Exploring the Intersection of Bayesian Deep Learning and Gaussian Processes

Bogdan Kozyrskiy

EXPLORING THE INTERSECTION OF BAYESIAN DEEP LEARNING AND GAUSSIAN PROCESSES

BOGDAN KOZYRSKIY

SUPERVISOR:
Maurizio Filippone

ABSTRACT

Deep learning played a significant role in establishing machine learning as a must-have instrument in several modern industries and sciences. However, the use of deep learning poses several new challenges. On the one hand, deep learning requires significant computational power for training models and applying them. Modern hardware comes to its limits when it is necessary to provide access to deep learning for millions of users. The enormous energy consumption of such kind of applications poses additional difficulties. Another problem with modern deep learning is its inability to estimate the uncertainty of the predictions, which creates significant obstacles to deploying deep learning in risk-sensitive applications. This thesis presents four projects to address these problems.

- We propose an approach making use of Optical Processing Units for deep learning applications. Optical Processing Units (OPUs) are computing devices that perform random projections of input data by exploiting the physical phenomenon of scattering a light source through a diffusive medium. This operation can be employed to construct kernel ridge regression models and serve as a component in deep neural networks. This novel hardware requires less energy and time to perform computations compared to classical hardware. However, OPUs require the input data to be binary and performs a non-differentiable operation, which poses problems when using this device on non-binary datasets or when integrating it into models trained with backpropagation. We overcome this difficulty by considering OPU as a black box and employing the REINFORCE gradient estimator. Then we integrate the OPU into a deep model, where the first layers serve as a binary encoder. REINFORCE gradient estimator allows us to calculate the gradient of the loss function with respect to the weights of the binary encoder and optimize these together with the parameters of kernel ridge regression with gradient-based optimization Naturally, our approach allows for the integration of OPUs into deep models trained with backpropagation to bring the advantages of this hardware to the domain of deep learning.

- We address the problem of uncertainty estimates for classification with Bayesian inference. We introduce techniques for shallow and deep models that can decrease the cost of Bayesian inference.

  Carrying out Bayesian inference over parameters of statistical models is intractable when the likelihood and the prior are non-conjugate. Variational bootstrap provides a way to obtain samples from the posterior distribution over model parameters, where each sample is the solution of a task where the labels are perturbed. For Bayesian linear regression with a Gaussian likelihood, variational bootstrap yields samples from the exact posterior, whereas for nonlinear models with a Gaussian likelihood, some guarantees of approaching the true posterior can be established. In this part of the thesis, we extend variational bootstrap to the Bernoulli likelihood to tackle

classification tasks. We use a transformation of the labels, which allows us to turn the classification task into a regression one. Then we apply variational bootstrap to obtain samples from an approximate posterior distribution over the parameters of the model. Variational bootstrap allows us to employ advanced gradient optimization techniques, which provide fast convergence. We provide experimental evidence that the proposed approach allows us to achieve classification accuracy and uncertainty estimation comparable with Markov Chain Monte Carlo (MCMC) methods at a fraction of the cost.

- Classical non-parametric Bayesian linear models, such as Gaussian Processes, are hard to scale to large datasets. We developed a novel framework to accelerate Gaussian process regression (GPR). We considered localization kernels at each data point to down-weigh the contributions from other data points that are far away, and we derived the GPR model stemming from the application of such localization operation. Through a set of experiments, we demonstrated the competitive performance of the proposed approach compared to full GPR, other localized models, and deep Gaussian processes. Crucially, these performances were obtained with considerable speedups compared to standard global GPR due to the sparsification effect of the Gram matrix induced by the localization operation.

- Specifying sensible priors for Bayesian neural networks (BNNs) is key to obtaining good predictive performance and sound predictive uncertainties. However, this is generally difficult because of the complex way prior distributions induce distributions over the functions that BNNs can represent. Switching the focus from the prior over the weights to such functional priors allows for the reasoning on what meaningful prior information should be incorporated. We propose enforcing such meaningful functional priors through Gaussian processes (GPs), which we view as a form of implicit prior over the weights. We employ scalable Markov chain Monte Carlo (MCMC) to obtain samples from the posterior distribution over BNN weights. Unlike previous approaches, our proposal does not require the modification of the original BNN model, it does not require any expensive preliminary optimization, and it can use any MCMC techniques and any functional prior that can be expressed in closed form. We illustrate the effectiveness of our approach with an extensive experimental campaign.

PUBLICATIONS

[1] Davit Gogolashvili, Bogdan Kozyrskiy, and Maurizio Filippone. "Locally Smoothed Gaussian Process Regression." In: *Procedia Computer Science* 207 (2022), pp. 2717–2726.

[2] Bogdan Kozyrskiy, Maurizio Filippone, Iacopo Poli, Ruben Ohana, Laurent Daudet, and Igor Carron. "Learning Binary Data Representation for Optical Processing Units." In: *Sensors & Transducers* 256.2 (2022), pp. 1–11.

[3] Bogdan Kozyrskiy, Dimitrios Milios, and Maurizio Filippone. "Variational Bootstrap for Classification." In: *Procedia Computer Science* 207 (2022), pp. 1222–1231.

[4] Bogdan Kozyrskiy, Dimitrios Milios, and Maurizio Filippone. "Deep Neural Networks with Intersection over Union Loss for Binary Image Segmentation." In: *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2023, Lisbon, Portugal, February 21-24, 2023*. SciTePress, in press 2023.

[5] Bogdan Kozyrskiy, Iacopo Poli, Ruben Ohana, Laurent Daudet, Igor Carron, and Maurizio Filippone. "Binarization for Optical Processing Units via REINFORCE." In: *Proceedings of the 3rd International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI'2021), Porto, Portugal*. 2021, pp. 23–27.

ACKNOWLEDGMENTS

During my PhD journey, I was fortunate to have met and been influenced by many remarkable people who have inspired and supported me in my work. I am now grateful for the opportunity to express my gratitude to everyone who made this journey possible.

First and foremost, I would like to express my immense gratitude to Maurizio Filippone for his unrelenting support, guidance, and patience throughout my PhD. I am genuinely thankful for the countless hours he spent reviewing drafts, providing suggestions, and offering constructive criticism.

I would also like to extend my gratitude to the 3IA Interdisciplinary Institute for Artificial Intelligence, and through them, to the French government for their confidence and financial support during my PhD.

My sincere thanks go to EURECOM for providing the administrative support and infrastructure that made my research possible. I am grateful for the research-friendly environment they provided, which made my dissertation work an enjoyable experience.

I am thankful to all of my colleagues, especially those I worked closely with. In particular, I am grateful to Dimitrios Milios for the numerous inspiring discussions and advice that allowed many of the ideas presented in this work to emerge. I want to thank Davit Gogolashvili for his valuable ideas and collaboration on Chapter 5 of this work. I am also grateful to Jonas Wacker and Simone Rossi for answering my endless questions and introducing me to the environment. Finally, I want to express my gratitude to Ba-Hien Tran for the daily conversations that provided a much-needed break from the intensity of research.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

BNN Bayesian Neural Network
VI Variational Inference
GP Gaussian Process
KRR Kernel Ridge Regression
RKHS Reproducing Kernel Hilbert Space
LRR Linear Ridge Regression
OPU Optical Processing Unit
RFF Random Fourier Features
RBF Radial Basis Function
DFA Direct Feedback Alignment
MSE Mean Squared Error
CE Cross Entropy
MAC Method of Auxiliary Coordinates
POVI Particle Optimization Variational Inference
MLP Multilayer Perceptron
MCMC Markov Chain Monte-Carlo
SGHMC Stochastic Gradient Hamiltonian Monte Carlo

1 INTRODUCTION

Recent advances in machine learning allowed to completely reshape a lot of fields of technology, starting from accessible consumer-level computer vision systems [54], dialog [98], translation systems [140] to complicated systems for controlling and improving production [77], and financial risk assessment [141]. In addition, modern machine learning techniques are used to drive scientific discoveries in various fields like experimental physics [30] and mathematics [74], medicine [58], chemistry [3], and many others [109].

One of the fascinating things about these advances is that machine learning as a field existed for a long time, but its applications were scarce. We can track the origins of machine learning to the middle of the 20th century. Arthur Samuel was one of the first who started to use the term "machine learning" [119]. His work in the late 40s introduces a self-learning computer program for playing checkers and introduces the first machine learning algorithms. Later, Rosenblatt combined Samuel's ideas with the model of brain cell interaction, proposed by [56], to introduce a perceptron model [116]. At that time, this method mainly had theoretical value as a step to understanding the principles of information processing in the brain. Still, the practical applicability of the method was limited to some image processing problems [91]. At the time, it seemed that the future of machine learning was in more sophisticated techniques from mathematical statistics.

One such statistical method that impacted machine learning was the classification algorithm proposed by [40]. It played a notable role in applied machine learning in the second half of the 20th century for developing various computer vision systems [137], [100].

In the 70s, Vapnik [138] and Chervonenkis [139] developed a powerful theoretical foundation for a new type of machine learning algorithms called Support Vector Machines. These algorithms proved themselves very powerful and successful in regression, classification, and clustering problems. Among the aforementioned methods, SVMs probably have had the most significant impact in industry at the time and anticipated the deep learning revolution. They were used for natural language processing [61], various tasks in computer vision [83], problems in biology [52], and other sciences [53].

The vanilla versions of the aforementioned methods are linear with respect to input features. Thus, their area of application was quite restricted. They owe their success on a practical level to the so-called kernel trick [1], wich allowed SVMs [11], and other classical statistical methods, like Ridge Regression [120], to tackle complex non-linear problems. At the same time, kernel methods made possible the application of Gaussian Processes, proposed by [62] to tackle machine learning problems [99], giving the linear models not only the flexibility to resolve non-linear problems but also an ability to estimate the uncertainty of the model.

Thus, we can conclude that the advances of practical machine learning in the second half of the 20th century were mainly based on the solid foundations of mathematical statistics and their introduction to the Computer Science community. This strong basis enabled accurate predictions but also a theoretical analysis of model's properties, like statistical guarantees for the risk on unseen data [16], model interpretability [128], and uncertainty quantification [114]. The overwhelming success of deep learning at the beginning of the 21st century resulted from a completely different approach based on vast amounts of data, computational resources, and the simple Perceptron model proposed half a century ago. Extended with gradient descent optimization and backpropagation to train deeper models, it enabled striking versatility. Deep learning reduces pipeline creation time by automatizing the burden of feature engineering due to its ability to train machine learning pipelines in an "*end-to-end*" fashion.

Deep neural networks can solve much more complicated problems, but with a cost. Theoretical guarantees of deep models are an open question in machine learning, except for some exceptional cases [59]. Predictions of these models tend to be overconfident [51] and hard to interpret. The community is constantly looking for effective solutions to counter these problems, but cheap and reliable solutions are still lacking. Nowadays, it may seem that most of the advances of machine learning in the 20th century did not play any role in the current state of the art.

This work proposes four separate projects connected by the idea of reintroducing some advances of classical machine learning and merging them with the power of deep neural networks. In this chapter, we will discuss the motivation and contributions of each project.

1.1 MACHINE LEARNING AT THE SPEED OF LIGHT

As mentioned above, the success of deep learning became possible due to significant advances in hardware. General purpose CPUs are not very efficient in matrix and tensor operations, which are a building block of all algorithms for inference and training of neural networks. Nowadays, GPUs and TPUs are absolutely necessary for research and practical applications of deep learning. The problem is that even this specialized hardware can face performance problems with applications that require high loads, such as requests from a significant amount of users or applications that require low latency.

One of the possible solutions to this problem is to use a completely different approach to perform computations. In Chapter 3 we consider a device that uses light to perform computations. It projects an input vector into a high-dimensional feature space using random projections. Then, it is possible to use these features to solve a linear regression or classification problem. From the mathematical point of view, this device allows solving a Kernel Ridge Regression (KRR) problem with a particular type of kernel: the *OPU-kernel*.

Utilization of the considered device for practical applications poses two main challenges. The first problem appears due to the fact that Optical Processing Units (OPUs) can work only with binary input vectors, and most real-life datasets, considered by machine learning, are not binary. It means that machine learning pipelines containing an OPU should also include a binarization mechanism for

input data. The second challenge, posed by the OPU architecture, is due to the fact that OPUs performs an operation that is non-differentiable. This means that it is not possible to integrate OPUs into the deep learning pipelines for "*end-to-end*" training using standard backpropagation. This situation becomes especially problematic when it is necessary to work with complicated feature extraction procedures that should be trained with the rest of the pipeline, like, for example, image data, that requires using convolutions to extract shift-invariant features.

Chapter 3 is dedicated to creating a training procedure for resolving these problems. The main contributions of Chapter 3 are:

- a training method that allows building heterogeneous pipelines with the OPU and training them in an "*end-to-end*" fashion;
- an empirical demonstration that our approach allows obtaining binary embeddings that improve the performance of the pipeline on classification or regression problems compared to non-supervised binarization techniques;
- an empirical demonstration that the proposed procedure allows integrating the OPU as a part of complex deep learning pipelines to address machine learning problems that require learnable feature extraction procedures.

1.2 PERFORMING CLASSIFICATION IN A BAYESIAN WAY

One of the common pitfalls of modern deep learning is overconfidence in predictions of deep models. Classification models take a vector as input and return a probability of belonging of this input vector to classes defined by the problem. But in practice, deep models assign a class label with high probability, even for input vectors that lie far away from data distribution and do not belong to any class of interest. We can consider a neural network trained to classify pictures of dogs and cats as a simple example. If we feed the model an image of a car, it will assign it to one of the classes with a probability close to 1, while the desired behavior would be 0.5 probability for each of the two classes.

The problem of overconfidence in predictions can be addressed with a Bayesian approach. Instead of producing a single prediction, Bayesian Neural Network produces the distribution of class probabilities or samples from this distribution. Analyzing the entropy of this distribution, it is possible to reason about the model's confidence level.

However, this property of Bayesian neural networks comes with a cost. There are multiple approaches to performing Bayesian inference for Bayesian Neural Networks, such as Variational Inference (VI) or Markov Chain Monte-Carlo (MCMC). VI approaches, in most cases, are unable to recover the true posterior predictive distributions, which is why they provide inadequate estimates of the uncertainty of the predictions. MCMC approaches provide good uncertainty estimates, but they require a lot of time to converge. Thus the main challenges for creating an inference procedure are obtaining reliable uncertainty estimates of the predictions and decreasing the time necessary for the inference procedure.

We consider VI and MCMC approaches in detail in Chapter 2

Chapter 4 is dedicated to addressing these challenges. In summary, in Chapter 4 we make the following contributions:

- we propose a novel algorithm for performing Bayesian inference for classification models;
- we demonstrated its applicability both for linear models and deep neural networks;
- we empirically demonstrated that the proposed method achieves a faster convergence rate than the MCMC approach while preserving comparable quality uncertainty estimations.

1.3 GAUSSIAN PROCESSES IN THE AGE OF BIG DATA

Nowadays, it is common to rely on big amounts of data to train statistical models. As mentioned, the success of deep learning owes much to a large amount of training data. And deep models can benefit from the big amount of data because of the vast amount of parameters that they have. If the big amount of data and parameters is the key to the machine learning method's success, can we extend the same principle to classical machine learning algorithms? Classical machine learning contains a category of methods based on this principle called non-parametric methods. This category includes many methods, like the aforementioned Gaussian Processes and other types of kernel machines. The general idea of all these methods is to grow the capacity of the model with the size of the training dataset. The problem is that most of these methods are not adapted to huge datasets of the deep learning era.

In Chapter 5 we consider one such method: Gaussian Process Regression. This algorithm possesses several nice properties, like flexibility, thanks to the use of kernels, which allows encoding various constraints on a modeling function, like smoothness, trend, or seasonality. In addition to this, Gaussian Processs (GPs) are Bayesian methods, which means, that they can provide information about the uncertainty of the predictions, and they have a natural mechanism of taking into account the information about measurement noise in training labels. However, classical GPs cannot deal with large data sets even using modern hardware. Another problem with this algorithm is that it uses that for most of the popular kernel functions, like Radial Basis Function (RBF) or Matérn, training points that lie far away from the input point of interest do not affect the prediction, while affecting the computational complexity of the algorithm.

We discuss GPs in detail in Section 2.2.1

There were multiple attempts to make GP scalable to large data sets, but most of these methods are based on distilling or synthesizing some small subset from training data and then making all predictions based on this small subset. We believe that this approach contradicts the philosophy that made deep learning successful. Indeed, modern neural networks very often have a larger number of parameters than the number of training samples. Thus, they do not perform a shrinking of any sort. In Chapter 5, we propose a method that extends the Gaussian Process Regression algorithm that is able to perform at scale while keeping all the training data available for predictions. To obtain this property, we leveraged the idea of localization. For each test data point, we use only the training points relevant to this specific region of space. From one point of view, that is how we avoid wasting computational resources on training points that do not affect this

We consider different approaches to build scalable GPs at Section 5.2

specific prediction. From another point of view, the test set, with points located in different regions of input space, is capable of leveraging the full training set.

The contributions of Chapter 5 are:

- the approach that allows performing Gaussian Process Regression at scale while effectively using computational resources and exploiting the full size of the training set;
- the empirical demonstration of the superiority of the proposed approach above classical GP Regression and other localization and sparsification techniques in terms of computational complexity and prediction performance.

1.4 HOW GAUSSIAN PROCESSES CAN IMPROVE BAYESIAN NEURAL NETWORKS

Bayesian neural networks have the same advantages as GP, like prediction uncertainty estimation and the ability to account for noise in training labels, but in addition they possess the expressiveness of deep models. The problem is that the Bayesian approach requires choosing the prior over the variable of interest. In the case of Bayesian Neural Network (BNN) it is the vector of neural network weights. There are a lot of approaches to impose different kinds of prior distributions over the weights. Still, most of them cannot address the main challenge of imposing meaningful functional properties on the model's output by imposing prior distributions over the model's weights. The described situation is a particular case of the problem, general for most deep learning techniques: lack of interpretability. But in the particular case of BNNs, it works the other way around. In standard deep learning, the functional properties of the model reflect the properties of a training set, but it is hard to interpret the intrinsic properties and structure of the weight. In BNNs, it is hard to interpret how the properties of the weights imposed by the prior affect the model's output.

We discuss in detail priors for BNNs in Section 6.2

In Chapter 6, we proposed a solution for this problem. Instead of manipulating the model's weights by imposing a prior, we decided to put prior distribution over the model's output. For this purpose, we need a distribution over functions. Such distribution can be provided by a GP. The idea of imposing the prior over BNNs is not novel, but we managed to propose a generalized approach that uses MCMC, standard for the BNN inference without any additional approximations. While our method does not provide exact posterior inference, we demonstrated empirically that it is competitive with other similar techniques. The main contributions of Chapter 6 are:

- A scalable approach for the inference of BNNs with a GP prior using Markov Chain Mote-Carlo
- Empirical demonstration of the effectiveness of the proposed approach compared to other methods of imposing a functional prior.

2 BACKGROUND

2.1 LINEAR METHODS IN MACHINE LEARNING

2.1.1 *Linear Ridge Regression*

Let $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_n$ be a set of input vectors $\mathbf{x} \in \mathbb{R}^d$ and let $\mathbf{y} = y_1, \ldots, y_n$ be a set of labels associated with the input vectors. Let's model the connection between the output y and the input vector $\mathbf{x}$ in the following way:

$$y_i = f(\mathbf{x}_i, \mathbf{w}) + \varepsilon, \ \varepsilon \sim \mathcal{N}(0, \sigma_{\text{noise}}), \tag{2.1}$$

where $f(\phi(\mathbf{x}), \mathbf{w})$ is a deterministic function with parameters $\mathbf{w}$ and $\phi(\cdot)$ is some function that extracts features from $\mathbf{x}$. In this section, we will consider f to be a linear function

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \phi(\mathbf{x}). \tag{2.2}$$

Given the Gaussian assumption about the noise ε, we can see that the likelihood of observing a response y given the input $\mathbf{x}$ is Gaussian.

$$p(y|\mathbf{x}) = \mathcal{N}(y|f(\phi(\mathbf{x}), \mathbf{w}), \sigma_{\text{noise}}). \tag{2.3}$$

Now we can find a maximum likelihood estimate of the parameters of the model $\mathbf{w}$. It is easier to minimize a log-likelihood instead of maximizing the likelihood itself. We will obtain an equivalent solution because the logarithm is a monotonically increasing function

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2\sigma^2_{\text{noise}}} \sum_{i=1}^{n} (y_i - \mathbf{w}^\top \phi(\mathbf{x}_i))^2 + \text{const}. \tag{2.4}$$

It is possible to introduce an L2 regularization term to the optimization objective that will penalize large values in $\mathbf{w}$. Thus, we can write the optimization objective of Linear Ridge Regression (LRR):

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} (y_i - \mathbf{w}^\top \phi(\mathbf{x}_i))^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2, \tag{2.5}$$

where parameter λ controls the level of regularization. In this formulation, it reflects the assumption regarding the variance of the observation noise $\lambda = \sigma^2_{\text{noise}}$.

2.1.2 *Kernel Ridge Regression*

Now we can consider a KRR algorithm, which is an extension of LRR. Kernel Ridge Regression (KRR) is a statistical model which constructs a functional relationship between the inputs and the labels which belongs to the so-called Reproducing

Kernel Hilbert Space (RKHS). The properties of such functions, such as smoothness, are characterized by choice of a so-called kernel function $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ [90], which is a positive semi-definite function of pairs of input points returning a scalar. The reproducing property of kernel functions is $\langle k(\mathbf{x},\cdot), k(\mathbf{y},\cdot)\rangle = k(\mathbf{x},\mathbf{y})$. According to Mercer theorem, positive definiteness of kernel functions implies that we can express $k(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$ for some set of (possibly infinite) basis functions $\phi(\cdot)$.

In order to derive the conventional formulation of kernel ridge regression, it is useful to start from linear regression, where a set of model parameters $\mathbf{w}$ is introduced to express a linear relationship between inputs and labels. Then, one introduces the following optimization problem:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2. \tag{2.6}$$

The objective function contains two terms; the first is a model fitting term, while the second is a regularization term, which prevents the weights from becoming too large. The solution to this optimization problem is available in closed form, given that the objective is quadratic with respect to the parameters, yielding:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{y}. \tag{2.7}$$

Using standard algebraic manipulations involving the Woodbury identity, we can re-express the solution as:

$$\hat{\mathbf{w}} = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})^{-1}\mathbf{y}. \tag{2.8}$$

While this is more costly than the previous expression in the common case where $d < n$ (inversion of a $n \times n$ matrix rather than a $d \times d$ matrix), this formulation is useful to derive kernel ridge regression.

Imagining to introduce basis functions $\phi(\cdot) = (\phi_1(\cdot), \ldots, \phi_D(\cdot))^\top$, we can solve this new optimization problem

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{w}\phi(\mathbf{x}_i)) + \frac{\lambda}{2}||\mathbf{w}||^2, \tag{2.9}$$

with solution

$$\hat{\mathbf{w}} = \mathbf{\Phi}^\top (\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda \mathbf{I})^{-1}\mathbf{y}. \tag{2.10}$$

Evaluating the model at a given input $\mathbf{x}_*$ yields:

$$\phi(\mathbf{x}_*)^\top \hat{\mathbf{w}} = \phi(\mathbf{x}_*)^\top \mathbf{\Phi}^\top (\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda \mathbf{I})^{-1}\mathbf{y}. \tag{2.11}$$

In this expression, we recognize the scalar product of vectors of basis functions. Then we can express these scalar products as a kernel function and obtain the following:

$$\phi(\mathbf{x}_*)^\top \hat{\mathbf{w}} = \mathbf{k}_*(\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{y}, \tag{2.12}$$

where $\mathbf{k}_* = [k(\mathbf{x}_1,\mathbf{x}_*), \ldots, k(\mathbf{x}_n,\mathbf{x}_*)]^\top$ and $K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$. In practice, one first chooses a kernel function, and this induces a set of basis functions; the beauty of

this formulation is that one never explicitly works with the set of basis functions, and all we need to use this model in practice is the evaluation of kernel functions among inputs.

The same method can be approached from a functional space point of view. In this case, it is necessary to solve an optimization problem in RKHS imposed by the kernel function $k(\cdot,\cdot)$ with respect to a regularization constraint over the RKHS norm of the function.

$$f^*(\cdot) = \underset{f\in\mathcal{H}}{\operatorname{argmin}} \frac{1}{2\sigma_n^2}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 + \frac{1}{2}||f||_{\mathcal{H}}^2. \tag{2.13}$$

The Representer theorem [89] shows that the solution to this problem has the following form:

$$f^*(\cdot) = \sum_{i=1}^{n}\alpha_i k(\cdot,\mathbf{x}_i). \tag{2.14}$$

We can find an expression for $f^*(\cdot)$ by substituting this result into (2.13) and solving it with respect to $\boldsymbol{\alpha}$

$$f^*(\mathbf{x}_*) = \mathbf{k}_*(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}, \tag{2.15}$$

which is equivalent to the expression (2.13) for $\sigma_n^2 = \lambda$.

The choice of the kernel for the KRR algorithm strongly depends on the specific problem. The most popular kernel is a RBF kernel, also known as the Gaussian kernel:

$$k(\mathbf{x},\mathbf{x}') = \alpha\exp(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{l}). \tag{2.16}$$

This kernel imposes a smoothness assumption on the model $f(\cdot)$, and it has two parameters α, l. Using these parameters, it is possible to control the amplitude and smoothness of the modeling function $f(\cdot)$.

Another popular choice for the kernel function is the Polynomial Kernel:

$$k(\mathbf{x},\mathbf{x}') = (\mathbf{x}^\top\mathbf{x}' + c)^d. \tag{2.17}$$

This kernel enables modeling of linear and non-linear relationships between input and output variables, depending on the degree d. In fact, it maps the input data into a higher-dimensional space, and it is particularly useful when the amplitude of the input vector $\mathbf{x}$ has significance for the task, as is often the case with computer vision tasks, for example.

2.1.3 *Random Features*

One of the main limitations of kernel methods is the scalability to large datasets. The problem arises from the need to evaluate and perform algebraic operations with the so-called Gram matrix $\mathbf{K}$. Because $\mathbf{K}$ is an $n \times n$ matrix, evaluating and storing $\mathbf{K}$ requires $\mathcal{O}(n^2)$ computations and storage, while any algebraic operations, such as factorization to handle the inverse of $\mathbf{K} + \lambda\mathbf{I}$, requires $\mathcal{O}(n^3)$ operations. These prevent the applicability of kernel methods in their exact form to datasets of

size beyond a few thousand. It is worth noting that some approaches have been proposed to solve algebraic operations in an iterative fashion and without the need to store $\mathbf{K}$ [39], [28], [144]. However, they still require $\mathcal{O}(n^2)$ computations for each iteration of their solvers. Furthermore, while the number of iterations of the solvers is much lower than n in practice, in the worst case, it can be $\mathcal{O}(n)$, leading to a worst-case complexity of $\mathcal{O}(n^3)$.

The literature offers a number of solutions to scale kernel methods to large data linearly in the number of data, such as Nyström approximations [43] and random features [111]. In this section, we consider the random features approach and in Chapter 3 we will consider a hardware implementation of random features in Optical Processing Units (OPUs) [118]. Random feature approximations form a class of approximations that attempt to construct a finite set of basis functions $\phi(\cdot) \in \mathbb{R}^D$ such that

$$k(\mathbf{x}_i, \mathbf{x}_j) \approx \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j). \tag{2.18}$$

There are different ways to construct such sets of basis functions, depending on the kernel. For example, so-called random Fourier features are commonly employed to approximate the Gaussian kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2). \tag{2.19}$$

Appealing to Bochner's theorem [111], this kernel, which is shift-invariant due to the dependence on $\boldsymbol{\tau} = \mathbf{x}_i - \mathbf{x}_j$, admits an alternative expression as:

$$k(\boldsymbol{\tau}) = \int p(\boldsymbol{\omega}) \exp(i2\pi\boldsymbol{\omega}\boldsymbol{\tau}) d\boldsymbol{\omega}, \tag{2.20}$$

where $p(\boldsymbol{\omega})$ is a proper density function and $i = \sqrt{-1}$. Interpreting this as an expectation under $p(\boldsymbol{\omega})$, it is possible to approximate the integral as an expectation using Monte Carlo:

$$k(\boldsymbol{\tau}) = \frac{1}{D} \sum_r \exp(i2\pi\boldsymbol{\omega}(r)\boldsymbol{\tau}), \tag{2.21}$$

with $\boldsymbol{\omega}^{(r)} \sim p(\boldsymbol{\omega})$. Furthermore, it is possible to use simple trigonometric identities to verify that the complex exponential can be broken down as a scalar product with terms depending on $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{D} \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j), \tag{2.22}$$

with

$$\phi_r(\mathbf{x}) = (\sin(\mathbf{x}^\top \boldsymbol{\omega}^{(r)}), \cos(\mathbf{x}^\top \boldsymbol{\omega}^{(r)})). \tag{2.23}$$

We refer the reader to [111], [24], [27], [143] for random features derived from alternative integral representation to the Fourier transform.

2.2 BAYESIAN METHODS

In classification and regression tasks, the objective is to tune the parameters $\mathbf{w}$ of a function $f(\mathbf{x}, \mathbf{w})$ so that it best explains a set of given observations. The Bayesian paradigm dictates that we specify a prior distribution $p(\mathbf{w})$, which captures any prior knowledge about the model. For a dataset $\mathcal{D}\{(\mathbf{x_i}, y_i) \mid i = 1 \dots n\}$ of n input vectors $\mathbf{x_i}$ and labels y_i, it is possible to define the posterior distribution over $\mathbf{w}$ after observing the data $\mathcal{D}$ by means of Bayes rule:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{\int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w}}, \tag{2.24}$$

where $p(\mathcal{D}|\mathbf{w})$ is known as the *likelihood* of the model, while the integral term in the denominator above is referred to as *marginal likelihood* or *evidence*.

2.2.1 *Bayesian Linear Models for Regression*

BAYESIAN LINEAR REGRESSION We can consider LRR from a Bayesian perspective. If we assume that the prior distribution over the parameters $\mathcal{N}(0, \sigma_{\text{prior}})$ is Gaussian, we can easily verify that the maximum a posteriori estimate of the parameters $\mathbf{w}$ corresponds to the LRR optimization objective.

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \phi(\mathbf{x}_i))^2 + \frac{1}{2}\frac{\sigma^2_{\text{noise}}}{\sigma^2_{\text{prior}}}\|\mathbf{w}\|_2^2. \tag{2.25}$$

But for the linear regression case, it is possible to go further. Due to the fact, that the Gaussian prior is conjugate to the Gaussian likelihood [7] it is possible to solve the integral in the denominator of (2.24) analytically and obtain an analytical expression for the posterior distribution of the parameters $\mathbf{w}$. After observing n data points, the posterior distribution of $\mathbf{w}$ will be Gaussian with parameters:

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{w}} &= \sigma^{-2}_{\text{noise}}\boldsymbol{\Sigma}_{\mathbf{w}}\boldsymbol{\Phi}^\top \mathbf{y}, \\ \boldsymbol{\Sigma}^{-1}_{\mathbf{w}} &= \sigma^{-2}_{\text{prior}}\mathbf{I} + \sigma^{-2}_{\text{noise}}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}. \end{aligned} \tag{2.26}$$

Having the posterior distribution over the weights it is possible to write the posterior predictive distribution given the test point $\mathbf{x}^*$

$$\begin{aligned} p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) &= \mathcal{N}(y_*|\boldsymbol{\mu}_{\mathbf{w}}^\top\phi(\mathbf{x}_*), \sigma_{y_*}), \\ \sigma^2_{y_*} &= \sigma^2_{\text{noise}} + \phi(\mathbf{x}_*)^\top\boldsymbol{\Sigma}_{\mathbf{w}}\phi(\mathbf{x}_*). \end{aligned} \tag{2.27}$$

GAUSSIAN PROCESS A GP regression can be seen as a Bayesian equivalent of the KRR method. And as the KRR method, GP regression can be approached from the parametric or the functional space points of view.

Let's consider a parametric approach first. We can re-express (2.27) using the kernel $k(\mathbf{x}, \mathbf{x}') = \sigma^2_w\phi(\mathbf{x})^\top\phi(\mathbf{x}')$. In this case, the mean of the predictive distribution for the input $\mathbf{x}_*$ is given by the following expression:

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{w}}^\top\phi(\mathbf{x}^*) =& (\sigma^{-2}_{\text{noise}}(\sigma^{-2}_{\text{prior}}\mathbf{I} + \sigma^{-2}_{\text{noise}}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y})^\top\phi(\mathbf{x}^*) = \\ & \sigma^2_{\text{prior}}\phi(\mathbf{x}_*)^\top\boldsymbol{\Phi}^\top(\sigma^2_{\text{noise}}\mathbf{I} + \sigma^2_{\text{prior}}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1}\mathbf{y} = \\ & \mathbf{k}_*(\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I})^{-1}\mathbf{y}. \end{aligned} \tag{2.28}$$

And the variance of the predictive distribution becomes:

$$\begin{aligned}\sigma^2_{y_*} &= \sigma^2_{\text{noise}} + \phi(\mathbf{x}_*)^\top(\sigma^2_{\text{noise}}\mathbf{I} + \sigma^2_{\text{prior}}\mathbf{\Phi}^\top\mathbf{\Phi})^{-1}\phi(\mathbf{x}_*) = \\ &\sigma^2_{\text{noise}} + \phi(\mathbf{x}_*)^\top(\sigma^2_{\text{prior}}\mathbf{I} - \sigma^2_{\text{prior}}\mathbf{\Phi}^\top(\sigma^2_{\text{noise}}\mathbf{I} + \sigma^2_{\text{prior}}\mathbf{\Phi}\mathbf{\Phi}^\top)^{-1}\sigma^2_{\text{prior}}\mathbf{\Phi})\phi(\mathbf{x}_*) = \\ &\sigma^2_{\text{noise}} + k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top(\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I})^{-1}\mathbf{k}_*.\end{aligned} \tag{2.29}$$

From the functional space point of view, GP is a distribution over functions.

