Generalised Variational Inference posteriors in Probabilistic Deep Learning

Georgios Felekis[1]

MSc Machine Learning

Supervised by:
Theodoros Damoulas (University of Warwick)
Brooks Paige (University College of London)

September 2020

[1]**Disclaimer:** This report is submitted as part requirement for the MSc Degree in Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

In this thesis we study probabilistic Deep Learning methods with a focus on Approximate Inference and offer a comparison against state of the art methods including standard Variational Inference, Monte-Carlo Dropout, Stochastic gradient Langevin dynamics and Deep Ensembles. The focal point of our work is the empirical inspection of the performance of these techniques, compared to a recently proposed framework called Generalised Variational Inference (GVI). GVI can be seen as a generalisation of Bayesian Inference that is specified by an optimisation problem over a space of probability measures with three independent arguments: a loss, a divergence and a variational family. By advocating an optimisation-view of Bayesian Modelling, GVI posteriors can be seen as the optimal $\mathcal{Q}$-constrained solution to the optimisation problem. We motivate GVI especially in the context of Bayesian Neural Networks where, most of the time, they suffer from certain inappropriate assumptions related to the prior, the likelihood and the computational resources. We find that, in certain cases, approximate posterior distributions derived from Generalised Variational Inference offer attractive properties with respect to uncertainty quantification, consistency and predictive performance.

Acknowledgements

I would like to thank my first supervisor, Prof. Theodoros Damoulas, for his useful advice, comments and his technical and mathematical remarks during this project. I would also like to express my gratitude to him for giving me a lot of freedom regarding the path this thesis would take. Furthermore, I would like to also thank my second supervisor, Prof. Brooks Paige, for giving me precious insights of the Bayesian deep learning world throughout our constant discussions. Without his passionate participation the completion of my dissertation would not have been possible. My sincere gratitude also goes to my family and my friends for their endless support in one of the most challenging years of my life. Last but not least, I'm extremely grateful to every single difficulty and obstacle that I faced this year and made me a better person.

To S.L.

Contents

Chapter 1

Introduction

1.1 Motivation

The ultimate goal of machine learning models is to make reliable predictions and automated decisions as efficiently as possible. Machine learning and especially deep learning has attracted a lot of attention from information engineering fields such as computer vision, voice recognition, language and image processing but also from scientific and critical decision making fields such as physics, biology and medical diagnosis. Representing model uncertainty in the latter ones is of utmost importance as we do not want our model to make mistakes by any chance, thus it is clear that *it is important to know what we don't know.* Nowadays the most popular neural network architectures (ReLU networks) only return point estimates of parameters and predictions, typically lacking a representation of uncertainty, and even when they return probability values (multi class classification) they suffer from overconfidence. So, there is a need to come up with new models (or adapt the current architectures) that not only provide point estimates, but also incorporate a confidence measure. In this direction Bayesian methods provide a natural probabilistic representation of uncertainty in deep learning. Bayesian modelling is the gold standard method to capture uncertainty in a really simple way. Instead of having point estimates as an output, in the Bayesian set up the output is a posterior distribution derived by our well-known Bayes rule. Usually, this is assumed to be a Gaussian distribution in advance and so all we need to do is to predict the mean and the variance of it. The main issue of Bayesian modelling is that most of the times it is extremely computationally expensive and time consuming as most of these posterior distributions are intractable. The recent success of these methods in practice, including the creation of Bayesian Deep Learning field, relies on Approximate Inference methods. These Approximation schemes perform Bayesian reasoning at a relatively low cost (with respect to time and memory) when compared to traditional methods, as they are not computing straight the posterior distribution but instead they create approximations of it either in a stochastic

way (MCMC) or in a deterministic one (Variational Inference) and thus allow Bayesian modelling to be applied to many practical tasks. In this work we are going to mainly focus on the Variational Inference methods, first because they are currently the most popular ones in the research community, but also to appoint core issues related with these methods and hence to show their inability to appropriately tackle real-world problems. We are going to do that to motivate a recently published framework called Generalised Variational Inference which is a generalisation of standard Bayesian and Variational Inference strategies and seems to overcome a lot of drawbacks of standard Variational Inference. The main purpose of this work as someone could see in the next section is to provide a thorough empirical comparative analysis of the different Variational Inference methods and emphasize the benefit of switching to a Generalised framework under certain circumstances.

1.2 Contributions

The main contributions of this work are the following:

1. We conduct a thorough literature review of the main Approximate Inference methods and motivate a recently proposed framework in this context called Generalised Variational Inference.

2. We carry out one of the most extensive empirical analysis of different GVI settings. We extend the work of the original GVI paper to multiple divergence measures and do a grid-search for the hyperparameter-dependent divergences like α-Divergence and α-Renyi Divergence. We found certain settings of different divergences, hyperparameters and neural network depths that markedly improve the performance of GVI compared to standard VI.

3. We extend the comparison of Generalised Variational Inference via a variety of different divergences, with not only Variational Inference methods (Bayes by Backprop) but also with standard Approximate Inference methods such as Monte Carlo Dropout, Stochastic Gradient Langevin Dynamics and Deep Ensembles.

4. We present a novel study of uncertainty quantification on an artificially-generated data from a Gaussian Process kernel for Generalised Variational Inference with a variety of different divergences. We also evaluate the epistemic and aleatoric uncertainty of them and asses model calibration.

1.3 Outline

The current work is divided into three main parts:

1. The first part (chapters 2 & 3) is an extensive literature review starting from the fundamentals of deep learning, to the uncertainty in deep learning, gradually diving into the world of Bayesian and Variational Inference. More precisely, we give the motivation of switching from point estimate predictions of classical Deep Learning to predictions in the form of distributions (posteriors) and then to approximations of them due to intractability. We thoroughly present the main objective of Variational Inference methods and we analytically present three of the most famous of its methods: Bayes by Backprop, Monte-Carlo Dropout and Stochastic Gradient Langevin Dynamics.

2. In the second part (chapter 4) of the work we present and advocate for a constrained optimisation view of the standard Variational Inference setting called Generalised Variational Inference. We later prove its optimality against alternative standard VI methods under a specific optimisation foundation called Rule of Three, specified by three separate and independent arguments that define the optimisation objective: a loss, a divergence and a space of feasible solutions.

3. Finally, having introduced all these inference strategies, in the final part of the work (chapter 5) we carry out an extensive empirical comparative analysis consisting of multiple regression experiments on different datasets comparing standard Variational Inference methods with Generalised Variational Inference for a wide variety of different discrepancy measures and neural network architectures. Additionally, performance of all approaches is assessed and ideas for further improvement are tested.

Chapter 2

Deep Learning

Deep learning [1], [2] is a subset of the broader family of Artificial Intelligence and the smaller one of machine learning. It constitutes of methods based on artificial neural networks with representation learning. The term "deep" refers to the number of processing stages of the information, known as layers, as the more layers a neural network contains, the deeper it is. Nowadays, the architecture of the state-of-the-art deep learning models consists of up to hundreds of layers.

2.1 Neural networks

Artificial Neural Networks (ANNs) or simply neural networks (NNs) as they called, are general models of connections and information processing, inspired by biological nervous systems of humans. Neural networks consist of nodes (artificial neurons), which are connected by weighted edges, referred as weights, exactly as the neural connections formed by axons and dendrites. We could think of neural networks as weighted graphs with an ordered set of layers, where every layer is a set of nodes. The first layer of the neural network is called the input layer, the last one is called the output layer and all the intermediate ones hidden layers. In the vanilla case of a neural network nodes belonging to one layer are connected to all the nodes in the following and the previous layers. Each layer may perform different transformations on the inputs. Information travels from the first layer to the last layer possibly after passing through the hidden layers multiple times. As a result, neural networks constitute a structure in which information is passed between nodes of different layers along the edges. These models have the interesting property to perform tasks without being explicitly programmed with certain rules. The process of "learning" in a neural net is the continuous effort to minimise the gap between the state of the net before and after processing an example. In cases that we feed a sufficient number of data examples to the model, then the net becomes more and more capable of predicting accurate results using the associations built from the training set. There is

a direct relationship between the amount and diversity of data processed by a neural network and the accuracy of its predictions. The predictions that they are making are numerical, in the form of vectors, into which all real-world data (images, sound, text etc) should be converted. Although the main concepts and mathematical formalisation of many neural network architectures (Feed-Forward, Convolutional [3], Long short-term memory [4]) have been around for many years they enjoy a revival in the last decade, due to the vast availability of data and computational resources. The starting point of Deep learning in the mainstream is considered to be a paper of Krizhevsky in 2012 [7]. Since then a lot of new architectures have been designed like Gated recurrent unit [5] and Generative adversarial networks [6]. Neural networks proved to be extremely efficient in many tasks and applications in a variety of areas like computer vision, speech and image recognition, machine translation, medical diagnosis, or even in activities that have been considered to belong to the human nature, like music synthesis or painting.

Linear regression

In order to introduce neural networks, we should start from one of the simplest algorithms that is being used in Machine Learning: Linear regression [Gauss, 1809; Legendre, 1805; Seal, 1967].

The set up of the algorithm is the following:
We are given a data set of size N $\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\}$ in the form of input-output pairs. We want to find a linear function that maps each $x_i \in \mathbb{R}^m$ to a $y_i \in \mathbb{R}^n$. In this case, the function is going to be a linear transformation of the inputs in the form of:

$$y = f(x) = Wx + b$$

with W some $m \times n$ matrix and b the bias term which is a real vector with n elements. Obviously, different parameters setups define different linear functions. Our goal is to find the best parameters that will fit the data and will be also able to generalise well. By the word "best" we mean that we want to find the parameters that would minimise a loss function. For example, one of the most popular is the Mean Squared Error:

$$\mathcal{L}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} ||y_i - (x_i \mathbf{W} + \mathbf{b})||^2$$

Perceptron

One of the first milestones in the machine learning world was the Perceptron model [8] in the late 1950s. The perceptron can be seen as the foundation of modern neural networks and is usually used to classify the data into two parts. Therefore, it is also known as a Linear Binary Classifier. More specifically, it can be shown that, using the Perceptron learning rule, any linearly separable

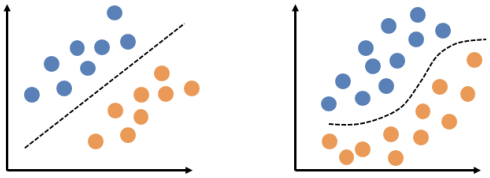


Figure 2.1: Two sets of binary-labeled dataset. Linearly seperable (left), Not linearly seperable (right).

binary classification problem can be learned by the algorithm. The parameters of the Perceptron are the weights $\mathbf{w} \in \mathbb{R}^n$ and the biases $b \in \mathbb{R}$. For an input $\mathbf{x} \in \mathbb{R}^n$ it returns a binary $y \in \{0, 1\}$ according to the following function:

$$y = \begin{cases} 1 & \mathbf{wx} + b \geq 0 \\ 0 & \mathbf{wx} + b < 0 \end{cases}$$

