrap, Alex Irpan, and James Davidson. Noise contrastive priors for functional uncertainty. *arXiv:1807.09289*, 2018.

[29] W Keith Hastings. *Monte Carlo sampling methods using Markov chains and their applications*. Oxford University Press, 1970.

[30] 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*, pages 770–778, 2016.

[31] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[32] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[33] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

[34] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[35] Jiri Hron, Alexander G de G Matthews, and Zoubin Ghahramani. Variational bayesian dropout: pitfalls and fixes. *arXiv preprint arXiv:1807.01969*, 2018.

[36] Jonathan H. Huggins, Mikoaj Kasprzak, Trevor Campbell, and Tamara Broderick. Validated variational inference via practical posterior error bounds. *arXiv preprint arXiv:1802.02538*, 2019.

[37] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[38] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016.

[39] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[40] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[41] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

[42] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[43] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

[44] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. *ProPublica (5 2016)*, 9, 2016.

[45] 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.

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

[47] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pages 2378–2386, 2016.

[48] Marco Lorenzi and Maurizio Filippone. Constraining the dynamics of deep probabilistic models. *arXiv preprint arXiv:1802.05680*, 2018.

[49] Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716, 2016.

[50] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org, 2017.

[51] Christos Louizos, Xiahan Shi, Klamer Schutte, and Max Welling. The functional neural process. In *Advances in Neural Information Processing Systems*, pages 8743–8754, 2019.

[52] David JC MacKay. Probable networks and plausible predictionsa review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469–505, 1995.

[53] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[54] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[55] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[56] Radford M Neal. *BAYESIAN LEARNING FOR NEURAL NETWORKS*. PhD thesis, University of Toronto, 1995.

[57] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[58] Bernt Øksendal. Stochastic differential equations. In *Stochastic differential equations*, pages 65–84. Springer, 2003.

[59] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. *Automatic differentiation in pytorch*. 31st Conference on Neural Information Processing Systems, 2017.

[60] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, 2014.

[61] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[62] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[63] Walter Rudin. *Real and complex analysis*. Tata McGraw-hill education, 2006.

[64] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. 2002.

[65] Dino Sejdinovic and Arthur Gretton. *What is an RKHS?* Citeseer, 2012.

[66] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[67] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[68] 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.

[69] Hongwei Sun. Mercer theorem for rkhs on noncompact sets. *Journal of Complexity*, 21(3):337–349, 2005.

[70] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

[71] Vladimir Vapnik and Alexey Chervonenkis. *Theory of pattern recognition*. Nauka, Moscow, 1974.

[72] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[73] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[74] Andrew Gordon Wilson. The case for bayesian deep learning. *arXiv preprint arXiv:2001.10995*, 2020.

[75] Wanqian Yang, Lars Lorch, Moritz A Graule, Srivatsan Srinivasan, Anirudh Suresh, Jiayu Yao, Melanie F Pradier, and Finale Doshi-Velez. Output-constrained bayesian neural networks. In *2019 ICML Workshop on Uncertainty and Robustness in Deep Learning (UDL)*, 2019.

[76] Xiao Yang, Roland Kwitt, Martin Styner, and Marc Niethammer. Quicksilver: Fast predictive image registration–a deep learning approach. *NeuroImage*, 158: 378–396, 2017.

[77] Jiayu Yao, Weiwei Pan, Soumya Ghosh, and Finale Doshi-Velez. Quality of uncertainty quantification for bayesian neural network inference. *arXiv:1906.09686*, 2019.

[78] Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Yes, but did it work?: Evaluating variational inference. *arXiv:1802.02538*, 2018.

[79] Huaiyu Zhu, Christopher KI Williams, Richard Rohwer, and Michal Morciniec. *Gaussian regression and optimal finite dimensional linear models*. Aston University, 1997.