Definition 1 *A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution [114].*

The distribution of a finite set of these variables is determined by the mean function $m(\mathbf{x})$ and the kernel function $k(\cdot,\cdot)$:

$$\begin{aligned}m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')].\end{aligned} \tag{2.30}$$

Let's consider a GP with a constant mean function $m(\mathbf{x}) = 0$. We can denote the GP as

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}')). \tag{2.31}$$

To apply this concept to regression problems, we consider the model

$$y_i = f(\mathbf{x}_i) + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma_{\text{noise}}). \tag{2.32}$$

We can choose a GP with $m(\mathbf{x}) = 0$ as a prior distribution over $f(\mathbf{x})$. Given the assumption about the Gaussainity of the noise case, the prior distribution over the outputs is also Gaussian.

$$\mathbf{y} \sim \mathcal{N}(0, \mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I}),\ \text{where } \mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j). \tag{2.33}$$

Now, given the dataset $\mathcal{D}\{(\mathbf{x_i}, y_i) \mid i = 1 \dots n\}$, it is possible to obtain a posterior distribution over $f(\cdot)$ at the test point $\mathbf{x}_*$. For a non-parametric model such as GP the posterior distribution plays a role of a predictive distribution of a parametric model. The posterior distribution over $f(\cdot)$ is also a GP:

$$f(\mathbf{x})|\mathbf{X}, \mathbf{y} \sim \mathcal{GP}(\hat{m}(\mathbf{x}), \hat{k}(\mathbf{x}, \mathbf{x}')). \tag{2.34}$$

Using the expression for a conditional Gaussian distribution [114], we can obtain the functions $\hat{m}(\mathbf{x})$ and $\hat{k}(\mathbf{x}, \mathbf{x}')$:

$$\begin{aligned}\hat{m}(\mathbf{x}) &= \mathbf{k}\left(\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I}\right)^{-1}\mathbf{y}, \\ \hat{k}\left(\mathbf{x}, \mathbf{x}'\right) &= k\left(\mathbf{x}, \mathbf{x}'\right) - \mathbf{k}\left(\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I}\right)^{-1}\mathbf{k}',\end{aligned} \tag{2.35}$$

where $\mathbf{k}^\top = [k(\mathbf{x}, \mathbf{x}_1), \dots, k(\mathbf{x}, \mathbf{x}_n)]$ and $\mathbf{k'}^\top = [k(\mathbf{x}', \mathbf{x}_1), \dots, k(\mathbf{x}', \mathbf{x}_n)]$. We can substitute the test point $\mathbf{x}_*$ to these expressions and use the property of the sum of Gaussian random variables to obtain the expressions (2.28) and (2.29).

2.2.2 *Bayesian model comparison and hyperparameter optimization*

Let's now consider Bayes Theorem (2.24) from a different point of view. We can use this theorem to perform model comparison and selection. Given a set of possible models $\mathcal{M}_l | l = 1, \dots, L$ and a dataset $\mathcal{D}$, our goal is to define which model describes the data in the best way. The set of the models is discrete so that we can write a posterior probability of each model given the data $\mathcal{D}$:

$$p(\mathcal{M}_l|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{M}_l)p(\mathcal{M}_l)}{\sum_{l=1}^{L} p(\mathcal{D}|\mathcal{M}_l)p(\mathcal{M}_l)}. \tag{2.36}$$

If it is necessary to choose only one model, we can find a maximum a posteriori model $\mathcal{M}_{\text{MAP}}$. And given the uniform prior over the models, the best model $\mathcal{M}_{\text{MAP}}$ is given by

$$\mathcal{M}_{\text{MAP}} = \underset{\mathcal{M}}{\text{argmax}}\, p(\mathcal{D}|\mathcal{M})p(\mathcal{M}) = \underset{\mathcal{M}}{\text{argmax}}\, p(\mathcal{D}|\mathcal{M}). \tag{2.37}$$

In this context, let's consider the models discussed in Section 2.2.1. The performance of the Bayesian method depends on the choice of the noise parameter σ_{noise} and the prior parameter σ_{prior}. We have shown their connection to the regularization parameter λ that controls regularization in frequentist linear methods in (2.25). Thus, we can conclude that an incorrect choice of these parameters will lead to the same problems typical for the incorrect choice of regularization. If σ_{prior} is too big, it will lead to model overfitting. If this parameter is too small, we will underfit the data. The opposite is true for σ_{noise}. Thus, we can claim that different values of these hyperparameters define different models, and we can use Bayesian model selection to optimize these parameters. Thus, we can write $p(\mathcal{D}|\sigma_{\text{noise}}, \sigma_{\text{prior}}) = p(\mathcal{D}|\mathcal{M})$.

To find optimal values for these parameters, we can consider the evidence of the model, i.e., the denominator of (2.24).

$$p(\mathcal{D}|\sigma_{\text{noise}}, \sigma_{\text{prior}}) = \int p(\mathcal{D}|\mathbf{w}, \sigma_{\text{noise}})p(\mathbf{w}|\sigma_{\text{prior}})d\mathbf{w}. \tag{2.38}$$

We can solve the optimization problem (2.37) with respect to the parameters σ_{noise} and σ_{prior}. For linear models with a Gaussian prior and likelihood, we can obtain an analytical expression for (2.38) using the property for the conditional distribution in a linear-Gaussian model. According to this property, the distribution $p(\mathcal{D}|\sigma_{\text{noise}}, \sigma_{\text{prior}}) = p(\mathbf{y}|\sigma_{\text{noise}}, \sigma_{\text{prior}})$ will also be Gaussian. For simplicity, we consider the prior $p(\mathbf{w})$ to be zero-centered.

$$p(\mathbf{y}|\sigma_{\text{noise}}, \sigma_{\text{prior}}) = \mathcal{N}(\mathbf{y}|0, \sigma^2_{\text{noise}}\mathbf{I} + \sigma^2_{\text{prior}}\mathbf{\Phi}\mathbf{\Phi}^\top). \tag{2.39}$$

Instead of maximizing the expression for $p(\mathbf{y}|\sigma_{\text{noise}}, \sigma_{\text{prior}})$ directly, we can maximize its logarithm. After some simple transformations, we can show that:

$$\begin{aligned}\log p(\mathbf{y}|\sigma_{\text{noise}}, \sigma_{\text{prior}}) = &-\frac{n}{2}\log\sigma^2_{\text{noise}} - \frac{d}{2}\log\sigma^2_{\text{prior}} - \\ &\frac{1}{2\sigma^2_{\text{noise}}}\|\mathbf{y} - \mathbf{\Phi}\boldsymbol{\mu}_{\mathbf{w}}\|^2 - \frac{1}{2\sigma^2_{\text{prior}}}\boldsymbol{\mu}^\top_{\mathbf{w}}\boldsymbol{\mu}_{\mathbf{w}} - \frac{1}{2}\log|\mathbf{\Sigma}_{\mathbf{w}}| - \frac{n}{2}\log(2\pi),\end{aligned} \tag{2.40}$$

where $\boldsymbol{\mu}_{\mathbf{w}}$ and $\boldsymbol{\Sigma}_{\mathbf{w}}$ are defined in (2.26). The expression for $\log p(\mathbf{y}|\sigma_{\text{noise}}, \sigma_{\text{prior}})$ can be maximized with respect to σ^2_{prior} and σ^2_{noise} using a gradient-based optimizer to obtain the optimal values for these parameters. The main difficulty here is the computation of $\boldsymbol{\mu}_{\mathbf{w}}$ due to the matrix inversion in (2.26), which has $\mathcal{O}(d^3)$ complexity. Given that we have to perform this computation for each step of the gradient-based optimizer, the overall complexity of obtaining the optimal values for $\sigma^2_{\text{prior}}, \sigma^2_{\text{noise}}$ is quite high.

We can use the same approach for optimizing the hyperparameters of GP regression. We can use the expression (2.39) directly by using the representation of the Gram matrix through basis functions $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$. The logarithm of the model evidence in this case becomes:

$$\begin{aligned}\log p(\mathbf{y}) = &-\frac{1}{2}\log|(\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I})| - \frac{1}{2}\mathbf{y}^\top(\mathbf{K} + \sigma_{\text{noise}}\mathbf{I})^{-1}\mathbf{y} \\ &-\frac{n}{2}\log(2\pi).\end{aligned} \tag{2.41}$$

The computational complexity for this expression is $\mathcal{O}(n^3)$. We can optimize $\log p(\mathbf{y})$ with respect to the noise parameter σ^2_{noise} and the parameters of the kernel $k(\cdot,\cdot)$, such as the lengthscale or output variance in (2.16).

2.2.3 *Bayesian Linear Models for Classification*

To address classification problems in a Bayesian way, we have to use another type of likelihood $p(\mathcal{D}|\mathbf{w})$ in the expression (2.24). The standard choice is categorical distribution:

$$p(\mathbf{y}|\mathbf{w}) = \prod_i^n \mathbb{1}[y_i = j]a_{ij}, \tag{2.42}$$

where $a_{ij} = \frac{\exp(\mathbf{w}_j^\top \mathbf{x}_i)}{\sum_{k=1}^c \exp(\mathbf{w}_k^\top \mathbf{x}_i)}$ for c-class problem, and $\mathbf{w}_j$ is a subset of $\mathbf{w}$, that corresponds to the output j. With this type of likelihood and a Gaussian prior $p(\mathbf{w})$, we can no longer leverage the property of conjugate distributions.

The easiest way to bypass this difficulty is to use the Laplace approximation. We can approximate the true posterior $p(\mathbf{w}|\mathcal{D})$ with an approximate posterior distribution $q(\mathbf{w}|\mathcal{D})$. As the approximate posterior $q(wvect|\mathcal{D})$, we can choose a Gaussian distribution.

$$q(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\mathbf{w}_{\text{MAP}}, \boldsymbol{\Sigma}_{\text{Laplace}}), \tag{2.43}$$

where $\boldsymbol{\Sigma}_{\text{Laplace}} = -(\nabla\nabla \log p(\mathbf{w}|\mathcal{D}))^{-1}$ and $\mathbf{w}_{\text{MAP}}$ is the mode of the posterior distribution $p(\mathbf{w}|\mathcal{D})$.

The idea of this approach is simple; we match the mode of the approximate distribution with the mode of the true posterior distribution. The Hessian of the log-posterior arises from the second-order Taylor approximation of the unnormalized log-posterior distribution at $\mathbf{w}_{\text{MAP}}$ [7].

There are plenty of other approaches to approximate Bayesian linear models, like Expectation Propagation [99], Evidence Lower Bound maximization, and Markov Chain Monte Carlo sampling, some of which we will consider later in this chapter.

In Chapter 4, we will consider another approach, based on a Dirichlet approximation, that allows replacing the classification problem with the regression one. The Laplace approximation is used in some experiments carried out in this work.

2.2.4 *Bayesian Neural Networks*

We can extend the Bayesian approach for linear models to non-linear ones, like neural networks. Let's consider a Multilayer Perceptron (MLP). We consider f to be defined as an MLP with L layers, each of which is given by the following formula:

$$f_l(\mathbf{x}) = \frac{1}{\sqrt{D_{l-1}}}\Big(\mathbf{W}_l\, \varphi(f_{l-1}(\mathbf{x}))\Big) + b_l, \qquad l \in \{1, ..., L+1\}, \tag{2.44}$$

where $\mathbf{W}_l$ and b_l are the weights and the biases of the l-th layer, φ is some non-linear function (i.e. ReLU), and D_l denotes the dimension of the input for the corresponding layer. $\mathbf{W}_l$ and b_l are the stochastic variables. By dividing each layer by $\sqrt{D_l}$, we ensure that the variance of the output does not explode in the limit where the width of the network tends to infinity; this specification is known as Neural Tangent Kernel parameterization [59]. We shall use $\mathbf{w}$ to refer to the set of parameters: $\mathbf{w} := \mathrm{vec}(\{\mathbf{W}_l, b_l\}_{l=1}^{L}) \in \mathbb{R}^m$ to be the corresponding vectorized form. The most popular choice of prior distribution $p(\mathbf{w})$ for BNNs is the fully factorized Gaussian. We will consider the subject of prior for BNNs more thoroughly in Chapter 6.

In this work, we will use mostly Bayesian MLPs, but we can note that the extension of the Bayesian framework to other architectures is straightforward.

2.2.5 *Inference techniques for Bayesian Neural Networks*

Performing inference for a BNN poses the same challenge as Bayesian Logistic Regression. We cannot obtain the posterior distribution over the weights $p(\mathbf{w}|\mathcal{D})$ in closed form because the prior and the likelihood are not conjugate. The two most popular directions to resolve this issue are Variational Inference and Markov Chain Monte Carlo sampling.

2.2.5.1 *Variational inference*

The main idea of the Variational Inference approach is to choose a family of distributions and find such an element $q(\mathbf{w})$ in this family that is closest to the true posterior distribution $p(\mathbf{w}|\mathcal{D})$ according to some metric. The most popular choice for this metric is the Kullback–Leibler divergence.

$$\mathbb{KL}(p(x)||q(x)) = \int p(x) \log \frac{p(x)}{q(x)} dx. \tag{2.45}$$

The classical approach to solving $\min_q \mathbb{KL}(q||p)$ is to maximize the Evidence Lower Bound (ELBO).

$$\begin{aligned} &\log p(\mathbf{w}|\mathcal{D}) \geq L(p(\mathbf{w})|\mathcal{D})), \\ &\text{where } L(p) = \mathbb{E}_{\mathbf{w}\sim q(\mathbf{w})} \log \frac{p(\mathbf{w}|\mathcal{D})}{q(\mathbf{w})}. \end{aligned} \tag{2.46}$$

It is possible to formulate an ELBO maximization objective as follows [69]:

$$\min_{q} -\mathbb{E}_{\mathbf{w}\sim q(\mathbf{w})}[\log p(\mathcal{D}|\mathbf{w}] + \mathbb{KL}(q(\mathbf{w})||p(\mathbf{w})). \tag{2.47}$$

The classical approach proposes to choose a parametric family of approximate distributions and solve this optimization problem with respect to the parameters of the approximate distribution $\boldsymbol{\theta}$.

The problem of this approach is how to choose a rich family for the approximate distribution that can approximate the true posterior well. From the other point of view, the second term in the RHS of (2.47) is analytically available for a small set of priors and approximate posteriors, which makes this approach quite restrictive.

Recently so-called Particle Optimization Variational Inference (POVI) [79] was proposed as an alternative approach which does not rely on parametric approximations but instead chooses an empirical distribution of particles, where each particle represents a particular realization of the random variable of interest. Good examples of this approach were investigated in [79], [29] and [88]. We will discuss in detail this particular type of VI in Chapters 4 and 6.

2.2.5.2 *Markov Chain Monte Carlo*

MCMC represents a completely different approach to characterize the posterior over model parameters in the Bayesian treatment of statistical models. This approach aims to find a sequence of samples generated by a Markov chain. A Markov chain is a sequence of random variables where the distribution of each variable depends only on the previous element of the sequence. In the case of Bayesian inference, the Markov chain is constructed in such a way as to generate samples from the posterior distribution.

To explain the details of this approach, we consider a Metropolis-Hastings algorithm, which is a very basic MCMC algorithm with slow convergence, and we consider it only for illustration purposes. Our goal is to generate samples from the posterior distribution $p(\mathbf{w}|\mathcal{D})$. For this algorithm, it is necessary to introduce an auxiliary distribution $q(\mathbf{w}'|\mathbf{w})$ that depends on the previous sample in the sequence $\mathbf{w}$, and it is referred to as the is called a proposal distribution. Starting from an initial point $\mathbf{w}_0$, we generate a new candidate $\mathbf{w}' \sim q(\mathbf{w}'|\mathbf{w})$ at each step. We accept $\mathbf{w}'$ and add it to the sequence with probability a

$$a = \min\left(1, \frac{p(\mathbf{w}'|\mathcal{D})q(\mathbf{w}'|\mathbf{w})}{p(\mathbf{w}|\mathcal{D})q(\mathbf{w}|\mathbf{w}')}\right). \tag{2.48}$$

If the candidate is not accepted, we add to the sequence old sample $\mathbf{w}$ to the sequence. It is easy to notice that for this algorithm, we do not need to compute the normalization constant in the denominator of (2.24) because it cancels out during the division operation in (2.48). Thus, the computation of the acceptance probability boils down to

$$a = \min\left(1, \frac{\hat{p}(\mathbf{w}'|\mathcal{D})q(\mathbf{w}'|\mathbf{w})}{\hat{p}(\mathbf{w}|\mathcal{D})q(\mathbf{w}|\mathbf{w}')}\right), \tag{2.49}$$

where $\hat{p}(\mathbf{w}|\mathcal{D})$ is unnormalized posterior probability density

$$\hat{p}(\mathbf{w}|\mathcal{D}) = p(\mathcal{D}|\mathbf{w})p(\mathbf{w}). \tag{2.50}$$

As mentioned before, the Metropolis-Hastings algorithm has poor convergence properties, and it requires computing the likelihood function $p(\mathcal{D}|\mathbf{w})$ over the whole dataset, which could be expensive in the case of large datasets. In this work, we widely use Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) because it has better convergence rates [21] and it allows for mini-batching, i.e., Monte Carlo estimates of the log-likelihood using mini-batches of data.

The SGHMC algorithm is based on Hamiltonian Monte Carlo (HMC) [94]. HMC considers the sampling procedure as a mechanical system with total energy, defined by the Hamiltonian, and position variables $\mathbf{w}$ and momentum variables $\mathbf{r}$:

$$H(\mathbf{w}, \mathbf{r}) = U(\mathbf{w}) + K(\mathbf{r}), \tag{2.51}$$

In this expression, the potential energy is given by

$$U(\mathbf{w}) = -\log \hat{p}(\mathbf{w}|\mathcal{D}), \tag{2.52}$$

and the kinetic energy is

$$K(\mathbf{r}) = \frac{1}{2}\mathbf{r}^\top \mathbf{M}^{-1}\mathbf{r}. \tag{2.53}$$

In most cases, the mass matrix $\mathbf{M}$ is set to the identity matrix $\mathbf{I}$. To generate a new sample for $\mathbf{w}$, it is necessary to sample $\mathbf{r} \sim \mathcal{N}(0, \mathbf{M})$, and then to simulate Hamiltonian dynamics of this system for some time t.

$$\begin{cases} d\mathbf{w} = \mathbf{M}^{-1}\mathbf{r}dt \\ d\mathbf{r} = -\nabla U(\mathbf{w})dt \end{cases} \tag{2.54}$$

To simulate this dynamics, it is necessary to use discrete integrators, which introduce discretization errors. That is why it is necessary to use Metropolis-Hastings correction at the end of the Hamiltonian simulation, i.e., accept the resulting $\mathbf{w}'$ with probability a given by (2.49).

The HMC algorithm has much better convergence properties but it still requires computing the exact likelihood $p(\mathcal{D}|\mathbf{w})$, giving it a limited scalability. The SGHMC algorithm allows resolving this issue using the augmented Hamiltonian dynamics:

$$\begin{cases} d\mathbf{w} = \mathbf{M}^{-1}\mathbf{r}dt \\ d\mathbf{r} = -\nabla U(\mathbf{w})dt - \mathbf{B}\mathbf{M}^{-1}\mathbf{r}dt + \mathcal{N}(0, 2\mathbf{B}dt), \end{cases} \tag{2.55}$$

where $\mathbf{B} = \frac{1}{2}\epsilon\mathbf{V}$, with the discretization step ϵ of the numerical integrator and the covariance $\mathbf{V}$, that comes from the mini-batched estimation of the gradient. The intuition behind this dynamics is the following. Utilization of the gradient estimates via mini-batches introduces some noise that, due to the central limit theorem, is assumed to have a Gaussian distribution. Because the dynamics have to preserve entropy, it is necessary to introduce a friction term $-\mathbf{B}\mathbf{M}^{-1}\mathbf{r}$ to the momentum dynamics.

A discretized version of this dynamics, implemented in practice, is the following:

$$\begin{aligned} \mathbf{w}_i &= \mathbf{w}_{i-1} - \epsilon_t \mathbf{M}^{-1}\mathbf{r}_{i-1} \\ \mathbf{r}_i &= \mathbf{r}_{i-1} - \epsilon_t \nabla \tilde{U}(\mathbf{w}_i) - \mathbf{C}\mathbf{M}^{-1}\mathbf{r}_{i-1} + \mathcal{N}(0, 2(\mathbf{C} - \hat{\mathbf{B}})\epsilon_t), \end{aligned} \tag{2.56}$$

where $\mathbf{C}$ is a user-defined friction parameter, $\tilde{U}(\mathbf{w}_i)$ is a mini-batched estimate of the $U((\mathbf{w}_i))$. The empirical noise covariance $\hat{\mathbf{B}}$ can be estimated using a method proposed in [127].

3

INPUT DATA BINARIZATION FOR OPTICAL PROCESSING UNITS

3.1 INTRODUCTION

Statistical models based on kernel methods offer powerful and theoretically well-understood tools for complex data modeling problems. The limitation of employing these kernel-based models in practice is that a naive implementation scales poorly with the size of the data set, and there has been a tremendous amount of work in the direction of mitigating this issue by introducing approximations.

In this context, Nyström approximations [150] and random features [111] are very popular techniques to scale kernel methods virtually to any number of data, thanks to mini-batch formulations [27], [57].

The focus of this chapter is on random feature approximations, whereby kernel-based models are "linearized" by an equivalent linear model with a set of suitably constructed random basis functions. The motivation behind this work is to considerably accelerate the construction of random features while reducing power consumption by resorting to dedicated hardware, which we refer to as Optical Processing Unit (OPU).

OPUs are computing devices that perform random projections of input vectors by exploiting the physical phenomenon of scattering a light source through a diffusive medium [118]. The random projection is then followed by a nonlinear operation, making the whole pipeline of computation exactly what is needed to construct random features to approximate kernel-based models. Crucially, OPUs offer the possibility to operate with a number of random features at the speed of light and with low power consumption, representing a unique solution to improve the scalability of kernel machines further. As an example, OPU-based random feature approximations have successfully been proposed to carry out approximate kernel ridge regression in [97], [17].

One limitation associated with working with OPUs is that, because of the hardware setup, input vectors need to be binarized. In addition, the random projection matrix characterizing the device is unknown and can only be retrieved through an expensive calibration procedure. In this paper, we propose a novel binarization strategy for OPUs, which is learned along with the regression/classification task in an end-to-end manner, meaning that the parameters of the binarization part are learned along with the kernel-based model parameters. In order to achieve this, we overcome the limitation that OPU projection matrices are unknown by employing the so-called REINFORCE gradient estimator, which allows us to treat the OPU as a black-box. Through experiments on several UCI classification/regression problems, we show that our proposal outperforms alternative unsupervised and supervised binarization techniques. We provide some analysis of the bounds on the objective functions of the proposed approaches, and also we consider several classes of kernels and image-based classification problems.

3.2 OVERVIEW OF THE SUBJECT FIELD

3.2.1 *Binary Neural Networks*

In neural networks, binarization is generally targeting intermediate layer activations, and it may also stem from the binarization of model parameters. In these cases, binarization is mostly introduced to reduce computational cost and memory consumption [107]. Neural networks with binary hidden layers find applications in binary autoencoders for hashing [20], data compression [132], and hard attention mechanism [154]. The binarization of layer activations is obtained by a suitable choice of activation functions; for instance, the sign or Heaviside functions for the deterministic case, or the sigmoid or *tanh* functions combined with the Bernoulli distribution for the stochastic case [26], [104]. The most popular technique to propagate gradients through such activation functions is the so-called straight-through estimator (STE) [6]. This technique assumes that the gradient of non-differentiable activation functions is equal to 1. Of course, this is quite a crude assumption and it leads to biased estimations of gradients of estimated variables, but it works surprisingly well in practice. More recently, there have been proposals to replace the STE with another estimator through a relaxation technique, also known as the Gumbel Softmax-trick [60].

Also, different kinds of target propagation are used to learn suitable targets for each binary layer and then train the associated parameters with relaxation techniques, or combinatorial optimization [44], [76], [19].

3.2.2 *Auxiliary models for binarization*

Focusing on OPUs, currently, the standard approach to binarize data makes use of a binary autoencoder [78], [132]. Such a binary autoencoder is trained independently from the OPU device, and it gives the possibility to perform the binarization operation by means of its encoder part. The autoencoder consists of a fully-connected encoder and decoder. The hidden layer has a Heaviside activation function, so its output is binary. The training procedure updates the weights of the decoder with backpropagation, and the weights of the encoder are forced to be equal to the weights of the decoder in order to be able to reproduce the input.

This task could be performed with a more advanced auxiliary model for binarization. The current state-of-the-art for autoencoders with latent binary space was proposed in [36]. While this paper is mostly focused on sampling in the latent Bernoulli space and generating new data, the proposed model itself provides very competitive image reconstruction performance even compared with non-binary autoencoders. At its core, it still uses the STE approach for training, but it introduces a novel method of sampling latent representations at the test phase. Thus, it could be beneficial to use this model to input binarization for OPU-based models.

3.2.3 *Optimization of heterogeneous models*

In this work, we aim to develop a supervised binarization model which is learned together with the supervised learning task. That is, we aim to provide a training

procedure for the heterogeneous model, consisting of the kernel ridge regression model approximated with random features and the binarization encoder before the OPU.

In the context of heterogeneous models, a general-purpose framework called Method of Auxiliary Coordinates (MAC) was proposed in [19] with examples of application in [20] and [25]. The authors propose to introduce auxiliary variables into a deep neural network. These auxiliary variables are assigned the role of pre-activations for each layer, and they get replaced during the forward pass. The first step of the optimization targets the auxiliary variables, and, after this step, the parameters of each layer are optimized to regress on these variables, which take the role of layer-specific labels. This is very beneficial when some layers are discrete and vanilla backpropagation is not applicable. In [25], this approach is used to train a fully connected network with binary activation functions, using an STE to propagate a learning signal through the non-differentiable parts. Reference [20] is especially interesting because the authors illustrate how discrete binary layers can be optimized within a larger, non-binary model.

While splitting the optimization of the binarization and the model is a viable option, we still need a way to train each part individually. There is a wide variety of ways to obtain a solution for kernel ridge regression with the random feature approximation, so the most difficult point is how to optimize the part consisting of the binary encoder and the OPU, because it combines a non-differentiable binarization function with an implicit random projection, which is a black-box function, so it is also non-differentiable at each point. These make the STE from [25] inapplicable. Also, we found that the combinatorial approach used in [20] and [44] is inapplicable to our case for two reasons. First, it is suitable only when the binary dimension is relatively small, which might be a limitation for a general solution. Second, the combinatorial approach combined with MAC converges in one iteration to poor local optima, and this happens because of the model setup, which is different from the ones in [20] and [44].

3.2.4 *Reinforcement learning approach*

From a different point of view, it is possible to view our problem through the lenses of reinforcement learning, where it is necessary to propagate binary codes through the OPU instead of discrete actions through the black-box environment. Instead of maximizing the reward from the environment, we are trying to minimize the loss function. The classical algorithm to solve this problem is REINFORCE [150]. This allows one to calculate the gradients of the reward with respect to the parameters of the policy that generates actions. The applicability of this method to other settings with black-box elements was shown in [112]. There are various versions of this algorithm intended to reduce the variance of the gradient of the parameters. Very frequently, they are based on relaxations of the non-differentiable sampling procedure [136], or approximation of the black-box part of the model [49]. It is also worth noting that there exist competitive alternatives to REINFORCE, such as the Augment-REINFORCE-Merge (ARM) [155], later extended with variance reduction [70] or relaxation [33]. However, the experimental results, reported in the publications dedicated to different modifications of ARM, show that the

performance gain is marginal compared to a REINFORCE method with a proper variance reduction technique. And given the fact that the REINFORCE method has proven itself as a very reliable technique applied to a large variety of tasks and datasets, we prefer to adopt this in the optimization of OPU-based models.

3.2.5 *Variance reduction for REINFORCE*

The most popular approaches that are used for variance reduction of REINFORCE-like estimators are conditioning [101] and control variates [102].

The conditioning technique is based on the Rao-Blackwell theorem. The technique is intended to reduce the variance of the estimator of the expectation $\mathbb{E}[f(x,y)]$ of the function f over the joint distribution $x,y \sim p(x,y)$. If it is possible to obtain the conditional expectation $\mathbb{E}[f(x,y)|y]$, it is possible to define an estimator $\hat{f}$

That is why, in the literature, it is often referred to as Rao-Blackwellization

$$\hat{f}(y) = \mathbb{E}[f(x,y)|y]. \tag{3.1}$$

First, we can note that

$$\mathbb{E}[\hat{f}(y)] = \mathbb{E}[f(x,y)]. \tag{3.2}$$

It means that it is possible to replace the standard empirical estimator of the expectation

$$\mathbb{E}[f(x,y)] \approx \frac{1}{n}\sum_{i}^{n} f(x,y),\ x,y \sim p(x,y) \tag{3.3}$$

with the conditional estimator

$$\mathbb{E}[f(x,y)] \approx \frac{1}{n}\sum_{i}^{n} \hat{f}(y),\ y \sim p(y). \tag{3.4}$$

We can easily compute the variance of the conditional estimator as:

$$\mathrm{var}(\hat{f}(y)) = \mathrm{var}(f(x,y)) - \mathbb{E}[(f(x,y) - \hat{f}(x))^2]. \tag{3.5}$$

Thus, the main motivation to do this replacement is the fact that the variance of the one-sample conditional estimator is always smaller than the variance of the original one.

$$\mathrm{var}\left(\frac{1}{n}\sum_{i=1}^{n} \hat{f}(y)\right) \leq \mathrm{var}\left(\frac{1}{n}\sum_{i=1}^{n} f(x,y)\right),\ x,y \sim p(x,y). \tag{3.6}$$

The main problem of this approach is that the straightforward analytical expression for $\mathbb{E}[f(x,y)|y]$ sometimes is not available for the chosen model.

The control variates technique provides another way to obtain an estimator with reduced variance. Unlike the previous method, this technique reduces the variance of the estimator of $\mathbb{E}[f(x)],\ x \sim p(x)$. So, it is applicable to functions that depend on one variable, and we don't have to compute any conditional expectation. For this approach, it is necessary to find an auxiliary function g that is somehow

similar to the target function f. The nature of this similarity will be revealed later. Let's consider the following estimator:

$$\tilde{f}(x) = f(x) - a(g(x) - \mathbb{E}[g(x)]). \tag{3.7}$$

There are different types of estimators that use control variates. The estimator (3.7) is one of the most popular ones because it allows for the optimization of variance reduction with respect to the parameter a. The main requirement for this type of control variates is the availability of the analytical form of $\mathbb{E}[g(x)]$.

Again, we can note that $\mathbb{E}[\hat{f}(x)] = \mathbb{E}[f(x)]$. And the most important thing is that this estimator has a lower variance compared to the original one:

$$\text{var}(\tilde{f}(x)) = \text{var}(f(x)) + a^2\text{var}(g(x)) - 2a\text{cov}(f(x), g(x)). \tag{3.8}$$

From this expression, we can see that by the similarity measure between f and g is the implied covariance between these functions. Also, we can find the minimum of the expression (3.8) with respect to the parameter a and find the optimal value a^*:

$$a^* = \frac{\text{cov}(f(x), g(x))}{\text{var}(g(x))}, \; x \sim p(x). \tag{3.9}$$

In practice, it is too expensive to compute $\text{cov}(f(x), g(x))$ and $\text{var}(g(x))$ over the whole distribution $p(x)$, so we have to use the Monte Carlo approximation.

3.3 RANDOM FEATURES ON OPTICAL PROCESSING UNITS

In this section, we discuss OPUs in the context of random features. In the previous section, we discussed random features as a way to approximate models involving kernels; for OPUs, instead, the device produces random features (fast and with little power consumption) and the question that we aim to address here is how to use these to implement approximate kernel machines.

OPUs are computing devices that exploit the physical process of scattering of light to perform a random projection operation of a given vector. In particular, given a binary vector $\mathbf{x}_i \in \mathbb{R}^d$, OPUs perform multiplication by a random matrix $\mathbf{R}$ and apply the nonlinear activation function $\|\cdot\|^2$. In other words,

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{D}}\|\mathbf{R}\mathbf{x}\|^2 \tag{3.10}$$

The matrix $\mathbf{R} \in \mathbb{C}^{D\times d}$ is a complex Gaussian matrix with elements $\mathbf{R}_{ij} \sim \mathcal{CN}(0, 1)$. Previous works have established that in the limit of an infinite number of random features, the equivalent kernel is the following [97]:

$$k(\mathbf{x}, \mathbf{y}) \approx \phi(\mathbf{x})\phi(\mathbf{y}) \overset{D\to\infty}{=} \|\mathbf{x}\|^2\|\mathbf{y}\|^2 + (\mathbf{x}^\top\mathbf{y})^2 \tag{3.11}$$

Therefore, when using OPUs for kernel ridge regression, we are implicitly working with this polynomial kernel.