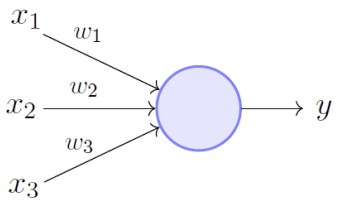


Figure 2.2: Perceptron model, Minsky-Papert in 1969

Multilayer Perceptron

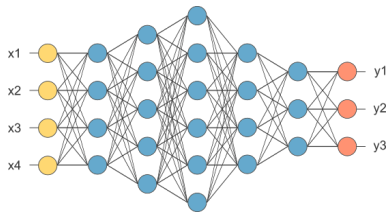


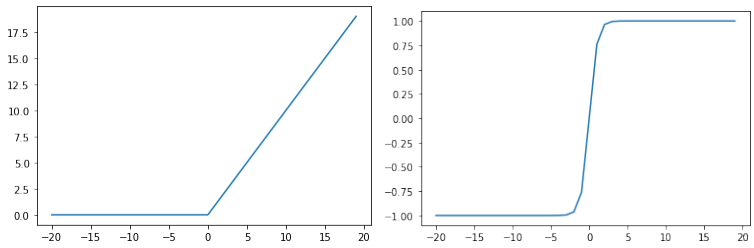
Figure 2.3: Multilayer Perceptron model

An extension of the Perceptron is the Multilayer Perceptron (MLP) or, as it is also called, Feedforward Neural Network. MLPs consist of multiple layers, each one of which consists of several Perceptrons each. We will first review a neural network model [9] for a single hidden layer and the following generalisation to multiple hidden layers will be straightforward. Let x be the model

input (referred to as input layer, a row vector with n elements), and transform it with an affine transformation to a row vector with K elements. We denote by W_1 the weight matrix, which is a linear map, and by b the bias term used to transform the input x to obtain $xW_1 + b$. Since the concatenation of linear transformations results as well in a linear transformation, an element-wise non-linearity $\sigma(\cdot)$ activation function (such as the rectified linear unit ReLU[1] or TanH) is then applied at the end of that layer, resulting in a hidden layer with each element referred to as network unit. This is followed by a second linear transformation with weight matrix W_2 mapping the hidden layer to the model output (referred to as output layer, a row vector with D elements). We thus have W_1 to be a $Q \times K$ matrix, W_2 to be a $K \times D$ matrix, b_1 is a K dimensional vector and b_2 is a D dimensional vector. Hence for a given input, a standard network would output

$$y = \sigma(xW_1 + b_1)W_2 + b_2$$

It is clear now that if we have L layers then the network's final output would



(a) ReLU (b) Tanh

Figure 2.4: Activation functions

be:

$$\hat{y}(x) = \sigma(...\sigma(\sigma(xW_1 + b_1)W_2 + b_2)...)W_L + b_L$$

where W_i and b_i the weight matrix and the bias term respectively of the $i-th$ layer. Now, based on the task that we want to do we define a different loss function:

- For regression tasks we usually use the Euclidean loss:

$$\mathcal{L}(\mathcal{D}_{train}) = \frac{1}{2N}\sum_{i=1}^{N} ||y_i - \hat{y_i}||^2$$

  where $(y_1, ..., y_N)$ are the N observed outputs and $(\hat{y_1}, ..., \hat{y_N})$ the model's outputs given the inputs $(x_1, ..., x_N)$. By minimizing $\mathcal{L}$ with respect to $\mathbf{W_i}$ and $\mathbf{b_i}$ we increase the model's accuracy in the predictions and make it more able to generalise well to unseen test data $\mathcal{D}_{test}$.

[1] $relu(x) = max(0, x)$

- For classification tasks we usually use the Softmax loss:

$$\mathcal{L}(\mathcal{D}_{train}) = \frac{1}{N}\sum_{i=1}^{N} log(p_{i,c_i})$$

where for a set of classes $\mathcal{C} = \{1, ..., C\}$ $c_i \in \mathcal{C}$ is the observed class for input x_i and p_{c_i} is the probability of this input being classified with a label in $\mathcal{C}$ given from the softmax function as follows:

$$p_c = \frac{exp(\hat{y_c})}{\sum_{i=1}^{C} exp(\hat{y_i})}$$

where $\hat{\mathbf{y}}$ is the model's output vector.

Backpropagation

In neural networks, learning is conducted by minimising the discrepancy between the predicted output and the target one according to the loss function equation. After a single forward pass through the neural network, the output will be a predicted value $\hat{y}$ and the loss value will be computed as we saw above. Now, given this loss value, we must update all parameters in the layers of the neural network. In order to do this, for any given layer l, we update the parameters by using the *Gradient Descent* algorithm with the following formulas:

$$W^{[l]} = W^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial W^{[l]}}$$

$$b^{[l]} = b^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial b^{[l]}}$$

where α is the learning rate.
Of course we assume that in this case the network uses differentiable activation functions and hence the error term is also differentiable. By making use of our well known chain rule we can successively calculate the gradients starting from the output layer and go all the way back to the first layer. This is why we call this algorithm backpropagation [10].

Overall, the learning process of a neural network consists of the following steps:

1. Randomly initialize the network's parameters (weight matrices and biases).

2. Input a set of the data points and pass them through the network's layers all the way to the output layer and get the network output.

3. Compare this obtained output with the values of the labels and compute the loss value for them.

4. Perform backpropagation to update the parameters of the neural network according to their partial contribution to the loss value with gradient descent, thus making sure that the total loss is reduced and a better model is obtained.

5. Keep iterating in the previous steps until convergence i.e. until the loss is considerably small or some criterion is met (maximum number of updatating steps, early stopping, etc).

Stochastic Gradient Descent (SGD)

The gradient descent algorithm states that we have to go through the entire dataset to take just one optimisation step. As we understand, this would have a high computational cost if the size of the dataset is large. However, there is an alternative algorithm that resolves this problem by updating the parameters whenever we encounter a new training example. Instead of computing the loss over the whole batch, we use the loss function over a single sample. **This algorithm is called Stochastic Gradient Descent (SGD)** [11], [12].

Algorithm 1 Stochastic Gradient Descent

Initialize: $W^{[l]}, b^{[l]}, \forall l \in L$

Repeat:

randomly choose a training example $(x_i, y_i) \in \mathcal{D}$

$$W = W - s\frac{\partial \mathcal{L}(f(x_i), y_i)}{\partial W}$$
$$b = b - s\frac{\partial \mathcal{L}(f(x_i), y_i)}{\partial b}$$

For SGD we have to take into consideration the step size, which in practice works well if it is fixed, but is also a common technique to use decreasing step sizes (dampens noise in step direction). Many classical convergence results depend on the so called *Robbins-Monro Conditions*:
For s_t the step size at the t-th step:

$$\sum_{t=1}^{\infty} s_t^2 < \infty \qquad \sum_{t=1}^{\infty} s_t = \infty$$

Finally, we understand that SGD requires a very large number of steps before even going through the whole dataset once. In order to solve this there is a smart algorithm that combines Gradient Descent and Stochastic Gradient Descent. **Mini-Batch Gradient Descent** randomly shuffles the dataset and

divides it into smaller batches with a common size. For every epoch of the training we have a new shuffling and a new partition of the training samples. With Mini-Batch Gradient Descent the model update frequency is higher than the regular gradient descent and the updates are smoother than SGD since the loss values are averaged across different data points, thus allowing for a more robust convergence.

Regularisation

A big issue that we often face in all the deep learning models is their tendency to overfit. Overfitting happens when a model learns too well the details and noise of the training data to the extent that it cannot generalize well for new unseen inputs. This is usually seen as the case where $\mathcal{L}(\mathcal{D}_{train})$ decreases whereas $\mathcal{L}(\mathcal{D}_{test})$ increases.
In order to prevent overfitting, a lot of regularisation techniques are used during optimisation like L_2-regularisation or Dropout [13]. For example, in L_2-regularisation we add a regularisation term for each parameter weighted by some weight decays λ_i for each layer's weights W_i, and hence the minimisation objective (for a neural network with L layers) becomes:

$$\mathcal{E}(\mathbf{W}, \mathbf{b}) = \mathcal{L}(\mathcal{D}_{train}) + \sum_{i=1}^{L} \lambda_i ||W_i||^2 + \lambda ||\mathbf{b}||^2$$

The equation above is known as the cost function of the neural network.
On the other side, Dropout introduces random perturbation during training as before every learning step,every input or hidden node is "disabled" or dropped with a given probability p. We are going to present the more in-depth mathematical modelisation of dropout in Chapter 3.

Universal Approximation Theorem (UAT)

One of the most centric results in the field of Theoretical Machine Learning is the Universal Approximation Theorem [14], [15] which literally constitutes the theoretical foundation of why neural networks work. According to the theorem, a neural network with one hidden layer containing a sufficient but finite number of neurons can approximate any continuous function on a compact (bounded, closed) set to a reasonable accuracy, under certain conditions for activation functions (being piecewise, sigmoidal etc). The brilliant point of the Universal Approximation Theorem is not defining complex mathematical relationships between the input and output, but instead using simple linear manipulations to divide the complicated function of the neural network into smaller, less complicated pieces, each of which are taken by one neuron. There are two main directions on the Universal Approximation research. The one that focuses on the approximation capabilities of neural networks with an arbitrary number of artificial neurons and the other that focuses on the case where there is an arbitrary number of hidden layers. Below we provide the original formal mathematical definition of UAT by Cybenko:

Theorem (UAT): Let $\mathcal{I}_n = [0,1]^n$ to be the n-dimensional unit cube and $C(\mathcal{I}_n)$ the space of continuous functions on $\mathcal{I}_n$ with the supremum norm $||\cdot||$. Also let σ be any continuous sigmoidal function. Then, finite sums of the form:

$$G(x) = \sum_{i=1}^{N} \alpha_i \sigma(w_i^T x + b_i), \quad \text{where } w_i \in \mathbb{R}^n \text{ and } \alpha_i, b_i \in \mathbb{R}$$

are dense in $C(\mathcal{I}_n)$.

In other words, given any $\epsilon > 0$ and $f \in C(\mathcal{I}_n)$, there is a sum $G(x)$ of the above from such that:

$$|G(x) - f(x)| < \epsilon, \quad \forall x \in \mathcal{I}_n$$

Of course, UAT is a theoretical result as in the real world it's infeasible to

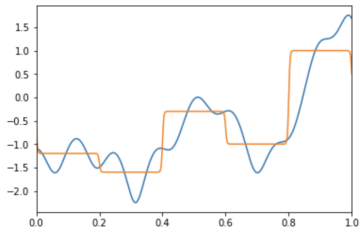


Figure 2.5: Approximating the weighted output of the hidden layer using sigmoid functions with very large weights.

continuously add neurons or layers to infinity in a neural network. However is up to the intuition and experience of each machine learning engineer to build proper neural network architectures for certain tasks, knowing that such a network exists.

2.2 Uncertainty in Deep Learning

In analysing data or making decisions, it is necessary to be able to tell how certain our model is about its output predictions, so as to transform it in a way that describes our observations as accurately as possible and be able to generalise well for new inputs. As it is normal understanding, if a model is underconfident or falsely over-confident can help us improve its performance by taking answers to certain questions like: How reliable are my observations? Should we use more diverse data? Should we change the model structure? However, with the modern deep learning models we mainly have only point estimates and predictions over the parameter space. Hence, it is not possible to answer the questions above, since most of the times we are not sure whether

a model is making sensible predictions or just random guesses. The possitive thing with this type of models is that they are tightly related to a family of probabilistic models which induce probability distributions over functions: the Gaussian process. A well known result is that given a one layer neural network with infinitely many weights, and placing a probability distribution over its parameter space then this would be equivalent to Gaussian process (see [16] and [17]). In the case that the parameter space is finite then we can still obtain model uncertainty in the form of a model called **Bayesian neural network** [18], [16]. The key distinguishing property between a standard neural network and a Bayesian one is that in the latter one, instead of having the parameters and the predictions to be represented by single, fixed values (point estimates), they are represented by distributions. In other words Bayesian approach is based on marginalisation rather than using a single setting of weights. Also, in Bayesian Neural Networks we introduce a prior distribution on the weights $p(W)$ and obtain the posterior distribution $p(W|D)$ through Bayesian learning (we are going to see how this works in practice in chapter 3). Wenzel et. al [19] provide evidence that a Gaussian prior choice of $p(W) \sim \mathcal{N}(0, I)$ is misspecified to any serious extent whereas the choice of $p(W) \sim \mathcal{N}(0, \alpha^2 I)$ for a range of α (easily determined through cross-validation) can provide high entropy across weights and is the most popular setting. Finally, in Bayesian Neural Networks regularisation arises naturally through the prior $p(W)$. Specifically, it has been proved that:

- A **Uniform** prior choice leads to a Maximum Likelihood training.
- A **Laplace** prior choice leads to a training with L1-regularisation.
- A **Gaussian** prior choice leads to a training with L2-regularisation.