Recently, a new version of OPUs has been proposed and developed in [34], which allows one to perform linear random feature projections. To avoid confusion with the polynomial OPU random features, we will denote them $\psi(\mathbf{x})$.

$$\psi(\mathbf{x}) = C\mathbf{R}\mathbf{x} \tag{3.12}$$

where C is a fixed constant. This novel type of OPU opens the possibility to approximate a wide variety of kernels by choosing an appropriate activation function [111], [143]. For example, it is possible to apply trigonometric activation functions to the outputs of the OPU:

$$\psi'(\mathbf{x}) = \begin{bmatrix} \sin(\psi(\mathbf{x})) \\ \cos(\psi(\mathbf{x})) \end{bmatrix} \tag{3.13}$$

This type of random features is called Random Fourier Features (RFF). It was proven in [111] that this kind of random features allows approximating RBF kernels.

$$\begin{aligned} &\frac{1}{D}\sum_{i=1}^{D}\psi'(\mathbf{x})^\top\psi'(\mathbf{y}) = \\ &\frac{1}{D}\sum_{r=1}^{D}\left(\begin{bmatrix} \sin(\psi(\mathbf{x})) \\ \cos(\psi(\mathbf{x})) \end{bmatrix}^\top \begin{bmatrix} \sin(\psi(\mathbf{y})) \\ \cos(\psi(\mathbf{y})) \end{bmatrix}\right) = \\ &\mathbb{E}_{\omega}[\cos(\omega(\mathbf{x}-\mathbf{y}))] = k_{\text{RBF}}(\mathbf{x},\mathbf{y}) \end{aligned} \tag{3.14}$$

As mentioned before, an important aspect of OPUs is that their input should be binary; this project proposes a novel way to carry out a binarization of its input along with the kernel ridge regression task in an end-to-end fashion.

3.4 COMBINATORIAL OPTIMIZATION

In order to be able to implement kernel ridge regression on OPUs we need to binarize the inputs $\mathbf{x}_i$. We propose to do so by employing an encoder implemented as a neural network and parameterized by a set of weights $\mathbf{W}_{\text{enc}}$. A combination of the OPU-based regression with the binarization network is a heterogeneous model.

Let's look at the application of the MAC on the OPU-based regression model. In this case, we have to propagate a temporary learning objective for the binary encoder through the optical random projection layer. Let's denote the desired output of the binary encoder as $\mathbf{s}$ and auxiliary variables corresponding to the activations of the OPU layer $\phi(\cdot)$ as $\mathbf{c}$.

In Fig. 3.1, we plotted the beginning of the optimization. Here, we use a 2D plane only for illustration purposes. In general, the discussed vectors are not two-dimensional. Vectors in the space of auxiliary variables are depicted in blue, and binary vectors that correspond to the output of the OPU are in red. Originally, we have an initial binary output of the encoder $\mathbf{s}^{(0)}$. We initialize the auxiliary variable as $\mathbf{c}^{(0)} = \phi(\mathbf{s}^{(0)})$. Auxiliary vector $\mathbf{c}^{(0)}$ is depicted collinear to the vector $\mathbf{s}^{(0)}$ only for demonstration purposes to emphasize the deterministic dependence of $\mathbf{c}^{(0)}$ from $\mathbf{s}^{(0)}$. And further, we will follow the same scheme of showing deterministic dependence between vectors. In general, these vectors lay in different spaces with different dimensionality. At the first step, we have to solve

$$\mathbf{c}^{(0+\frac{1}{2})} = \underset{\mathbf{c}}{\operatorname{argmin}}[\|y - \mathbf{c}\mathbf{W}^{(0)}_{\text{regr}}\|^2 + \mu\|\mathbf{c} - \phi(\mathbf{s}^{(0)})\|^2]. \tag{3.15}$$

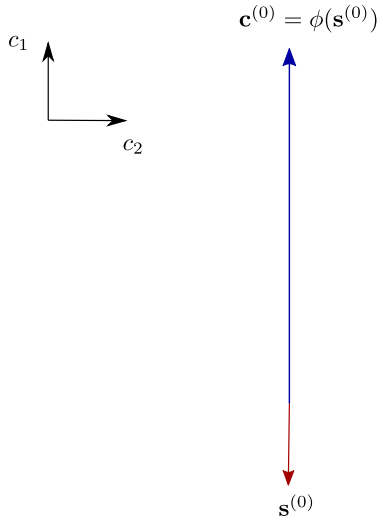

Figure 3.1: Initialization of the MAC procedure

Here $\mathbf{W}_{\text{regr}}^{(0)}$ are the initial weights of the regression model and μ is a regularization parameter. Let's note, that $\mathbf{c}^{(0+\frac{1}{2})}$ belongs to a span of vectors $\mathbf{c}^{(0)}$ (contribution of regularization) and $\mathbf{c}^{*(0)}$ (contribution of data), where

$$\mathbf{c}^{*(0)} = \underset{\mathbf{c}}{\text{argmin}}[\|y - \mathbf{c}\mathbf{W}_{\text{regr}}^{(0)}\|^2] \tag{3.16}$$

. Then, we update the weights of the binary encoder.

$$\mathbf{W}_{\text{regr}}^{(1)} = \underset{\mathbf{W}_{\text{regr}}}{\text{argmin}}[\|y - \mathbf{c}^{(0+\frac{1}{2})}\mathbf{W}_{\text{regr}}\|^2] \tag{3.17}$$

At the end of the iteration, we perform a combinatorial search for the optimal $\mathbf{s}^{(1)}$ among all possible candidates $\mathbf{s}_{\text{candidate}}^{(1)}$, which generates the closest to $\mathbf{c}^{(0+\frac{1}{2})}$ activations $\phi(\mathbf{s})$ (Fig. 3.2). It is possible to argue that an exhaustive combinatorial search is not a very practical solution, but for illustrative purposes, in this example, we assume that we have access to some discrete optimization procedure that gives us a solution for the following optimization objective:

$$\mathbf{s}^{(1)} = \underset{\mathbf{s}}{\text{argmin}} \|\mathbf{c}^{(0+\frac{1}{2})} - \phi(\mathbf{s})\|^2. \tag{3.18}$$

On the next iteration, the new vector $\mathbf{c}^{(1+\frac{1}{2})}$ will be a solution of (3.15), and it will lay within a span of the vector $\mathbf{c}^{(*1)} = \mathbf{c}^{(0+\frac{1}{2})}$ of the data contribution and the vector $\mathbf{c}^{(1)} = \phi(\mathbf{s}^{(1)})$ of the regularization contribution.

It means that the closest vector of activations generated by vectors in discrete space of binary encodings will be again $\mathbf{c}^{(1)}$. The same will happen for all next iterations (Fig. 3.3). This optimum is mostly defined by $\mathbf{W}^{(0)}, \mathbf{s}^{(0)}, \mu$

This argumentation shows that the MAC approach is not applicable for the models that involve OPUs because it almost immediately gets stuck in a poor local optimum. That is why we have to switch to another approach to train the binary encoder for input data.

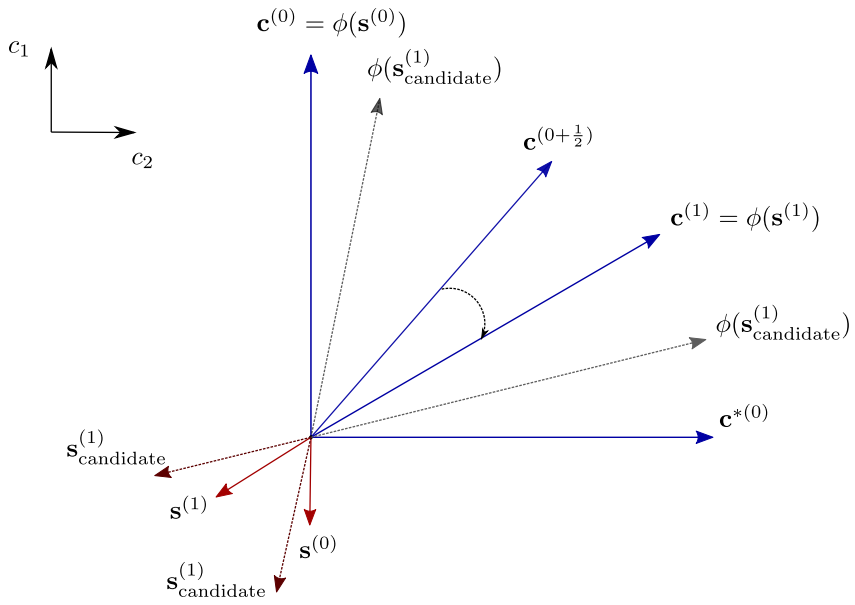

Figure 3.2: The state of the MAC algorithm after one iteration

3.5 REINFORCE FOR KERNEL REGRESSION WITH BINARIZED INPUTS

Another approach to implement kernel ridge regression on OPUs is using stochastic binarization of the inputs $\mathbf{x}_i$, and we propose to do so by employing an encoder, implemented as a neural network with a stochastic output, parameterized by a set of weights $\mathbf{W}_{\text{enc}}$. The encoder transforms the inputs to kernel-based models $\mathbf{x}_i$ and turns them into a set of Bernoulli-distributed binary random variables $\mathbf{z}_i$.

In particular, we denote by $f_k(\mathbf{x}, \mathbf{W}_{\text{enc}})$ the function implemented by the encoder which parameterizes the Bernoulli distribution associated with the kth element of the output, that is z_k.

Recalling the random feature formulation of linear regression of Chapter 2, we propose the following approach to construct an approximate kernel-based model with binary inputs:

$$y = \mathbb{E}_{\mathbf{z}}[\mathbf{w}_{\text{regr}}^{\top} \phi(\mathbf{z})] + \epsilon, \tag{3.19}$$

where $\mathbf{z} \sim \text{Bernoulli}(f(\mathbf{x}, \mathbf{W}_{\text{enc}}))$ and $\mathbf{w}_{\text{regr}}$ are the parameters of the linearized regression model. Note how in this formulation, the binary vectors $\mathbf{z}$ are treated stochastically due to the expectation under the Bernoulli distribution induced by the encoder. The reason for this is that it allows us to employ the so-called REINFORCE gradient estimator, as we discuss next.

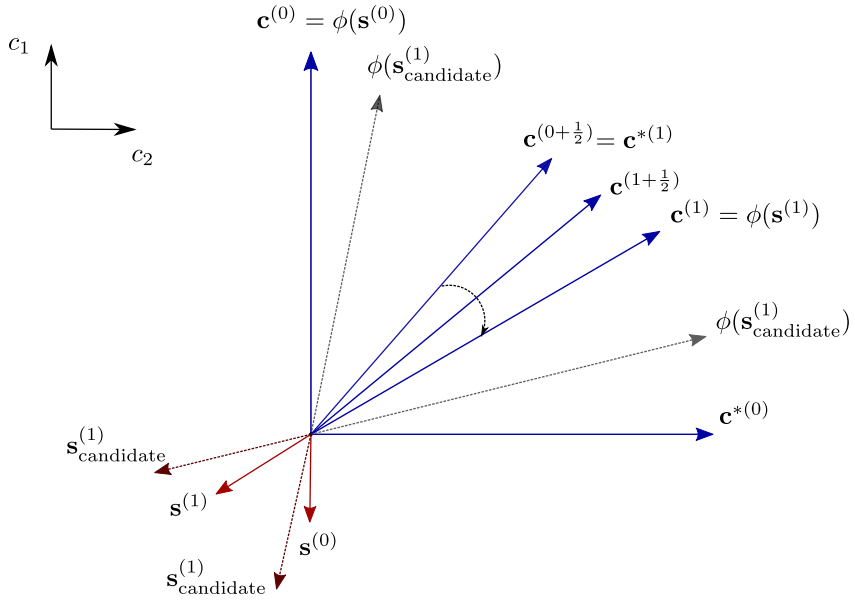

Figure 3.3

REINFORCE, also known as the log-derivative trick or score function estimator, offers a way to estimate the gradient of the expectation of a non-differentiable function $f(z)$ under the distribution of the input random vector variables z:

$$\nabla_\theta \mathbb{E}_{p(z;\theta)} f(z) = \nabla_\theta \int p(\mathbf{z};\theta) f(z) dz = \tag{3.20}$$

$$\int \nabla_\theta p(z;\theta) f(z) dz = \int p(z;\theta) \frac{\nabla_\theta p(z;\theta)}{p(z;\theta)} f(z) dz = \tag{3.21}$$

$$\mathbb{E}_{p(z;\theta)} \nabla_\theta \log p(z;\theta) f(z) \approx \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta \log p(z;\theta) f(z) \tag{3.22}$$

where M is number of samples drown from $p(z,\theta)$. Applying REINFORCE to our approximate kernel-based model yields the following optimization objective:

$$\begin{aligned} \min_{\mathbf{w}_{\text{regr}}, \mathbf{W}_{\text{enc}}} \mathbb{E}_{\mathbf{Z} \sim \text{Bernoulli}(f(\mathbf{X}, \mathbf{W}_{\text{enc}}))} [\mathcal{L}(\mathbf{y}, \phi(\mathbf{Z}) \mathbf{w}_{\text{regr}})] + \\ \lambda_{\text{enc}} \|\mathbf{W}_{\text{enc}}\|^2 + \lambda_{\text{regr}} \|\mathbf{w}_{\text{regr}}\|^2 \end{aligned} \tag{3.23}$$

In this expression, we denoted by $\mathcal{L}(\mathbf{y}, \tilde{\mathbf{y}})$ the loss function associated with the task at hand and by $\mathbf{Z}$ the matrix that contains binary encoded variables for the whole training set $\mathbf{X}$. We can optimize this objective by means of gradient-based techniques; for this, we require that we are able to compute the gradient of the objective with respect to all parameters. The gradient of the first term of the objective with respect to $\mathbf{W}_{\text{enc}}$, which is the most involved part, is:

$$\begin{aligned} &\nabla_{\mathbf{W}_{\text{enc}}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})} [\mathcal{L}(y, \mathbf{w}_{\text{regr}}^\top \phi(\mathbf{z}))] \approx \\ &\approx \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}(y, \mathbf{w}_{\text{regr}}^\top \phi(\mathbf{z}_i)) \nabla_{\mathbf{W}_{\text{enc}}} \log q(\mathbf{z}_i) \end{aligned} \tag{3.24}$$

while the derivatives of the other terms are straightforward to compute. With this derivation, we observe that it is then possible to jointly optimize all parameters, leading to what it is commonly referred to as an end-to-end approach. In the remainder of this paper, we refer to this method as End-to-End SE, where SE stands for Supervised Encoder.

3.6 VARIANCE REDUCTION

A straightforward application of the expressison (3.24) for gradient optimization will lead to a slow convergence due to the variance of the REINFORCE gradient estimator. In this section, we will discuss the application of variance reduction techniques for the proposed OPU pipeline.

For the proposed approach, there is no straightforward way of applying the Rao-Blackwellization technique because, in our case, the stochastic activations of the encoder are mutually independent. It means that the conditional expectation of each dimension of the activation vector $\mathbf{z}$ with respect to all other dimensions is equal to its marginal distribution, and the conditional estimator becomes useless. However, the Rao-Blackwellization technique can still be applied, but it will require changing the original pipeline or replacing the original Bernoulli stochastic activations with auxiliary variables [155]. It means that for this particular setting, we have to use control variates.

There are multiple ways to apply of the control variates technique.As a first attempt, we could apply the regression estimator (3.7), which is one of the most popular approaches. In this scenario, we have to find a control variate for $h(\mathbf{z}) = \nabla_{\mathbf{W}_{\text{enc}}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})} [\mathcal{L}(y, \mathbf{w}_{\text{regr}}^\top \phi(\mathbf{z}))]$. The obvious choice for the control variate is the score function:

$$g(\mathbf{z}) = \nabla_{\mathbf{W}_{\text{enc}}} \log q(\mathbf{z}) \tag{3.25}$$

This choice of control variate has an analytically available form for the expectation $\mathbb{E}[h(\mathbf{z})] = 0$. And it has a computational benefit because the score function $\nabla_{\mathbf{W}_{\text{enc}}} \log q(\mathbf{z})$ should be computed anyway for the gradient estimator itself (3.24). Thus, the modified gradient estimator has the following form:

$$\begin{aligned} &\nabla_{\mathbf{W}_{\text{enc}}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})} [\mathcal{L}(y, \mathbf{w}_{\text{regr}}^\top \phi(\mathbf{z}))] \approx \\ &\approx \frac{1}{M} \sum_{i=1}^{M} (\mathcal{L}(y, \mathbf{w}_{\text{regr}}^\top \phi(\mathbf{z}_i)) - a) \nabla_{\mathbf{W}_{\text{enc}}} \log q(\mathbf{z}_i). \end{aligned} \tag{3.26}$$

Then we have to compute an optimal value for the parameter a according to (3.9), which implies computation of the empirical variance $\widehat{\text{var}}(h(\mathbf{z}))$ and the empirical covariance $\widehat{\text{cov}}(h(\mathbf{z}), g(\mathbf{z}))$. The key element, which is necessary for this computation, is the evaluation of the $\nabla_{\mathbf{W}_{\text{enc}}} \log q(\mathbf{z}_i)$ with respect to each sample $\mathbf{z}_i \sim q(\mathbf{z})$. All these evaluations should be stored and then used for the computation of the variance and covariance. This observation shows that this gradient estimator is computationally inefficient for the proposed pipeline because we have to evaluate separately the gradients for each data point and for each binary representation of this data point sampled from the encoder.

In order to reduce the variance of this estimator, we employ control variates [70]. In this approach, we add a set of random variables to the estimator, such that these

variables have zero means, so they do not alter the expectation of the gradient. The aim is to construct such variables so as to reduce the overall variance of the estimator:

$$\begin{aligned}&\nabla_{\mathbf{W}_{\text{enc}}}\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}[\mathcal{L}(y, \mathbf{w}_{\text{regr}}^{\top}\phi(\mathbf{z}))] \approx \\ &\frac{1}{M}\sum_{i=1}^{M}\nabla_{\mathbf{W}_{\text{enc}}}\log q(\mathbf{z}_i)(\mathcal{L}(y, \mathbf{w}_{\text{regr}}^{\top}\phi(\mathbf{z}_i)) - \mathbf{v}_i) \\ &\text{where } \mathbf{v}_i = \frac{1}{M-1}\sum_{i\neq j}\mathcal{L}(y, \mathbf{w}_{\text{regr}}^{\top}\phi(\mathbf{z}_j))\end{aligned} \tag{3.27}$$

3.7 LOWERING THE COST OF REINFORCE

The estimation of the gradient of the End-to-End SE with respect to $\mathbf{W}_{\text{enc}}$ can be expensive when the number of random features is large. This is due to the fact that this requires multiple samples to be passed from the encoder through the random projection and the approximate kernel ridge regression model. In this section, we propose a strategy to reduce the complexity of REINFORCE applied to our model, whereby we average the set of basis functions under the resampling of the binary variables as follows:

$$\tilde{y} = \mathbf{w}_{\text{regr}}^{\top}\mathbb{E}[\phi(\mathbf{z})] + \epsilon \tag{3.28}$$

where $\mathbf{z} \sim \text{Bernoulli}(f(\mathbf{x}, \mathbf{W}_{\text{enc}}))$.

With this new modeling assumption, the training is based on a modified optimization problem as follows:

$$\begin{aligned}\min_{\mathbf{w}_{\text{regr}}, \mathbf{W}_{\text{enc}}} \mathcal{L}(\mathbf{y}, \mathbb{E}_{\mathbf{Z}\sim\text{Bernoulli}(f(\mathbf{X}, \mathbf{W}_{\text{enc}}))}[\phi(\mathbf{Z})]\mathbf{w}_{\text{regr}}) + \\ \lambda_{\text{enc}}\|\mathbf{W}_{\text{enc}}\|^2 + \lambda_{\text{regr}}\|\mathbf{w}_{\text{regr}}\|^2\end{aligned} \tag{3.29}$$

Again, we can perform gradient-based optimization. Focusing on the first term, which is the nontrivial one to differentiate in the objective, we obtain

$$\nabla_{\mathbf{W}_{\text{enc}}}\mathcal{L} = \frac{d\mathcal{L}}{d(\mathbb{E}[\phi(\mathbf{z}]))}\nabla_{\mathbf{W}_{\text{enc}}}\mathbb{E}[\phi(\mathbf{z})] + 2\lambda_{\text{enc}}\sum_{ij}\mathbf{W}_{\text{enc}_{ij}} \tag{3.30}$$

where $\nabla_{\mathbf{W}_{\text{enc}}}\mathbb{E}[\phi(\mathbf{z})]$ is calculated with the REINFORCE estimator. In the remainder of the paper, we will refer to this method as Isolated Supervised Encoder (SE).

Regarding the comparison of the End-to-End SE and Isolated SE, we can note the following relationship between these models in the case of regression problems. In the data term of the optimization objective (3.29) we can put an expectation over the whole matrix product of the random features map and the regression weights instead of an expectation over the random features only. Then we can move the expectation in such a way that it is taken over the whole term within the squared norm. We can do this because, within one gradient step iteration, neither $\mathbf{y}$ nor $\mathbf{W}_{\text{enc}}$ are considered as random variables. In this case, the data term looks as follows:

$$\begin{aligned}\|\mathbf{y} - \mathbb{E}[\phi(\mathbf{Z})]\mathbf{w}_{\text{regr}}\|^2 &= \|\mathbf{y} - \mathbb{E}[\phi(\mathbf{Z})\mathbf{w}_{\text{regr}}]\|^2 \\ &= \|\mathbb{E}[\mathbf{y} - \phi(\mathbf{Z})\mathbf{w}_{\text{regr}}]\|^2\end{aligned} \tag{3.31}$$

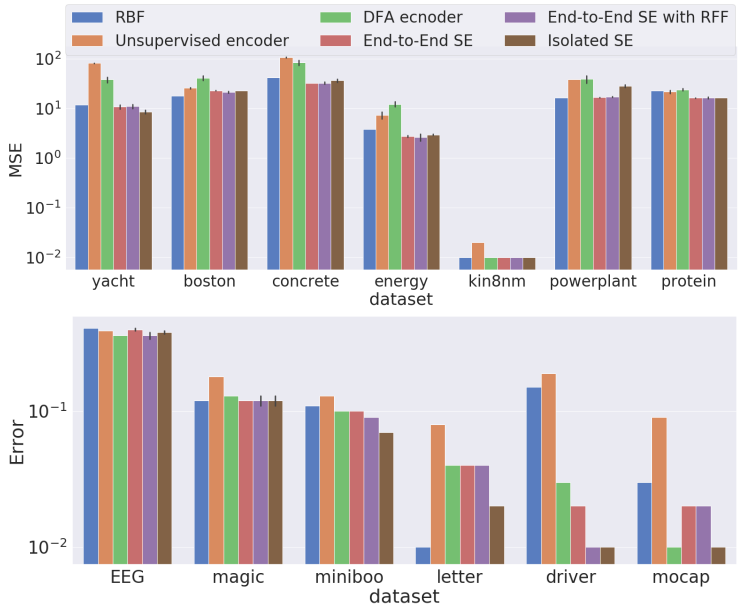

Figure 3.4: Mean Squared Error (MSE) for regression (top) and negative error on classification (bottom) datasets comparison.

In turn, End-to-End SE has the following data term as part of its optimization objective (3.23):

$$\mathbb{E}[\|\mathbf{y} - \phi(\mathbf{Z})\mathbf{w}_{\text{regr}}\|^2] \tag{3.32}$$

We can note that the squared loss is a convex function. Thus, we can apply Jensen's inequality to obtain the following expression:

$$\begin{aligned}\mathbb{E}[\|\mathbf{y} - \phi(\mathbf{Z})\mathbf{w}_{\text{regr}}\|^2] &\geq \|\mathbb{E}[\mathbf{y} - [\phi(\mathbf{Z})]\mathbf{w}_{\text{regr}}]\|^2 \\ &= \|\mathbf{y} - \mathbb{E}[\phi(\mathbf{Z})]\mathbf{w}_{\text{regr}}\|^2\end{aligned} \tag{3.33}$$

As a result, End-to-End SE optimizes the upper bound of the Isolated SE objective.

3.8 EXPERIMENTS

3.8.1 *Experiments on the UCI datasets*

We compared the performance of the proposed approaches for a non-linear OPU (End-to-End SE and Isolated SE) and a linear OPU that uses trigonometric activations (End-to-End SE with RFF) against a model based on unsupervised autoencoder proposed in [132], an encoder trained with Direct Feedback Alignment (DFA) [96] and a KRR based on a RBF.

Results are reported in Fig. 3.4 for several UCI regression and classification problems [34]. We want to emphasize that the main competitors of the proposed methods are the ones based on unsupervised autoencoder and encoder trained by DFA, because kernel ridge regression is unable to work with large datasets, and OPU-based regression just approximates this method and is intended to replace it on large datasets.

For the KRR experiments, we used MSE as a loss function. To apply KRR to the classification problems, we replaced 0 and 1 in class labels with -1, 1 and solved a classification problem as a regression one using MSE loss as an optimization objective. For all other models, we used MSE loss for the regression problems and Cross Entropy (CE) loss for the classification problems. We used a logistic activation function on the last layer for the classification tasks.

For Isolated SE and End-to-End SE as an encoding function $f(\mathbf{x}, \mathbf{W}_{\text{enc}})$ providing parameters for the Bernoulli distribution, we chose a single linear layer with sigmoid activation.

$$f(\mathbf{x}, \mathbf{W}_{\text{enc}}) = \sigma(\mathbf{W}_{\text{enc}}^{\top}\mathbf{x}) \tag{3.34}$$

All hyperparameters for the DFA encoder, End-to-End SE, and Isolated SE models (size of binary embedding, learning rate, L2 regularization for the encoder and the regression layer) were chosen with a random search during cross-validation. Kernel parameters of KRR were tuned by random search with cross-validation. This poses computational challenges for the large datasets (MiniBoo, MoCap), so we resort to RFF approximations for these cases.

For the models involving random features (both Fourier and OPU-generated ones), we have tuned the variance of the distribution that generates these random features. Concretely, assuming that the elements of the $\mathbf{R}$ matrix generating the random projections are distributed through the standard Normal distribution, we can obtain a new random matrix $\mathbf{R}'$ by multiplying $\mathbf{R}$ by any variance, for instance:

$$\phi'(\mathbf{x}) = c|\mathbf{R}'\mathbf{x}|^2 = c|\frac{\mathbf{R}}{\alpha}\mathbf{x}|^2 = c\frac{1}{\alpha^2}|\mathbf{R}\mathbf{x}|^2 \tag{3.35}$$

It is enough to multiply the output of the OPU by an additional parameter γ, such that $\gamma^2 = \frac{1}{\alpha^2}$, and α optimize them with standard gradient descent. The parameter γ is not equivalent to the lengthscale parameter of the RBF kernel. In practice, it has an effect on the outputscale parameter of the RBF kernel, as it has simply a scaling effect on the kernel.

On the regression problems, both proposed methods outperformed their main competitors. On the classification problems, the DFA-based approach was better only on one dataset, and on all other datasets, the proposed methods performed better or equally well. Regarding the type of kernel approximated by the OPU, the experiments show that the linear OPU with trigonometric activations performs as good as OPU kernel for most of the datasets. It gives performance gains only for some classification problems. Considering the comparison between the proposed methods, we see that End-to-End SE is more stable and requires a significantly fewer number of samples from the encoder, although Isolated SE showed slightly better results on classification problems.

We considered including results obtained by running these models on the real OPU (Fig. 3.5). Unfortunately, the regression problems required such a large number of epochs that we could not perform the experiments in a reasonable amount of time.

We also tested the performance of our approach with respect to the number of samples required to employ REINFORCE. We found that End-to-End SE can achieve good results with a small number of samples from the encoder, and the increase of amount of samples does not seem to improve performance.

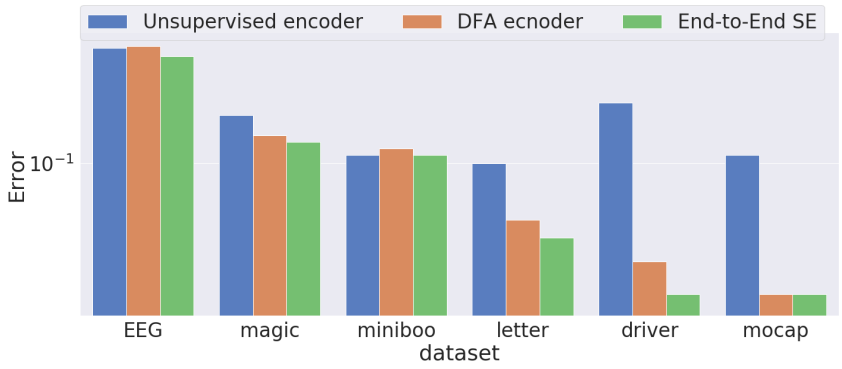

Figure 3.5: Error comparison on classification (bottom) datasets for experiments on the real hardware.

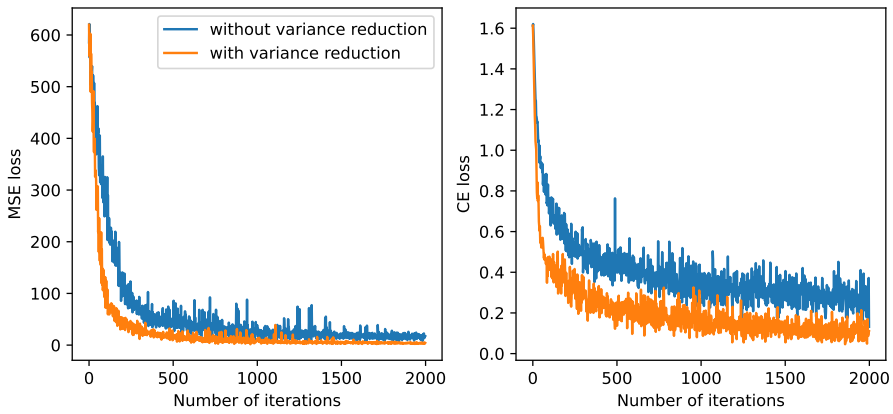

Figure 3.6: Convergence of the training procedure on regression problem: boston dataset (left) and classification problem: mocap dataset (right).

Finally, we evaluated the effect of variance reduction on convergence speed and performance for the End-to-End SE model. In Fig. 3.6, we report results for one classification and one regression problem. The convergence curves indicate that the convergence speed benefits from the gradient variance reduction.

3.8.2 *Experiments on image data*

In this section, we evaluate an optical random feature regression approach for image classification tasks with several different binarization techniques, including the proposed methods.

The kernel generated by OPU (3.11) is an example of a polynomial kernel. Polynomial kernels, unlike more popular RBF kernels, take into account interaction between different feature dimensions. This property is especially important for image data because a relative alignment of pixels is crucial for image classification. Of course, when we are working with OPUs, the kernel takes into account an alignment of different dimensions of the binary embedding of an image instead of the pixels. The relative alignment of different dimensions of the binary embedding most probably does not contain exactly the same information as the mutual

Table 3.1: LeNet-based binary encoder architecture

Layer	Dimensions
Conv2D	5×5, 6 filters
MaxPooling	2×2
Conv2D	5×5, 16 filters
MaxPooling	2×2
Linear	576×512
Linear	$512 \times d_{\text{binary}}$
Binarization layer	d_{binary}

Table 3.2: Image datasets used in the experiments

Dataset	Train/test size	C	D
MNIST	60000/10000	10	$1 \times 28 \times 28$
F-MNIST	60000/10000	10	$1 \times 28 \times 28$
CIFAR10	50000/10000	10	$3 \times 32 \times 32$

alignment of pixels. But until the binary encoder does not have disentanglement properties, the mutual interaction of dimensions of the binary embedding has to contain additional information about the image. That is why it is still important to use a kernel that is capable to take into account these relationships.

For the experiments on image data, we used two convolutional architectures of the binary encoder. The first architecture was inspired by the LeNet model (Table 3.1). We performed experiments on three classical image classification datasets (Table 3.2). We compared End-to-End SE with a model that used an autoencoder to train the encoder (AE) and a model that used direct feedback alignment (DFA) for the same purpose. The results are shown in Table 3.3.

Table 3.3: Classification error obtained by the model with the LeNet encoder

Dataset	AE	DFA	End-to-End SE
MNIST	0.06±0.02	0.31±0.03	0.01±0.00
F-MNIST	0.20±0.01	0.47±0.01	0.09±0.00
CIFAR10	0.55±0.01	0.81±0.02	0.32±0.01

We used the same encoder architecture with the same hyperparameters for each binarization method. The size of the binary embedding was set to 400, except of the unsupervised AE method.

The reason we trained the unsupervised AE differently for these experiments is because we were using complex convolutional models, and it was hard to adapt the method proposed in [132]. This binarization approach requires using exactly the same values of weights both for the encoder and for the decoder models. It

means that this method requires building a decoder model that is symmetrical to the encoder model. Achieving this property for convolutional neural networks is difficult because, for the decoder, it is difficult to pick equivalent symmetric operations for convolutional and pooling layers in the encoder. Transposed convolutions and interpolation operations that are used in decoders for image data are suitable for training the autoencoder by end-to-end backpropagation. They learn operations that are not symmetric to convolutions and pooling layers of the encoder. The method proposed in [132] assumes that only the decoder is trained, and the encoder copies weights from it, which is impossible to do for asymmetric operations in the decoder and encoder. That is why we trained the autoencoder model in a different way. The encoder of the AE model used an *tanh* activation function at the output. We used a parameter β that controls the steepness of the *tanh* function. We slowly increased the value of this parameter from $\beta = 1$ to $\beta = 100$ in the process of training. At the end of the training, the function *tanh* with a high value of the parameter β is almost equivalent to a shifted and scaled Heaviside function.

$$2h(x) - 1 \approx tanh(\beta x), \beta \to \infty \tag{3.36}$$

The results of the DFA approach signify that this type of gradient update is not suitable for convolutional models. This observation is supported by other researchers [5, 75]

Unsupervised AE was able to provide acceptable accuracy for the MNIST dataset, but on CIFAR-10, its performance dropped significantly. A possible explanation is that simple convolutional AE is unable to extract a reasonable binary representation of complex images. The AE model used in the experiments was able to reconstruct simple images from the MNIST dataset. But the reconstruction quality of the same model was much worse for the CIFAR-10 dataset. When the dimensionality of the binary embedding was equal to 400, the AE model was unable to generate any sensible images. Thus, we had to increase the size of the binary embedding to 1024. But the reconstructed images were very blurry even with this modification.