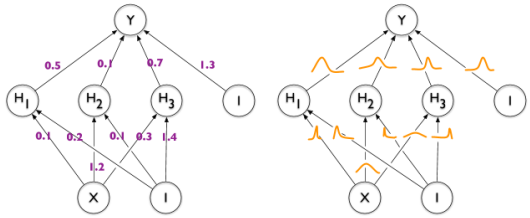


Figure 2.6: Standard NN (left) and Bayesian NN (right) [Blundell, C. et al. Weight Uncertainty in Neural Networks]

Recently, there's a lot of progress on these ideas mainly through variational techniques by Blundell [20], Graves [21], Kingma and Welling [22]. In general, we have two types of uncertainty that we have to take into consideration:
Epistemic uncertainty accounts for uncertainty in the model and is due to limited data and knowledge. Specifically, in the deep learning models epistemic uncertainty has to do with the uncertainty in model parameters that best explain the observed data and the architecture of the model. A larger number of

data points is able to decrease this kind of uncertainty.
The second type of uncertainty is the one arising from the natural stochasticity of observations and is called *Aleatoric uncertainty*. In other words, it is the uncertainty that is increased when our observed labels are noisy. This type of uncertainty cannot be reduced even when we increase the number of our data samples. We often distinguish between two types of aleatoric uncertainty. *Homoscedastic* uncertainty, which denotes the variance that stays constant for all inputs and *Heteroscedastic* uncertainty, which denotes the variance that can possibly vary for different inputs and can also be predicted as a model output. For example in a regression task, typically the model has the following form:

$$y(x) = f(x) + \sigma(x)\epsilon$$

where $f(x)$, $\sigma(x)$ are deterministic functions and ϵ is a noise variable (e.g. standard gaussian noise). Now, $\sigma(x)$ is a function that tells us you how strong the noise is at each x. If $\sigma(x)$ is constant, the model is homoscedastic, whereas if $\sigma(x)$ is allowed to vary, then it is heteroscedastic. Overall, aleatoric and epistemic uncertainty can then together express the overall predictive uncertainty, and hence the confidence we have in our predictions.

Chapter 3

Inference

3.1 Maximum Likelihood Estimation

A model P is a set of probability distributions where we index each distribution by a parameter value $\theta \in \Theta$. The model can be written as:

$$P = \{P_\theta | \theta \in \Theta\}$$

where the set Θ is called the parameter space of the model. In the parametric models, we usually notate P_θ as a probability density function $p(x|\theta)$.
For the Maximum Likelihood Estimation [R. A. Fisher 1912] set up, given a data set $\mathcal{D} = \{x_1, x_2, ..., x_n\}$ and a parametric model $P = \{p(x|\theta)|\theta \in \Theta\}$ we want to find the best distribution in P (i.e. the one which best explains the data set). More specifically we want to find the parameter $\hat{\theta}$ which maximizes the joint density of the data. Therefore, the maximum likelihood estimator is defined as:

$$\hat{\theta}_{ML} = \underset{\theta \in \Theta}{\operatorname{argmax}} p(x_1, x_2, ..., x_n|\theta)$$

Most of the times we make the assumption that the data is independent and identically distributed (i.i.d.). This means that the joint density decomposes as:

$$L(\theta) = p(x_1, x_2, ..., x_n|\theta) = \prod_{i=1}^{n} p(x_i|\theta)$$

We call the latter one the *likelihood function* and, thus, the maximum likelihood estimate is given by

$$\hat{\theta}_{ML} = \underset{\theta \in \Theta}{\operatorname{argmax}} L(\theta)$$

Instead of maximizing the likelihood function, we often maximize the log-likelihood:

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log p(x_i|\theta)$$

which results in the same estimate because the logarithm function is monotonically increasing.
The analytic criterion for a maximum likelihood estimator (under the i.i.d. assumption) is:

$$\frac{\partial}{\partial\theta}L(\theta) = 0$$

3.2 Bayesian Inference

As we saw Maximum Likelihood estimation is a point estimate prediction of the optimal model's values $\hat{\theta}$ and only computes the value that maximises the likelihood function. In contrast, the Bayesian inference setting, inspired by the principles of Bayesian statistics, take into consideration prior beliefs for the model parameters, i.e. beliefs about the distribution of the parameters before seeing any data. Specifically, a bayesian model consists of two pieces:

- A parametric family $\{p(D|\theta)|\theta \in \Theta\}$ of likelihood functions.
- A prior distribution $p(\theta)$.

Of course our ultimate goal is to approximate the value of $p(\theta|D)$, known as the posterior distribution, which represents the updated beliefs after seeing the data D. The easiest way to do this is by using the famous *Bayes rule*:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

where $p(D)$ is called the marginal likelihood or evidence.
For a fixed dataset D we can drop factors that are independent to θ and hence:

$$p(\theta|D) \propto p(D|\theta)p(\theta)$$

The posterior distribution can now be used to obtain a single point estimations, either by by evaluating the posterior mean:

$$\theta^* = \mathbb{E}[\theta|D]$$

or by taking the *Maximum a Posteriori* estimate:

$$\theta^* = \underset{\theta\in\Theta}{\text{argmax}} p(\theta|D)$$

Finally, we can also make predictions based on this Bayesian setting by taking the following formula:

$$p(x_{new}|D) = \int p(x_{new}|\theta, D)P(\theta|D)d\theta$$

Here it is important to mention a few thing about the prior choice. We can divide priors in different categories as follows:

- Objective priors: These are non informative priors that attempt to capture ignorance.
- Subjective priors: These capture our beliefs as well as possible.
- Hierarchical priors: These constitute of different levels of priors (i.e. prior for parameters, prior for their hyperparameters, etc).
- Empirical priors: These priors, in contrast to standard Bayesian strategy where the prior distribution is fixed before any data are observed, they are estimated from the data.

Conjugate priors

Although the process seems quite straightforward, updating the prior to form the posterior is not an easy task for most of the times. Actually, the task is really difficult and the posterior is usually intractable if we don't use what is a called conjugate prior.
Let π be a family of prior distributions on θ and P a parametric family distributions with parameter space Θ.

Definition: A family of distributions π is *conjugate* to model P for any prior in π if the posterior is always in π.

Choosing a conjugate prior is computational efficient, and depending on the parameters a conjugate prior can be empirical, or non-informative, etc. However, hierarchical priors probably are not conjugate.

3.3 Variational inference

Although Bayesian inference seems to provide more information than the MLE, as we do not have just point estimates for parameters and also for future predictions, we are often faced with analytical solutions for the posterior distribution $p(\theta|D)$ that are intractable, caused mainly by the evidence term of the Bayes rule, especially in higher dimensions (curse of dimensionality). When we refer to intractable quantities, we mean that they are either unavailable in closed form or that they do not fall in the Cobham–Edmonds thesis [23] that states that tractable problems are the ones that require polynomial time to be computed that is, if they lie in the complexity class $\mathcal{P}$. and therefore cause a problem in the approximate inference process. As a result, we cannot compute the posterior for many interesting models. Thus, it is important to find other ways to approximate the true posterior distribution. One way is to apply the widely used Markov Chain Monte Carlo (MCMC) sampling methods that by simulating a large number of samples from a distribution that approaches the true posterior [24], [25]. In the MCMC approach we simply sample from an ergodic Markov chain whose stationary distribution is $p(Z|D)$ where $Z = \{z_1, z_2, ..., z_m\}$ latent

variables and then we approximate the exact posterior with an empirical estimate from a subset of the collected samples. These methods provide really good solutions in this direction but as it is reasonable they are extremely computationally expensive.
Variational Inference is an alternative approach for approximating the true posterior distribution and is proved to be faster and easier to scale to large data. Instead of sampling, Variational Inference is based on optimisation. The main idea is the following:

Main Idea:
For observations $D = \{x_1, x_2, ..., x_n\}$ and latent variables $Z = \{z_1, z_2, ..., z_m\}$:

- We pick a restricted family of distributions $\mathcal{Q}$ (approximate densities) over the latent variables with each own parameters,

$$q(Z|\kappa)$$

- Then, we seek the member of the family that makes q close enough to the true posterior.
- Finally, we approximate the posterior with the optimal member of the family $q^*(\cdot)$.

For the restricted variational family we have to keep in mind that it is important for the family to be restricted sufficiently in order to have only tractable functions but also it should be "rich" enough so it can provide a good approximation to the true posterior.

Kullback-Leibler Divergence

It is vital to mathematically define what do we mean by "close enough to the true posterior". In Variational Inference we measure the closeness of two distributions with a statistical measure of discrepancy called Kullback-Leibler Divergence (KL). The Kullback–Leibler divergence was introduced by Solomon Kullback and Richard Leibler in 1951 as the directed divergence between two distributions [40].

The KL divergence for Variational inference is given by the following equation:

$$KL(q||p) = \mathbb{E}_q \left[log \frac{q(Z)}{P(Z|D)} \right] = \int_Z q(z) \log \frac{q(z)}{p(z|x)} dz \quad (3.1)$$

We can easily see that:

$$KL(q||p) = 0 \iff q \equiv p$$

Hence, we want to find the member of $\mathcal{Q}$ that minimizes the KL divergence to the exact posterior.

$$q^*(z) = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} KL(q(Z)||p(Z|D)) \quad (3.2)$$

Evidence Lower Bound (ELBO)

However, the truth is that we actually cannot minimize the KL divergence direct, and thus we minimize a function that is equal to it up to a constant. This is the evidence lower bound (ELBO). Specifically,

$$KL(q||p) = \mathbb{E}_q\left[\log \frac{q(Z)}{P(Z|D)}\right] = \mathbb{E}_q[\log q(Z)] - \mathbb{E}_q[\log p(Z|D)]$$

$$= \mathbb{E}_q[\log q(Z)] - \mathbb{E}_q[\log p(Z, D)] + \log p(D) \quad (3.3)$$

This reveals the dependence on $\log p(D)$.
As we said we cannot compute the KL divergence and instead we optimize an alternative objective, which is the following:

$$ELBO(q) = \mathbb{E}_q[\log p(Z, D)] - \mathbb{E}_q[\log q(Z)] \quad (3.4)$$

It is obvious that the ELBO is the negative KL divergence of equation (3.3) plus $\log p(D)$, which is a constant w.r.t q(Z) and hence, minimizing the KL divergence is the same as maximizing the ELBO. Here we examine two of the most insightful properties of the ELBO:

1. If we rewrite equation (3.4) we have:

$$ELBO(q) = \mathbb{E}_q[\log p(Z)] + \mathbb{E}_q[\log p(D|Z)] - \mathbb{E}_q[\log q(Z)]$$

$$= \mathbb{E}[\log p(D|Z)] - KL(q(Z)||p(Z))$$

   By making the ELBO the sum of these two proportions we can see that the $\mathbb{E}[\log p(D|Z)]$ term encourages densities that place their mass on configurations of the latent variables that best explain the data, whereas, the $KL(q(Z)||p(Z))$ term encourages densities close to the prior. So, we have this trade off between how much I should count on the data and how much on my beliefs about the model.

2. Furthermore, the ELBO is a lower bound on $\log p(D)$ because the Kullback-Leibler divergence is always a non-negative quantity:

$$\log p(D) \geq ELBO(q)$$

   because $\log p(D) = KL(q(Z)||p(Z||D)) + ELBO(q)$.
   The later one could be also derived through Jensen's inequality [27]:

$$f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$$

   It is interesting that in the literature the variational bound between the ELBO and $\log p(D)$ has been used as a model selection criterion. Although selecting a model based on a lower bound seems quite odd, in practice it proved to be really efficient as is a good approximation of the marginal likelihood, which provides a basis for selecting a model.

Mean-Field Variational Family (MFV)

Previously, we defined the optimisation objective, which is the maximisation of the ELBO. Now, it's important to describe the variational family $\mathcal{Q}$ that we keep refering to it, in order to complete the specification of the optimisation problem. The most standard type of family for variational inference is the Mean-Field variational family. In this type of variational inference, we assume the variational distribution over the latent variables factorizes as:

$$q(Z) = \prod_{i=1}^{m} q_i(z_i)$$

and we refer to $q_i(z_i)$ as the local approximation for a single latent variable. We can also "group" the latent variables as follows:
Suppose we partition the elements of Z into k-disjoint groups denoted by $Z_i, i = 1, 2, ..., k$ and $k \leq m$. This means that the latent variables are mutually independent and certain groups of them governed by a distinct factor in the variational density. Then a member of this family has the following form:

$$q(Z) = \prod_{i=1}^{k} q_i(Z_i)$$

Now, amongst all distributions $q(Z)$ we seek the one that maximizes the ELBO of equation (3.4). Hence, we want to make optimisation of $ELBO(q)$ w.r.t all $q_i(Z_i)$. Finally, we have to mention that we do not need to specify a certain parametric form for the individual variational factors. This depends on the corresponding random variable of each problem. The advantage of the MFV is that it can capture any possible marginal density of the latent variables but, on the other hand, it completely fails to appoint correlations between them.
We now want to optimize the ELBO in the MFV inference set up. The most popular optimisation algorithm for this task is the Coordinate Ascent Variational Inference (CAVI) optimisation [41], where we iteratively optimize each factor of the MFV density while holding the others fixed until the ELBO converges to a local optimum. CAVI can also be seen as a "message passing" algorithm [42] as it updates each variable's parameters based on the parameters of the variables in its Markov blanket.

Below we present the CAVI algorithm as this is presented in Bishop 2006:

Algorithm 2 Coordinate Ascent Variational Inference

Input: A model $p(D, Z)$ and a dataset D.

Output: A variational density $q(Z) = \prod_{i=1}^{m} q_i(z_i)$

Initialize: Variational factors $q_i(z_i)$
While the ELBO has not converged **do**:

. **for** $i \in \{1, 2, ..., m\}$ **do**:
. Set $q_i(z_i) \propto exp\{\mathbb{E}_{-i}[\log p(z_i|z_{-i}, D)]\}$
. **end**

Compute $ELBO(q) = \mathbb{E}[\log p(Z, D)] - \mathbb{E}[\log q(Z)]$
end
Return: q(Z)

An important observation is that the function of ELBO is most of the times a non-convex function. Hence, CAVI only guarantees convergence to a local optimum, which can be sensitive to initialisation. In that sense, multiple random initialisations can proved to be extremely efficient.
Finally, even if there are also other algorithms to optimize the ELBO (i.e. gradient ascent, using the natural gradient) CAVI is nowadays the most used one by the research community these days.
Overall, the task of Variational Inference is to approximate a conditional density of latent variables given observed ones, the exact posterior. Instead of solving this inference problem with sampling methods as MCMC, it solves it via an optimisation approach. By taking a restricted variational family, parametrised by "variational parameters", variational inference finds the the optimal setting of the parameters which is closest to the true posterior distribution by minimizing the KL divergence.

Research on Variational Inference

Variational inference methods for Bayesian modelling can be divided into two separate tracks that have been developed in parallel. The one of Peterson and Anderson [26] which was the first variational method for a neural network model and a model in general, while others that followed this research track were Ghahramani et al. [27]. The other track was the one of Hinton and Van Camp [28] and Hinton and Neal [29]. The latter ones made crucial connections of the variational inference methods to the Expectation Maximisation algorithm. As a result, they paved a new course for the application of VI to other types of models [30], [31] and [32]. In recent years modern variational inference research differs significantly from earlier formulations. The main focus is now on scalable approaches due to the availability of massive datasets [33], [34]. Also there is a lot of research going on around black box VI algorithms [35] apart from the

restricted class of conjugate exponential family models that we only mentioned here. In addition, lately a lot of work has been done on modern Bayesian deep learning architectures such as Variational Autoencoders (VAE) . Finally, this trend of generalisation has motivated researchers to start looking at different divergence measures [36], [37], [38] and structured variational families [39] on the top of the typical setting of KL divergence and MFV family. An extensive overview of the foundation and all the modern alternative VI methods can be found in the exceptional work of Blei, Kucukelbir and McAuliffe in [43].

3.3.1 Methods

Bayes by Backpropagation

The first of the methods that we are going to examine is the Bayes by Backprop (BBB) algorithm [20]. BBB is a a variational backpropagation inference scheme for learning posterior distribution over the weights of a neural network. It does this by regularising the weights with the minimisation of a compression cost known as Variational Free Energy. BBB provides richer representations and predictions compared to the ones of standard model averaging. Typically this posterior distribution is taken to be a Gaussian $\mathcal{N}(\theta|\mu, \sigma^2)$ and our goal is to infere μ and σ^2 by using a diagonal covariance matrix.
BBB compared to the CAVI algorithm that we introduced above has the advantage that can easily be applied to "bigger" models, and models which do not have conjugate exponential family relationships between the different distributions (e.g. with neural networks as deterministic components of the model), but on the other hand it doesn't easily applicable to models with discrete latent variables or non-differentiable deterministic functions because it requires reparameterization. Therefore it's a tradeoff between requiring conjugate exponential families (CAVI) and requiring continuous latent variables and differentiability (BBB).
As we mentioned before the network is trained by minimizing the variational free energy:

$$\mathcal{F}(\theta) = \mathbb{E}\left[\log \frac{q(\theta)}{p(y|\theta, D)p(\theta)}\right]$$

where $p(y|\theta, D)$ is the log-likelihood of the model and $p(\theta)$ the prior on the parameters.
Equivalently,

$$\mathcal{F}(\theta) = KL(q(\theta)||p(\theta)) - \mathbb{E}_q[\log p(y|\theta, D)]$$

Here we can observe that the first term is a prior dependent term whereas the second one a data-dependent term, representing the trade off between satisfying the complexity of the data and the simplicity of the prior.
We approximate our cost function using sampled weights:

$$\mathcal{F}(\theta) = \sum_{i=1}^{n} \log q(w^{(i)}|\theta) - \log p(w^{(i)}) - \log p(D|w^{(i)})$$

where $w^{(i)}$ is the i-th Monte Carlo sample drawn from the variational posterior $q(w^{(i)}|\theta)$. Below we present the algorithm of how we can learn both the mean and the standard deviation using the parametrisation trick (Step 2) as this is defined in the original paper.

Algorithm 3 Bayes by Backprop

0. Sample $\epsilon \sim \mathcal{N}(0, I)$ and define a learning rate α.
1. Let the variational parameters to be: $\theta = (\mu, \rho)$
2. Let $w = \mu + log(1 + exp(\rho)) \cdot \epsilon$
3. Let $f(W, \theta) = \log q(W|\theta) - \log p(W)p(D|W)$
4. Calculate the gradients with respect to θ:

$$\Delta_\mu = \frac{\partial f(W,\theta)}{\partial W} + \frac{\partial f(W,\theta)}{\partial \mu}$$

$$\Delta_\rho = \frac{\partial f(W,\theta)}{\partial W} \frac{\epsilon}{1 + exp(-\rho)} + \frac{\partial f(W,\theta)}{\partial \rho}$$

5. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha\Delta_\mu$$

$$\rho \leftarrow \rho - \alpha\Delta_\rho$$

Note that, $\frac{\partial f(W,\theta)}{\partial W}$ is the derivatives that we find by the usual backpropagation algorithm.

Monte-Carlo Dropout

As we briefly saw in chapter 2 *dropout* [13] is a widely used regularisation technique in neural networks that helps them to avoid overfitting. The initial implementation of dropout only applies in the training time and not in the test time. The formal way to introduce dropout is the following:
For a one hidden layer MLP with input dimension N, weight matrix for the input layer W_1, hidden layer dimension K with weight matrix W_2 for it and a bias term b:

1. We sample two binary vectors $\epsilon_1 \in \mathbb{R}^{N\times 1}, \epsilon_2 \in \mathbb{R}^{K\times 1}$. The elements of these vectors take value 0 with probability p_i, where $0 \leq p_i \leq 1, i = 1, 2$. Hence, $\epsilon_i \sim Bernoulli(p_i)$.

2. For an input x we set $x = x \cdot \epsilon_1$. Therefore, the output of the first layer is given by:
$$h = \sigma(xW_1 + b)$$

3. Then we again apply p_2 and now $h = h \cdot \epsilon_2$.

4. Finally, we get the model's output from the following equation:

$$y = hW_2$$

For every input data point we have different samples for ϵ_i and use the same values for the backpropagation in order to optimize model parameters $\theta = \{W_1, W_2, b\}$.
The generalisation to more hidden layers is straightforward as we just repeat the previous chain rule.
However, Monte-Carlo Dropout [44], which was first introduced in [45] uses the dropout method to introduce uncertainty to the predictions by evaluating the same input multiple times, resulting in a set of different predictions that can be used to estimate a distribution of the posterior instead of a single point estimate by returning its mean and standard deviation values. One interesting result is that a neural network that is already trained with dropout is a Bayesian Neural Network and has all its properties. In other words the optimal weight values that we found from the optimisation of a dropout neural network are similar to the variational ones of a Bayesian Neural Network. Specifically, in [45] a transformation of dropout's noise from the feature space to the parameter space, where the uncertainty comes from was proposed in the Bayesian Neural networks, as follows:

$$y = hW_2 = (h \cdot \epsilon_2)W_2 = (h \cdot diag(\epsilon_2))W_2 = h(diag(\epsilon_2)W_2)$$

$$= \sigma(xW_1 + b)(diag(\epsilon_2)W_2) = \sigma((x \cdot \epsilon_1)W_1 + b)(diag(\epsilon_2)W_2)$$

$$= \sigma(x(diag(\epsilon_1)W_1) + b)(diag(\epsilon_2)W_2)$$

By now writing, $\hat{W}_1 = diag(\epsilon_1)W_1$ and $\hat{W}_2 = diag(\epsilon_2)W_2$ we have :

$$\hat{y} = \sigma(x\hat{W}_1 + b)\hat{W}_2$$

Recall the standard neural network's objective from chapter 2. For the MC Dropout we end up with a loss function of the following form:

$$\mathcal{L}_{\text{dropout}}(W_1, W_2, b) = \frac{1}{N}\sum_{i=1}^{N} ||\hat{y}(x_i) - y_i||^2 + \lambda_1||W_1||^2 + \lambda_2||W_2||^2 + \lambda_3||b||^2$$

where $\lambda > 0$, is a weight decay factor. It has been shown (Srivastava et al 2014 7.5) that MC Dropout can be approximated by averaging the weights of the network by multipling with p_i at test time all the weight matrices W_i. In the literature this is known also as Bayesian Model Averaging and in our case uncertainty can by obtained by performing T stochastic forward passes and averaging the results. The mean prediction is estimated by averaging these passes while the uncertainty is estimated by computing the variance of them which is equivalent to the model's predictive variance. Specifically, while we are doing inference:

Let $\hat{y}_t^*$ the sampled outputs via MC Dropout and recall that our approximate predictive distribution is given by:

$$q_\theta^*(y^*|x^*) = \int p(y^*|\hat{y}_t^*) q_\theta^*(\omega) d\omega$$

where $\omega = \{W_i\}_{i=1}^L$.
Then, given $p(y^*|\hat{y}_t^*) \sim \mathcal{N}(y^*; \hat{y}_t^*, \tau^{-1} I)$ for some $\tau > 0$:

$$\mathbb{E}[y^*] = \frac{1}{T} \sum_{i=1}^{T} \hat{y}_t^*(x^*)$$

and

$$Var[y^*] = \tau^{-1} I + \frac{1}{T} \sum_{i=1}^{T} \hat{y}_t^*(x^*)^T \hat{y}_t^*(x^*) - \mathbb{E}[y^*]^T \mathbb{E}[y^*]$$

with $\tau = \frac{(1-p) l_i^2}{2N\lambda_i}$ the model precision, where λ is the weight decay factor, l_i is the so called prior length scale and p the probability of units not being dropped.
In the same work [45] Gal came up with a condition (KL condition) under which MC-Dropout and standard VI are identical optimisation procedures. In particular, we recall that the objective of standard VI is the maximisation of the ELBO, or equivalently the minimisation of the negative ELBO:

$$-ELBO(q) = -\mathbb{E}[\log p(D|Z)] + KL(q(Z)||p(Z))$$

Hence, in our notation:

$$\mathcal{L}_{\text{VI}} = -\sum_{i=1}^{N} \int q_\theta(\omega) \log p(y_i|\hat{y}_i) d\omega + KL(q_\theta(\omega)||p(\omega))$$

then KL condition is the following:

$$\frac{\partial}{\partial\theta} \mathcal{L}_{\text{Dropout}} = \frac{1}{N\tau} \frac{\partial}{\partial\theta} \mathcal{L}_{\text{VI}}$$

Stochastic gradient Langevin dynamics

Another widely used novel method is Stochastic Gradient Langevin Dynamics - SGLD [46] which bridges optimisation and Bayesian learning approaches. This framework combines stochastic optimisation(Robbins 1951) with Langevin Dynamics (Neal 2010) to learn from large scale datasets. The difference of the two approaches is that Stochastic Optimisation (SO) looks for the MAP solution, whereas Langevin Dynamics (LD) seeks to simulate from the posterior distribution. Let's see the main idea of the algorithm:

Given a dataset $D = \{x_1, ..., x_n\}$, where for every iteration t we pick a subset $X_t \subseteq D$, $|X_t| = m \leq n$ and a parameter θ with a prior distribution $p(\theta)$ at

each iteration stochastic optimisation updates θ based on the following formula:

$$\Delta\theta_t = \frac{\epsilon_t}{2}(\nabla \log p(\theta_t) + \frac{n}{m}\sum_{i=1}^{m} \nabla \log p(x_{ti}|\theta_t))$$

where ϵ_t a sequence of step sizes that have to satisfy the Robbins-Monro Conditions we saw on chapter 2 to ensure convergence. Furthermore, Langevin Dynamics is a class of MCMC techniques and is similar to stochastic optimisation with the small difference that it adds a Gaussian noise into the parameter updates, in order to avoid collapsing to the MAP solution and explore the entire parameter space. Specifically,

$$\Delta\theta_t = \frac{\epsilon}{2}(\nabla \log p(\theta_t) + \sum_{i=1}^{n} \nabla \log p(x_i|\theta_t)) + \eta_t$$

where $\eta_t \sim \mathcal{N}(0, \epsilon)$.