Because of the poor performance of the unsupervised AE baseline, we decided to add another baseline to the comparison. For this experiment, we used a RESNET-based convolutional network as the encoder. LBAE approach proposed in [36] implements an autoencoder with a latent binary space. The training procedure of this method is based on the straight-through gradient estimator. The architecture of the binary encoder is represented in Table 3.4 and Table 3.5. This encoder used leaky RELU as an activation function.

Table 3.6 contains the results of the comparison between the LBAE-based and End-to-End SE-based encoders in terms of classification error.

Both binarization approaches used the same RESNET-based architecture of the encoder network. We dropped the DFA approach from the comparison because of its poor performance.

The results of this experiment showed interesting properties of the unsupervised approach for training the binary encoder. The LBAE-based encoder with a deeper network performed worse than the simpler autoencoder with *tanh* annealing in terms of classification error. At the same time, the LBAE approach was better in the

Table 3.4: Architecture of the RESNET-based binary encoder

Layer	Dimensions
Conv2D	3×3, 64 filters
Conv2D	4×4, 64 filters
Residual Block	3×3, 64 filters
Conv2D	4×4, 64 filters
Residual Block	3×3, 64 filters
Conv2D	4×4, 128 filters
Linear	$4096 \times d_{\text{binary}}$
Binarization layer	d_{binary}

Table 3.5: Residual block structure. The number of filters is specified in Tab. 3.4

Layer	Dimensions
Conv2d	3×3
Conv2d	3×3

image reconstruction task. It seems that the binary latent projection of image data, which is suitable for image reconstruction, is unsuitable for image classification.

As with the UCI data, we evaluated the effect of variance reduction.

As we can see, variance reduction plays a crucial role in image classification. Without this technique, the proposed method is unable to train the model for CIFAR10.

3.9 DISCUSSION

Recent advances in alternatives to transistor-based hardware are bringing a new wave of sustainable computing solutions for machine learning [153]. This paper focuses on optical-based computing through OPUs [118], which performs random-

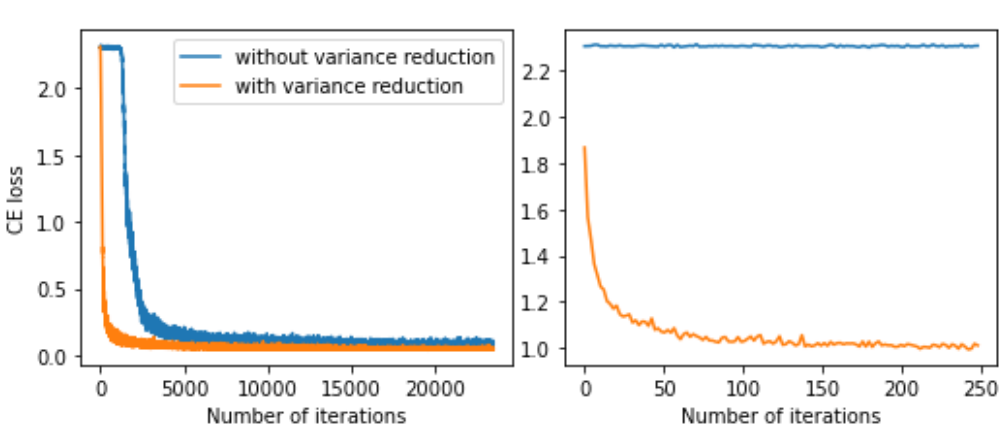

Figure 3.7: Training loss with and without variance reduction

Table 3.6: Classification error for models with the RESNET encoder

Dataset	LBAE	End-to-End SE
MNIST	0.15±0.01	0.01 ± 0.00
F-MNIST	0.24±0.02	0.06±0.01
CIFAR10	0.63±0.02	0.17±0.01

ized projections of binary input vectors at the speed of light with low energy consumption. In this paper, we considered these randomized computations to implement kernel-based models for regression and classification tasks through random feature approximations. In particular, we proposed a novel strategy to binarize the inputs of the given task to be able to employ OPUs inspired by reinforcement learning. The proposed strategy uses an encoder to map the inputs to a set of binary variables and employs the REINFORCE gradient estimator to estimate its parameters jointly with the parameters of the kernel-based model. We also explored ways to reduce the variance of the gradient estimator and accelerate convergence, which is critical in a number of challenging modeling tasks such as image classification. Through a series of experiments, we showed that our proposal outperforms competitors based on unsupervised binarization and those that do not employ gradient information. We are currently investigating our approach in the context of other kernel-based models, such as Gaussian Processes [114], and their extension to deep models, such as Deep Kernel Learning [152] and Deep Gaussian processes [27].

4

BAYESIAN APPROACH TO CLASSIFICATION

4.1 INTRODUCTION

The Bayesian treatment of statistical models is desirable in applications where quantification of uncertainty is a primary requirement. For many classes of models, this is analytically intractable, and one needs to resort to approximations. Given a statistical model with parameters on which a prior distribution is assumed, such approximations yield an approximation to the posterior distribution over these parameters, either in closed form or in the form of samples. Popular approaches include the Laplace Approximation, Variational Inference, and Markov chain Monte Carlo.

In this work, we focus on an alternative approach called *variational bootstrap* [88]. Variational bootstrap works by producing a set of replicas of the data set with perturbed labels. Then, each of these perturbed problems is solved by maximum-a-posteriori-type optimization. Perhaps not surprisingly, in the Bayesian linear regression case, it is possible to introduce perturbations in a way such that the set of solutions to the perturbed problems is distributed exactly as the posterior over the parameters. One remarkable property of this approach is that it transforms the problem of characterizing the posterior distribution over model parameters into a set of easily parallelizable optimization problems.

Milios et al. [88] provide an extension of this result to the case of regression with deep neural networks featuring ReLU activations, where certain theoretical guarantees are given for the minimization of the KL divergence between the approximate and the true posterior.

The main limitation of this approach is that no guarantees are provided for non-Gaussian likelihoods, such as Bernoulli or Multinomial, which are associated with classification problems.

In this work, we broaden the scope of variational bootstrap by extending it to classification problems. We propose to transform Bernoulli/Multinomial distributed labels to a latent representation with Gaussian noise. We can then apply a model with a Gaussian likelihood to solve a regression problem within this latent space. To carry out this transformation, we follow the method in [87], where classification labels are interpreted as the output of a Dirichlet distribution. We thus propose the combination of variational bootstrap with Dirichlet-based classification; this allows one to obtain reliable uncertainty estimates for the classification model at a lower cost than other Bayesian inference methods, and it enables easy parallelization.

We study the proposed extension to the nonlinear case on deep neural networks for classification with ReLU activations. The transformation of the labels from discrete to continuous allows us to borrow the results obtained in the Gaussian likelihood case. As a result, we obtain a method for which we have guarantees that the optimization of the perturbed problems yields an improvement of the

approximation to the posterior over model parameters. In the experiments, we showcase results on various data sets, demonstrating that this is a competitive approach to characterize the posterior over model parameters. Crucially, our proposal is extremely easy to parallelize, and we view this as a considerable advantage compared to alternatives to obtain the posterior over model parameters.

The proposed extension is also relevant for Bayesian logistic regression applied to large datasets with a large number of features. An interesting application of this approach that we showcase in the paper is the approximation of Gaussian process classifiers with random features [110], which turns the model into a Bayesian logistic regression model. The larger the set of random features, the better the approximation, but this has a negative impact on the computational cost. The proposed approach yields a very practical and effective method to overcome these difficulties.

4.2 RELATED WORK

In recent years, neural networks (NN) have gained popularity due to their effectiveness on a broad variety of tasks, including image classification, [55], natural language processing [31], and many others [13, 47, 95, 105]. One of the most persistent challenges is that NNs tend to make overconfident decisions [51, 66, 72]. In the literature, overconfident decisions are treated by introducing uncertainty to the outputs of a NN. In practice, it means that the model produces a predictive distribution of outputs for each input vector. This practice offers ways to quantify the uncertainty of predictions, as high predictive variance implies a lack of confidence. Methods that provide uncertainty quantification for NNs include deep ensembles [73], Monte Carlo dropout [46], and Bayesian Neural Networks (BNN) [82].

In this work, our focus is on BNNs. In contrast with the deterministic networks, BNNs consider parameters to be random variables that are associated with some prior distribution. Instead of a training procedure based on optimization, this treatment requires characterizing a posterior distribution over their weights given observations by means of Bayes theorem. The main challenge is that this procedure is intractable for nonlinear models, which has motivated the development of approximations using, for instance, Variational Inference (VI) [10, 45]. These techniques, while being relatively cheap in terms of computational resources, require selecting a family of distributions that is used for approximation of the posterior distribution. The choice of a family of approximate posteriors has a crucial effect on the performance of the model and on the reliability of uncertainty estimations. Because different models and tasks require different families of approximate posterior distributions, there are no reliable heuristics for this task. As an alternative to approximate inference, it is possible to use Markov Chain Monte Carlo (MCMC) techniques, which allows one to get samples from the true posterior distribution of the weights. The recently proposed Stochastic Gradient Hamiltonian Monte-Carlo (SGHMC) method [21] allows obtaining samples from a true posterior distribution over the weights for large-scale problems using mini-batching. Despite a significant reduction of the time complexity compared to other MCMC methods, SGHMC may still be slow to converge.

Another family of Bayesian inference techniques is particle optimization variational inference (POVI) [29, 79, 88]. Methods from this family use particles that are different instances of the model, which are optimized independently. The resulting set of optimized particles serves as an approximate posterior distribution that is more flexible than VI solutions. In this work we focus on variational bootstrap [88], which provides a theoretical connection between POVI and parametric bootstrap [35] for regression tasks, and our contribution is to extend this method to classification problems.

4.3 VARIATIONAL BOOTSTRAP FOR CLASSIFICATION MODELS

4.3.1 *Considered Models*

BAYESIAN NEURAL NETWORKS In this chapter, we will use Bayesian MLP described in Section 2.2.4, expression (2.44). The method that we propose is quite general and does not depend on the choice of the BNN architecture. It can easily be applied to convolutional and recurrent networks.

RANDOM FOURIER FEATURES APPROXIMATION OF GAUSSIAN PROCESSES We shall also examine the case of linear models, as they can serve as scalable approximations to another class of Bayesian models, namely Gaussian processes [114]. Following the random Fourier features (RFF) approximation [110], we consider $\phi(\mathbf{x}_i) \in \mathbb{R}^{D\times 1}$ to be the projection of an input point $\mathbf{x}_i \in \mathbb{R}^{d\times 1}$ onto a feature space of D trigonometric basis functions. Then $\mathbf{\Phi} \in \mathbb{R}^{D\times N}$ denotes the design matrix of the entire training set in the feature space. In this case, the model parameters can be directly described as a vector: $\theta := \mathbf{w} \in \mathbb{R}^m$. Given a Gaussian likelihood $\mathcal{N}(y; f(\mathbf{x}, \mathbf{w}), \sigma^2)$ and a Gaussian prior over the weights $\mathbf{w} \sim \mathcal{N}(0, \alpha^2 \mathbf{I_m})$, then the posterior distribution over $\mathbf{w}$ after observing the dataset $\mathcal{D}$ is known to be Gaussian, yielding the following predictive mean and variance for a test point $\mathbf{x}_*$:

$$\begin{aligned} &\mathrm{E}[f(\mathbf{x}_*)] = \frac{1}{\sigma^2}\phi(\mathbf{x}_*)^\top \mathbf{A}^{-1}\mathbf{\Phi}\mathbf{y}, \\ &\mathrm{Var}[f(\mathbf{x}_*)] = \phi(\mathbf{x}_*)^\top \mathbf{A}^{-1}\phi(\mathbf{x}_*), \\ &\text{where } \mathbf{A} = \frac{1}{\sigma^2}\mathbf{\Phi}\mathbf{\Phi}^\top + \frac{1}{\alpha^2}I. \end{aligned} \tag{4.1}$$

We observe that in order to make a prediction, one has to solve two $D \times D$ linear systems: $\mathbf{A}^{-1}\mathbf{\Phi}\mathbf{y}$ is solved only once, but $\mathbf{A}^{-1}\phi(\mathbf{x}_*)$ has to be solved for every new test point $\mathbf{x}_*$. So for n_* test points, this translates to $\mathcal{O}(n_* \times D^2)$ complexity. If the number of test points is large, then it is preferable to directly calculate the decomposition of the matrix $\mathbf{A}$, so that it can be reused to solve the linear systems needed to calculate the predictive distribution for new test points. In many problems however, a large number of features might be required to obtain an accurate approximation of a GP. Later, we show that variational bootstrap can keep the computational cost down when both D and n_* are large.

4.3.2 *Variational Bootstrap for Neural Network Regression*

In its original formulation, variational bootstrap was defined as an alternative to Bayesian inference for regression tasks. More specifically, the objective is to approximate the posterior distribution of a nonlinear model with a Gaussian likelihood and a Gaussian prior as follows:

$$p(\mathcal{D}|\mathbf{w}) = \prod_{i=1}^{n} \mathcal{N}(y_i; f(\mathbf{x}_i, \mathbf{w}), \sigma^2) \qquad \text{and} \qquad p(\mathbf{w}) = \mathcal{N}(0, \alpha^2 \mathbf{I}_m). \tag{4.2}$$

For this model, variational bootstrap yields a set of samples that represents an empirical distribution q and approximates the posterior distribution of the parameters $\mathbf{w}$. These samples are obtained by optimization of a set of particles. Each particle is defined as the *maximum a posteriori* (MAP) estimate for a regression task on a perturbed version of the joint log-likelihood, with its own set of training labels and mean parameters of the prior distribution. The training labels for each particle are obtained by a parametric bootstrap procedure. For each given label y_i, we generate a perturbed label $\tilde{y}_i$ according to the likelihood function, that is Gaussian with variance σ^2 and mean y_i:

$$\tilde{y}_i^{(k)} \sim \mathcal{N}\left(y_i, \sigma^2\right), \; k = 1...K \tag{4.3}$$

Also, each particle is associated with a unique sample from the prior distribution $\tilde{\mathbf{w}}^{(k)}$. So the new prior for each model in the ensemble becomes as follows:

$$p(\mathbf{w}^{(k)}, \tilde{\mathbf{w}}^{(k)}) \sim \mathcal{N}(\tilde{\mathbf{w}}^{(k)}, \alpha^2 \mathbf{I}_m), \text{ where } \tilde{\mathbf{w}}^{(k)} \sim \mathcal{N}(0, \alpha^2 \mathbf{I}_m), \; k = 1...K \tag{4.4}$$

After resampling K perturbed sets of the labels and parameters of the prior distribution, we can obtain K samples from the approximate posterior by solving K optimization problems of the form:

$$\text{argmin}_{\mathbf{w}^{(k)}} \frac{1}{2\sigma_n^2} \sum_{i=1}^{N} (\tilde{y}_i^{(k)} - f(\mathbf{x}_i, \mathbf{w}^{(k)}))^2 + \frac{1}{2\alpha^2} ||\mathbf{w}^{(k)} - \tilde{\mathbf{w}}^{(k)}||^2, \qquad k = 1...K \tag{4.5}$$

The parameters of the particles are updated by a gradient descent algorithm. For the case of NNs under the additional assumption of linear or piecewise linear activation functions, it is shown in [88] that each gradient step optimizes the joint log-likelihood of each model and moves the distribution of parameters $q(\mathbf{w})$ closer to the true posterior $p(\mathbf{w}|\mathcal{D})$.

The main benefit of this procedure is that it allows to use modern gradient optimization techniques, like Adam, to approximate the posterior distribution of the parameters of a nonlinear model.

4.3.3 *Variational Bootstrap for Random Fourier Features*

We demonstrate here that variational bootstrap can induce computational advantages also for the RFF approximation of Gaussian processes. If the labels are

perturbed according to the likelihood so that $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, and the regularization term is a sample from the prior so that $\tilde{\mathbf{w}} \sim \mathcal{N}(0, \alpha^2 \mathbf{I}_m)$, then the MAP solution is:

$$\hat{\mathbf{w}} = \frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{\Phi}(\mathbf{y} + \varepsilon) + \frac{1}{\alpha^2}\mathbf{A}^{-1}\tilde{\mathbf{w}}, \tag{4.6}$$

The MAP estimate $\hat{\mathbf{w}}$ is a Gaussian random vector, as it is a linear combination of two Gaussian random perturbations: ε and $\tilde{\mathbf{w}}$. According to [88], we can calculate the expectation and the covariance of $\hat{\mathbf{w}}$ to obtain the true posterior mean and covariance for the weights of the linear model:

$$\mathrm{E}_{\varepsilon,\tilde{\mathbf{w}}}[\hat{\mathbf{w}}] = \frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{\Phi}\mathbf{y} =: \bar{\mathbf{w}}, \qquad \mathrm{E}_{\varepsilon,\tilde{\mathbf{w}}}[(\hat{\mathbf{w}} - \bar{\mathbf{w}})(\hat{\mathbf{w}} - \bar{\mathbf{w}})^\top] = \mathbf{A}^{-1} \tag{4.7}$$

Equation (4.6) yields samples from the true posterior distribution. From a computational perspective, we observe that it behaves differently from the predictive posterior in (4.1). Here, exactly two linear systems have to be solved for every sample, as opposed to every test point as in (4.1). Let K denote the number of samples; then the complexity becomes $\mathcal{O}(K \times D^2)$. If the number of test points is larger than the number of posterior samples, this can induce significant computational gains, as we demonstrate in the experimental section.

4.3.4 *Dirichlet Label Transformation*

The goal of Bayesian classification is to estimate the distribution of class probabilities for an input data point. The Gaussian likelihood model we discussed in the previous section is not appropriate for a classification task; it is more reasonable to use a Multinomial likelihood instead. For a C-class classification problem, the class label $\mathbf{y}$ for the input point $\mathbf{x}$ is a sample from the Multinomial distribution $\mathbf{y} \sim \mathrm{Cat}(\boldsymbol{\pi})$. The authors of [87] propose to use a C-dimensional Dirichlet distribution to model the distribution of class probabilities $\boldsymbol{\pi} \sim \mathrm{Dir}(\boldsymbol{\alpha})$, with parameters $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_C]^T$.

$$p(\pi_1, \ldots, \pi_C; \alpha_1, \ldots, \alpha_C) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})}\prod_{i=1}^{K}\pi_i^{\alpha_i - 1}, \ \mathrm{B}(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K}\Gamma(\alpha_i)}{\Gamma(\alpha_0)} \tag{4.8}$$

Any label observations are treated as Dirichlet distributions: if an input point $\mathbf{x}$ belongs to a class k, then it corresponds to the following Dirichlet parameters:

$$\alpha_i = \begin{cases} 1 + \alpha_\epsilon, & \text{if } i = k. \\ \alpha_\epsilon, & \text{if } i \neq k. \end{cases} \tag{4.9}$$

The term $\alpha_\epsilon > 0$ represents a small quantity added to the Dirichlet parameters in order to guarantee a valid Dirichlet distribution. Then it is possible to represent samples from the C-dimensional Dirichlet distribution as samples from C Gamma distributions: $\pi_i = \frac{z_i}{\sum_{c=1}^{C} z_c}$, where $z_i \sim \mathrm{Gamma}(\alpha_i, 1)$, for $i \in \{1, ..., C\}$.

The original paper proposes to approximate the Gamma distribution by a Lognormal$(\tilde{y}_i, \sigma_i^2)$ distribution whose parameters are determined by moment matching. Because the logarithm of the log-normally distributed random variable

has a Gaussian distribution $\mathcal{N}(\tilde{y}_i, \sigma_i^2)$, it becomes possible to use a Gaussian likelihood and thus transform the classification problem into a regression problem: the transformed labels $\tilde{y}_i$ become the new targets for the inputs $\mathbf{x}_i$ and σ_i^2 becomes the variance parameter of the Gaussian likelihood.

$$\sigma_i^2 = \log(1/\alpha_i + 1) \qquad \hat{y}_i = \log \alpha_i - \sigma_i^2/2 \tag{4.10}$$

It is worth mentioning that the expression (4.10) produces different noise parameters σ^2 for each observation $\mathbf{y}$. This means that after the transformation, we are dealing with heteroskedastic linear regression.

In order to train the whole classification model, it is required to solve C regression problems, one for each dimension of the Dirichlet distribution. To make predictions, one has to apply a softmax transformation to the outputs of the C regression models $\mathbf{f} = [f_1, ..., f_C]^T$, as follows:

$$\mathbb{E}[\pi_i|\mathbf{x}] = \int \frac{\exp(f_i(\mathbf{x}))}{\sum_{c=1}^{C} \exp(f_c(\mathbf{x})} p(f_i(\mathbf{x})|X) d\mathbf{f}(\mathbf{x}) \tag{4.11}$$

During the inference stage this approach consideres the new type of transformation, which is different from the common sigmoid or softmax transformations. This choice is motivated by the opportunity of approximating the posterior distribution of the transformed labels $\tilde{y}_i$ with a Gaussian distribution.

4.3.5 *Classification with Variational Bootstrap*

One of the key components of variational bootstrap for regression is the data resampling via parametric bootstrap. In the case of classification, it is not straightforward to adjust this strategy to produce perturbed versions of the class labels in a way that reflects the nature of the Bernoulli (or the Multinomial) likelihood. For example, a class label y can take values in $\{0, 1\}$; if we locally fit a distribution $Bern(p)$ to each y, the maximum-likelihood parameter will be the one-sample mean, i.e., $p = 0$ or $p = 1$. If we use this fitted model (i.e. Bernoulli with parameter 0 or 1) to resample new labels, this will deterministically produce either 0 or 1, depending on the original label.

Therefore, we have adopted a strategy that combines variational bootstrap with the Dirichlet labels transformation. Consider a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ of input vectors $\mathbf{x}_i$ and labels y_i for a C-class classification problem. As a first step, we transform each label y_i into a pair $\{\hat{\mathbf{y}}_i, \boldsymbol{\sigma}_i\}$, $\hat{\mathbf{y}}_i \in \mathbb{R}^C$, $\boldsymbol{\sigma}_i \in \mathbb{R}^C$ by the means of the Dirichlet label transformation. The C-class classification problem is then transformed into a C-dimensional heteroskedastic regression problem with labels $\hat{\mathbf{y}}_i$ and observation noise variances $\boldsymbol{\sigma}_i^2$; each training sample and each dimension of the output has observation noise with its own variance.

The second step involves an application of the variational bootstrap method, where we generate K independent sets of regression labels, as well as K sets of prior parameters. Perturbations of each transformed label $\hat{\mathbf{y}}_i$ from the training set are sampled from the corresponding distribution with variance σ_i^2 as follows:

$$\tilde{\mathbf{y}}_i \sim \mathcal{N}(\hat{\mathbf{y}}_i, \text{Diag}(\sigma_i^2)) \qquad \text{where} \qquad \text{Diag}(\sigma_i^2) = \mathbf{I} \odot (\sigma_i^2 \mathbf{1}^\top) \tag{4.12}$$

Algorithm 1 Variational bootstrap for MLP

Input: $\mathbf{X}$, $\mathbf{y}$, α, α_ϵ, h
Output: $\mathbf{w} \sim q(\mathbf{w})$
for $i \leftarrow 1$ **to** N **do**
    $\hat{\mathbf{y}}_i, \sigma_i \leftarrow \text{DirichletTransform}(y_i, \alpha_\epsilon)$ ▷ Eq. (4.10)
end for
for $k \leftarrow 1$ **to** K **do**
    $\tilde{\mathbf{y}}_1^{(k)}, \dots \tilde{\mathbf{y}}_N^{(k)} \sim \mathcal{N}(\hat{\mathbf{y}}_1, \sigma_1^2), \dots, \mathcal{N}(\hat{\mathbf{y}}_N, \sigma_N^2)$
    Draw sample $\tilde{\mathbf{w}}$ from $\mathcal{N}(0, \alpha^2\mathbf{I})$
    Initialize $\mathbf{w}^{(k)} \leftarrow \tilde{\mathbf{w}}$
    $\mathbf{w}^{(k)} \leftarrow$ for each output dimension **optimize** (4.5)
end for

Algorithm 2 Variational bootstrap for RFF

Input: $\mathbf{\Phi}$, $\mathbf{y}$, α, α_ϵ
Output: $\{\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(K)}\} \sim p(\mathbf{w}|\mathbf{\Phi}, \mathbf{y})$
for $i \leftarrow 1$ **to** N **do**
    $\hat{\mathbf{y}}_i, \sigma_i \leftarrow \text{DirichletTransform}(y_i, \alpha_\epsilon)$
end for
for $k \leftarrow 1$ **to** K **do**
    $\tilde{\mathbf{y}}_1^{(k)}, \dots \tilde{\mathbf{y}}_N^{(k)} \sim \mathcal{N}(\hat{\mathbf{y}}_1, \sigma_1^2), \dots, \mathcal{N}(\hat{\mathbf{y}}_N, \sigma_N^2)$
    Draw sample $\tilde{\mathbf{w}}$ from $\mathcal{N}(0, \alpha^2\mathbf{I})$
    $\mathbf{w}^{(k)} \leftarrow$ for each output dimension **solve** (4.6)
end for

The procedure for applying variational bootstrap to classification is summarized in Algorithm 1 for MLP models and in Algorithm 2 for the RFF Gaussian process approximation. In both cases, parameters of the prior distribution for each particle are sampled according to (4.4), while the labels are perturbed as in (4.12). Then in Algorithm 1, we optimize K models independently. For Algorithm 2, the main difference is that we obtain the solution of regression problem (4.5) analytically instead of using gradient optimization. After optimization of the K particles, it is possible to perform predictions using a Monte Carlo approximation of Equation (4.11). As a final remark, our choice to employ the Dirichlet label transformation and to treat the classification problem as regression implies that our methods enjoy any convergence guarantees that can be proven for the regression case.

4.3.6 *Experiments: Toy dataset*

We first demonstrate the application of variational bootstrap on a synthetic one-dimensional binary classification problem. We considered a Gaussian process classifier based on a radial basis function (RBF) kernel, and we approximated it with a Bayesian logistic regression model on a set of random features. We used Random Fourier Features approximation of the RBF kernel proposed in [110]. For our experiment, we considered $5,000$ random features. On the left panel of Fig. 4.1, we show the distribution over functions corresponding to the approximate

posterior distribution over the parameters obtained with variational bootstrap with the Dirichlet label transformation; optimization was performed via the L-BFGS algorithm. In the middle panel of the figure, we report the distribution over functions obtained by the same approximation of the model, but where inference is carried out by Markov chain Monte Carlo (MCMC). On the right panel of the figure, we show the distribution of functions obtained with the Laplace approximation. For the toy dataset, we used the Metropolis-Hastings algorithm with 100 chains. For the prediction, we took the last sample from each chain. R-hat convergence diagnostic [14] showed that it takes around 10^6 steps before convergence. The comparison shows a remarkable property of the proposed approach to accurately approximate the posterior over model parameters without the need for expensive or excessively long computations. In fact, the L-BFGS algorithm converged for variational bootstrap after 16 iterations only.

The Laplace approximation is described in Section 2.2.3.

4.3.7 *Experiments: UCI Datasets*

Table 4.1: UCI datasets used for evaluation.

Dataset	Classes	Training instances	Test instances	Dimensionality
Magic	2	14020	5000	10
HTRU2	2	12898	5000	8
MiniBoo	2	120064	10000	50
Drive	11	48509	10000	54
Letter	26	15000	5000	16
Mocap	5	68095	10000	37

We evaluated our method on several UCI classification problems outlined in Table 4.1. We applied variational bootstrap with the Dirichlet label transformation on two different models. First, we considered a two-hidden layer MLP model with a ReLU activation function and 512 neurons in each hidden layer. We used Adam [68] to optimize the parameters of the model. This model is referred to as VBoot-MLP. The second model we considered is the RFF approximation [110] of a GP, where 5,000 random features were used to approximate an RBF kernel with fixed hyperparameters. In this case, we generated 50 samples as prescribed in

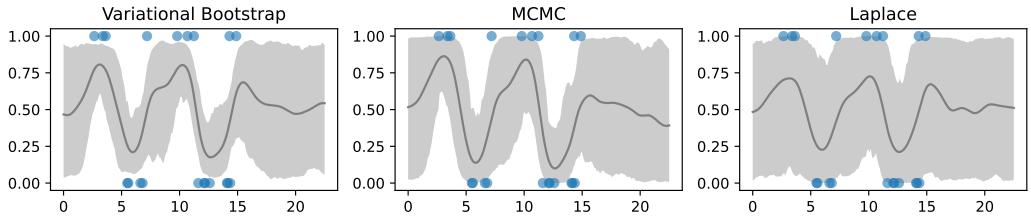

Figure 4.1: Comparison between the predictions with the regression weights, obtained by the variational bootstrap (left), MCMC (middle) and Laplace approximation (right)

Equation (4.6); this required solving 50 linear systems of order $\mathcal{O}(D^2)$, instead of thousands as required by Equation (4.1). This model is referred to as VBoot-RFF.

The performance of variational bootstrap is compared against a regular BNN, whose posterior has been approximated by means of MCMC sampling, and in particular, SGHMC [21]. The baseline BNN uses the same prior distribution over the parameters and a Bernoulli likelihood, while its output is given by a softmax activation function. VBoot-RFF is compared against the Dirichlet transformation with a Sparse GP approach proposed in [87]. The hyperparameters of this model were optimized as in the original paper. We evaluate performance using several metrics, including classification error and mean negative log-likelihood (MNLL). The results are outlined in Tables 4.2 and 4.3, respectively. The models considered are referred to as VBoot-MLP, VBoot-RFF, Sparse GP, and MCMC-MLP, respectively.

Considering the MLP models, our experiments show that the proposed approach is competitive when compared against a principled framework such as MCMC. Most importantly, our approach has significantly different behavior in terms of convergence speed, as it relies on optimization rather than sampling. We demonstrate this property empirically by monitoring the progression of validation error for variational bootstrap and MCMC. In Fig. 4.2, we report classification error on held-out data over the first 200 training epochs. These results indicate that the variational bootstrap technique provides much faster convergence compared to SGHMC. On the other hand, only in some cases SGHMC eventually settles to significantly lower validation error. We used the Wilcoxon test [148] to compare classification errors and MNLLs for VBoot-MLP and MCMC-MLP methods. The test did not show any statistically significant difference regarding the performance of these two methods within 0.05 significance level. The lack of statistical significance appears partly due to the small number of used data splits. Nevertheless, variational bootstrap is shown to achieve a good trade-off between accuracy and efficiency. This trade-off can be further exploited in practice, as variational bootstrap is trivially parallelizable.

Regarding the GP-based models, we also see that VBoot-RFF is highly competitive against traditional sparse GPs that rely on inducing points. In fact, according to the one-sided Wilcoxon signed-rank test, the VBoot-RFF results are slightly better than Sparse GP in a statistically significant way for some of the datasets (marked with "*" in Tables 4.2 and 4.3). Our variational bootstrap framework allowed us to use a large number of random features (i.e., $5,000$), which would not be possible for many of the datasets considered. This can be seen in Fig. 4.3, which shows the computation time for VBoot-RFF and Sparse GP models as a function of the number of random features and inducing points correspondingly. Of course, using more random features/inducing points results in a better approximation of the full GP model, but it also increases computational complexity. We see that VBoot-RFF has better scalability properties compared to sparse GPs. Regarding the computation times reported for Sparse GPs, we note that we have excluded the initial K-Means step needed to initialize the inducing locations, so in practice, sparse GPs are more expensive than what we report here. Interestingly, sparse GPs could not scale beyond $1,000$ or $2,000$ inducing points for some datasets due to memory errors.

Table 4.2: Classification error for variational bootstrap with the Dirichlet transformation and Markov-chain Monte Carlo approaches.

Dataset	VBoot-MLP	MCMC-MLP	VBoot-RFF	Sparse GP
Magic	0.12±0.01	0.12±0.01	0.13±0.01	0.13±0.01
HTRU2	0.02±0.01	0.02±0.00	0.02±0.00	0.02±0.01
MiniBoo	0.11±0.01	0.09±0.01	0.08±0.01	0.08±0.01
Drive	0.01±0.00	0.001±0.000	0.01±0.00*	0.02±0.01
Letter	0.05±0.01	0.03±0.01	0.05±0.01*	0.08±0.01
Mocap	0.005±0.000	0.007±0.000	0.02±0.00*	0.03±0.01

Table 4.3: MNLL for variational bootstrap with the Dirichlet transformation and Markov-chain Monte Carlo approaches.

Dataset	VBoot-MLP	MCMC-MLP	VBoot-RFF	Sparse GP
Magic	0.31±0.00	0.29±0.00	0.33±0.01*	0.35±0.01
HTRU2	0.08±0.00	0.07±0.00	0.07±0.01	0.07±0.01
MiniBoo	0.25±0.01	0.22±0.00	0.20±0.01*	0.21±0.01
Drive	0.06±0.00	0.01±0.00	0.08±0.03	0.08±0.01
Letter	0.31±0.00	0.24±0.00	0.28±0.01	0.25±0.01*
Mocap	0.02±0.00	0.03±0.00	0.09±0.01*	0.13±0.01

4.3.8 *Discussion of the approach*

In this work, we proposed a novel way to carry out Bayesian inference for classification models based on Neural Networks and Gaussian processes. For NNs, this is important because, while they achieve state-of-the-art performance in many tasks, they lack a principled way to characterize uncertainty in predictions, so they represent a class of models for which a Bayesian treatment is highly desirable but mathematically and computationally challenging. For Gaussian processes, instead, while the Bayesian treatment is at the core of its formulation, classification tasks are difficult to handle because they require expensive approximations. Our work provides a practical and easily parallelizable way to tackle all these limitations.

We are currently investigating parallel implementations of the proposed approach to considerably accelerate inference of large-scale problems by operating on clusters of computing machines. In addition, we are exploring the application of our approach to problems involving optical-based computing hardware, also known as Optical Processing Units [118]. OPUs offer a fast and low-power way to approximate Gaussian processes through random features, and because they are capable of generating millions of these at the speed of light, we believe that our approach could be the key to exploit these computations effectively.

Another possible direction is the adaptation of our framework towards a more efficient marginal likelihood maximization for GPs, which is a standard practice to

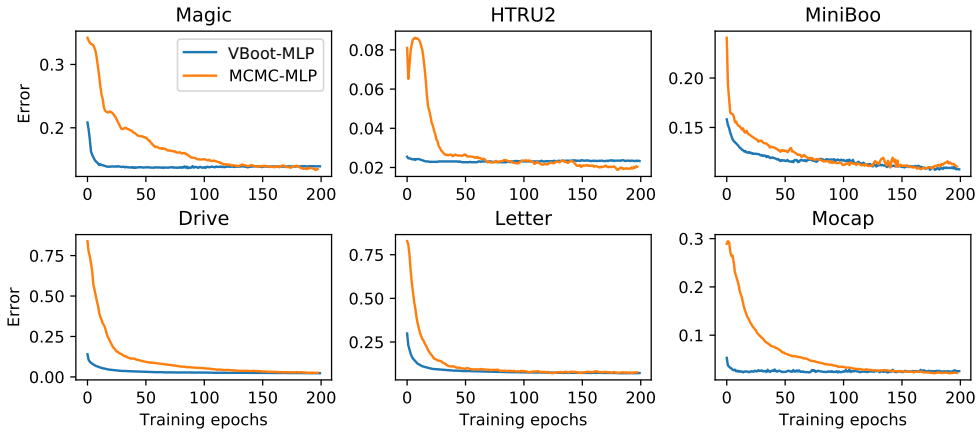

Figure 4.2: Convergence of the classification error on validation data for variational bootstrap and SGHMC methods

Figure 4.3: Computation time complexity for VBoot-RFF and Sparse GP as a function of the number of random features and inducing points correspondingly.

tune kernel hyperparameters [114]. In this work, we have treated GPs by means of fixed feature map approximations, which correspond to fixed hyperparameter values. Estimating marginal likelihoods through samples is an open research question, and it can be the subject of future work.

4.4 IMPROVING THE DIRICHLET-TRANSFORMATION

4.4.1 *Motivation*

We aim to develop a method that provides well-calibrated predictions and accurate uncertainty estimates for Gaussian Processes Classification (GPC) by improving on the idea proposed in [87] and described in Sec. 4.3.4. The original method proposes to use a Log-normal distribution to approximate a Gamma distribution, which is used for sampling from a Dirichlet distribution. In fact, the Log-normal distribution poorly approximates Gamma distributions with a small value of the

parameter α of the Gamma distribution. The experiments in the original paper [87] show that the best performance of the original method is achieved exactly for such cases. This means that an approximate posterior predictive distribution poorly approximates the true posterior predictive distribution when the classification error is the smallest. Also, this conclusion is indirectly supported by the visualization of the predictive posterior for 1D synthetic datasets, provided in [87], where we can see an underestimation of the variance of the predictive distribution.

The proposed work replaces the Log-normal approximation of the Dirichlet distribution used in the original work. The idea is to construct a transformation that allows to transform a Dirichlet distributed random variable to a normally distributed random variable. With this transformation, it will be possible to project the Dirichlet distributed class probability $\boldsymbol{\pi}$ to a latent representation in such a way that the latent representation is Gaussian. To obtain a class probability distribution for a test data point, it is necessary to solve a Gaussian process regression in a latent space with projected training labels. Then we have to apply the inverse of the initial transform to the prediction for the test data point representation in the latent space.

4.4.2 *Cumulative Density Function Transformation*

Considering a two-class classification problem, let's assume that there is a dataset $\mathcal{D}\{(\mathbf{x_i}, y_i)\}$ of N input vectors $\mathbf{x_i}$ and binary labels y_i. In this case, the distribution of class probabilities for each input $\mathbf{x}$ is one-dimensional. Thus, it is possible to use a Beta distribution to model class probabilities instead of a Dirichlet distribution. For each training data point, there is a Beta distributed random variable π responsible for a class probability. If $y = 0$, the corresponding distribution of π will be $\text{Beta}(\alpha_\epsilon, 1 + \alpha_\epsilon)$, if $y = 1$, the corresponding distribution of π will be $\text{Beta}(1 + \alpha_\epsilon, \alpha_\epsilon)$. The hyperparameter α_ϵ refers to the uncertainty in class labels.

The goal is to transform class probability π to a normally distributed latent random variable f centered at $-\mu$ for one class and μ for another class. We impose this symmetry in the latent space to reflect the symmetry that exists between $\text{Beta}(\alpha_\epsilon, 1 + \alpha_\epsilon)$ and $\text{Beta}(1 + \alpha_\epsilon, \alpha_\epsilon)$.

$$F_{\text{Beta}(\alpha_\epsilon, 1+\alpha_\epsilon)}(\pi) = 1 - F_{\text{Beta}(\alpha_{1+\epsilon}, \alpha_\epsilon)}(1 - \pi) \tag{4.13}$$

After the transformation, the symmetric one should be as follows:

$$F_{\mathcal{N}(-\mu, \sigma^2)}(f) = 1 - F_{\mathcal{N}(\mu, \sigma^2)}(-f) \tag{4.14}$$

It is known that it is possible to transform a random variable X with a cumulative density function F_X to a random variable Y with a cumulative density function F_Y as follows:

$$Y = F_Y^{-1}(F_X(X)) \tag{4.15}$$

We can apply this expression to transform a Beta distributed random variable into a normally distributed random variable.

$$\begin{aligned} g_1(x) &= F^{-1}_{\mathcal{N}(-\mu, \sigma^2)}(F_{\text{Beta}(\alpha_\epsilon, 1+\alpha_\epsilon)}(x)) \\ g_2(x) &= F^{-1}_{\mathcal{N}(\mu, \sigma^2)}(F_{\text{Beta}(1+\alpha_\epsilon, \alpha_\epsilon)}(x)) \end{aligned} \tag{4.16}$$

Also, the requirement of symmetry provides an intuition for another constraint. When we observe a latent prediction that lies in the middle between the centers of the Gaussians, we make a decision that the model is unable to assign a specific class label to this prediction. Thus, we have to force the inverse transformation to return 0.5 for this latent prediction. This will signify that the model assigns 0.5 probability for both classes.

$$\begin{aligned} &g_1^{-1}(0) = 0.5 \\ &F^{-1}_{\text{Beta}(\alpha_\epsilon, 1+\alpha_\epsilon)}(F_{\mathcal{N}(-\mu,\sigma^2)}(0)) = 0.5 \end{aligned} \tag{4.17}$$

The same derivations applied for the transformation g_2 will give the same result. Thus, we have obtained a connection between parameters μ and σ for the latent representations of the distributions of class probability.

$$\mu = \sigma\sqrt{2}(\text{erf}^{-1}(2F_{\text{Beta}(\alpha_\epsilon, 1+\alpha_\epsilon)}(0.5) - 1) \tag{4.18}$$

This constraint gives the following property. Ratios between a probability mass of the whole distribution and the probability mass belonging to a region of overlap with the symmetric distribution are equal for the distributions of class probability and for their latent representations.

$$\begin{aligned} \frac{1}{1 - F_{\mathcal{N}(-\mu,\sigma^2)}(0) + F_{\mathcal{N}(\mu,\sigma^2)}(0)} &= \frac{1}{1 - F_{\text{Beta}(\alpha_\epsilon,1+\alpha_\epsilon)} + F_{\text{Beta}(1+\alpha_\epsilon,\alpha_\epsilon)}(0.5)} \\ \frac{1}{2F_{\mathcal{N}(-\mu,\sigma^2)}(0)} &= \frac{1}{2F_{\text{Beta}(\alpha_\epsilon,1+\alpha_\epsilon)}(0.5)} \end{aligned} \tag{4.19}$$

The latter equality is satisfied because of equality (4.17).

Now there are two transformations instead of one. This fact poses the question: which one is necessary to use for the inverse operation? When we observe in the latent space a GP prediction with parameters $-\mu$, σ^2, it is better to use g_1, and g_2 for the symmetric case. There is an intuition that it is better to use the transformation g_1^{-1} if the latent prediction is closer to the Gaussian, centered at $-\mu$ and the transformation g_2^{-1}, when the latent prediction is closer to the Gaussian centered at μ.

Both transformations g_1^{-1}, g_1^{-1} represent some activation functions that maps an interval $[-\infty, \infty]$ into $[0, 1]$ (Fig. 4.4).

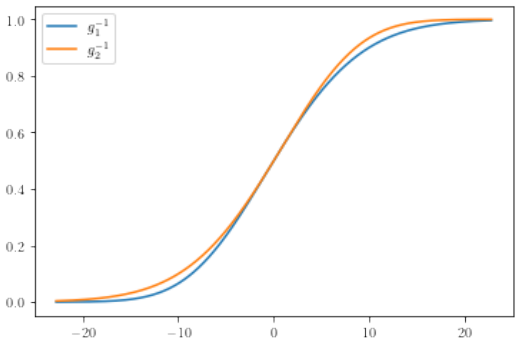

Figure 4.4: The plot of the activation functions g_1^{-1}, g_2^{-1} for $\mu = 12.74$, $\sigma = 10$

There is an asymmetry of each function with respect to the point $[0, 0.5]$. This asymmetry imposes the following problem. If we apply both transformations to the latent predictions with a distribution $\mathcal{N}(-\mu, \sigma^2)$ the goal is to obtain a distribution of class probabilities $\text{Beta}(\alpha_\epsilon, 1 + \alpha_\epsilon)$ for each transformation.

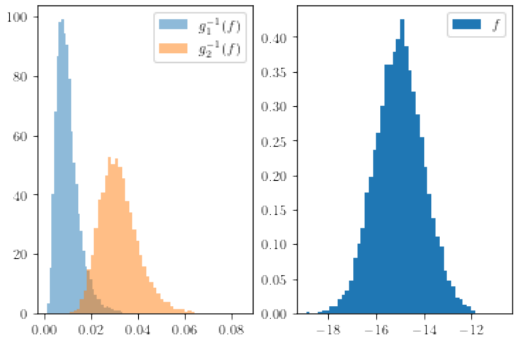

Figure 4.5: The histograms of predictions in the latent space (right) and corresponding transformed values in the space of class probabilities (left).

It is possible to see that the transformation g_2^{-1} gives an incorrect distribution of class probabilities when we observe the distribution of the latent predictions, corresponding to $y = 0$, which is $\mathcal{N}(-\mu, \sigma)$. The symmetric situation happens for $y = 1$ with g_1^{-1} transformation. It is possible to overcome the described difficulty by constructing a hybrid transformation that combines transformations g_1 and g_2

$$g(x) = \begin{cases} g_1(x), & \text{if } x < 0.5 \\ g_2(x), & \text{if } x >= 0.5 \end{cases} \tag{4.20}$$

The inverse transform is as follows:

$$g^{-1}(x) = \begin{cases} g_1^{-1}(x), & \text{if } x < 0 \\ g_2^{-1}(x), & \text{if } x >= 0 \end{cases} \tag{4.21}$$

This transformation is symmetric and thus produces almost correct distributions of class probabilities π for the extreme cases in the latent space when the GP predictions are distributed according to $\mathcal{N}(-\mu, \sigma^2)$ or $\mathcal{N}(\mu, \sigma^2)$. The quality of this approximation for non-extreme cases will be analyzed empirically in the next section.

4.4.3 *Experiments*

At the beginning, the method was tested on a toy classification dataset. Simple GP regression was used to solve the classification problem in a latent space. The hyperparameters of the kernel were optimized by log marginal likelihood maximization. Obtained predictions were compared with the predictions obtained with the log-normal approximation proposed in [87] (Fig. 4.6). The same noise hyperparameter $\alpha_\epsilon = 0.2$ was used for both methods. As we see in Fig. 4.6, the proposed method assigns higher uncertainty estimates for the same input points.

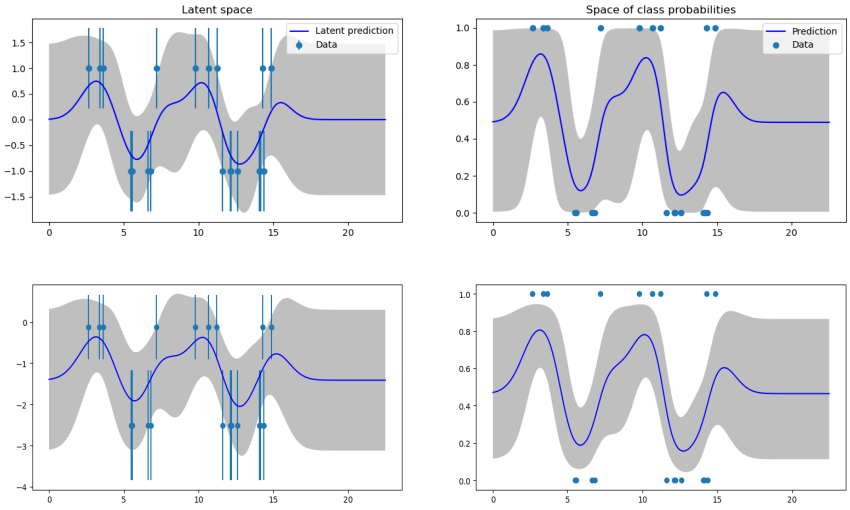

Figure 4.6: Toy dataset predictions obtained with the proposed transformation (top) and with the log-normal approximation (bottom)

It is worth mentioning that for the toy dataset, the choice of μ hyperparameter for the latent space does not affect the distribution of class probabilities. Different values of parameter μ in a range $[0.01, 50]$ were tested, and all of them gave exactly the same predictions in the space of class probabilities.

Also, the proposed method was tested on several UCI classification datasets and compared with the log-normal approximation in terms of classification error, mean negative log-likelihood (MNLL), and expected calibration error (ECE), which is outlined in Tables 4.4, 4.5, 4.6. In the tables, the proposed method is denoted as Hybrid Transformation, and the baseline is denoted as Log-normal approximation. The classification performances of both methods are very similar, but the proposed method requires solving only one regression problem in a latent space instead of two, as in the case of the baseline.

Table 4.4: Classification error of the proposed method and log-normal approximation

Dataset	Hybrid Transformation	Log-normal approximation
EEG	0.18±0.01	0.21±0.03
Magic	0.133±0.002	0.139±0.003
HTRU2	0.0224±0.0006	0.022±0.001
MiniBoo	0.083±0.003	0.085±0.003

4.4.4 *Discussion of the approach*

The general goal of this work was to obtain an accurate posterior predictive distribution. The proposed method still uses an approximation, and it does not allow obtaining the true posterior distribution. But the most problematic part of this approach is multi-class classification because, in this case, we have to construct a C-dimensional activation function that maps logits to the $[0, 1]$ interval. The

Table 4.5: MNLL of the proposed method and log-normal approximation

Dataset	Hybrid Transformation	Log-normal approximation
EEG	0.40±0.03	0.43±0.04
Magic	0.336±0.006	0.351±0.005
HTRU2	0.074±0.005	0.081±0.005
MiniBoo	0.226±0.008	0.205±0.006

Table 4.6: ECE of the proposed methods and log-normal approximation

Dataset	Hybrid Transformation	Log-normal approximation
EEG	0.027±0.005	0.022±0.007
Magic	0.0263±0.0009	0.022±0.004
HTRU2	0.043±0.001	0.039±0.001
MiniBoo	0.0274±0.0003	0.0286±0.0009

empirical method of construction of the activation function proposed in this work is not applicable in a C-dimensional space.

The preliminary experiments conducted in this work for two class datasets show some advantages of the proposed technique compared to the Dirichlet label transformation in terms of negative log-likelihood and accuracy, but they are ambiguous in terms of Calibration Error. Because of these two arguments, this proposed approach is considered promising while requiring more investigation.

5 LOCAL GAUSSIAN PROCESS REGRESSION

5.1 INTRODUCTION

Function estimation is a fundamental problem in Machine Learning. In supervised learning tasks applied to a data set composed of observed input data and labels, the goal of function estimation is to establish a mapping between these two groups of observed quantities. Function estimation can be approached in various ways, and we can broadly divide algorithms in two categories, as *global* and *local*. Examples of global algorithms are Neural Networks [93] and kernel machines [123], which impose a functional form yielding a global representation of the function. The functional form is parameterized by a set of parameters that are optimized or inferred based on all the available data. The estimated model can later be used to query the function at any input points of interest. In local algorithms such as K-Nearest Neighbors (KNN), instead, the target point is fixed, and the corresponding value of the function is estimated based on the closest data available.

Obviously, any global algorithm can be made local by training it only for the few training points located in the vicinity of the target test point. While it may seem that the idea of localizing global algorithms is not a very profound one, empirical evidence shows that localization could improve the performance of the best global models [12]. The idea of localization was therefore applied to global models such as SVMs [8, 9]. In addition to performance gains, by operating on smaller sets of data points, these local approaches enjoy computational advantages, which are particularly attractive for kernel machines for which scalability with the number of data points is generally an issue [23, 121, 122].

In this work, we develop novel ideas to implement a localization of Gaussian processes (GPs) in order to obtain performance gains, as well as computational ones. GPs are great candidates to benefit from computational speedups given that a naïve implementation requires expensive algebraic computations with the covariance matrix; denoting by n the number of input data, such operations cost $\mathcal{O}(n^3)$ operations and require storing $\mathcal{O}(n^2)$ elements, hindering the applicability of GPs to data sets beyond a few thousand data points [108]. Another issue with GPs is how to choose a suitable kernel for the problem at hand so as to avoid problems of model misspecification. Both of these issues have been addressed in various ways, by proposing scalable approximations based on inducing points [57] and random features [27, 110], and by composing GPs to obtain a rich and flexible class of equivalent kernels [151].

In this work, we explore an alternative way to address scalability and kernel design issues by localizing GPs. In particular, we show how the localization operation leads to a particular form for the localized GP and what is the effect on the kernel of this model. Furthermore, the localization makes it apparent how to implement the model with considerable gains compared to other approaches to

approximate GPs. We demonstrate such performance gains on regression tasks on standard UCI benchmarks [34].

5.2 RELATED WORK

Local learning algorithms were introduced by Bottou and Vapnik [12], with the main objective of estimating the optimal decision function for each single testing point. Examples of local learning algorithms include the well-known K-Nearest Neighbor regression [2] and local polynomial regression [37]. These methods provide simple means for solving regression problems for the cases where training data are nonstationary or their size is prohibitively large for building a global model. However, neither of these methods provides ways to quantify uncertainty in predictions, which is a highly desirable feature in cost-sensitive applications.

Gaussian Process Regression (GPR) [114] is a popular nonparametric regression method based on Bayesian principles, which provides uncertainty estimates for its predictions. Similarly to other kernel methods (e.g., SVMs and KRR), GPR is a global method, meaning that it takes into account the whole dataset at prediction time. Thus, GPR inherits the computational complexity of global kernel methods, which is prohibitive for large datasets. Among the large class of scalable approximations for GPR, successful ones are based on Random Fourier Features [110] and on sparsification of the Gram matrix induced by the kernel [114].

Random feature approximation of the kernel proposed in [110] is based on the Bochner theorem and allows representing the kernel function as a dot product of (possibly infinite) feature maps applied to the input data. In practice, infinite feature maps are replaced by a finite Monte Carlo approximation. The disadvantage of this approach is that it is necessary to construct a specific random feature mapping for each type of kernel. While random feature approximations are known for popular kernels such as RBF [110], and polynomial [103], there is no straightforward application of this method to approximate arbitrary kernels.

The Gram matrix sparsification approach is based on the idea of introducing so-called inducing points in order to approximate the full Gram matrix. One of the most popular methods in this family is the Nyström approximation [114]. The main drawback of this approach is that a low number of inducing points might lead to a poor approximation of the original model, which affects predictive performance. An important advancement within this family of approaches which provides a scalable variational formulation, was proposed in [133].

While providing good performance and scalability for large datasets, these approaches still require some design choices for the kernel. For stationary kernels, they assume that the same kernel is suitable for all the regions of input space, and if data are nonstationary, this may harm the predictive performance. The literature has a wide range of proposals to address kernel design by incorporating ideas from deep learning [27, 151].

Recently, partitioning strategies have also gained some attention. The main idea is to divide the input space into regions where local estimators are defined [18, 86, 92, 131]. In partition-based methods, the main challenge is to define an effective partitioning of the space.

There are several approaches that use the idea of local learning for training GP models. The method proposed in [85] and extended in [84] mostly focuses on Bayesian parametric linear regression. The methods in these papers build an ensemble of local models centered at several fixed points, where each training point is weighted accordingly to the distance from the center of the model. Predictions are computed as a weighted sum of the local models. The authors claim that their approach extends to GPR, but in this case, each local model considers the full training set. This means that these methods use localization to address nonstationarity but poorly scale to large datasets. The method proposed in [126] proposes to build local GP models that use only subsets of the training data, but it lacks a mechanism that assigns importance weight for the training points for each local model according to the distance from the center of the model. That is why the model can make overconfident predictions for the points that lay far away from the centers of the local models. In [48], in order to obtain fast approximate prediction at a target point, the Authors propose a forward step-wise variable selection procedure to find the optimal sub-design.

5.3 GAUSSIAN PROCESSES, KERNEL RIDGE REGRESSION, AND LOCALIZATION

5.3.1 *Gaussian Process Regression*

We already discussed GP regression in Chapter 2, Section 2.2. The problem with this approach is that it is required solving a linear system involving a matrix of size $n \times n$ to obtain the parameters of a predictive distribution (2.28), (2.29). Direct methods to solve these operations require $O\left(n^3\right)$ operations and storing $O\left(n^2\right)$. Iterative solvers, instead, can reduce these complexities by relying exclusively on matrix-vector products, which require $O\left(n^2\right)$ operations per iteration and do not need to store the Gram matrix [28, 38]. However, a quadratic time complexity may still be prohibitive for large-scale problems.

There is rich literature on approaches that recover tractability by introducing approximations. One popular line of work introduces m so-called inducing points as a means to approximate the whole GP prior [108]. This treatment of GPs was later extended within a scalable variational framework [71, 133], making the complexity cubic in the number of inducing points m. Another approach proposes ways to linearize GPs by obtaining an explicit set of features so as to obtain a close approximation to the original kernel-based model. Within this framework, a popular approach is based on random features [110]. Denoting by $\mathbf{\Phi}$ the $n \times D$ matrix obtained by applying a set of D random basis functions to the inputs in $\mathbf{x}_1 \ldots, \mathbf{x}_n$, these approximations are so that $\mathbf{\Phi}\mathbf{\Phi}^\top$ approximates in an unbiased way the original kernel matrix $K_{\mathbf{X}\mathbf{X}}$, that is $E[\mathbf{\Phi}\mathbf{\Phi}^\top] = K_{\mathbf{X}\mathbf{X}}$. For the Gaussian kernel, for example, a Fourier analysis shows that the basis functions that satisfy this property are trigonometric functions with random frequencies [110]. This approach has been applied to GPs in [80] and later made scalable by operating on mini-batches in [27].

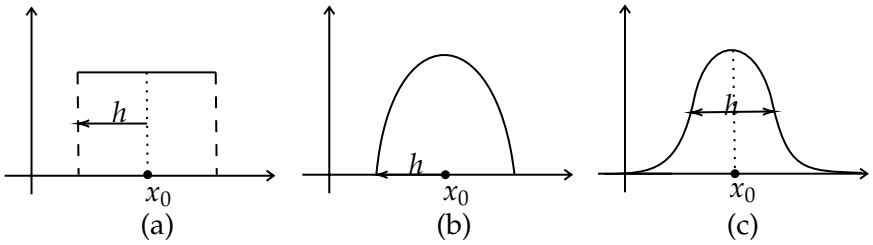

Figure 5.1: Examples of local kernels: (a) Rectangular kernel, (b) Epanechnikov kernel, (c) Gaussian kernel.

5.3.2 *Locally Smoothed Gaussian Process Regression*

GP regression, as formulated above, is an example of a *global learner*; for a fixed training data $\mathcal{D}$, it builds a posterior distribution over functions that can be used to calculate the predictive distribution for any test inputs. To construct a predictive distribution for *any* input point, it is necessary to use all the information available from the data $\mathcal{D}$, resulting in the need to do algebraic operations with the matrix $\left(K_{\mathbf{XX}} + \sigma^2 I_n\right)$. However, if we focus on the prediction problem locally, at a *given* target input $\mathbf{x}_0$, most of the information carried by the (potentially large) covariance matrix might be neglected with little loss of information. The main idea behind localized GPs is to down-weight the contribution of the data points far from $\mathbf{x}_0$, so that the structure of the covariance matrix is more adapted to the prediction task at a given point. To make this general idea work, we need to tackle two challenges. First, we need to specify what it means to be far or close to a given point; second, the change of the structure of the covariance matrix must give us a valid covariance matrix, i.e., the resulting covariance matrix should be symmetric and positive definite. Note that a simple truncation of the covariance function to obtain a compact-support covariance function may generally destroy positive definiteness [65].

We accomplish the localization of GPs in a straightforward manner as follows. We localize the target and the prior in the model (2.32) by multiplying them by the square root of the weighting function:

$$k_h(\mathbf{x}, \mathbf{x}_0) := \frac{1}{h} k\left(\frac{\|\mathbf{x} - \mathbf{x}_0\|}{h}\right),$$

where $k : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ is a non-negative, integrable function satisfying $\int K(\mathbf{x}) d\mathbf{x} = 1$ and $\|\cdot\|$ is Euclidean norm on $\mathbb{R}^d$. Considering the square root of the weighting function will be convenient later when we discuss the link between local GPs and local Kernel Ridge Regression. Some classical examples of the weighting functions are given in Fig. 1. Because of the linearity of the weighting operation, the resulting model is another zero-mean Gaussian process $\tilde{f}(\mathbf{x})$ with covariance function given by

$$\tilde{K}(\mathbf{x}, \mathbf{x}'; \mathbf{x}_0) = k_h^{\frac{1}{2}}(\mathbf{x}, \mathbf{x}_0) K(\mathbf{x}, \mathbf{x}') k_h^{\frac{1}{2}}(\mathbf{x}', \mathbf{x}_0). \tag{5.1}$$

In this formulation, we have localized the relationship between noisy targets and function realizations as

$$\tilde{y}_i = \tilde{f}(\mathbf{x}_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2), \tag{5.2}$$

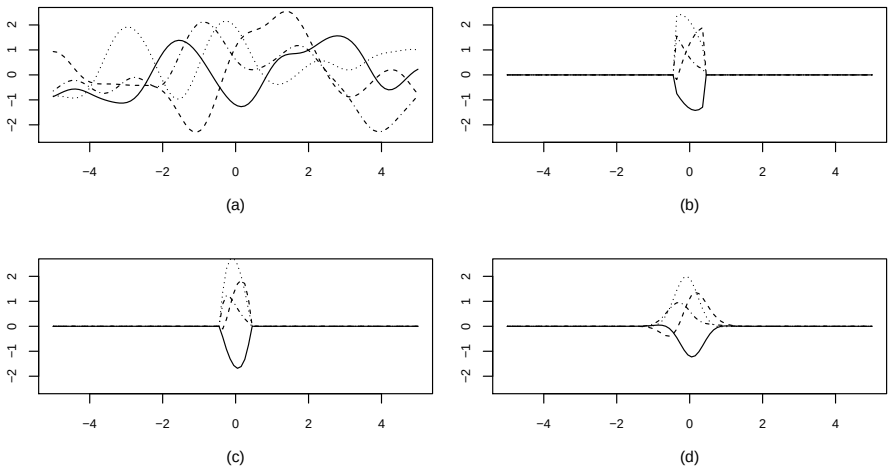

Figure 5.2: (a) Samples from a global GP prior with the exponential kernel. (b) Samples from a GP prior with exponential kernel localized by rectangular smoother centered at $x_0 = 0$. (c) Samples from a GP prior with exponential kernel localized by Epanechnikov smoother centered at $x_0 = 0$. (d) Samples from a GP prior with exponential kernel localized by Gaussian smoother centered at $x_0 = 0$.

with $\tilde{y}_i = \sqrt{k_h(\mathbf{x}_i, \mathbf{x}_0)}y_i$, and the prior is given by a zero-mean GP with the localized covariance kernel (5.1). The model (5.2) can be alternatively written as a model with heteroscedastic noise

$$y_i = f(\mathbf{x}_i) + \frac{1}{\sqrt{k_h(\mathbf{x}_i, \mathbf{x}_0)}}\varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2).$$

Making the noise parameter location-dependent can significantly improve the performance for problems where the assumption of a homoscedastic noise is not satisfied.

Proposition 1 *Let $I = \{i : \|\mathbf{x}_i - \mathbf{x}_0\| \leq h\}$, $\mathbf{X}_I = \{\mathbf{x}_i : i \in I\}$ and $\mathbf{y}_I = \{y_i\}_{i \in I} \in \mathbb{R}^{|I|}$. Assume that (5.2) holds for the fixed target point $\mathbf{x}_0$. Then $f(\mathbf{x}_0) \mid \mathbf{y}_I$ is a Gaussian random variable with mean and variance given by*

$$\tilde{m}(\mathbf{x}_0) = K_{\mathbf{x}_0\mathbf{x}_I}\left(K_{\mathbf{x}_I\mathbf{x}_I} + \sigma^2\mathbf{W}_{\mathbf{x}_0}^{-1}\right)^{-1}\mathbf{y}_I \tag{5.3}$$

$$\tilde{\mathcal{K}}(\mathbf{x}_0, \mathbf{x}_0) = K(\mathbf{x}_0, \mathbf{x}_0) - K_{\mathbf{x}_0\mathbf{x}_I}\left(K_{\mathbf{x}_I\mathbf{x}_I} + \sigma^2\mathbf{W}_{\mathbf{x}_0}^{-1}\right)^{-1}K_{\mathbf{x}_I\mathbf{x}_0} \tag{5.4}$$

where $\mathbf{W}_{\mathbf{x}_0}$ is the diagonal matrix with main diagonal entries $k_h(\mathbf{x}_i, \mathbf{x}_0)$, $\mathbf{x}_i \in \mathbf{X}_I$.

Proof: Let $\mathbf{x}_0$ be any fixed target point. Then the observations $\mathbf{y}_I \in \mathbb{R}^{|I|}$ and GP-function value at target point $f_0 = f(\mathbf{x}_0) \in \mathbb{R}$ are jointly Gaussian such that

$$\begin{bmatrix} \mathbf{y} \\ f_0 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0}_I \\ 0 \end{bmatrix}, \begin{pmatrix} K_{\mathbf{x}_I\mathbf{x}_I} + \sigma^2\mathbf{W}_{\mathbf{x}_0}^{-1} & K_{\mathbf{x}_I\mathbf{x}_0} \\ K_{\mathbf{x}_0\mathbf{x}_I} & K(\mathbf{x}_0, \mathbf{x}_0) \end{pmatrix}\right).$$

Then the proposition follows from the basic formula for conditional distributions of Gaussian random vectors (see, e.g., [113], Appendix A.2).

Compared to global GPs, in the local formulation, in order to compute the posterior mean and variance, we need to invert $\left(K_{\mathbf{X}_I\mathbf{X}_I} + \sigma^2\mathbf{W}_{\mathbf{x}_0}^{-1}\right)$. This might give a key advantage when dealing with large data sets, as the localization by the compactly supported kernel (local) could significantly sparsify the Gram matrix corresponding to $K_{\mathbf{XX}}$. Denoting by s_0 the number of inputs for which the localizing weights are nonzero for a test point $\mathbf{x}_0$, the complexity of performing such an inversion is $\mathcal{O}(s_0^3)$. Another interesting observation is that the kernel function $\tilde{K}(\mathbf{x}, \mathbf{x}')$ is potentially more flexible than the original kernel function; this is due to the multiplication by the localizing weighting function, which may introduce some interesting nonstationarity even for kernel functions which are stationary, depending on the choice of the weighting function.