Hence, SGLD which combines the ideas of the two approaches, allows us to capture the uncertainty in a Bayesian manner. The update rule of SGLD is the following:

$$\Delta\theta_t = \frac{\epsilon_t}{2}(\nabla \log p(\theta_t) + \frac{n}{m}\sum_{i=1}^{n} \nabla \log p(x_{ti}|\theta_t)) + \eta_t$$

where $\eta_t \sim \mathcal{N}(0, \epsilon_t)$.
Therefore, we can conclude that doing Bayesian inference can be as simple as running noisy Stochastic Gradient Descent. In the original paper of Welling and Teh there is an interesting proof that when the number of iterations goes to infinity then the SGLD algorithm converges to the posterior distribution, meaning that θ_t will approach samples from the posterior.
It's important to note here that SGLD as an approximate inference method is not a VI algorithm, but rather an MCMC algorithm because instead of performing optimisation over a space of distributions, with some notion of similarity between the posterior and the approximating family, it attempts to draw samples from the posterior.

Chapter 4

Generalised Variational Inference

In this chapter we are going to present an optimisation-centric view of Bayesian Inference and prove the optimality of Variational Inference methods on it, meaning that for a well specified optimisation problem maximising the ELBO is always preferable to alternative approximations of the posterior. Also, we are going to present a generalisation of the Bayesian Inference setting named Generalised Variational Inference [47] which seems to be efficient in solving many of the issues that Standard Variational Inference suffers from and also generalise the Variational Inference approach. In particular, we are going to split the inference problem into three elements: A loss, a divergence and a space of feasible solutions. This split is called *Rule of Three (RoT)* and it is true that many of the existing approximation methods can be interpreted by RoT. The biggest advantage of Rule of Three is its modularity as each component serves a specific purpose. Finally, we are going to explicitly define Generalised Variational Inference as a large and tractable family of belief distributions and as a special case of the RoT.
In order to be consistent to our comparisons we tried to use the notation introduced in the original GVI paper for the rest of this chapter.

4.1 Divergences

Definition 1: A statistical divergence $D(p||q)$ is a measure of discrepancy between two probability densities p and q on the same parameter space Θ, or more generally on the same statistical manifold, with the following properties:

1. $D(p||q) \geq 0 \;\; \forall q \in P(\Theta)$
2. $D(p||q) = 0 \iff p = q$

We can understand that a divergence need not to be a distance metric as it doesn't need to be symmetric (most of the times $D(p||q) \neq D(q||p)$), and need

not to satisfy the triangle property. Obviously, any distance metric on $P(\Theta)$ is a divergence measure.

Definition 2: For a convex function f such that $f(1) = 0$, the **f-divergence** of P from Q is defined as:

$$D_f(P||Q) = \mathbb{E}_Q\left[f\Big(\frac{dP}{dQ}\Big)\right] = \int_\Omega f\Big(\frac{dP}{dQ}\Big)dQ$$

In this case that P and Q are both absolutely continuous with respect to a common dominating measure μ on Ω (reference distribution which in our case is the Lebesgue measure) then the f-divergence can be written w.r.t their probability densities as:

$$D_f(P||Q) = \int_\Omega f\Big(\frac{p(x)}{q(x)}\Big)q(x)dx$$

where $dP = pd\mu$ and $dQ = qd\mu$.
It is interesting here to see the information interpretation. Specifically, we can think that there exist two classes $\mathcal{C}_p$(i.e., hypothesis is that $x \sim p$) and $\mathcal{C}_q$ (i.e., hypothesis is that $x \sim q$). Then,

- $f\Big(\frac{p(x)}{q(x)}\Big)$is the information in x discriminating $\mathcal{C}_p$ and $\mathcal{C}_q$.
- $D_f(p||q)$ is the arithmetic mean information for discriminating $\mathcal{C}_p$ and $\mathcal{C}_q$ (mean relative to $x \sim q$).

The familly of f-divergences enjoy some invariance properties investigated in [147, 148] (see also [25]) and a universal monotonicity property known under the name of generalised data processing theorem [165, 249]. (https://hal.inria.fr/inria-00542337/document) properties are investigated in [63].

We are going to briefly present some of the main theorems around the f-divergence familly:

Theorem 1: (Main inequality). Let f be the same as the one in Definition 2 for f, P, Q then we have:

$$D_f(p||q) \geq 0 \;\; \forall q \in P(\Theta)$$

Proof:

$$D_f(p||q) = \sum_x q(x) f\Big(\frac{p(x)}{q(x)}\Big) \overset{\text{Jensen}}{\geq} f\Big(\sum_x \frac{p(x)q(x)}{q(x)}\Big) = f(1) = 0$$

Theorem 2: (Monotonicity). Let f be the same as the one in Definition 2 for f, P, Q. Denoting by A, B two real random variables we have:

$$D_f(p_{A,B}||q_{A,B}) \geq D_f(p_A||q_A)$$

Proof:

$$D_f(p_{A,B}||q_{A,B}) = \sum_{a,b} q_{A,B}(a,b) f\Big(\frac{p_{A,B}(a,b)}{q_{A,B}(a,b}\Big)$$

$$= \sum_a q_A(a) \sum_b q_{B|A}(b|a) f\Big(\frac{p_{B|A}(b|a)p_A(a)}{q_{B|A}(b|a)q_A(a)}\Big)$$

$$\overset{\text{Jensen}}{\geq} \sum_a q_A(a) f\Big(\frac{p_A(a)}{q_A(a)}\Big) = D_f(p_A||q_A)$$

This leads us to one of the most famous inequalities on the Information Theory field:

Theorem 3:(Data Processing Inequality, DPI). Let f be the same as the one in Definition 2 for f, P, Q. Denoting by X, Y two real random variables we have:

$$D_f(p_X||q_X) \geq D_f(p_Y||q_Y)$$

Proof:

$$D_f(p_{X,Y}||q_{X,Y}) = \sum_{x,y} q_{X,Y}(x,y) f\Big(\frac{p_{X,Y}(x,y)}{q_{X,Y}(x,y}\Big)$$

But we know that:

$$\frac{p_{X,Y}(x,y)}{q_{X,Y}(x,y} = \frac{p_X(x)p_{Y|X}(y|x)}{q_X(x)q_{Y|X}(y|x)} = \frac{p_X(x)}{q_X(x)}$$

It is obvious that the last ratio does not depend on y and hence,

$$\implies D_f(p_{X,Y}||q_{X,Y}) = \sum_x q_X(x) f\Big(\frac{p_X(x)}{q_X(x}\Big) = D_f(p_X||q_X)$$

But from Theorem 2 we know that $D_f(p_{X,Y}||q_{X,Y}) \geq D_f(p_Y||q_Y)$. Therefore,

$$D_f(p_X||q_X) \geq D_f(p_Y||q_Y)$$

Below we present some of the most common f-divergences:

Kullback-Leibler Divergence

For $f(x) = x\log(x)$,

$$KL(p||q) = \mathbb{E}_Q[\log p(x) - \log q(x)] = \int_{-\infty}^{+\infty} p(x) log\Big(\frac{p(x)}{q(x)}\Big) dx$$

Reverse Kullback-Leibler Divergence

For $f(x) = -\log(x)$,

$$RKL(p||q) = KL(q||p) = \int_{-\infty}^{+\infty} q(x) \log\Big(\frac{q(x)}{p(x)}\Big) dx$$

α-Divergence

For $f(x) = -\log_\alpha(x)$,

$$D_A^{(\alpha)}(p||q) = \mathbb{E}_q[\log_\alpha(p(x) - log_\alpha q(x)]$$

where $\log_\alpha(x)$ is a generalised log function with $\log_\alpha(x) = \frac{1}{\alpha(1-\alpha)}(x^{\alpha-1} - 1)$ and obviously, $\lim_{\alpha\to 1} \log_\alpha(x) = \log(x)$.
Overall, the basic α-divergence can be defined as:

$$D_A^{(\alpha)}(p||q) = \frac{1}{\alpha(1-\alpha)} \int (p^\alpha(x) q^{1-\alpha}(x) - \alpha p(x) + (\alpha - 1) q(x)) dx$$

For certain values of α in the α-Divergence we can retrieve other members of the f-divergences family. Specifically,

- For $\alpha = \frac{1}{2}$ we get the squared **Hellinger distance**:

$$D_A^{(1/2)}(p||q) = 2D_H(p||q) = 2 \int (\sqrt{p(x)} - \sqrt{q(x)})^2 dx$$

- For $\alpha = 2$ we get the **Pearson χ^2-square divergence**:

$$D_A^{(2)}(p||q) = 2\chi^2(p||q) = \frac{1}{2} \int \frac{(p(x) - q(x))^2}{q(x)} dx$$

- We can easily see that when the limit is evaluated (using the L'Hopital's rule) for $\alpha \to 1$, we obtain the Kullback–Leibler divergence. In other words,

$$\lim_{\alpha\to 1} D_A^{(\alpha)}(p||q) = KL(p||q)$$

Jensen-Shannon Divergence

For $f(x) = x \log(\frac{2x}{x+1}) + \log(\frac{2}{x+1})$,

$$JS(p,q) = KL\Big(p||\frac{p+q}{2}\Big) + KL\Big(q||\frac{p+q}{2}\Big)$$

Total Variation Distance

For $f(x) = \frac{1}{2}|x-1|$,

$$TV(p,q) = \frac{1}{2}\,\mathbb{E}_Q\left[\left|\frac{dp}{dq} - 1\right|\right] = \frac{1}{2}\int |dp - dq|$$

Apart from the f-divergences we also have other discrepancy measures that have proved to be useful for our study:

α-Rényi Divergence

We define a divergence called α–Rényi divergence as follows:

$$D_{AR}^{(\alpha)}(p||q) = \frac{1}{(1-\alpha)}\log\left(\mathbb{E}\left[\left(\frac{p(x)}{q(x)}\right)^{\alpha-1}\right]\right)$$

$$\implies D_{AR}^{(\alpha)}(p||q) = \frac{1}{(1-\alpha)}\log\int p(x)q(x)^{1-\alpha}dx$$

Surprisingly, the Renyi divergence is closely related to the α-divergence:

$$D_{AR}^{(\alpha)}(p||q) = \frac{1}{\alpha(1-\alpha)}\log\left(1+\alpha(\alpha-1)D_{A}^{(\alpha)}(p||q)\right)$$

$$\implies D_{AR}^{(\alpha)}(p||q) = \frac{1}{\alpha(1-\alpha)}\log\Big(\int(p^{\alpha}q^{1-\alpha} - \alpha p + (\alpha-1)q)dx + 1\Big), \alpha \in \mathbb{R}\backslash\{0,1\}$$

An interesting observation here is that for $\alpha = 1$ and $\alpha = 0$ the α-Renyi divergence simplifies to Kullback–Leibler and Reverse Kullback–Leibler divergences respectively.

$$KL(p||q) = \lim_{\alpha\to 1} D_{AR}^{(\alpha)}(p||q)$$

$$RKL(p||q) = KL(q||p) = \lim_{\alpha\to 0} D_{AR}^{(\alpha)}(p||q)$$

Also,

- For $\alpha = \frac{1}{2}$, we can retrive a function of the square Hellinger distance:

$$-2\log\left(1 - D_H[p||q]\right)$$

- For $\alpha = \frac{1}{2}$, we can retrive a function of the χ^2-divergence:

$$-\log\left(1 - \chi^2[p||q]\right)$$

Finally, there are other families of divergences, like **Bregman ϕ-divergences**, introduced in [39] and they are defined for vectors, matrices, functions and probability distributions. The Bregman ϕ-divergence between probability densities is defined as:

$$D_{\phi}(p||q) = \int(\phi(p(x)) - \phi(q(x)) - (p(x) - q(x))\phi'(q(x))dx$$

where with $\phi : \mathbb{R}^d \to \mathbb{R}$ a differentiable strictly convex function. Bergman divergence has a lot of interesting properties, such that can be written as the limit t of a Jensen difference [27, 17]:

$$D_\phi(p||q) = \lim_{\beta \to 1} \frac{1}{1-\beta} \mathcal{J}_\phi^\beta(p, q)$$

where $\mathcal{J}_\phi^\beta = \beta\phi(p) + (1-\beta)\phi(q) - \phi(\beta p + (1-\beta)q)$.
For $\beta = \frac{1}{2}$ Jensen difference is the **Burbea-Rao divergence** [47].