The calculation of the predictive distribution with Locally Smoothed Gaussian Process Regression (LSGPR) is described in Algorithm 3 The parameter selection in probabilistic models given by GPs is based on the *marginal log-likelihood* maximization, which, in our local formulation, can be defined as follows:

$$\begin{aligned}\log p(\mathbf{y}_I|\mathbf{X}_I) = &-\frac{1}{2}\mathbf{y}_I^\top \left(K_{\mathbf{X}_I\mathbf{X}_I} + \sigma^2\mathbf{W}_{\mathbf{x}_0}^{-1}\right)^{-1}\mathbf{y}_I \\ &-\frac{1}{2}\log\left|K_{\mathbf{X}_I\mathbf{X}_I} + \sigma^2\mathbf{W}_{\mathbf{x}_0}^{-1}\right| - \frac{n}{2}\log(2\pi)\end{aligned} \tag{5.5}$$

Unfortunately, gradient-based optimization cannot be used to find the optimal localization parameter h, as the marginal log-likelihood is not continuously differentiable w.r.t. this parameter when compactly supported local kernels are used. The simplest way to resolve this problem is by using grid search for the localization parameter h, while kernel parameters can be optimized by gradient-based methods for any given h.

Algorithm 3 LSGPR

Input: $\mathbf{X}$, $\mathbf{y}$, σ^2, h, $\mathbf{x}_0$
Output: $\tilde{m}(\mathbf{x}_0)$, $\tilde{\mathcal{K}}(\mathbf{x}_0, \mathbf{x}_0)$
$I := \{i : ||\mathbf{x}_i - \mathbf{x}_0|| \leq h\}$
$\mathbf{X}_I := \{\mathbf{x}_i : i \in I\}$
$\mathbf{y}_I := \{y_i : i \in I\}$
$\mathbf{W}_{\mathbf{x}_0} := \text{diag}(\{k_h(\mathbf{x}_i, \mathbf{x}_0) : i \in I\})$
$\mathbf{L} := \text{Cholesky}\left(K_{\mathbf{X}_I\mathbf{X}_I} + \sigma^2\mathbf{W}_{\mathbf{x}_0}^{-1}\right)$
$\boldsymbol{\alpha} := (\mathbf{L}^{-1})^\top\mathbf{L}^{-1}\mathbf{y}$
$\tilde{m}(\mathbf{x}_0) := K_{\mathbf{x}_0,\mathbf{X}_I}\boldsymbol{\alpha}$
$\mathbf{v} := \mathbf{L}^{-1}K_{\mathbf{x}_0,\mathbf{X}_I}$
$\tilde{\mathcal{K}}(\mathbf{x}_0, \mathbf{x}_0) := K(\mathbf{x}_0, \mathbf{x}_0) - \mathbf{v}^\top\mathbf{v}$

5.3.3 *Local Kernel Ridge Regression*

For every Gaussian process $f(\mathbf{x})$ with covariance function $K(\mathbf{x}, \mathbf{x}')$, there is a unique corresponding Hilbert space $\mathcal{H}_K$. This is commonly referred to as a *reproducing kernel Hilbert space* (RKHS) and constructed as a completion of the linear space of all functions:

$$\mathbf{x} \mapsto \sum_{i=1}^{k} \alpha_i K(\mathbf{a}_i, \mathbf{x}), \quad \alpha_1, \ldots, \alpha_k \in \mathbb{R}, \mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathcal{X}, k \in \mathbb{N}$$

relative to the norm induced by the inner product

$$\left\langle \sum_{i=1}^{k} \alpha_i K\left(\mathbf{s}_i, \cdot\right), \sum_{j=1}^{l} \beta_j K\left(\mathbf{t}_j, \cdot\right) \right\rangle_{\mathcal{H}_K} = \sum_{i=1}^{k} \sum_{j=1}^{l} \alpha_i \beta_j K\left(\mathbf{s}_i, \mathbf{t}_j\right).$$

It is well known that the posterior mean of Gaussian process regression can be alternatively derived by minimizing the regularized empirical risk over the RKHS [67]; see, e.g., [64] for a recent review. For local GPs, this corresponds to a weighted least square minimization over the RKHS with the weights given by $k_h(\mathbf{x}, \mathbf{x}_0)$, that is

$$\tilde{m}(\mathbf{x}) = \arg\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i)\right)^2 k_h\left(\mathbf{x}_i, \mathbf{x}_0\right) + \frac{\sigma^2}{n} \|f\|_{\mathcal{H}_k}^2. \tag{5.6}$$

Note that in the local formulation, for a given point $\mathbf{x}_0$ one has to estimate both the parameters of the reproducing kernel and the width of the local kernel h. Here are two examples of well-known classical local methods, which are the solution to the empirical risk minimization problem (5.6).

K-NEAREST NEIGHBORS This model corresponds to the noise-free case ($\sigma = 0$) with a positive constant reproducing kernel and a rectangular local kernel whose width is adjusted to contain exactly k data points. The solution of the minimization problem (5.6) is the mean of the outputs corresponding to the k closest to $\mathbf{x}_0$ input points.

LOCAL POLYNOMIAL REGRESSION If we use the polynomial kernel $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}\mathbf{x}')^k$ for the space $\mathcal{H}_K$, and use any smooth local kernel (i.e. exponential), then in the noise-free case the solution of the minimization problem (5.6) is so called local polynomial regression [135]. In this special case, when the degree of the polynomial is 0, we have Nadaraya-Watson regression, which is the minimizer of the local squared loss over the constant function.

5.4 EXPERIMENTS

5.4.1 *Toy dataset*

In order to illustrate the behavior of the proposed Locally Smoothed Gaussian Process (LSGPs), we start from a toy dataset generated from the Doppler function (Fig. 5.3).

$$y(x) = \sqrt{x(1-x)} \sin\left(\frac{2.1\pi}{x + 0.05}\right) + \varepsilon, \quad 0 \leq x \leq 1, \quad \varepsilon \sim \mathcal{N}(0, 0.1) \tag{5.7}$$

For this experiment, we used the RBF kernel, and the Epanechnikov localizing kernel

$$k(x) = \frac{3}{4}(1 - |x|^2)\mathbb{I}(|x| \leq 1). \tag{5.8}$$

We tuned the lengthscale parameter of the RBF kernel by optimizing the marginal log-likelihood of the model. We used the L-BFGS algorithm for gradient optimization [106]. We chose the value of the parameter h of the localizing kernel that gave

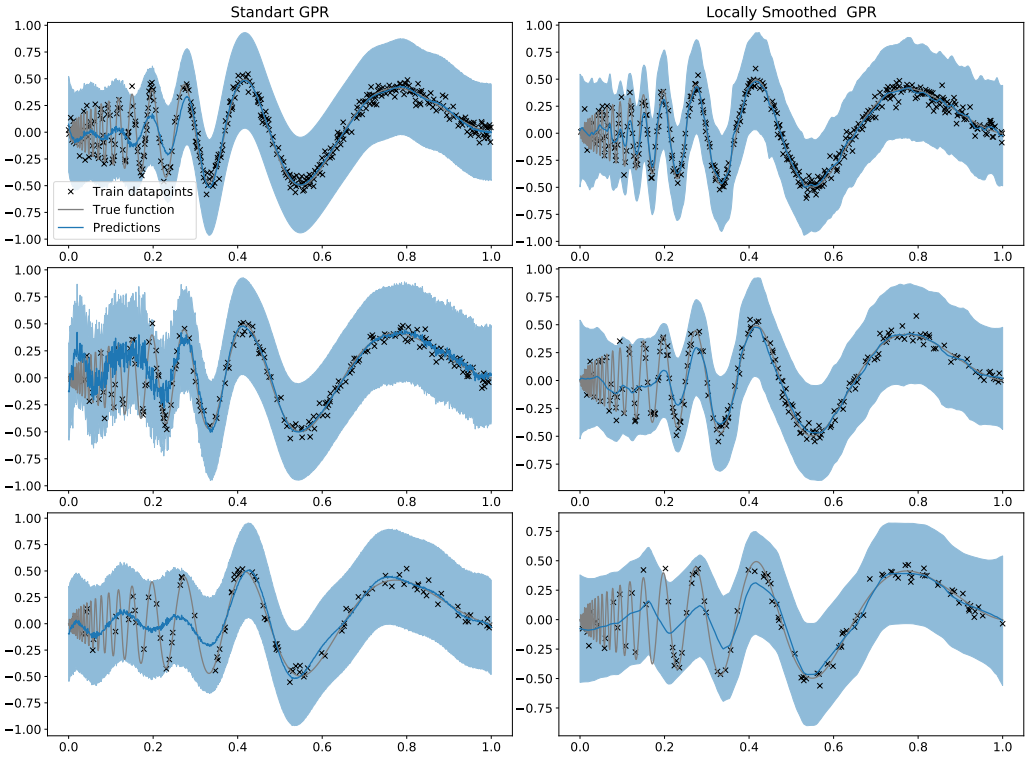

Figure 5.3: Illustration of the predictive distribution of GPs (left) and LSGPs (right) applied to data sampled from the Doppler function for 400 training points (top), 200 training points (middle), and 100 training points.

the best Mean Squared Error (MSE) on a validation set. The LSGP model used on average 7 training points to make a prediction. We compared the predictions of LSGP with the predictions of standard GP regression.

As we can see from Fig. (5.3), the Gaussian Process with RBF kernel is unable to make reasonable predictions in the region where the target function contains high-frequency components. While some nonstationary covariance functions might be appropriate for this example, the combination of a standard stationary covariance function with the localization approach offers substantial modeling improvements.

5.4.2 *UCI datasets*

We evaluated the performance of the LSGP method on several problems from the UCI datasets collection and compared it against standard GPR, Deep GPs approximated with random features [27], and k-nearest neighbors (KNN) regression. In particular, we aim to compare the predictive performance offered by the localization against the baseline of exact GPR, and to verify that any performance gains are not just due to localization, meaning that we expect to outperform KNN.

Table 5.1: UCI datasets used for evaluation.

Dataset	Training instances	Dimensionality
Yacht	308	6
Boston	506	13
Concrete	1030	8
Kin8nm	8192	8
Powerplant	9568	4
Protein	45730	9

Because our model is more flexible than standard GPR, we also added to the comparison Deep GP models based on random features expansion. The size and dimensionality of these problems are outlined in Table 5.1. Since the Euclidean norm used in the local kernels depends on the units in each coordinate, all the datasets except the Protein were scaled within the $[0, 1]$ range. We used a standardization procedure for the Protein dataset because the baseline model worked much better with this type of preprocessing. For the Deep GP model, we used the hyperparameters and the data preprocessing described in the original paper.

The LSGPR method requires creating a new local model with its own set of hyperparameters for each input point where the prediction has to be made. During the optimization, the kernel parameters of each model were constrained to be equal among all local models. Considering the hyperparameter h, we found that it is hard to find values of h that perform well across all regions of the input space. Thus, for each input point of interest, we chose values of h that ensured that the localizer considers at least m neighboring training points. In this experiment, we used 3-fold cross-validation to choose the noise variance σ^2, the lengthscale of the GP kernel, and the parameter m of the localizing kernel. We report the results on the held-out test set.

In this experiment, we also used the Hilbert localizing kernel [32, 124]

$$k(\mathbf{x}) = \|\mathbf{x}\|^{-1}\mathbb{I}(\|\mathbf{x}\| \leq 1), \tag{5.9}$$

which showed good performance for most of the datasets. In Table 5.2, locally smoothed Gaussian Process Regression based on Hilbert kernel is referred to as LSGPR Hilbert, while the same model based on Epanechnikov kernel is referred to as LSGPR Epanechnikov. GP regression, Deep GP regression, and KNN regression are referred to as GP, DeepGP and KNN, respectively.

The results indicate that LSGP offers competitive performance with respect to GPR and Deep GP baselines. The results also clearly show that LSGP offers superior performance to KNN, suggesting that the localization alone is not enough to obtain good performance and that this works well in combination with the GP model. To make a comparison with the baselines, we used the one-sided Wilcoxon test [148]. For each method, we measured its performance on 10 data splits, and we used exactly the same splits for testing the performance of each method, so we matched samples of the MSEs. Then we used the test to compare methods in pairs, where the alternative hypothesis was that the MSE of any given method is smaller

Table 5.2: Comparison in terms of test set MSE between LSGPR a standard GPR.

Dataset	LSGPR Hilbert	LSGPR Epanechnikov	GP	DeepGP	KNN
Yacht	**0.63±0.12**	2.02±0.58	1.09±0.05	0.93±0.13	57.80±16.65
Boston	14.78±0.88	15.30±1.28	17.94±0.71	**7.92±0.14**	23.30±2.58
Concrete	**34.79±1.07**	40.43±3.16	37.81±0.61	130.94±3.93	94.23±7.89
Kin8nm	0.01±0.000	0.01±0.00	0.01±0.000	0.06±0.00	0.01±0.00
Powerplant	14.65±0.34	**14.40±0.67**	16.85±0.69	14.57±0.15	15.35±0.49
Protein	36.85±2.15	**12.50±0.26**	17.03±0.57	16.94 ±0.16	19.89±4.26

than the MSE of competitors. We used confidence level $\alpha = 0.05$. In Table 5.2, the results that are statistically better than the competitors are marked in bold.

5.5 DISCUSSION OF THE METHOD

In this work, we developed a novel framework to localize Gaussian processes (GPs). We focused, in particular, on Gaussian Process Regression (GPR), and we derived the GP model after applying the localization operation through the down-weighting of contributions from input points that are far away from a given test point. The form of the localized GP maintains positive definiteness of the covariance, and it allows for considerable speedups compared to standard global GPR due to the sparsification effect of the Gram matrix.

The proposed method may suffer from the curse of dimensionality. In this work we chose the localization hyperparameter to ensure, that the local model considers at least m neighboring training points. This is a reasonable choice for low-dimensional problems, but it may not be sufficient for high-dimensional problems.

The proposed method requires cross-validation to tune the scale parameter of the localizing kernel, while other GP-based techniques use a less expensive marginal log-likelihood (MLL) gradient optimization to tune these types of parameters. We found MLL gradient optimization problematic because of the discontinuity of the local kernel with respect to the scale parameter, which in turn makes the MLL function discontinuous with respect to this parameter. It would be interesting to investigate ways to extend the idea of localization for GPR to other tasks, such as classification.

6

IMPOSING FUNCTIONAL PRIORS ON BAYESIAN NEURAL NETWORKS

6.1 INTRODUCTION

Artificial Neural Networks (NN) currently represent a general class of successful models for various machine learning tasks, including computer vision, natural language processing, and many others. Bayesian Neural Networks (BNN) combine the representation power of NNs with Bayesian inference, making them an attractive choice in applications where predictive performance and accurate uncertainty quantification in parameter estimates and predictions are both important. However, BNNs are typically difficult to work with due to the intractability of the posterior over model parameters, which requires one to resort to approximations. Furthermore, recent works point to the need to choose sensible priors over model parameters in order to be able to obtain good performance [41, 134]. In BNNs, the prior over the weights and the network architecture determine a distribution over the outputs of such BNNs [130], and we refer to this induced prior as a *functional prior*.

The functional prior should encode any prior information on the conditional distribution of the labels given the inputs. However, it is unclear how to encode this type of information when having to specify a prior distribution over the weights.

In this paper, we propose a novel framework to enforce such meaningful functional priors. In particular, we rely on scalable Markov chain Monte Carlo (MCMC) sampling from an approximation to the posterior distribution over BNN weights, and we specify the prior over the weights implicitly through a prior over the induced functional prior. Our approach is different from the literature on Implicit Process Priors (IPPs) [81], where the goal is to obtain an approximate framework to handle the functional prior implicitly induced by choice of a prior distribution over the weights. In our work, we operate in the opposite direction by *imposing* a functional prior, which implicitly determines a prior over the weights; we do not know such a prior over the weights in closed form, but we implicitly determine it through the specification of the induced functional prior.

Stochastic Processes are natural mathematical objects suitable to define distributions over functions [63], and GPs represent popular examples that are routinely used in numerous machine learning tasks. This type of stochastic process is well investigated and has strong theoretical foundations [114]. There are theoretical guarantees for the generalization error of GP regression, and this method has a strong connection with non-Bayesian Kernel Ridge Regression (KRRs) [64]. Also, it was shown in [93] that in the infinite width limit, shallow BNNs are equivalent to GPs. We propose to use GPs to impose functional priors over BNNs because GPs provide a flexible set of instruments for encoding different types of beliefs about functions, such as periodicity or smoothness, through the specification of kernels.

However, our approach is not restricted to GPs, and it can handle any functional priors that can be written down in the closed form.

6.2 A REVIEW OF TYPES OF PRIOR IN BNN

A popular way of choosing prior distributions for BNNs is to employ a Gaussian distribution over the weight of the model [50, 93]. While this choice of prior does not render inference tractable, it does offer some practical advantages, for instance, when employing Variational Inference (VI) [50]. In this case, it is possible to calculate part of the VI objective without the need to resort to Monte Carlo sampling, and it requires a linear complexity in the number of parameters when making a mean-field assumption. This computational benefit comes with a cost of poor approximation of the true posterior due to the lack of flexibility of the family of the approximate distributions.

Even when adopting more advanced and generally more accurate inference techniques, such as Hamiltonian Monte Carlo (HMC) [94], and Stochastic-Gradient HMC (SGHMC) [22], the Gaussian assumption on the prior over model parameters is still very common. By studying the entropy of the predictive distribution, it was shown by [42] that Gaussian priors are problematic in terms of model performance and the ability to detect Out-of-Domain (OOD) input examples. This work also shows how Gaussian priors over the weights could be responsible for the cold posterior effect described by [147]; this effect is characterized by the necessity of applying temperature scaling to the prior density term in Bayes theorem in order to obtain good performance.

The poor performance associated with the choice of Gaussian priors gave rise to alternatives involving flexible distributions, such as a mixture of Gaussians [10], Student's t-distribution [42], hierarchical Gaussian distribution [22], and many others [41]. However, all these types of prior distributions imposed over the weights of the model share the same issue of preventing from understanding the effect that they impose on the outputs of the model.

Another way of encoding the desired output prior constraints into the model is to use an auxiliary probabilistic model which learns the desired distribution of parameters to fulfill the required output constraints and to use this distribution as a prior over the weights of the original model [4].

Instead of focusing on the prior over the weights of NNs, an alternative is to study their effect on the distribution on the NNs output, which we refer to as functional priors. [130] proposes to use VI to tackle the problem of finding a Bayesian posterior in the space of functions for a functional prior, defined by a stochastic process. They introduce a variational objective, which includes the evaluation of the KL divergence between the approximate predictive posterior and the functional prior using a finite set of evaluations of the function. It was shown that the supremum of the KL divergence over all possible sets of input points is equal to the true KL divergence in functional space. In this setting, the model is then trained in an adversarial manner, which means that the optimization procedure simultaneously minimizes the optimization objective with respect to the parameters of the model and maximizes the KL term with respect to the input data points, which makes the optimization process unstable. Also, the opti-

mization objective requires evaluating the gradient of the approximate posterior density, which is performed with the Stein gradient estimator [125], and this, in turn, requires a careful choice of a kernel function. The work in [81] focuses on representing the functional prior as a BNN and uses GPs to obtain an approximate posterior over functions. The problem with this approach is that GPs may yield a poor approximation quality for the true functional posterior. The authors in [130] and [81] use VI to find an approximate posterior distribution, which means that the optimization objective contains a functional KL divergence term. However, in [15, 117], it is claimed that the KL divergence between the functional approximate posterior and the GP process functional prior is problematic as it may go to infinity. On the other hand, they acknowledge that it does not mean that parametric models cannot approximate GPs well.

The authors of [134] propose to impose GP functional priors so as to constrain the parametric prior over the weights of BNNs. They propose several options to define such a prior, including Gaussian, Hierarchical Gaussian, and Normalizing Flows [115]. They propose to optimize the parameters of the prior over the weights so as to minimize the Wasserstein distance between the outputs of the untrained BNN and samples from the GP prior. Then, the posterior over the weights is characterized by means of MCMC.

In our work, we aim to avoid the computation of the KL divergence or any other distance metric in functional space. Instead, we propose to enforce the choice of a functional prior directly when carrying out approximate inference of BNN weights.

6.3 IMPOSING FUNCTIONAL PRIORS ON BNNS

Consider a supervised learning task with a dataset $\mathcal{D}\{(\mathbf{x}_i, y_i)\}_{i=1\ldots n}$ of n input vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1\ldots,n}$ and corresponding labels $\mathbf{y} = \{y_i\}_{i=1\ldots,n}$, and imagine employing a NN-based model with parameters $\mathbf{w}$ to establish a parametric mapping between inputs and labels. We denote the input/output mapping by $f_{\mathbf{w}}(\mathbf{x})$, and for convenience we also define $\mathbf{f}^\top = [f_{\mathbf{w}}(\mathbf{x}_1), \ldots, f_{\mathbf{w}}(\mathbf{x}_N)]$ and $\mathbf{f}^{*\top} = [f_{\mathbf{w}}(\mathbf{x}_1), \ldots, f_{\mathbf{w}}(\mathbf{x}_N), f_{\mathbf{w}}(\tilde{\mathbf{x}}_1), \ldots, f_{\mathbf{w}}(\tilde{\mathbf{x}}_M)]$ as the evaluation of the function $f_{\mathbf{w}}(\mathbf{x})$ at the inputs $\mathbf{X}$ and an augmented set of inputs $\mathbf{X}^* = [\mathbf{X}, \tilde{\mathbf{X}}]$, respectively. The set $\mathbf{X}^*$ has cardinality $N^* = N + \tilde{N}$, and the $\tilde{N}$ inputs in $\tilde{\mathbf{X}}$ are drawn from a given $p(\mathbf{x})$. Note that the sets $\mathbf{X}$ and $\mathbf{X}^*$ can be disjoint, but in order to keep the notation uncluttered, we assume $\mathbf{X} \subset \mathbf{X}^*$

A Bayesian treatment NNs requires specifying a prior distribution $p(\mathbf{w})$ over the parameters and a likelihood function for the labels given the inputs, that is $p(\mathbf{y}|\mathbf{X}, \mathbf{w})$. For this BNN, it is possible to write down an expression for the posterior distribution over model parameters as:

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{\int p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})d\mathbf{w}} \tag{6.1}$$

Carrying out inference in BNNs is extremely difficult for at least two reasons. One main difficulty stems from the complex way in which parameters affect the likelihood function, and this requires approximation techniques to characterize the posterior over model parameters; popular approaches involve MCMC and

variational approximations. A second and more subtle challenge is how to specify priors for BNNs because it is difficult to establish what is the effect of prior parameters on the distribution over the functions that BNNs can represent. In this work, we propose a novel way to address the challenge of choosing sensible priors for BNNs by working with implicit priors over the weights induced by the choice of functional priors while we follow the recent trend to employ MCMC techniques to address the intractability of the inference process. We begin by focusing on the distribution over the functions represented by BNNs. In particular, we consider the distribution of $\mathbf{f}^*$, which is the distribution of $f_{\mathbf{w}}(\mathbf{x})$ evaluated at the set of input points $\mathbf{X}^*$, and we impose a prior over this set of variables which encourages functions to behave in a sensible way *a priori*. Later we will study in particular Gaussian process priors, but any functional prior can be incorporated as long as it can be expressed in closed form.

We now rewrite the likelihood function in terms of $\mathbf{f}$ rather than $\mathbf{w}$:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \rightarrow p(\mathbf{y}|\mathbf{f}). \tag{6.2}$$

The main idea behind our work is to now define a prior over $\mathbf{f}$ instead of $\mathbf{w}$, and to perform inference over $\mathbf{w}$. With this change of variables, we should account for the change of measure through a Jacobian term. However, such a change of variables involves groups of variables of different dimensions in general, and even when this is not the case, computing this term would be computationally costly. For this reason, we are going to ignore the Jacobian accepting to settle for an approximate posterior over $\mathbf{w}$. With this choice, we rewrite Bayes theorem as:

$$\log p(\mathbf{f}^*|\mathbf{y}, \mathbf{X}^*) = \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}^*|\mathbf{X}^*) + \text{const.} \tag{6.3}$$

Note that in this equation, we introduced the functional prior:

$$p(\mathbf{f}^*|\mathbf{X}^*) = \int p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{w}) p(\mathbf{w}) d\mathbf{w}, \tag{6.4}$$

where $p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{w})$ is a Dirac's delta placed at the evaluation of $f_{\mathbf{w}}(\mathbf{x})$ at the inputs $\mathbf{X}^*$ due to the deterministic way in which inputs are mapped into outputs in NNs. Again, we stress that while we focus on the distribution of functions represented by BNNs, we actually use the objective in eq. 6.3 to perform MCMC sampling in the space of the weights $\mathbf{w}$. Note that we carry out inference over $\mathbf{w}$ through MCMC, but given that we are working with an approximation to the posterior over $\mathbf{w}$, we could alternatively employ other fast approximate inference techniques such as Variational Inference. Here, we focus on MCMC so as to isolate the effect of the way we impose functional priors compared with alternatives that try to characterize the exact posterior over $\mathbf{w}$ [134].

BAYESIAN INTERPRETATION. From a Bayesian point of view, imposing a prior over function by specifying a prior over $\mathbf{f}^*$ induces an implicit prior over the weights through eq. 6.4. In other words, the prior over $\mathbf{f}^*$ is, in practice, a prior over a deterministic transformation of $\mathbf{w}$, and this is implemented by the NN. It is interesting to note that in the literature eq. 6.4 is usually interpreted in the opposite way; that is, one uses eq. 6.4 starting from a prior over the weights $p(\mathbf{w})$ to define a functional prior in an implicit way [81]. The likelihood function establishes what

is the likelihood of the labels $\mathbf{y}$, and it is conditioned on $\mathbf{f}$ or equivalently on $\mathbf{w}$ and $\mathbf{X}$. Therefore, the expression in eq. 6.3 can be seen as an expression for the (approximate) posterior over the weights $\mathbf{w}$ (due to the lack of a Jacobian term), where the prior is assumed over a transformation of such weights. In this paper, we take this view to carry out Bayesian inference over $\mathbf{w}$ using MCMC techniques. We also note that our approach has some close similarity with the Product of Expert approach proposed in [146] for inference of parameters of Ordinary Differential Equations using Gaussian Processes.

REGULARIZATION INTERPRETATION. While we proceed with a Bayesian treatment of $\mathbf{w}$, it is useful to interpret eq. 6.3 as a regularized objective in the following way. The first term $\log p(\mathbf{y}|\mathbf{f})$ is the negative loss, which can be equivalently seen as a function of $\mathbf{w}$ and $\mathbf{X}^*$, so this provides a constraint on $\mathbf{w}$ because the objective promotes values of $\mathbf{f}$ which are compatible with the labels $\mathbf{y}$, and $\mathbf{f}$ depends on $\mathbf{w}$ and $\mathbf{X}^*$. The second term is a regularization term, which penalizes functions deviating from a behavior established by the functional prior. Because $\mathbf{f}^*$ is a function of $\mathbf{w}$ and $\mathbf{X}^*$, this translates into a regularization term for $\mathbf{w}$.

6.4 GAUSSIAN PROCESS PRIORS

The proposed formulation focusing on functional representations has the advantage of putting the emphasis on the functions that BNNs can represent, and for which it is possible to assume sensible priors. Here we specify how to operate in the case of Gaussian processes (GPs), which yield a prior term in eq. 6.3 as:

$$\log p(\mathbf{f}^*|\mathbf{X}^*) = -\frac{1}{2}\mathbf{f}^{*\top}\mathbf{C}^{-1}\mathbf{f}^* + \text{const}, \tag{6.5}$$

where the covariance matrix is $\mathbf{C} = (\mathbf{K}_{\mathbf{X}^*\mathbf{X}^*} + \sigma_n^2)$, and $\mathbf{K}_{\mathbf{X}^*\mathbf{X}^*}$ contains the evaluation of the kernel function κ among all the inputs in $\mathbf{X}^*$. For simplicity, we assumed a zero-mean GP, but other mean functions can be easily included.

In the next subsections, we elaborate on how to use this GP prior in practice by proposing a way to operate with mini-batches for scalability purposes, by discussing hyper-parameter optimization, and by discussing the properties of the proposed approach when N^* goes to infinity.

6.4.1 *Mini-batching*

In this work, we aim to employ advanced MCMC sampling methods based on stochastic gradients, and in particular Stochastic Gradient Hamiltonian Monte Carlo (SG-HMC) [22] to sample from the weights $\mathbf{w}$ of BNNs. In order to do so, we need to formulate our MCMC objective in a way that is suitable for mini-batching. However, extending the previous formulation to operate with mini-batches without care would produce a biased estimation of the quadratic term $\mathbf{f}^{*\top}\mathbf{C}^{-1}\mathbf{f}^* \neq \mathbb{E}[\mathbf{f}_b^{*\top}\mathbf{C}_b^{-1}\mathbf{f}_b^*]$, where $\mathbf{f}_b$ and $\mathbf{C}_b$ are computed over a mini-batch $\mathbf{X}_b$.

The main difficulty of full batch training is the necessity of solving linear systems with the matrix $\mathbf{C}$, which has $\mathcal{O}(N^{*3})$ complexity in the number of

inputs in $\mathbf{X}^*$. The literature on GPs offers many cues on how to circumvent this problem. In particular, there exist formulations of GPs based on inducing points [57] and random features [80, 110] which operate on mini-batches [27]. In this work, we focus on approximations based on random features, but inducing points formulations is also possible.

The Random Features (RF) expansion of the kernel $\kappa(\cdot,\cdot)$ allows us to obtain a finite-dimensional representation for an explicit feature map that approximates its value. Using this expansion, we can express the Gram matrix as a dot product of feature maps computed over the data $\mathbf{K} \approx \mathbf{\Phi}\mathbf{\Phi}^\top$. We can use this property and the Woodbury identity to rewrite the quadratic term as follows:

$$\mathbf{f}^{*\top}\mathbf{C}^{-1}\mathbf{f}^* = \mathbf{f}^{*\top}(\mathbf{\Phi}\mathbf{\Phi}^\top + \sigma_f^2\mathbf{I})^{-1}\mathbf{f}^* = \\ \frac{1}{\sigma_f^2}\mathbf{f}^{*\top}\mathbf{f}^* - \frac{1}{\sigma_f^2}\mathbf{f}^{*\top}\mathbf{\Phi}(\mathbf{\Phi}^\top\mathbf{\Phi} + \sigma_f^2\mathbf{I})^{-1}\mathbf{\Phi}^\top\mathbf{f}^*. \tag{6.6}$$

In this case, instead of inverting the matrix of size $N^* \times N^*$, we invert the matrix of size $D \times D$, where D is the dimensionality of the RF vector. But this approach has two drawbacks. First, it is extremely unstable when $\sigma_f^2 \to 0$, because after application of the Woodbury identity the term $\frac{1}{\sigma_f^2}\mathbf{f}^{*\top}\mathbf{f}^* \to \infty$. Second, this approach still does not allow mini-batch training.

We can reformulate our MCMC objective by replacing the nonparametric term pertaining to the GP with a parametric one based on RFs. For the set $\mathbf{f}^*$, we can factorize its prior probability as:

$$p(\mathbf{f}^*|\mathbf{X}^*) = \int p(\mathbf{f}^*|\boldsymbol{\beta}, \mathbf{X}^*)p(\boldsymbol{\beta})d\boldsymbol{\beta}, \tag{6.7}$$

where $\boldsymbol{\beta}$ are the parameters of RF approximation of the GP, that is $p(\boldsymbol{\beta}) \sim \mathcal{N}(0, \mathbf{I})$ and $p(\mathbf{f}^*|\boldsymbol{\beta}, \mathbf{X}^*) \sim \mathcal{N}(\mathbf{\Phi}\boldsymbol{\beta}, \sigma_f^2\mathbf{I})$. In this case it is easy to verify that $p(\mathbf{f}^*) = \mathcal{N}(0, \mathbf{\Phi}\mathbf{\Phi}^\top + \sigma_f^2\mathbf{I})$ and according to the property of the RF approximation, the covariance matrix coincides with the prior term of the objective in eq. 6.6. Instead of sampling directly from the unnormalized posterior $p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{y})$ marginalized over $\boldsymbol{\beta}$, we can sample from the joint density $p(\mathbf{f}^*, \boldsymbol{\beta}|\mathbf{X}, \mathbf{y})$ and discard samples over $\boldsymbol{\beta}$:

$$p(\mathbf{f}^*, \boldsymbol{\beta}|\mathbf{X}^*, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}^*|\boldsymbol{\beta}, \mathbf{X}^*)p(\boldsymbol{\beta}). \tag{6.8}$$

Again, when we refer to the fact that we sample $\mathbf{f}^*$, in practice, we sample $\mathbf{w}$. This RF-based approach avoids the necessity of inverting the matrix $(\mathbf{\Phi}\mathbf{\Phi}^\top + \sigma_f^2\mathbf{I})$ during the computation of the objective.

Resuming, the expression for the unnormalized log-posterior in eq. 6.3, where the GP regularization is approximated using RFs, is as follows:

$$\log \hat{p}(\mathbf{f}^*, \boldsymbol{\beta}|\mathbf{X}^*, \mathbf{y}) = \log p(\mathbf{y}|\mathbf{f}) - \\ \frac{1}{2\sigma_f^2}\|\mathbf{f}^* - \mathbf{\Phi}\boldsymbol{\beta})\|^2 - \frac{||\boldsymbol{\beta}||^2}{2} + \text{const.} \tag{6.9}$$

It is straightforward to verify that this MCMC objective can be written as a sum of terms involving individual input points, and it is therefore, amenable to mini-batching. It is also easy to verify that one can proceed with a Gibbs sampling

scheme whereby $\mathbf{f}^*$ (that is $\mathbf{w}$) is sampled from the conditional $\hat{p}(\mathbf{f}^*|\boldsymbol{\beta}, \mathbf{X}^*, \mathbf{y})$ using SG-HMC and $\boldsymbol{\beta}$ is sampled directly from $\hat{p}(\boldsymbol{\beta}|\mathbf{f}^*, \mathbf{X}^*, \mathbf{y})$, which has a Gaussian form.