4.2 Rule of Three: A new Bayesian paradigm

For the rest of the chapter we are going to follow the notation below:

- Θ is the **parameter space** and $\theta \in \Theta$ the parameter value.
- $P(\Theta)$ is the set of all probability measures on Θ.
- $\mathcal{Q}$ is a **variational family** (i.e. parametrised subset of $P(\Theta)$).
- $q : \Theta \to \mathbb{R}_+$ is the **posterior** density (i.e., known after data are seen).
- $\pi : \Theta \to \mathbb{R}_+$ is the **prior** density (i.e., known before data are seen).

The main focus of this chapter is to challenge the traditional Bayesian inference paradigm and generalise it in a way that can confront its main problems and tackle more complex real-world problems. In particular, the main issues that modern machine learning models suffer from under the standard Bayesian inference framework are the following:

1. **Prior misspecification.** We usually assume that the prior distribution $\pi(\theta)$ reflects our beliefs exactly and encodes our best judgment about the parameters θ of the model. As it is reasonable this is not the case especially for models like neural networks where the number of parameters is extremely large. Also, it is interesting that in neural networks we usually make the extra assumption that the prior is unimodal (there's only one exact most likely reparametrisation of the weights) and the weights are uncorrelated (pairwise and mutually independent). We can easily understand that these are very naive assumptions for such a complex model like a neural network. Intuitively, we understand that especially the correlations amongst the different layers of the neural network is of utmost importance in order for the model to extract certain features from a large data set. What is happening in practice is that the experts use the prior to perform regularisation rather than to capture their exact beliefs about the model and hence they end up using a sub-optimal prior.

2. **Likelihood misspecification.** When we are setting up our model we oftenly make the arbitary assumption that there exists a specific θ^* that

makes our likelihood function equal to the actual data generating mechanism. Although while parametrizing our model adequately the over-parametrised likelihood functions can fit the data set well enough there are, however, severe problems with untypical data points like outliers and therefore cannot generalise well. In that aspect it might be better to try to encode the typical behaviour of the data than try to fully model it. The most common way used by people dealing with this issue nowadays is that they either let the outliers to get encoded in the likelihood function or they define proper distributions like t-Student which has heavy tails and can explain these untypical points by having mass even far away from the mean. However, as we are going to see in the next section, a method called Generalised Variational Inference, has a brilliant way to deal with this problem using robust divergences and ignoring the outlier if this has a profound effect. There's a lot of work in this direction starting from β-divergences by Basu et al. [48] and has extended also to other divergences like [49], [50] and [51].

3. **Computational limitations.** In the most theoretical approaches of Bayesian inference the analysis is based on the fact that the computational resources are infinite which of course is not true in the real world. As we mention in chapter 3 an intractable quantity is a quantity who needs exponential time to be computed. This problem is one of the main reasons that Variational Inference took over and also inspired a lot of other methods like Laplace approximations etc.

We recall the ELBO formula from chapter 3:

$$ELBO(q) = \mathbb{E}[\log p(D|Z)] - KL(q(Z)||p(Z))$$

we can rewrite the above in the form of the negative ELBO:

$$-ELBO(q) = -\mathbb{E}[\log p(D|Z)] + KL(q(Z)||p(Z))$$

Hence maximizing the ELBO is equivalent to minimizing the negative ELBO. It has been shown by Csiszar [52] and Donsker [53] that the Bayesian inference objective can be seen as the solution to an infinite dimensional optimisation problem and therefore every posterior distribution is the result of a well defined problem of this nature. The first to strictly formalise this idea was Zellner in 1998 [54] by proposing that the Bayesian posterior can be computed from the following equation [1]:

$$q^*(\theta) = \underset{q \in P(\Theta)}{\operatorname{argmin}} \left\{ \mathbb{E}_{q(\theta)} \left[-\sum_{i=1}^{n} \log p(x_i|\theta) \right] + KL(q||\pi) \right\}$$

Many years later Bissiri et. al [55] extensively discussed a generalised Bayesian solution of Zellner's equation inspired by the Fenchel's conjugate of KL diver-

[1] Here, we use the asterisk symbol to emphasize that we are solving an optimisation problem

gence and restated the problem as:

$$q^*(\theta) = \underset{q \in P(\Theta)}{\operatorname{argmin}} \left\{ \mathbb{E}_{q(\theta)} \left[\sum_{i=1}^{n} l(\theta, x_i) \right] + KL(q||\pi) \right\} \quad (1)$$

where $q^*(\theta) = q(\theta|\kappa^*)$ for some optimal parameter $\kappa^* \in K$, the variational parameter space.

It is obvious that in the last equation the first term $\sum_{i=1}^{n} l(\theta, x_i)$ is minimised by $\delta_{\hat{\theta}_n}(\theta)$ where $\hat{\theta}_n$ is the empirical risk minimizer and δ the dirac function. Also, the second term is minimised by $q = \pi$.

Overall, all the above should make us see Bayesian inference as an optimisation problem over $P(\Theta)$ about a parameter θ specified via a loss l and regularised by the KL divergence.

As we have seen there are a lot of assumptions in the traditional Bayesian inference framework that are not a good basis to deal with real-world challenges. Hence, there was a need to design a novel generalised Bayesian inference framework that is consistent with a set of axioms and flexible enough in order to deal with a wider spectre of problems. In this direction Knoblauch et al. [47] proposed the **Rule of Three (RoT)** foundation which provides enough flexibility to tackle most of the inappropriate assumptions especially by its modularity nature. As we saw previously Bisiri et al. proposed a generalisation of Zellner's representation for the standard Bayesian posterior by replacing $-\log p(x_i|\theta)$ with a general loss function $l(\theta, x_i)$. Here the authors took a step further by letting also the divergence measure to be a general divergence function and not only the KL one as in equation (1). Hence, they proposed a generalised representation of Bayesian inference via the following definition:

Definition 3: For given observations $x_{1:n}$, a prior $\pi(\theta)$, a space $\Pi \subseteq \mathcal{P}(\Theta)$, a loss function $l : \Theta \times \mathcal{X} \to \mathbb{R}$ and a divergence $D(\cdot||\pi) : \Pi \to \mathbb{R}_+$ we say that posteriors have been constructed via the *Rule of Three* if they can be written as:

$$q^*(\theta) = \underset{q \in \Pi}{\operatorname{argmin}} \left\{ \mathbb{E}_{q(\theta)} \left[\sum_{i=1}^{n} l(\theta, x_i) \right] + D(q||\pi) \right\} = P(l, D, \Pi)$$

Here we can observe that instead of having a regularised loss-minimisation interpretation of inference as before, we have regularised and constrained minimisation interpretation of it. Taking a more detailed look at the constituent elements of RoT $P(l, D, \Pi)$ it becomes clear that they serve a specific and separate from each other purpose. Specifically,

- $\Pi \subseteq P(\Theta)$ is the set of the admissible posterior beliefs. It answers the question *"Which beliefs are allowed?"*. If Π is a variational family then we write $\Pi = \mathcal{Q}$
- The divergence measure $D : P(\Theta) \times P(\Theta) \to \mathbb{R}_+$ is acting as the uncertainty quantifier/ regularizer and gives us information about how q^* looks

like.

- The loss $l : \Theta \times \mathcal{X} \to \mathbb{R}$ tells us which parameter θ we care about.

All the above make it clear that $P(l, D, \Pi)$ has a modular interpretation and decomposes into three separate parts, each of them serving a certain purpose. The following Theorem of the original paper formalise this modularity by making it more precise:

Theorem 4: Hold $\pi(\theta)$ and Π fixed and take $q_1^*(\theta) \in \Pi$ as a posterior derived from $P(l, D, \Pi)$. If we want to compute an alternative posterior $q_2^*(\theta) \in \Pi$ through the RoT:

1. Which deals with model misspecification then we have to change l.
2. Which is robust to prior misspecification, without changing θ, then we have to change D.
3. Which affects the uncertainty quantification, without changing θ, then we have to change D.

While RoT is intuitively appealing, it can also be derived axiomatically. Specifically, we define the following four axioms:

Axiom 1 (Representation):
Bayesian inference infers posteriors q on Θ by:

1. Measuring how q fits a sample $\mathcal{X}$ via the expectation of a loss $l(\theta, x)$.
2. Quantifying uncertainty about the parameter of interest θ^* via a divergence D between prior π and q.
3. Optimizing q over a space of probability distributions $\Pi \subseteq P(\Theta)$.

Axiom 2 (Information Difference):
$P(l, D, \Pi)$ produces different posteriors for $\mathcal{X} = x_{1:n}$ and $\mathcal{X}' = x_{1:n+m}$.

Axiom 3 (Prior Regularisation):
The distribution q is regularised against π by penalizing the divergence $D(q||\pi)$.

Axiom 4 (Translation Invariance):
For constants C, M and $l' = l + C$, $D' = D + M \implies P(l', D, \Pi) = P(l, D, \Pi)$.

In the original paper there is an analytic proof of a theorem that states that based only on these axioms then the objective of RoT is uniquely identified as we define above.
The table below presents a variety of Bayesian methods that can be recovered by the RoT framework :

Method	$\ell(\boldsymbol{\theta}, x_i)$	D	Π
Standard Bayes	$-\log p(x_i\|\boldsymbol{\theta})$	KLD	$\mathcal{P}(\boldsymbol{\Theta})$
Power Likelihood Bayes	$-\log p(x_i\|\boldsymbol{\theta})$	$\frac{1}{w}$KLD, $w<1$	$\mathcal{P}(\boldsymbol{\Theta})$
Composite Likelihood Bayes	$-w_i \log p(x_i\|\boldsymbol{\theta})$	KLD	$\mathcal{P}(\boldsymbol{\Theta})$
Divergence-based Bayes	divergence-based ℓ	KLD	$\mathcal{P}(\boldsymbol{\Theta})$
PAC/Gibbs Bayes	any ℓ	KLD	$\mathcal{P}(\boldsymbol{\Theta})$
VAE	$-\log p_{\boldsymbol{\zeta}}(x_i\|\boldsymbol{\theta})$	KLD	$\mathcal{Q}$
β-VAE	$-\log p_{\boldsymbol{\zeta}}(x_i\|\boldsymbol{\theta})$	$\beta \cdot$ KLD, $\beta>1$	$\mathcal{Q}$
Bernoulli-VAE	continuous Bernoulli	KLD	$\mathcal{Q}$
Standard VI	$-\log p(x_i\|\boldsymbol{\theta})$	KLD	$\mathcal{Q}$
Power VI	$-\log p(x_i\|\boldsymbol{\theta})$	$\frac{1}{w}$KLD, $w<1$	$\mathcal{Q}$
Utility VI[9]	$-\log p(x_i\|\boldsymbol{\theta}) + \log u(h, x_i)$	KLD	$\mathcal{Q}$
Regularized Bayes	$-\log p(x_i\|\boldsymbol{\theta}) + \phi(\boldsymbol{\theta}, x_i)$	KLD	$\mathcal{Q}$
Gibbs VI	any ℓ	KLD	$\mathcal{Q}$
Generalized VI	any ℓ	any D	$\mathcal{Q}$

Figure 4.1: $P(l, D, \Pi)$ for existing methods.

4.3 Different views of Variational Inference

In this final section we are going to see the different views of Variational Inference and prove that the standard VI method of maximising the ELBO is always optimal for a fixed variational family relatively to methods based on alternative divergences to approximate the exact Bayesian posterior. Finally, we are going to motivate GVI and argue about its generation of appealing posterior beliefs.