6.4.2 *Hyper-parameter optimization*

The choice of a GP prior opens to the need to specify its kernel parameters. In the absence of any way to determine such hyper-parameters, we propose to optimize them by marginal log-likelihood (MLL) optimization, which is a popular way to proceed with GP models. In our case, the random feature approximation lends itself to a scalable solution, avoiding the need to invert large matrices. In particular, the marginal likelihood is:

$$p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \boldsymbol{\Phi}\boldsymbol{\Phi}^\top). \tag{6.10}$$

Again, using Woodbury matrix identities, it is possible to rewrite the marginal likelihood so that the cost of computing it is cubic in the number of random features instead of cubic in the number of input points.

6.4.3 *Classification*

While for regression, it is natural to specify functional priors through GPs and to obtain a tractable framework to scale these through random features, for other likelihoods, things may become more involved. For instance, in classification problems, we may wish to specify functional priors such that the distribution over classes is uniform *a priori*.

Alternatively, following an empirical Bayes approach, we could optimize the GP prior hyper-parameters so as to maximize the marginal likelihood. In this case, the random feature approximation of GPs leads to so-called Generalized Linear Models (GLMs) and this requires approximations to be able to compute the marginal likelihood. For classification tasks, there exist solutions to bypass the need to work directly with Bernoulli or Multinoulli likelihoods $p(\mathbf{y}|\mathbf{w}, \mathbf{X})$. Here we follow the idea proposed by [87], in which labels are transformed so that classification models can be replaced by regression models with heteroscedastic observation noise. In particular, for each one-hot encoded label $\mathbf{y}$ we can obtain real-valued vectors $\tilde{\mathbf{y}}, \sigma_n^2$ (see [87] for details):

$$\tilde{y}_i = \log(\alpha_i) - \frac{\tilde{\sigma}_i^2}{2}; \qquad \tilde{\sigma}_i^2 = \log\left(\frac{1}{\alpha_i} + 1\right). \tag{6.11}$$

In this case, we can use a Gaussian likelihood, and thus we can obtain a closed-form solution for the marginal likelihood of the model using the fact that the Gaussian likelihood and Gaussian prior are conjugate.

6.4.4 *Connections with Kernel Ridge Regression*

As already pointed out, in the proposed approach, we are free to choose the set $\mathbf{X}^*$. In this section, we try to get insights as to what is the behavior of the

Gaussian process functional prior as a function of the cardinality of $\mathbf{X}^*$, that is N^*. In particular, we aim to show that the term associated with the functional prior in eq. 6.3, which is the quadratic form $\mathbf{f}^{*\top}\mathbf{C}^{-1}\mathbf{f}^*$, does not grow indefinitely with N^*. To do so, we resort to some connections with Kernel Ridge Regression. According to [149] we can express

$$\begin{aligned}&\mathbf{f}^{*\top}\mathbf{C}^{-1}\mathbf{f}^* = \\ &\min_{g\in\mathcal{H}} \frac{1}{\sigma_f^2}\sum_{i=1}^{N^*}(f_{\mathbf{w}}(\mathbf{x}_i^*) - g(\mathbf{x}_i^*))^2 + ||g||_{\mathcal{H}}^2.\end{aligned} \tag{6.12}$$

In the case when $f_{\mathbf{w}}$ belongs to the RKHS induced by the kernel κ of the GP prior, we can easily see that the quadratic term on the LHS of (6.12) is bounded by $\|f\|_{\mathcal{H}}^2$.

Indeed, if $\hat{g}$ is a unique minimum of the RHS of eq. 6.12, we can write

$$\begin{aligned}&\frac{1}{\sigma_f^2}\sum_{i=1}^{N^*}(f_{\mathbf{w}}(\mathbf{x}_i^*) - \hat{g}(\mathbf{x}_i^*))^2 + ||\hat{g}||_{\mathcal{H}}^2 \leq \\ &\frac{1}{\sigma_f^2}\sum_{i=1}^{N^*}(f_{\mathbf{w}}(\mathbf{x}_i^*) - f_{\mathbf{w}}(\mathbf{x}_i^*))^2 + ||f_{\mathbf{w}}||_{\mathcal{H}}^2 = ||f_{\mathbf{w}}||_{\mathcal{H}}^2.\end{aligned} \tag{6.13}$$

This means that if $||f_{\mathbf{w}}||_{\mathcal{H}}^2$ is bounded, the quadratic term $\mathbf{f}^{*\top}\mathbf{C}^{-1}\mathbf{f}^*$ is bounded as well.

As an illustration of this fact, we picked a 1D function that has a following form:

$$f(\cdot) = \sum_i^M \alpha_i \kappa(\cdot, x_i). \tag{6.14}$$

We used $M = 20$, $x_i \sim \text{Unifrom}(-1, 1)$, $\alpha_i \sim \text{Unifrom}(-1, 1)$. For this function, we can compute the Hilbert norm analytically. Then, for this function, we evaluated the value of the quadratic term $\mathbf{f}^{*\top}\mathbf{C}^{-1}\mathbf{f}^*$ for different amounts of points x_i (Fig. 6.1).

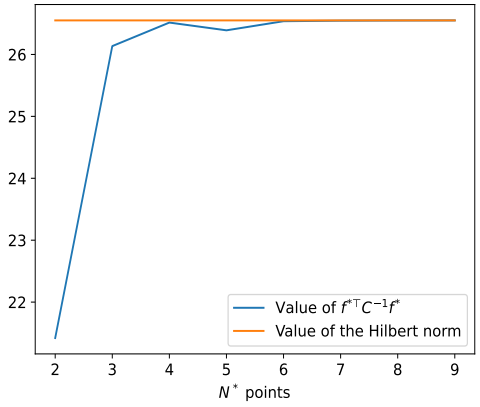

Figure 6.1: The value of the quadratic regularization term with respect to the number of points used for evaluation of the function.

We can also reuse a result from [149], which allows us to express the projection of $f_{\mathbf{w}} \in \mathcal{H}$ on a subset of $\mathcal{H}$, $\mathcal{H}' \subset \mathcal{H}$. And $\mathcal{H}'$ is a linear span of functions $\phi_i(\cdot) = k(\cdot, \mathbf{x}_i)$, $i \in \{1, \ldots, N^*\}$.

$$\mathbb{P}(f_{\mathbf{w}}) = \arg\min_{g \in \mathcal{H}'} ||f_{\mathbf{w}} - g||^2_{\mathcal{H}} = k_{\mathbf{X}^*}(\cdot)\mathbf{K}^{-1}_{\mathbf{X}^*\mathbf{X}^*}\mathbf{f}^*, \tag{6.15}$$

and we can note that

$$\|\mathbb{P}(f_{\mathbf{w}})\|^2_{\mathcal{H}'} = \mathbf{f}^{*\top}\mathbf{K}^{-1}_{\mathbf{X}^*\mathbf{X}^*}\mathbf{f}^*. \tag{6.16}$$

By increasing the number of basis functions in $\mathcal{H}'$ to infinity we get

$$\min_{g \in \mathcal{H}'} ||f_{\mathbf{w}} - g||^2_{\mathcal{H}} = ||f_{\mathbf{w}} - P(f)||^2_{\mathcal{H}} \xrightarrow{N \to \infty} 0. \tag{6.17}$$

Using the expression for $\mathbb{P}(f)$, we obtain:

$$\begin{aligned} &||f_{\mathbf{w}} - \mathbb{P}(f)||^2_{\mathcal{H}} = \\ &||f_{\mathbf{w}}||^2_{\mathcal{H}} + \mathbf{f}^{*\top}\mathbf{K}^{-1}_{\mathbf{X}^*\mathbf{X}^*}\mathbf{f}^* - 2\langle f_{\mathbf{w}}, k_X(\cdot)\mathbf{K}^{-1}_{\mathbf{X}^*\mathbf{X}^*}\mathbf{f}^*\rangle = \\ &||f_{\mathbf{w}}||^2_{\mathcal{H}} - \mathbf{f}^{*\top}\mathbf{K}^{-1}_{\mathbf{X}^*\mathbf{X}^*}\mathbf{f}^*. \end{aligned} \tag{6.18}$$

In the last step, we used the reproducing property of the RKHS $\mathcal{H}$. Substituting this result into (6.17) we obtain:

$$\mathbf{f}^{\top}\mathbf{K}^{-1}_{\mathbf{X}^*\mathbf{X}^*}\mathbf{f} \xrightarrow{N \to \infty} ||f_{\mathbf{w}}||^2_{\mathcal{H}}. \tag{6.19}$$

In cases where the function $f_{\mathbf{w}}$ is represented by NNs with smooth and bounded activation functions, like *tanh* or *sigmoid*, at the layer before output, the output function of the NN $f_{\mathbf{w}}$ is smooth and bounded for any bounded $\mathbf{w}$. This is because a linear combination of smooth and bounded functions with bounded coefficients is smooth and bounded too. This means that if we are using kernels that have universal approximation properties, like RBF, exponential or binomial kernels [129], each function generated by such NNs can be modeled by functions belonging to the RKHS imposed by this kernel.

With a finite amount of points in $\mathbf{X}^*$, the overall expression for the unnormalized log-posterior becomes:

$$\begin{aligned} &\log \hat{p}(\mathbf{f}|\mathbf{y}) = \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}||\hat{g}||^2_{\mathcal{H}} + \text{const}, \\ &\hat{g} = \arg\min_{g \in \mathcal{H}'} ||f_{\mathbf{w}} - g||^2_{\mathcal{H}}. \end{aligned} \tag{6.20}$$

This means that, in fact, we are using an approximation of the correct unnormalized log-posterior, but still, this objective guarantees that for the points in $\mathbf{X}^*$, we will obtain evaluations of functions sampled from the true posterior.

6.5 EXPERIMENTAL EVALUATION OF MCMC FOR GP PRIORS

6.5.1 *Toy regression dataset*

We begin by testing our approach on a synthetic 1D dataset. We employ a two-hidden layer NN with *tanh* activation function and with 256 neurons at each layer.

For the functional GP prior, we use an RBF kernel with a length-scale parameter $l = 1$ and output variance $\sigma^2_{out} = 1$. In Fig. 6.2, we report functions sampled from the predictive posterior of the BNN with such a GP prior (GP in the figure) along with the same GP prior approximated with 100 random features with and without mini-batching (GP RFF and GP RFF mini-batch in the figure). As a regularization set $\tilde{\mathbf{X}}$, we used a grid of 200 equally spaced points. These points were used as a test set. For comparison, we also include the approach proposed in [134] where the prior over the weights is optimized so that the functional prior of the BNN minimizes the Wasserstein distance to the GP prior (WDGPi-G in the figure).

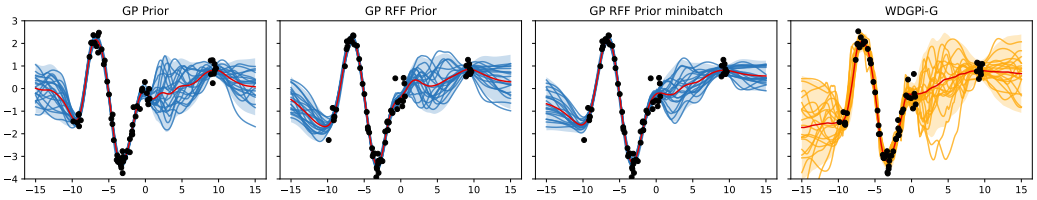

Figure 6.2: Sampled predictions of BNNs where the GP functional prior is imposed implicitly (our work) and by means of the optimization of the Wasserstein distance with the functional BNN prior (WDGPi-G).

6.5.2 *UCI regression datasets*

To evaluate the effectiveness of our approach on more challenging problems, we consider a suite of UCI datasets [34]. For these tests, we use a two-hidden layer NN with *tanh* activation function and with 100 neurons at each layer for all data sets, except for the Protein dataset, for which we used 200 neurons. We imposed a GP prior with an RBF kernel. Input vectors and labels were initially standardized. For all experiments, we used the extended dataset $\mathbf{X}^*$, which contained 90% of real trained data and 10% of uniformly sampled vectors from the input domain. The uniform distribution has support bounded at each dimension by minimum and maximum values of the training samples over this dimension. For all the datasets, we used full-batch training, while for Kin8nm, Power, and Protein, we used mini-batch training with a batch-size size of 512.

As a baseline, we consider the method proposed in [134], and we choose the best-performing version from their paper, which is called GPi-G. We denote this as WDGPi-G here to emphasize the fact that this approach optimizes the Wasserstein distance of the BNN functional prior to the GP prior to determine the prior over the BNN weights. This method uses a Gaussian distribution as a parametric form for the prior over the weights and a Hierarchical GP with a LogNormal distribution over the length-scale and output variance of the GP kernel. The comparison of our method and WDGPi-G in terms of RMSE is shown in Table 6.1. We also report comparisons with deep ensembles [73], where each model in the ensemble had the same architecture as the NN used in the proposed method.

From these results, we see that our method outperforms WDGPi-G for most of the datasets. It is particularly interesting to note that WDGPi-G uses a form of Hierarchical Gaussian Process as a functional prior, whereas we consider a simple

Table 6.1: Average RMSE for UCI regression datasets

Dataset	Functional MCMC	WDGPi-G	Deep Ensembles
Boston	2.73±0.02	2.83±0.92	3.69±1.15
Concrete	4.06±0.12	4.80±0.41	3.07±0.26
Energy	0.48±0.18	0.34±0.07	1.37±0.32
Kin8nm	0.04±0.00	0.06±0.00	0.06±0.00
Power	3.24±0.06	3.72±0.18	3.86±0.21
Protein	3.61 ±0.04	3.65±0.02	4.45±0.02
Wine	0.60±0.01	0.60±0.04	0.62±0.02

Table 6.2: MNLL for UCI regression datasets

Dataset	Functional MCMC	WDGPi-G	Deep Ensembles
Boston	2.45±0.01	2.48±0.12	3.19±1.12
Concrete	2.74±0.16	3.03±0.05	3.07±0.26
Energy	0.80±0.05	0.35±0.15	2.07±0.98
Kin8nm	-1.46±0.11	-1.23±0.01	-1.32±0.08
Power	2.73±0.08	2.74±0.04	2.74±0.05
Protein	2.73±0.01	2.75±0.00	2.80±0.01
Wine	0.76±0.04	0.92±0.06	1.08±0.20

GP in our case. Hierarchical GPs represent a richer functional prior than standard GPs used in our method, but we still obtain competitive performance.

6.5.3 *Toy classification dataset*

We now illustrate how the proposed approach behaves on a 2D toy example involving the so-called banana dataset. Again we used a two-hidden layer NN with *tanh* activation function and with 256 neurons at each layer. We transform the labels using the approach from [87] to obtain labels which allow us to employ a Gaussian likelihood, as explained in Sec. 3. We then use an RBF kernel $\sigma_{\text{out}} = 5$ with different values for the lengthscale and show how these choices affect the final results. As a regularization set $\tilde{\mathbf{X}}$, we used a grid of 40×40 equally spaced points. Also, these points were used as a test set. For comparison, we report results with the WDGPi-G approach from [134]. As we can see on the plot, WDGPi-G method fails to incorporate the GP prior for a smaller lengthscale (l=0.1) and the prediction function behaves more smoothly than it should.

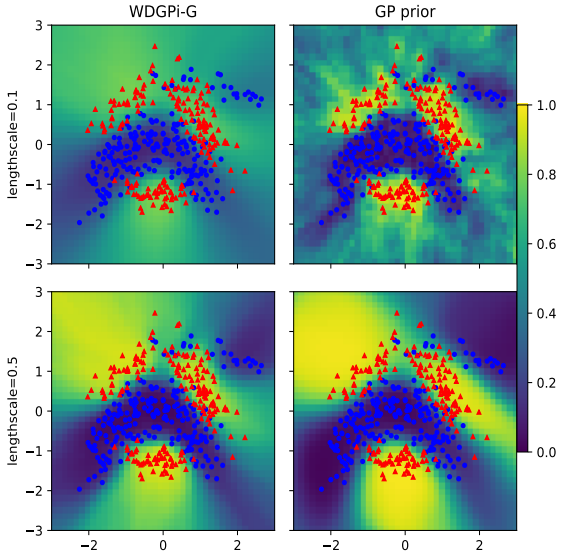

Figure 6.3: Sampled predictions of the neural network with GP prior using Mahalanobis regularization and WDGPi-G methods

6.5.4 *UCI classification datasets*

In this section, we test the effectiveness of our approach on various UCI classification datasets. We use a two-hidden layer NN with *tanh* activation function. For the EEG, HTRU2, Letter, and Magic, we use 100 neurons in each hidden layer, while we use 200 neurons for Miniboo, Drive, and Mocap. Because all the datasets are rather large, in all cases, we use the Random Feature approximation of the functional GP prior with $D = 1000$ random features and mini-batches of size 512. GP hyper-parameters are optimized using the marginal likelihood after the label transformation presented in Sec. 3. We found that using the same transformation for the classification task with the BNN itself gave slightly better results than using

Table 6.3: Average classification accuracy for UCI classification datasets

Dataset	Functional MCMC	WDGPi-G	Deep Ensembles
EEG	92.51±1.82	94.13±1.96	89.04 ± 5.01
HTRU2	98.10±0.26	98.03±0.24	98.03 ± 0.20
Magic	88.16±0.33	88.37±0.29	87.90 ± 0.24
Miniboo	92.54±0.21	92.74±0.39	91.49 ± 0.19
Letter	98.22±0.18	96.90±0.29	96.38 ± 0.30
Drive	99.45±0.09	99.69±0.04	99.33 ± 0.05
Mocap	99.10±0.12	99.24±0.10	99.10 ± 0.08

Table 6.4: Average test NLL for UCI classification datasets

Dataset	Functional MCMC	WDGPi-G	Deep Ensembles
EEG	0.33±0.04	0.18±0.04	0.24 ± 0.10
HTRU2	0.06±0.002	0.06±0.00	0.07 ± 0.01
Magic	0.31±0.00	0.29±0.00	0.30 ± 0.01
Miniboo	0.18±0.01	0.18±0.00	0.20 ± 0.01
Letter	0.09±0.01	0.17±0.00	0.15 ± 0.01
Drive	0.08±0.01	0.03±0.00	0.05 ± 0.01
Mocap	0.19±0.00	0.03±0.00	0.04 ± 0.00

classification likelihoods, and therefore we report these results in the table. We attribute this to the fact that the GP hyper-parameters are optimized with the transformed labels.

As baselines, we chose the same methods as in the regression experiments. The results in terms of classification accuracy and mean negative test log-likelihood are presented in Tables 6.3 and 6.4. The results show that even in the classification setting, the performance of the proposed approach is competitive with the state-of-the-art. While we perform optimization of GP hyper-parameters, our approach allows us to get rid of the Wasserstein optimization phase, which is used in WDGPi-G, without a significant loss in classification performance.

6.6 LIMITATIONS OF THE APPROACH

While we consider our approach quite elegant in encoding prior information in the form of functional priors, we believe that it is important to point out some limitations compared to other works.

First, we point out that the form of posterior distribution we are targeting is approximate. Then, the main limitation is that the functional prior density needs to have a closed form. Even though the class of functional priors which have this

property is large, this might be too restrictive in applications where it is possible to sample from such priors, but no closed form is available. Prior works which perform a preliminary optimization of the prior over the weights (e.g., [134]) can operate on samples from functional priors without the need to express these in closed form.

Another limitation related to the choice of a GP functional prior is how to set hyper-parameters. In this work, we resort to marginal likelihood optimization, but it is possible that this choice induces overfitting. One way around this would be to include hyper-parameters in the set of variables to be sampled in SG-HMC to obtain samples from their posterior at the expenses of having to deal with a more costly MCMC sampling. Having said that, there are situations where functional priors are easy to elicit and express without the need to carry out hyperparameter optimization.

6.7 DISCUSSION OF THE MCMC INFERENCE FOR FUNCTIONAL PRIORS

In this paper, we proposed a novel way to incorporate prior knowledge in Bayesian NNs (BNNs) in the form of functional priors. In our view, such functional priors implicitly determine priors over BNN weights, and the proposed formulation yields an approximate posterior over the weights from which it is possible to sample through MCMC or any other approximate inference techniques. In this paper, we studied the scenario where functional priors are expressed in the form of Gaussian processes (GPs), but our formulation can handle any functional prior, which can be expressed in closed form. We then discussed how to scale our approach to handle large data sets by operating on mini-batches, despite the complications stemming from the use of GP priors.

We can also point out that if we look at the proposed method from a different perspective, we find out that it produces samples from the true posterior distribution $p(\mathbf{w}|\mathbf{X}^*, \mathbf{y})$ of another probabilistic model, which is depicted on Fig. 6.4. In this model, we additionally introduced latent variables $\mathbf{f}_1, \mathbf{f}_2$. The variable $\mathbf{f}_1$

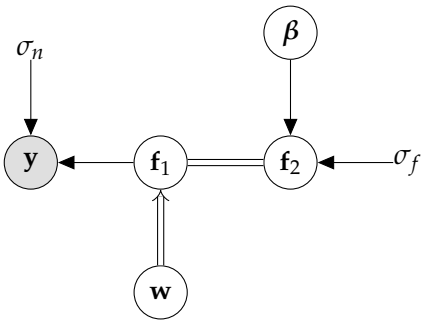

Figure 6.4: Graphical model representation of an alternative view on the proposed method. Double line connections are used to indicate a deterministic relationship.

deterministically depends on $\mathbf{w}$, so

$$p(\mathbf{f}_1|\mathbf{w}) = \delta(\mathbf{f}_1 - f(\mathbf{X}^*, \mathbf{w})), \tag{6.21}$$

where $f(\mathbf{X}^*, \mathbf{w})$ is an output of the BNN. Let's derive a joint distribution of variables $\mathbf{w}, \boldsymbol{\beta}, \mathbf{f}_1, \mathbf{f}_2, \mathbf{y}$:

$$
\begin{aligned}
&p(\mathbf{w}, \boldsymbol{\beta}, \mathbf{f}_1, \mathbf{f}_2, \mathbf{y} | \sigma_n, \sigma_f) = \\
&p(\mathbf{w}) p(\mathbf{f}_1 | \mathbf{w}) p(\mathbf{y} | \mathbf{f}_1, \sigma_n) p(\boldsymbol{\beta}) p(\mathbf{f}_2 | \boldsymbol{\beta}, \sigma_f) p(\mathbf{f}_1, \mathbf{f}_2) = \\
&p(\mathbf{w}) p(\mathbf{f}_1 | \mathbf{w}) p(\mathbf{y} | \mathbf{f}_1, \sigma_n) p(\boldsymbol{\beta}) p(\mathbf{f}_2 | \boldsymbol{\beta}, \sigma_f) \delta(\mathbf{f}_1 - \mathbf{f}_2).
\end{aligned}
$$

In the third line we used the fact that $\mathbf{f}_1$ and $\mathbf{f}_2$ are deterministically connected, thus $p(\mathbf{f}_1, \mathbf{f}_2) = \delta(\mathbf{f}_1 - \mathbf{f}_2)$. Using the property of the Dirac delta function, we can note that:

$$
p(\mathbf{f}_2 | \boldsymbol{\beta}, \sigma_f) \delta(\mathbf{f}_1 - \mathbf{f}_2) = p(\mathbf{f}_1 | \boldsymbol{\beta}, \sigma_f).
$$

Also, we can substitute the expression for the $p(\mathbf{f}_1 | \mathbf{w})$ and obtain

$$
\begin{aligned}
&p(\mathbf{w}, \boldsymbol{\beta}, \mathbf{f}_1, \mathbf{y} | \sigma_n, \sigma_f) = \\
&p(\mathbf{w}) \delta(\mathbf{f}_1 - f(\mathbf{X}^*, \mathbf{w})) p(\mathbf{y} | \mathbf{f}_1, \sigma_n) p(\boldsymbol{\beta}) p(\mathbf{f}_1 | \boldsymbol{\beta}, \sigma_f).
\end{aligned}
$$

Then, using the same property of the Dirac delta function

$$
p(\mathbf{w}, \boldsymbol{\beta}, \mathbf{y} | \sigma_n, \sigma_f) = p(\mathbf{w}) p(\mathbf{y} | f(\mathbf{X}^*, \mathbf{w}), \sigma_n) p(\boldsymbol{\beta}) p(f(\mathbf{x}, \mathbf{w}) | \boldsymbol{\beta}, \sigma_f)
$$

Here we note that this expression is exactly the sampling objective that was used in the proposed method, with an assumption that the prior distribution over the weights of the BNN $p(\mathbf{w})$ is uniform. Thus, we can conclude that the proposed method actually produces samples from the true posterior distribution but from a different type of model with certain assumptions.

We tested our proposal on regression and classification tasks and compared it with state-of-the-art approaches to carry out inference and prior optimization for BNNs. Our results demonstrate that the proposed approach is competitive in terms of performance and quantification of uncertainty while being easy to implement.

We are currently investigating ways to handle GP priors with priors over hyper-parameters for increased flexibility and alternative ways to specify functional priors. Furthermore, we are investigating applications of BNNs for image classification tasks for which BNN architectures use convolutional layers.

6.8 VARIATIONAL BOOTSTRAP FOR BNNS WITH FUNCTIONAL PRIORS

In this section, we develop another type of VI for BNNs with a GP prior over the outputs of the model. As it was mentioned before [15, 117], the concept of explicit utilization of KL divergence as an objective for functional VI, like in [130], [145], is questionable. One of the possible ways to avoid this problem is to abandon the explicit KL computation and switch to methods that optimize the divergence implicitly. Recently, a family of Particle Optimization Variational Inference (POVI) methods that exploit exactly this trick [29], [79] was proposed.

POVI allows one to obtain samples from the posterior of the BNN without modification of the model. To achieve this property, methods from this family use particles which are different instances of the model. The particles are optimized

independently, with special constraints for each particle. These constraints provide convergence of a set of particles to an approximate posterior distribution while preserving their diversity.

The performance of POVI methods depends on a choice of a kernel, which is independent of a type of prior in general. This work aims to establish some connection between a kernel used to construct a prior distribution of functions and a kernel used to solve a POVI problem.

The proposed work takes the idea of Variational Bootstrap (VB) from [88]. We already described this idea in Section 4.3.2. In the proposed work, the diversity of the particles is introduced by perturbations of the training labels and different samples from the functional prior distribution. Mahalanobis regularization, proposed in Section 6.4, ensures the closeness of the set of particles to samples from the functional prior distribution. The set of perturbed labels and the samples from the functional prior are unique for each particle.

6.8.1 *Linear model case*

If there is a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, a likelihood of the model $p(\mathcal{D}|f)$ and a functional prior $\mathcal{GP}$, our goal is to get samples from the predictive posterior distribution of the model with a parametric bootstrap procedure.

Let's assume, that the functional prior is $\mathcal{GP}\{0, \kappa\}$ with some kernel κ,a likelihood of the model is Gaussian $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_n^2)$, where $\mathbf{f}$ is a vector of latent function values and σ_n^2 is an observational noise. To get samples from a functional posterior, we can apply a functional version of the parametric bootstrap. Each particle represents a kernel ridge regression (KRR) problem with perturbed data labels $\tilde{\mathbf{y}} \sim \mathcal{N}(\mathbf{y}, \sigma^2)$ and regularization $||f - \tilde{f}||_{\mathcal{H}}^2$, where $\tilde{f} \sim \mathcal{GP}(0, \kappa)$.

$$\underset{f}{\operatorname{argmin}} \frac{1}{\sigma_n^2} \sum_{i=1}^{N} (\tilde{y}_i - f(\mathbf{x}_i))^2 + ||f - \tilde{f}||_{\mathcal{H}}^2 \tag{6.22}$$

The solution for each particle is as follows:

$$f(\mathbf{x}_*) = \mathbf{k}^\top (\mathbf{K} + \mathbf{I})^{-1} (\tilde{\mathbf{y}} - \tilde{\mathbf{f}}) + \tilde{f}(\mathbf{x}_*) \tag{6.23}$$

where $\tilde{\mathbf{f}}$ is a vector of evaluations of the function $\tilde{f}$ on the training set. It is easy to see that

$$\mathbb{E}_{\tilde{f} \sim p(0,\kappa)}[f(\mathbf{x}_*)] = \mathbf{k}^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{6.24}$$

and

$$\operatorname{var}(f(\mathbf{x}_*)) = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k} \tag{6.25}$$

So, the mean and the variance of the predictions of the particles are equivalent to the mean and the variance of the GP prediction.

6.8.2 *Approximation of the Hilbert space norm*

Let's consider functions $f, \tilde{f} \in \mathcal{H}$ and denote $f - \tilde{f}$ as g. A vector $\mathbf{g}$ is an evaluation of the function g at the training points $\{\mathbf{x}\}_{i=1}^N$. Let's consider a function h

$$h = \sum_{i=1}^{N} \alpha_i \kappa(\cdot, \mathbf{x}_i)$$

with coefficients $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = \mathbf{g}^T \mathbf{K}^{-1}$$

We can observe that

$$||h||_{\mathcal{H}}^2 = \mathbf{g}^T \mathbf{K}^{-1} \mathbf{g}$$

and that

$$\begin{aligned} &h = \underset{\tilde{g}}{\operatorname{argmin}} \, ||\tilde{g}||_{\mathcal{H}}^2 \\ &s.t. \; \tilde{g}(\mathbf{x}) = g(\mathbf{x}) \; \forall \mathbf{x} \in \{\mathbf{x}\}_{i=1}^N \end{aligned} \tag{6.26}$$

So, we can claim that $\mathbf{g}^T \mathbf{K}^{-1} \mathbf{g}$ is the Hilbert norm of the smoothest function that passes through all the points $\mathbf{f} - \tilde{\mathbf{f}}$. So, this quadratic form gives a lower bound on $||f - \tilde{f}||_{\mathcal{H}}^2$. For the KRR, we can reformulate the optimization objective as:

$$\underset{f}{\operatorname{argmin}} \frac{1}{\sigma_n^2} \sum_{i=1}^{N} (\tilde{y}_i - f(\mathbf{x}_i))^2 + (\mathbf{f} - \tilde{\mathbf{f}})^T \mathbf{K}^{-1} (\mathbf{f} - \tilde{\mathbf{f}}) \tag{6.27}$$

This optimization problem will have exactly the same solution as the initial formulation of the KRR. We can note that the regularization term is equivalent to a squared Mahalanobis distance. To avoid numerical problems, we can relax the equality constraint for (6.26)

$$\begin{aligned} &h' = \underset{\tilde{g}}{\operatorname{argmin}} \, ||\tilde{g}||_{\mathcal{H}}^2 \\ &\text{s.t. } ||\tilde{g}(\mathbf{x}) - g(\mathbf{x})||^2 < c, \; \forall \mathbf{x} \in \{\mathbf{x}\}_{i=1}^N \end{aligned}$$

The optimization objective for this problem is as follows:

$$h' = \underset{\tilde{g}}{\operatorname{argmin}} \frac{1}{\lambda} \sum_{i=1}^{N} (\tilde{g}(\mathbf{x}_i) - g(\mathbf{x}_i))^2 + ||\tilde{g}||_{\mathcal{H}}^2 \tag{6.28}$$

Thus we can obtain a lower bound for the $||h||_{\mathcal{H}}^2$

$$||h'||_{\mathcal{H}}^2 = \mathbf{g}^T (\mathbf{K} + \frac{1}{\lambda}\mathbf{I})^{-1} \mathbf{K} (\mathbf{K} + \frac{1}{\lambda}\mathbf{I})^{-1} \mathbf{g}$$

It is possible to train a neural network with the relaxed Mahalanobis regularization. It is necessary to add M points from the input space to the regularization term to ensure the regularization constraint in the target regions of the input space. We denote the extended vectors of functions values as $\mathbf{f}_*$ and $\tilde{\mathbf{f}}_*$, their difference as $\mathbf{g}_*$, and the extended Gram matrix as $\mathbf{K}_*$. The optimization objective will be as follows:

$$\operatorname{argmin}_{\mathbf{w}} \frac{1}{\sigma_n^2} \sum_{i=1}^{N} (\tilde{y}_i - f(\mathbf{x}_i, \mathbf{w}))^2 + \mathbf{g}_*^T (\mathbf{K}_* + \frac{1}{\lambda}\mathbf{I})^{-1} \mathbf{K}_* (\mathbf{K}_* + \frac{1}{\lambda}\mathbf{I})^{-1} \mathbf{g}_* \tag{6.29}$$

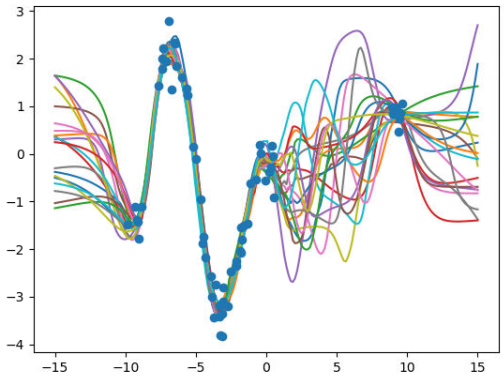

Figure 6.5: FBVI predictions of 20 particles

Dataset	FBVI	WDGPi-G
Boston	2.82±0.06	2.80±0.40
Concrete	4.88±0.16	4.80±0.14
Energy	0.49±0.05	0.34±0.07
Wine	0.62±0.04	0.60±0.04

Table 6.5: RMSE for GP regression (GPR), the proposed method (FBVI) and WDGPi-G

6.8.3 *Emprical evaluation*

In the beginning, the Functional Bootstrap Variational Inference (FBVI) was tested on a toy regression dataset. A fully connected network was used for the experiments on synthetic data. The network had two hidden layers with 64 neurons in each layer and used *tanh* activation function. An equally spaced grid of input points was used for the Mahalanobis regularization term. Then the proposed method was tested on several UCI regression datasets. A network with the same architecture was used. It had 100 neurons in each layer, and it used the ReLU activation function. We performed the experiments only for the small UCI datasets, where full-batch training is available because the proposed method in its current state is not adopted for mini-batch training.

6.8.4 *Discussion of the FVBI method*

The proposed idea gave good-looking predictions on toy data (Fig. 6.5, that are much more similar to a GP than the ones produced by WDGPi-G method (Fig. 6.2). But the FBVI method failed to outperform the WDGPi-G in the preliminary experiments on the UCI data, and it demonstrated worse results than MCMC for GP functional priors proposed earlier in this chapter (Section 6.3). Besides, this method does not require a Wasserstein distance minimization phase like WDGPi-G. And as we see in Chapter 4, Fig. 4.2, Variational Bootstrap allows obtaining faster convergence than MCMC. Also, this method could be extended to mini-batch training by means of Random Features similarly to the idea in Section 6.4.1.

7 CONCLUSIONS

7.1 SUMMARY OF CONTRIBUTIONS AND OPEN QUESTIONS

Optical Processing Hardware (Chapter 3)
We proposed a unified approach for integrating OPUs into deep learning models that allows addressing two issues posed by this device. On the one hand, the proposed approach allows for dealing with the restrictions that OPUs put on the input data type. The ability to work exclusively with binary input data made OPU applicable only to a very narrow set of problems, which was a significant obstacle to the commercialization and widespread use of the technology. The proposed approach allows binarizing any input data of any type using an appropriate neural network in an optimal fashion regarding the loss of the specific machine learning problem. On the other hand, the proposed method allows integrating OPUs into deep learning models so that the whole pipeline can be trained in an End-to-End fashion. It allows for solving problems that could not be solved with the Kernel Ridge Regression method implemented by the device. As we emphasized in Chapter 1, machine learning became so successful due to feature extraction procedures trainable with backpropagation. And nowadays, it is the only way to achieve high performance for many machine learning tasks. That is why it was crucial to resolve the problem of non-differentiability of the OPU.

While this work proposes solutions for some problems of the OPU, there are still challenges that have to be addressed. One of the most problematic things with optical hardware is the high latency of the data exchange between the OPU and other devices, such as CPU, GPU, and RAM. This drawback significantly alleviates the benefits of the OPU. The approach proposed in this work necessitates a considerable exchange of data between the device and the GPU. By conducting the data exchange in a batch manner, the proposed method prioritizes high bandwidth over low latency. However, the high latency of the OPU is still a challenge because it results in a slower training process compared to models that only use the GPU. Thus, one of the possible directions for future research could be decreasing the number of data exchanges between OPUs and other hardware. Another issue that requires further consideration is the possibility of stacking several OPUs in one pipeline. As it was said, the main benefit of optical hardware is the ability to extract features in nanoseconds, so if OPUs work on top of deep learning pipelines, the primary time complexity will come from GPUs. Moving as many computations as possible to OPUs will significantly decrease the inference time for the whole pipeline.

Efficient and accurate inference for Bayesian classification (Chapter 4)
Neural networks are widely used for achieving state-of-the-art performance in various tasks. However, they lack a systematic way to characterize uncertainty in predictions. This makes a Bayesian treatment of this type of models highly desirable

but also mathematically and computationally challenging. On the other hand, GPs are founded on Bayesian principles, but classification tasks using GPs require computationally expensive approximations. Inspired by the idea of variational bootstrap, which allows performing fast and accurate Bayesian inference for regression models, like GPs, we proposed a novel method for conducting Bayesian inference in classification models such as neural networks and Gaussian Processes. Our method provides a practical solution to the abovementioned limitations by offering an easily parallelizable approach for Bayesian inference.

One of the possible directions for further research on the proposed method is various parallel implementations of our approach to significantly speed up inference for large-scale problems through the use of clusters of computing machines. Another possible improvement for the proposed method is to find a better way to approximate a Dirichlet likelihood used in the approach. Currently, the approximation is not accurate for the cases when the level of confidence in training labels is high. This may lead to the degradation of the model's performance in terms of classification accuracy and uncertainty estimation. We made some steps to improve the quality of approximation in Section 4.4, but this problem requires further investigation. And the third possible research direction for the proposed method is combining it with the methods presented in Chapter 3 for training Bayesian models on OPUs. The proposed method is preferable for the OPU setting because it provides a faster inference procedure compared to other Bayesian inference techniques.

Making Gaussian Processes scalable with localization (Chapter 5)
In Chapter 5, we introduced a new approach for scaling up GPs. We focused on regression problems and created a novel GP-based model by applying the localization technique. We achieved the localization effect by reducing the contributions of training points that are distant from a target test point. The training points outside the radius, defined by the localization hyperparameter, do not contribute to the prediction. The resulting GP maintains the positive definiteness property for a covariance matrix and allows for a significant performance improvement compared to a vanilla GP model that uses the full training set for each test point, thanks to the effect of sparsification of the Gram matrix. As a result, the model uses different subsets of training data to perform predictions for test points located in different regions of the input space.

Currently, the main drawback of the Local Gaussian Process Regression is the necessity to choose the localization hyperparameter with cross-validation. It is possible to leverage the marginal log-likelihood optimization approach to select the appropriate values for this hyperparameter. In a current setting applying this approach to the proposed method is problematic because the MLL function is not smooth with respect to the localizing hyperparameter. Addressing this issue could be one of the directions for future research.

Another direction for future research is extending this approach to random feature approximation of the localized GP. Some steps in this direction were already made in [142], but this work addresses a very special combination of GP kernel and localizing kernel. A more general way of obtaining such approximations for

localizing GPs is an open question.

Bayesian neural networks with functional priors (Chapter 6)
In this chapter, we proposed a novel approach for incorporating functional prior knowledge into Bayesian neural networks using GP priors. Our approach connects the prior distribution over functions with a prior distribution over BNN weights. Using such type of priors allows for encoding into the deep model some functional properties, like smoothness, seasonality, or trend, which adds an additional mechanism to control the behavior of the deep model, which is often considered as a black-box from an interpretability point of view.

Our method allows for estimating the approximate posterior distribution of the weights using MCMC. Also, we showed the possibility of using a variational bootstrap approach for the same purpose. In the proposed work, we used GPs as a functional prior, but our formulation allows for extending the method to any type of functional distribution that has an analytical expression for the probability density. For the case of GP prior, we managed to make our approach scalable to large datasets with mini-batched training by using random feature approximation to address the scalability issues typical for GPs.

As it was mentioned in Chapter 6, the proposed method performs approximate posterior inference, and the main challenge for future research is making this procedure exact. The most promising direction, in this case, is POVI methods that, under some conditions, can provide an exact solution [29].

Another possible extension of the proposed approach is speeding up the inference by variational bootstrap. We made some steps in this direction in Section 6.8. In this case, we are unable to obtain an exact posterior distribution over the weights, but our experiments in Chapter 4 showed that the variational bootstrap provides a satisfactory approximate posterior in terms of the performance of the model and its uncertainty estimation.

7.2 CONCLUDING REMARKS

In conclusion, this work explored some intersections between deep learning, Bayesian methods, and linear models, with a particular focus on Gaussian Processes. In four separate projects, we demonstrated how these approaches can be combined to enhance their flexibility, improve inference speed and scalability, and incorporate desirable functional properties into a model.

Deep learning showed outstanding results in complex machine learning problems, but its utilization in risk-sensitive applications is arguable due to the lack of uncertainty quantification and control over the properties of its predictions. Bayesian methods allow addressing both of these issues by forcing it to produce a predictive distribution instead of single predictions and by introducing prior knowledge into the model. The first property allows reasoning about the level of confidence of the predictions, and the second property allows forcing the model to respect some given constraints, which can be essential for some specific task.

As we explored in this thesis, Bayesian linear models can be modified to stay relevant in the age of big data on their own and also to work together with deep models, compensating for some of their drawbacks. The latter gives hope that the

extensive theoretical framework behind these classical models will play some role in understanding the theoretical properties of deep models.

BIBLIOGRAPHY

[1] Mark A Aizerman. "Theoretical foundations of the potential function method in pattern recognition learning." In: *Automation and remote control* 25 (1964), pp. 821–837.

[2] Naomi S Altman. "An introduction to kernel and nearest-neighbor nonparametric regression." In: *The American Statistician* 46.3 (1992), pp. 175–185.

[3] Nongnuch Artrith, Keith T Butler, François-Xavier Coudert, Seungwu Han, Olexandr Isayev, Anubhav Jain, and Aron Walsh. "Best practices in machine learning for chemistry." In: *Nature chemistry* 13.6 (2021), pp. 505–508.

[4] Andrei Atanov, Arsenii Ashukha, Kirill Struminsky, Dmitriy Vetrov, and Max Welling. "The Deep Weight Prior." In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=ByGuynAct7.

[5] Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. "Assessing the Scalability of Biologically-Motivated Deep Learning Algorithms and Architectures." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/63c3ddcc7b23daa1e42dc41f9a44a873-Paper.pdf.

[6] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation." In: *CoRR* (2013). arXiv: 1308.3432. URL: http://arxiv.org/abs/1308.3432.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[8] Enrico Blanzieri and Farid Melgani. "An adaptive SVM nearest neighbor classifier for remotely sensed imagery." In: *2006 IEEE International Symposium on Geoscience and Remote Sensing*. IEEE, 2006, pp. 3931–3934.

[9] Enrico Blanzieri and Farid Melgani. "Nearest neighbor classification of remote sensing images with the maximal margin principle." In: *IEEE Transactions on geoscience and remote sensing* 46.6 (2008), pp. 1804–1811.

[10] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. "Weight Uncertainty in Neural Network." In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. PMLR, 2015, pp. 1613–1622.

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers." In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. New York, NY, USA: Association for Computing Machinery, 1992, pp. 144–152. ISBN: 089791497X. DOI: 10.1145/130385.130401. URL: https://doi.org/10.1145/130385.130401.

[12] Léon Bottou and Vladimir Vapnik. "Local learning algorithms." In: *Neural computation* 4.6 (1992), pp. 888–900.

[13] Dimitri Bourilkov. "Machine and Deep Learning Applications in Particle Physics." In: *International Journal of Modern Physics A* 34.35 (2019), p. 1930019.

[14] Stephen P Brooks and Andrew Gelman. "General Methods for Monitoring Convergence of Iterative Simulations." In: *Journal of computational and graphical statistics* 7.4 (1998), pp. 434–455.

[15] David R. Burt, Sebastian W. Ober, Adrià Garriga-Alonso, and Mark van der Wilk. "Understanding Variational Inference in Function-Space." In: *Third Symposium on Advances in Approximate Bayesian Inference*. 2021. URL: https://openreview.net/forum?id=7P9y3sRa5Mk.

[16] Andrea Caponnetto and Ernesto De Vito. "Optimal Rates for the Regularized Least-Squares Algorithm." In: *Found. Comput. Math.* 7.3 (2007), pp. 331–368. DOI: 10.1007/s10208-006-0196-8. URL: https://doi.org/10.1007/s10208-006-0196-8.

[17] Alessandro Cappelli, Ruben Ohana, Julien Launay, Laurent Meunier, Iacopo Poli, and Florent Krzakala. "Adversarial Robustness by Design Through Analog Computing And Synthetic Gradients." In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 3493–3497. DOI: 10.1109/ICASSP43922.2022.9746671.

[18] Luigi Carratino, Stefano Vigogna, Daniele Calandriello, and Lorenzo Rosasco. "Park: Sound and efficient kernel ridge regression by feature space partitions." In: *Advances in Neural Information Processing Systems* 34 (2021).

[19] Miguel Carreira-Perpinan and Weiran Wang. "Distributed optimization of deeply nested systems." In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Ed. by Samuel Kaski and Jukka Corander. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, 2014, pp. 10–19. URL: https://proceedings.mlr.press/v33/carreira-perpinan14.html.

[20] Miguel Á. Carreira-Perpiñán and Ramin Raziperchikolaei. "Hashing with binary autoencoders." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 557–566. DOI: 10.1109/CVPR.2015.7298654.

[21] Tianqi Chen, Emily Fox, and Carlos Guestrin. "Stochastic Gradient Hamiltonian Monte Carlo." In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. Proceedings of Machine Learning Research 2. PMLR, 2014, pp. 1683–1691.

[22] Tianqi Chen, Emily Fox, and Carlos Guestrin. "Stochastic Gradient Hamiltonian Monte Carlo." In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, 2014, pp. 1683–1691. URL: https://proceedings.mlr.press/v32/cheni14.html.

[23] Haibin Cheng, Pang-Ning Tan, and Rong Jin. "Localized support vector machine and its efficient algorithm." In: *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM. 2007, pp. 461–466.

[24] Youngmin Cho and Lawrence Saul. "Kernel Methods for Deep Learning." In: *Advances in Neural Information Processing Systems*. Ed. by Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta. Vol. 22. Curran Associates, Inc., 2009. URL: https://proceedings.neurips.cc/paper/2009/file/5751ec3e9a4feab575962e78e006250d-Paper.pdf.

[25] Anna Choromanska et al. "Beyond Backprop: Online Alternating Minimization with Auxiliary Variables." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1193–1202. URL: https://proceedings.mlr.press/v97/choromanska19a.html.

[26] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. *Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1*. 2016. DOI: 10.48550/ARXIV.1602.02830. URL: https://arxiv.org/abs/1602.02830.

[27] Kurt Cutajar, Edwin V. Bonilla, Pietro Michiardi, and Maurizio Filippone. "Random Feature Expansions for Deep Gaussian Processes." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 884–893. URL: https://proceedings.mlr.press/v70/cutajar17a.html.

[28] Kurt Cutajar, Michael Osborne, John Cunningham, and Maurizio Filippone. "Preconditioning Kernel Matrices." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 2529–2538. URL: https://proceedings.mlr.press/v48/cutajar16.html.

[29] Francesco D'Angelo and Vincent Fortuin. "Repulsive Deep Ensembles are Bayesian." In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021.

[30] Denis Derkach, Nikita Kazeev, Fedor Ratnikov, Andrey Ustyuzhanin, and Alexandra Volokhova. "Cherenkov Detectors Fast Simulation Using Neural Networks." In: *CoRR* (2019). arXiv: 1903.11788. URL: http://arxiv.org/abs/1903.11788.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019-06, pp. 4171–4186.

[32] Luc Devroye, Laszlo Györfi, and Adam Krzyżak. "The Hilbert kernel regression estimate." In: *Journal of Multivariate Analysis* 65.2 (1998), pp. 209–227.

[33] Zhe Dong, Andriy Mnih, and George Tucker. "DisARM: An Antithetic Gradient Estimator for Binary Latent Variables." In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/d880e783834172e5ebd1868d84463d93-Abstract.html.

[34] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[35] Bradley Efron. "Bayesian Inference and the Parametric Bootstrap." In: *Ann. Appl. Stat.* 6.4 (2012-12), pp. 1971–1997.

[36] Jiri Fajtl, Vasileios Argyriou, Dorothy Monekosso, and Paolo Remagnino. "Latent Bernoulli Autoencoder." In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2964–2974. URL: https://proceedings.mlr.press/v119/fajtl20a.html.

[37] Jianqing Fan and Irene Gijbels. *Local polynomial modelling and its applications*. Routledge, 2018.

[38] Maurizio Filippone and Raphael Engler. "Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE)." In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11, 2015*. 2015, pp. 1015–1024.

[39] Maurizio Filippone and Raphael Engler. "Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE)." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1015–1024. URL: https://proceedings.mlr.press/v37/filippone15.html.

[40] Ronald A Fisher. "The use of multiple measurements in taxonomic problems." In: *Annals of eugenics* 7.2 (1936), pp. 179–188.

[41] Vincent Fortuin. "Priors in bayesian deep learning: A review." In: *International Statistical Review* (2022).

[42] Vincent Fortuin, Adrià Garriga-Alonso, Florian Wenzel, Gunnar Ratsch, Richard E Turner, Mark van der Wilk, and Laurence Aitchison. "Bayesian Neural Network Priors Revisited." In: *Third Symposium on Advances in Approximate Bayesian Inference*. 2021. URL: https://openreview.net/forum?id=xaqKWHcoOGP.

[43] C. Fowlkes, S. Belongie, and J. Malik. "Efficient spatiotemporal grouping using the Nystrom method." In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990481.

[44] Abram L. Friesen and Pedro Domingos. "Deep Learning as a Mixed Convex-Combinatorial Optimization Problem." In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=B1Lc-Gb0Z.

[45] Yarin Gal and Zoubin Ghahramani. "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference." In: *4th International Conference on Learning Representations (ICLR) workshop track*. 2016.

[46] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning." In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. PMLR, 2016, pp. 1050–1059.

[47] Erik Gawehn, Jan A Hiss, and Gisbert Schneider. "Deep learning in drug discovery." In: *Molecular informatics* 35.1 (2016), pp. 3–14.

[48] Robert B Gramacy and Daniel W Apley. "Local Gaussian process approximation for large computer experiments." In: *Journal of Computational and Graphical Statistics* 24.2 (2015), pp. 561–578.

[49] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. "Backpropagation through the Void: Optimizing control variates for black-box gradient estimation." In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=SyzKd1bCW.

[50] Alex Graves. "Practical Variational Inference for Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf.

[51] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. "On Calibration of Modern Neural Networks." In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1321–1330.

[52] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. "Gene selection for cancer classification using support vector machines." In: *Machine learning* 46.1 (2002), pp. 389–422.

[53] T Hastie and J Zhu. "Discussion of "Support vector machines with applications" by Javier Moguerza and Alberto Munoz." In: *Statistical Science* 21.3 (2006), pp. 352–357.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[55] 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*. 2016, pp. 770–778.

[56] Donald O Hebb. "The first stage of perception: growth of the assembly." In: *The Organization of Behavior* 4 (1949), pp. 60–78.

[57] James Hensman, Nicolò Fusi, and Neil D. Lawrence. "Gaussian Processes for Big Data." In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. UAI'13. Bellevue, WA: AUAI Press, 2013, pp. 282–290.

[58] Khalid M. Hosny, Mohamed A. Kassem, and Mohamed M. Foaud. "Skin Cancer Classification using Deep Learning and Transfer Learning." In: *2018 9th Cairo International Biomedical Engineering Conference (CIBEC)*. 2018, pp. 90–93. DOI: 10.1109/CIBEC.2018.8641762.

[59] Arthur Jacot, Franck Gabriel, and Clement Hongler. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks." In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

[60] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax." In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=rkE3y85ee.

[61] Thorsten Joachim. "Text categorization with support vector machines." In: *Proceedings of the tenth European Conference on Machine Learning, 1998*. 1998.

[62] Marc Kac and AJF Siegert. "An explicit representation of a stationary Gaussian process." In: *The Annals of Mathematical Statistics* 18.3 (1947), pp. 438–442.

[63] Olav Kallenberg and Olav Kallenberg. *Foundations of modern probability*. Vol. 2. Springer, 1997.

[64] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. "Gaussian processes and kernel methods: A review on connections and equivalences." In: *arXiv preprint arXiv:1807.02582* (2018).

[65] Cari G. Kaufman, Mark J. Schervish, and Douglas W. Nychka. "Covariance Tapering for Likelihood-Based Estimation in Large Spatial Data Sets." In: *Journal of the American Statistical Association* 103.484 (2008), pp. 1545–1555.

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

[67] George S Kimeldorf and Grace Wahba. "A correspondence between Bayesian estimation on stochastic processes and smoothing by splines." In: *The Annals of Mathematical Statistics* 41.2 (1970), pp. 495–502.

[68] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[69] Diederik Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: *ICLR* (2013-12).

[70] Wouter Kool, Herke van Hoof, and Max Welling. "Buy 4 REINFORCE Samples, Get a Baseline for Free!" In: *Deep Reinforcement Learning Meets Structured Prediction, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=r1lgTGL5DE.

[71] Karl Krauth, Edwin V. Bonilla, Kurt Cutajar, and Maurizio Filippone. "AutoGP: Exploring the Capabilities and Limitations of Gaussian Process Models." In: *Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, August 11-15, 2017, Sydney, Australia*. 2017.

[72] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[73] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles." In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[74] Guillaume Lample and François Charton. "Deep Learning For Symbolic Mathematics." In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=S1eZYeHFDS.

[75] Julien Launay, Iacopo Poli, and Florent Krzakala. "Principled Training of Neural Networks with Direct Feedback Alignment." In: *CoRR* (2019). arXiv: 1906.04554. URL: http://arxiv.org/abs/1906.04554.

[76] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. "Difference Target Propagation." In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, Carlos Soares, João Gama, and Alípio Jorge. Cham: Springer International Publishing, 2015, pp. 498–515. ISBN: 978-3-319-23528-8.

[77] Liangzhi Li, Kaoru Ota, and Mianxiong Dong. "Deep learning for smart industry: Efficient manufacture inspection system with fog computing." In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4665–4673.

[78] LightOn. *Lightonml documentation*. URL: https://docs.lighton.ai/index.html.

[79] Qiang Liu and Dilin Wang. "Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm." In: *Advances in Neural Information Processing Systems*. Vol. 29. 2016.

[80] M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. "Sparse Spectrum Gaussian Process Regression." In: *Journal of Machine Learning Research* 11 (2010), pp. 1865–1881.

[81] Chao Ma, Yingzhen Li, and Jose Miguel Hernandez-Lobato. "Variational Implicit Processes." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 4222–4233. URL: https://proceedings.mlr.press/v97/ma19b.html.

[82] David JC MacKay. "Bayesian neural networks and density networks." In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 354.1 (1995), pp. 73–80.

[83] Abhishek Maity. "Supervised classification of RADARSAT-2 polarimetric data for different land features." In: *arXiv preprint arXiv:1608.00501* (2016).

[84] Franziska Meier, Philipp Hennig, and Stefan Schaal. "Incremental local Gaussian regression." In: *Advances in Neural Information Processing Systems* 27 (2014).

[85] Franziska Meier, Philipp Hennig, and Stefan Schaal. "Local Gaussian regression." In: *arXiv preprint arXiv:1402.0645* (2014).

[86] Mona Meister and Ingo Steinwart. "Optimal learning rates for localized SVMs." In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 6722–6765.

[87] Dimitrios Milios, Raffaello Camoriano, Pietro Michiardi, Lorenzo Rosasco, and Maurizio Filippone. "Dirichlet-based Gaussian Processes for Large-scale Calibrated Classification." In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

[88] Dimitrios Milios, Pietro Michiardi, and Maurizio Filippone. "A Variational View on Bootstrap Ensembles as Bayesian Inference." In: *AABI 2021, 3rd Symposium on Advances in Approximate Bayesian Inference, January-February 2021*. 2021.

[89] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN: 026201825X.

[90] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[91] ALBERT E Murray. "Perceptron applications in photo interpretation." In: *Photogrammetric Engineering* 27.4 (1961).

[92] Nicole Müecke. "Reducing training time by efficient localized kernel regression." In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 2603–2610.

[93] Radford M. Neal. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. 1st ed. Springer, 1996-08-09.

[94] Radford M Neal et al. "MCMC using Hamiltonian dynamics." In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2.

[95] Hoang Nguyen, Le-Minh Kieu, Tao Wen, and Chen Cai. "Deep learning methods in transportation domain: a review." In: *IET Intelligent Transport Systems* 12.9 (2018), pp. 998–1004.

[96] Arild Nøkland. "Direct Feedback Alignment Provides Learning in Deep Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/d490d7b4576290fa60eb31b5fc917ad1-Paper.pdf.

[97] Ruben Ohana, Jonas Wacker, Jonathan Dong, Sebastien Marmin, Florent Krzakala, Maurizio Filippone, and Laurent Daudet. "Kernel Computations from Large-Scale Random Features Obtained by Optical Processing Units." In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 9294–9298. DOI: 10.1109/ICASSP40776.2020.9053272.

[98] OpenAI. *ChatGPT*. https://chat.openai.com/chat. Accessed: 2023-01-26. 2021.

[99] Manfred Opper and Ole Winther. "Gaussian processes for classification: Mean-field algorithms." In: *Neural computation* 12.11 (2000), pp. 2655–2684.

[100] Nobuyuki Otsu. "A threshold selection method from gray-level histograms." In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[101] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

[102] John Paisley, David M. Blei, and Michael I. Jordan. "Variational Bayesian Inference with Stochastic Search." In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML'12. Edinburgh, Scotland: Omnipress, 2012, pp. 1363–1370. ISBN: 9781450312851.

[103] Jeffrey Pennington, Felix Xinnan X Yu, and Sanjiv Kumar. "Spherical Random Features for Polynomial Kernels." In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015.

[104] Jorn W.T. Peters, Tim Genewein, and Max Welling. *Probabilistic Binary Neural Networks*. 2019. URL: https://openreview.net/forum?id=B1fysiAqK7.

[105] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. "Deep Learning for Audio Signal Processing." In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (2019), pp. 206–219.

[106] Radoslaw Pytlak. *Conjugate gradient algorithms in nonconvex optimization*. Vol. 89. Springer Science & Business Media, 2008.

[107] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. "Binary neural networks: A survey." In: *Pattern Recognition* 105 (2020), p. 107281. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2020.107281. URL: https://www.sciencedirect.com/science/article/pii/S0031320320300856.

[108] Joaquin Quinonero-Candela and Carl Edward Rasmussen. "A unifying view of sparse approximate Gaussian process regression." In: *The Journal of Machine Learning Research* 6 (2005), pp. 1939–1959.

[109] Maithra Raghu and Eric Schmidt. "A survey of deep learning for scientific discovery." In: *arXiv preprint arXiv:2003.11755* (2020).

[110] Ali Rahimi and Benjamin Recht. "Random Features for Large-Scale Kernel Machines." In: *Advances in Neural Information Processing Systems 20*. Ed. by J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis. Curran Associates, Inc., 2008, pp. 1177–1184.

[111] Ali Rahimi and Benjamin Recht. "Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning." In: *Advances in Neural Information Processing Systems*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Vol. 21. Curran Associates, Inc., 2008. URL: https://proceedings.neurips.cc/paper/2008/file/0efe32849d230d7f53049ddc4a4b0c60-Paper.pdf.

[112] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black Box Variational Inference." In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Ed. by Samuel Kaski and Jukka Corander. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, 2014, pp. 814–822. URL: https://proceedings.mlr.press/v33/ranganath14.html.

[113] Carl E. Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[114] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[115] Danilo Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1530–1538. URL: https://proceedings.mlr.press/v37/rezende15.html.

[116] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[117] Tim G. J. Rudner, Zonghao Chen, and Yarin Gal. "Rethinking Function-Space Variational Inference in Bayesian Neural Networks." In: *Third Symposium on Advances in Approximate Bayesian Inference*. 2021. URL: https://openreview.net/forum?id=KtY5qphxnCv.

[118] A. Saade, F. Caltagirone, I. Carron, L. Daudet, A. Drémeau, S. Gigan, and F. Krzakala. "Random projections through multiple optical scattering: Approximating Kernels at the speed of light." In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 6215–6219. DOI: 10.1109/ICASSP.2016.7472872.

[119] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers." In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210.

[120] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem." In: *Neural Comput.* 10.5 (1998), pp. 1299–1319. DOI: 10.1162/089976698300017467. URL: https://doi.org/10.1162/089976698300017467.

[121] Nicola Segata and Enrico Blanzieri. "Fast and Scalable Local Kernel Machines." In: *Journal of Machine Learning Research* 11.6 (2010).

[122] Nicola Segata, Edoardo Pasolli, Farid Melgani, and Enrico Blanzieri. "Local SVM approaches for fast and accurate classification of remote-sensing images." In: *International Journal of Remote Sensing* 33.19 (2012), pp. 6186–6201.

[123] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[124] Donald Shepard. "A two-dimensional interpolation function for irregularly-spaced data." In: *Proceedings of the 1968 23rd ACM national conference*. 1968, pp. 517–524.

[125] Jiaxin Shi, Shengyang Sun, and Jun Zhu. "A Spectral Approach to Gradient Estimation for Implicit Distributions." In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4644–4653. URL: https://proceedings.mlr.press/v80/shi18a.html.

[126] Edward Snelson and Zoubin Ghahramani. "Local and global sparse Gaussian process approximations." In: *Artificial Intelligence and Statistics*. PMLR. 2007, pp. 524–531.

[127] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. "Bayesian Optimization with Robust Bayesian Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/a96d3afec184766bfeca7a9f989fc7e7-Paper.pdf.

[128] Alexander Statnikov, Douglas Hardin, and Constantin Aliferis. "Using SVM weight-based methods to identify causally relevant and non-causally relevant variables." In: *sign* 1.4 (2006), pp. 474–484.

[129] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[130] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. "FUNCTIONAL VARIATIONAL BAYESIAN NEURAL NETWORKS." In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rkxacs0qY7.

[131] Rashish Tandon, Si Si, Pradeep Ravikumar, and Inderjit Dhillon. "Kernel ridge regression via partitioning." In: *arXiv preprint arXiv:1608.01976* (2016).

[132] Julien Tissier, Christophe Gravier, and Amaury Habrard. "Near-Lossless Binarization of Word Embeddings." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 7104–7111. DOI: 10.1609/aaai.v33i01.33017104. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4692.

[133] Michalis K. Titsias. "Variational Learning of Inducing Variables in Sparse Gaussian Processes." In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*. Vol. 5. JMLR Proceedings. JMLR.org, 2009, pp. 567–574.

[134] Ba-Hien Tran, Simone Rossi, Dimitrios Milios, and Maurizio Filippone. "All You Need is a Good Functional Prior for Bayesian Deep Learning." In: *Journal of Machine Learning Research* 23.74 (2022), pp. 1–56. URL: http://jmlr.org/papers/v23/20-1340.html.

[135] A. Tsybakov. *Introduction to Nonparametric Estimation*. New York: Springer, 2009.

[136] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. "REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/ebd6d2f5d60ff9afaeda1a81fc53e2d0-Paper.pdf.

[137] Matthew Turk and Alex Pentland. "Eigenfaces for recognition." In: *Journal of cognitive neuroscience* 3.1 (1991), pp. 71–86.

[138] Vladimir N Vapnik. "Estimation of Dependencies Based on Empirical Data Springer-Verlag." In: *New York, NY* (1982).

[139] Vladimir Vapnik and Alexey Chervonenkis. *Theory of pattern recognition*. 1974.

[140] 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*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[141] Nhi NY Vo, Xuezhong He, Shaowu Liu, and Guandong Xu. "Deep learning for decision making and the optimization of socially responsible investments and portfolio." In: *Decision Support Systems* 124 (2019), p. 113097.

[142] Jonas Wacker and Maurizio Filippone. "Local Random Feature Approximations of the Gaussian Kernel." In: *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES-2022, Verona, Italy and Virtual Event, 7-9 September 2022*. Ed. by Matteo Cristani, Carlos Toro, Cecilia Zanni-Merk, Robert J. Howlett, and Lakhmi C. Jain. Vol. 207. Procedia Computer Science. Elsevier, 2022, pp. 987–996. DOI: 10.1016/j.procs.2022.09.154. URL: https://doi.org/10.1016/j.procs.2022.09.154.

[143] Jonas Wacker, Motonobu Kanagawa, and Maurizio Filippone. "Improved Random Features for Dot Product Kernels." In: *CoRR* (2022). arXiv: 2201.08712. URL: https://arxiv.org/abs/2201.08712.

[144] Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. "Exact Gaussian Processes on a Million Data Points." In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/01ce84968c6969bdd5d51c5eeaa3946a-Paper.pdf.

[145] Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. "Function Space Particle Optimization for Bayesian Neural Networks." In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=BkgtDsCcKQ.

[146] Philippe Wenk, Alkis Gotovos, Stefan Bauer, Nico S. Gorbach, Andreas Krause, and Joachim M. Buhmann. "Fast Gaussian process based gradient matching for parameter identification in systems of nonlinear ODEs." In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1351–1360. URL: https://proceedings.mlr.press/v89/wenk19a.html.

[147] Florian Wenzel, Kevin Roth, Bastiaan Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. "How Good is the Bayes Posterior in Deep Neural Networks Really?" In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 10248–10259. URL: https://proceedings.mlr.press/v119/wenzel20a.html.

[148] Frank Wilcoxon. "Individual Comparisons by Ranking Methods." In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83. ISSN: 00994987.

[149] Veit Wild, Motonobu Kanagawa, and Dino Sejdinovic. "Connections and Equivalences between the Nyström Method and Sparse Variational Gaussian Processes." In: *arXiv preprint arXiv:2106.01121* (2021).

[150] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[151] Andrew G. Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. "Deep Kernel Learning." In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Vol. 51. Proceedings of Machine Learning Research. Cadiz, Spain: PMLR, 2016-05, pp. 370–378.

[152] Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. "Stochastic Variational Deep Kernel Learning." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/bcc0d400288793e8bdcd7c19a8ac0c2b-Paper.pdf.

[153] Qiangfei Xia et al. "Roadmap on emerging hardware and technology for machine learning." In: *Nanotechnology* 32 (2020-07). DOI: 10.1088/1361-6528/aba70f.

[154] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 2048–2057. URL: https://proceedings.mlr.press/v37/xuc15.html.

[155] Mingzhang Yin and Mingyuan Zhou. "ARM: Augment-REINFORCE-Merge Gradient for Stochastic Binary Networks." In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=S1lg0jAcYm.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*".

Final Version as of April 25, 2023 (classicthesis v4.6).