ELBO view (Model selection)

We have seen before that the standard way of deriving Variational Inference is by maximizing the evidence in the data by picking the element from the approximation family $\mathcal{Q}$ that maximizes the ELBO and we ended up obtaining the posterior as the solution on the following optimisation problem:

$$q^*(\theta) = \underset{q \in P(\Theta)}{\operatorname{argmin}} \left\{ \mathbb{E}_{q(\theta)} \left[\sum_{i=1}^{n} l(\theta, x_i) \right] + KL(q||\pi) \right\}$$

Obviously, this interpretation can give an insight/criterion for model selection by comparing the log-likelihoods.

Discrepancy-minimisation view (DVI)

It is obvious that the Bayesian posterior is not a unique solution to the optimisation problem and hence, we could see its solution as the minimisation of a distance (usually in the form of divergences). Specifically, if we want to approximate the standard Bayesian posterior $q_B^*(\theta)$ with a variational distribution $q(\theta)$ we could write the solution of this problem as:

$$q^*(\theta) = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} KL(q||q_B^*)$$

In fact, the same objective function that maximizes the ELBO is the one that minimizes the distance of $\mathcal{Q}$ and $q_B^*(\theta)$ in the KL divergence sense by just rearranging the terms of the ELBO equation.
This way of approaching the Bayesian posterior approximation has motivated a large body of research that tries to approximate the posterior by minimizing divergences, different from the KL, between the family $\mathcal{Q}$ and $q_B^*(\theta)$. In particular, for a divergence measure $D : P(\Theta) \times P(\Theta) \to \mathbb{R}_+$ we can define a new class of methods called *Discrepancy Variational Inference (DVI) methods*. DVI methods' objective is of the following form:

$$q^*_{\text{DVI}}(\theta) = \underset{q \in \mathcal{Q}}{\text{argmin}} D(q||q^*_B), \qquad \text{D} \neq \text{KL}$$

Some of the most interesting work are for:

- D = Renyi's α-divergence (Li and Turner, 2016; Saha et al., 2017)
- D = χ-divergence (Dieng et al., 2017)
- D = Wasserstein distance (Ambrogioni et al., 2018)

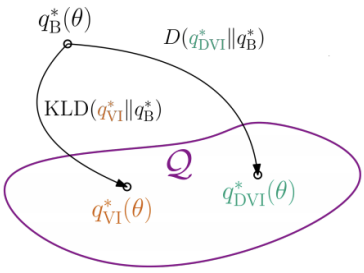


Figure 4.2: Distinction between the two interpretations of VI.

Constrained optimisation VIEW (GVI)

The third view of VI is its generalisation approach and which was first presented at Knoblauch, Damoulas et al. [47] and treats the VI solution as the best $\mathcal{Q}$-constrained solution. Specifically, we recall that:

$$q^*_{\text{VI}}(\theta) = \underset{q \in \mathcal{Q}}{\text{argmax}} ELBO(q) = \underset{q \in \mathcal{Q}}{\text{argmin}} - ELBO(q)$$

$$= \underset{q \in P(\Theta)}{\text{argmin}} \left\{ \mathbb{E}_{q(\theta)} \left[- \sum_{i=1}^{n} \log p(x_i|\theta) \right] + KL(q||\pi) \right\}$$

Hence, VI solves a problem specified by the Rule of Three $P(l, D, \Pi)$ for $l = \log p(x_i|\theta)$, $D = KL(q||\pi)$ and $\Pi = \mathcal{Q}$.

$$\implies q^*_{\text{VI}}(\theta) = P(\log p(x_i|\theta), KL(q||\pi), \mathcal{Q})$$

This $\mathcal{Q}$-constrained version of original Bayes problem presents the novel view of VI, the Generalised Variational Inference view, which we'll analyze in-depth below.

Theorem 5: For a fixed variational family $\mathcal{Q}$ standard Variational Inference produces the uniquely optimal posterior belief in $\mathcal{Q}$ solving the $P(l, KL(q||\pi), P(\Theta))$.

Proof: The proof is straightforward if we just observe that q^*_{VI} is the minimizer of this objective relative to $\mathcal{Q}$ while q^*_{B} is the minimizer relative to $P(\Theta)$ and that $\mathcal{Q} \subseteq P(\Theta))$. But if VI is optimal what about the DVI methods? Why they are useful? It is true that for a fixed variational family $\mathcal{Q}$ methods different from VI are sub-optimal relative to $P(l, KL(q||\pi), P(\Theta))$, i.e. produce worse posterior beliefs. The proof for this is based on contradiction and is again straightforward.

However, there is an interesting thing here. In practice, DVI methods often outperform the VI ones. So, from the one hand VI is the best approximation to the standard Bayesian posterior but on the other hand DVI produces more desirable posterior inferences. Is this a contradiction? The answer is No. In particular what is actually happening here is that DVI methods target a better inference problem than $P(l, KL(q||\pi), P(\Theta))$ which is **non-standard**. Therefore, another reasonable question arises: Why don't we design these non-standard objectives/targets explicitly in order to generate desirable posterior beliefs via VI? In order to do that we have to solve two main issues:

- Inference should be optimal relative to something (i.e. $l, D, \mathcal{Q}$).
- Ingredients of non-standard problem should be interpretable.

Generalised Variational Inference combines the best of both VI and DVI methods and offers this novel constrained optimisation view of approximating the Bayesian posterior. Specifically,

Definition 4: Any Bayesian inference method solving a RoT form $P(l, D, \mathcal{Q})$ for $\mathcal{Q} = \{q(\theta|\kappa) : \kappa \in K\} \subseteq P(\Theta)$ is a procedure called **Generalazided Variational Inference (GVI)**.

Hence, GVI **like VI** has the form $P(l, D, \mathcal{Q})$ and satisfies the RoT modularity property and **like DVI** targets non-standard posteriors without conflating l and D. In other words, it combines the advantages of both approaches. Changing the divergence in the GVI-sense affects only the uncertainty quantification in contrast with the DVI-sense where by changing the divergence we can accidentially interfere with the way that the best parameter is found (e.g. interfere

with the loss function). Three of the main areas that GVI has prooved to be really efficient are the following:

1. Robust alternatives to $l(\theta, x_i) = -\log p(x_i|\theta)$.
2. Prior-robust uncertainty quantification via D.
3. Adjusting marginal variances via D.

M-closed/M-open Inference

Finally, we are going to give a high-level explanation on why GVI works in practice.

Definition 5: We call an inference procedure **M-closed** if we assume that both the model and the prior are correct. In other words,

$$\exists \theta^* \in \Theta : x_i \sim p(x_i|\theta^*)$$

Therefore, with the M-closed view we can focus on computing the exact Bayesian posterior

$$q^*(\theta) \propto \prod_{i=1}^{N} p(x_i|\theta)\pi(\theta)$$

which is the solution of $P(-\sum_{i=1}^{n} -\log p(x_i|\theta), D, P(\Theta))$.

If we take the M-closed view then $P(l, D, P(\Theta))$ is the best we can do because VI is the best $\mathcal{Q}$-constrained approximation and any other DVI method would be sub-optimal to it.

Definition 6: We call an inference procedure **M-open** if we assume that both the model and the prior are correct. In other words,

$$\nexists \theta^* \in \Theta : x_i \sim p(x_i|\theta^*)$$

In traditional statistics, in that case we would try to come up with better models until we get close enough to the true model. However, from the Machine Learning standpoint we tend to use black box models like Bayesian Neural Networks or Deep Gaussian Processes and we don't care too much about the exact posterior. If we assume an M-open view of the inference procedure then it would make sense if DVI methods outperform VI in practice because they would target better targets. However, GVI could target better alternative targets but instead of doing it implicitly it is going to do it explicitly. Specifically, if the model is not good then we should adjust our scoring rule by changing the loss l. Also, if the prior is bad or misspecified we should change the discrepancy penalisation from π and change D. Overall, if the model is good enough then using exact inference or VI makes sense. However, if the model is not good which usually

leads to a better performance by DVI methods, this could be addressed by GVI if we know what is causing the issue.
Even though Generalised Variational Inference approach is only recently proposed it seems to be very promising as it already has many applications that outperform the traditional VI approaches (in BNNs, DGPs, BOCD etc). We encourage the reader to spend some time on the experiments chapter of the original GVI paper [47] for a more in-depth understanding.

Chapter 5

Experiments and Analysis

As mentioned before, as for certain scientific fields it is of utmost importance to move from point estimate predictions of traditional Deep Learning to predictions that will contain confidence information about themselves and thus we use Bayesian modeling. By switching from optimisation to marginalisation we can get posterior distributions over the parameter space. However we saw that these distributions are usually intractable and hence we should find a way to approximate them. For this reason we moved to Approximate and Variational Inference methods which can scale well for large data sets and are more computationally efficient than the standard Bayesian approach. Although VI methods for Bayesian Neural Networks seem to perform quite well even in complex tasks, they suffer from three major problems that we have highlighted in Chapter 4: prior misspecification, likelihood misspecification and computational limitations. Generalised Variational Inference, given its modularity property, seems to overcome most of these problems by restating the VI problem as a $\mathcal{Q}$-constrained optimization problem. In this chapter we are going to understand the benefits and drawbacks of all these methods on Bayesian Neural Networks and provide some insights that could be really fruitful for future work on the field. In particular, we are going to thoroughly present the experiments that we have conducted and perform on them qualitative and quantitative analysis. Finally, in Figure 5.1 we give a "roadmap" of the experiments that we run and analyse in this section.

The code for all the experiments can be found in the following public Github repository: **https://github.com/gfelekis/MSc-Dissertation.git**

Divergence	Notation
KL Divergence	KL
Reverse KL Divergence	RKL
α-Divergence	A
α-Renyi Divergence	AR
param. α-Renyi Divergence	αAR
Jensen–Shannon Divergence	JS
Total Variation Distance	TV
Fisher Distance	F

Description	Methods	Metrics/ Criteria	Datasets
Cross-Validation for hyperparameter-dependent divergencies.	GVI with D = A, AR, aAR.	NLL, RMSE	Boston, Concrete, Energy, Yacht
Comparison of standard VI with GVI of multiple divergences across different NN depths.	Standard VI and GVI with D = RKL, A, AR, aAR, JS, F, TVL, TVU	NLL, RMSE	Boston, Concrete, Energy, Yacht
Comparison of GVI with other approximate inference methods across different NN depths.	GVI with D = KL, A, aAR, JS and MC Dropout, SGLD and Deep Ensemble.	NLL, RMSE	Boston, Concrete, Energy, Yacht
Uncertainty quantification of different GVI divergences across different NN depths.	GVI with D = KL, RKL, A, AR, aAR, JS, F, TVL, TVU	BIC, AIC, HQC, NLL	GP with RBF kernel.

Figure 5.1

5.0.1 Regression on UCI data sets

Experiment description

In this first part of the chapter we carry out multiple regression experiments on four different data sets from the UCI Machine Learning repository [56] (Boston, Concrete, Energy and Yacht) in order to compare different divergence measures and different neural network depths on the GVI setting. Bayesian Neural Networks, as we saw before, are prone to suffer from prior misspecification. Thanks to the modularity of RoT, and as a result of GVI, in order to tackle this issue we just need to focus on trying different discrepancy measures while keeping the loss function fixed. Indeed, in the experiments that we have conducted for this work

we expand the research of [47] to almost all the members of the f-divergence family that were described on the previous chapter and also Fisher distance, while we kept the loss function fixed to the usual log-likelihood one. Under this framework, we ended up comparing Standard VI (KL Divergence) with GVI for multiple divergence measures by performing an extensive empirical analysis of them based on their RMSE and NLL values, not only across the different data sets but also across different number of hidden layers. We also expand the research of [47] as for α-Divergence and α-Renyi Divergence we first run a thorough grid-search for their respective hyperparameter α by again compairing their RMSE and NLL trying 11 different α values and kept the ones with the best performance. Note that we also used a parametrised form of α-Renyi Divergence which equation is similar with the standard α-Renyi Divergence with the only difference that is multiplied by α.
Across all the experiments we kept the structure of the neural network fixed in order to make the comparisons as fair as possible. In particular, we use a Multi-Layer Perceptron with [100] hidden units for the one hidden layer case, [100, 100] hidden units for the two hidden layer case and [100, 100, 100] hidden units for the three hidden layer case. The activation function was a ReLU function and inference was performed via Bayes by backprop and the Adam optimiser. For the training of each model we run 100 epochs and perform 30 random splits of each data set with a split of 90%-10% (train-test). All the models were evaluated on the test sets using the average negative log likelihood (NLL) as well as the average root mean square error (RMSE). Also, for each of the 30 splits, the predictions are computed based on 100 samples from the variational posterior. Note that the priors and variational posteriors are both fully factorised normal distributions and thus our model was predicting the regression mean $\mu(x)$ and the log-standard deviation $\log \sigma(x)$. Also, that helped us of having all of our divergences in a closed Gaussian form. Below we present the closed form of the divergences that were used:

For a prior $\pi \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and approximate posterior $q \sim \mathcal{N}(\mu_2, \sigma_2^2)$ distributions we have the following closed forms:

- Kullback-Leibler Divergence:

$$KL(\pi||q) = \frac{1}{2\sigma_2^2}\big((\mu_1 - \mu_2)^2 + \sigma_1^2 - \sigma_2^2\big) + \ln \frac{\sigma_2}{\sigma_1}$$

- Reverse Kullback-Leibler Divergence:

$$RKL(\pi||q) = KL(q||\pi)$$

- α-Divergence:

$$D_A^{(\alpha)}(\pi||q) = \frac{1}{\alpha(1-\alpha)}\left(1 - \frac{\sigma_2^{\alpha}\sigma_1^{1-\alpha}}{\sqrt{\alpha\sigma_2^2 + (1-\alpha)\sigma_1^2}} e^{-\frac{\alpha(1-\alpha)}{\alpha\sigma_2^2+(1-\alpha)\sigma_1^2}\frac{(\mu_1-\mu_2)^2}{2}}\right)$$

- Jensen-Shannon Divergence:

$$JS(\pi, q) = KL\Big(p||\frac{p+q}{2}\Big) + KL\Big(q||\frac{p+q}{2}\Big)$$

- Total Variation Distance:
  For the TV distance there is no closed form and hence we approximate it by its bounds. In particular, the following inequality holds:

$$\frac{1}{200}\min\left\{1, \max\left\{\frac{|\sigma_1^2-\sigma_2^2|}{\sigma_1^2}, \frac{40|\mu_1-\mu_2|}{\sigma_1}\right\}\right\} \leq TV(\pi, q) \leq \frac{3|\sigma_1^2-\sigma_2^2|}{2\sigma_1^2} + \frac{|\mu_1-\mu_2|}{2\sigma_1}$$

  Hence we defined an approximation for each bound (upper U, lower L):

$$TVU(\pi, q) = \frac{1}{200}\min\left\{1, \max\left\{\frac{|\sigma_1^2-\sigma_2^2|}{\sigma_1^2}, \frac{40|\mu_1-\mu_2|}{\sigma_1}\right\}\right\}$$

$$TVL(\pi, q) = \frac{3|\sigma_1^2-\sigma_2^2|}{2\sigma_1^2} + \frac{|\mu_1-\mu_2|}{2\sigma_1}$$

- α-Rényi Divergence:

$$D_{AR}^{(\alpha)}(p||q) = \ln\frac{\sigma_2}{\sigma_1} + \frac{1}{2(\alpha-1)}\ln\big(\frac{\sigma_2^2}{(\sigma^2)_\alpha^*}\big) + \frac{1}{2}\frac{\alpha(\mu_1-\mu_2)^2}{(\sigma^2)_\alpha^*}$$

  where:

$$(\sigma^2)_\alpha^* = \alpha\sigma_2^2 + (1-\alpha)\sigma_1^2 > 0$$

- Fisher Distance:

$$F(\pi, q) = \sqrt{2}ln\big(\frac{\mathcal{F}((\mu_1, \sigma_1^2), (\mu_2, \sigma_2^2)) + (\mu_1-\mu_2)^2 + 2(\sigma_1^2+\sigma_2^2)}{4\sigma_1\sigma_2}\big)$$

  where:

$$\mathcal{F}((\mu_1, \sigma_1^2), (\mu_2, \sigma_2^2)) = \sqrt{((\mu_1-\mu_2)^2 + 2(\sigma_1-\sigma_2)^2)((\mu_1-\mu_2)^2 + 2(\sigma_1+\sigma_2^2))}$$

Experiment analysis

Before we start comparing GVI to standard VI, we take an intermediate step and perform a grid-search for the hyperparameter α for α-Divergence, α-Renyi Divergence and parametrised α-Renyi Divergence. An open problem for all the inference methods that has to do with discrepancy minimisation is that, for the divergences that depend on a hyperparameter, it is difficult to make this parameter part of the inferential process by placing a prior and trying to approximate its posterior. Thus, cross-validation seems to be the most promising and reasonable strategy. For this reason, in order to decide which is the best parameter for each of the three divergences we conduct this analytic grid-search for 11 different α values:

$$\alpha \in \{0.25, 0.5, 0.75, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0\}$$

In Figure 5.2 we summarise all the results of each divergence where at the top row are the RMSE values and at the bottom row the NLL ones, across a range of data sets using Bayesian Neural Networks. The dots correspond to means, and the error bars to the variance i.e. standard errors. We can observe that the differences amongst different α values are really small but with a closer look we see that the predictive performance improves for $\alpha > 1$ and especially for values in the range $[1.75, 3.0]$ for all the cases.
The next step now is to compare the performance of different divergence measures, while performing GVI, with standard VI. Although presenting the results in the form of error bars as before is a reasonable way, we have observed that there were some splits that were causing problems during optimisation by returning relatively big RMSE or NLL values compared to the others causing very uncertain predictions i.e. predictions with high variance. This might happen for a couple of reasons but it has mainly to do with the general difficulty of Bayesian Deep Learning, that most of the times demands a lot of engineering of the hyperparameters and extensive hands-on analysis of the code in order to spot the problem. In particular, in a Bayesian Deep Learning task there are a lot of possible sources of errors that could create such numerical issues, like: the approximations of the variational family, the Monte Carlo sampling and also the general nature of Deep Learning and its optimisation process. Therefore, it is important to keep these things in mind and do any possible heuristics to improve the performance of the model. Below we propose some of them:

1. Print the variational posterior values in order to have a better intuition about how well the parameters that we approximate are.

2. Print the ELBO function to see if it presents any problematic behaviour (Figure 5.3).

3. Try different seeds to confirm that the possible problem has nothing to do with the optimisation or initialisation but rather with something else.

4. Especially in the GVI case, take into consideration the expression of the different divergences and how they might affect the result.

Thus, for the reasons mentioned above, we think that the representation would be way more fruitful if we present the results in the form of boxplots (Figure 5.4) and graphically represent the RMSE and NLL values of each divergence through their quartiles. In particular, a box plot displays the five-number summary of a set of data including the minimum, first quartile, median, third quartile, and the maximum. Hence, instead of looking at the mean and the variance we focus on the median (the mid-point of the data) and the interquartile range that shows the middle 50% of the scores. The sceptical reader would probably notice that here we have picked a fixed α-value for each of the A, AR and αAR divergences. It is true that this choice is quite naive as we should have an optimal α-value not only for each divergence but also for each data set. Hence we encourage the reader to take a more analytic view at the results of the grid-search plots and

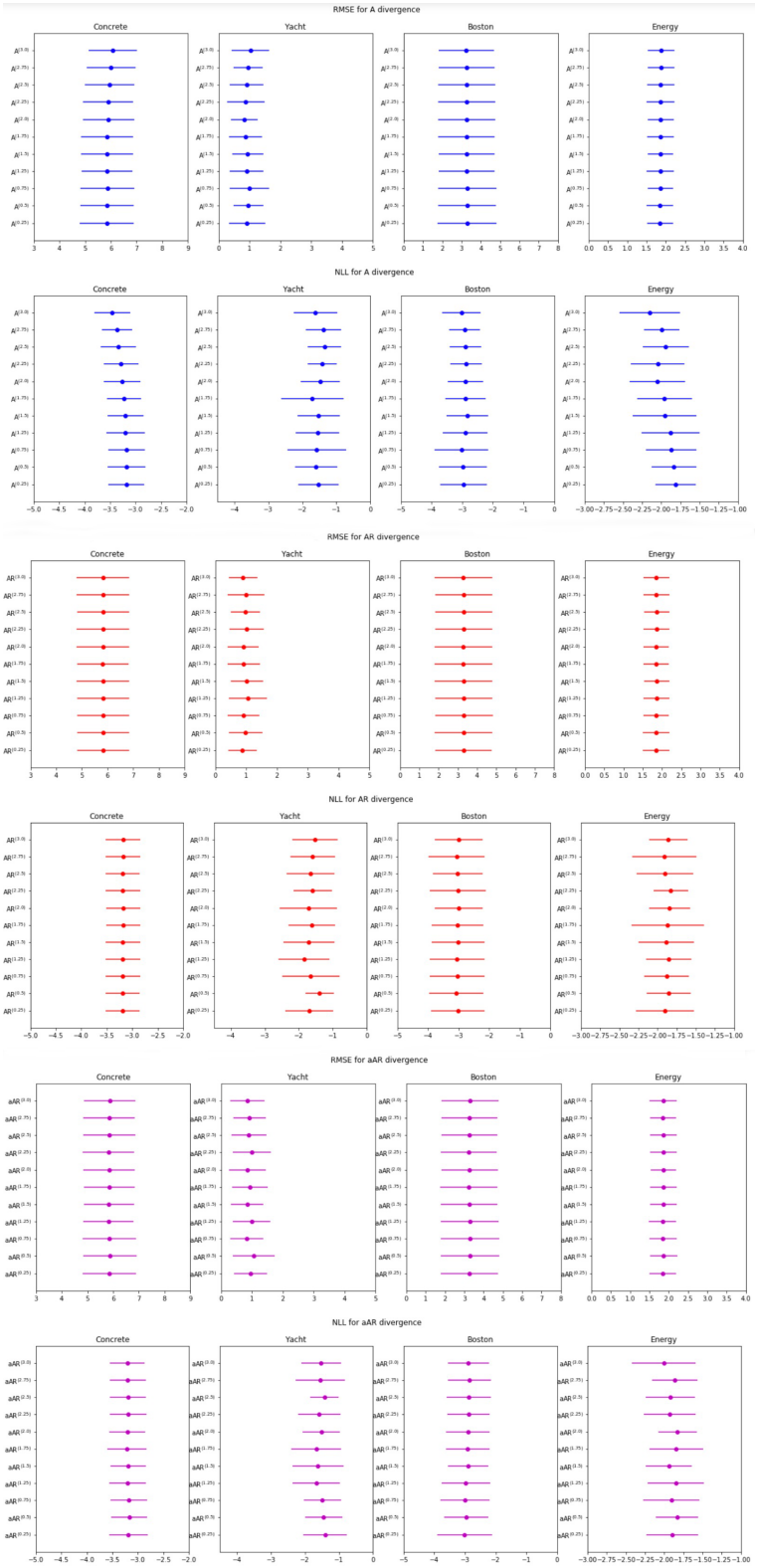

RMSE for A divergence
Concrete
Yacht
Boston
Energy
NLL for A divergence
Concrete
Yacht
Boston
Energy
RMSE for AR divergence
Concrete
Yacht
Boston
Energy
NLL for AR divergence
Concrete
Yacht
Boston
Energy
RMSE for aAR divergence
Concrete
Yacht
Boston
Energy
NLL for aAR divergence
Concrete
Yacht
Boston
Energy

Figure 5.2

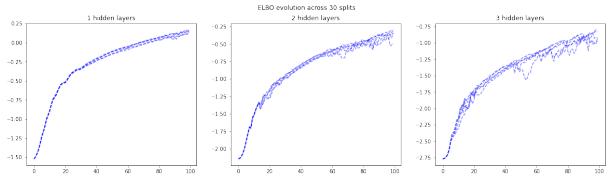


Figure 5.3

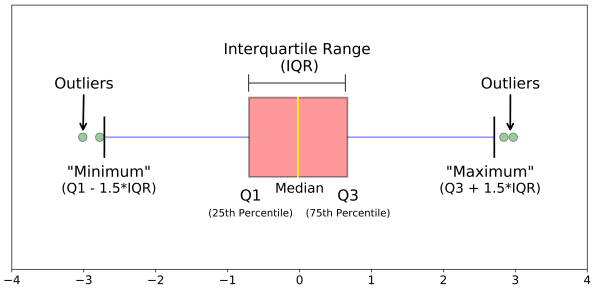


Figure 5.4

find the best value for each and every one of them.
In Figure 5.5 we summarise the results and we can make the following remarks:

- For the **one hidden layer** case: we can confirm the finding of the original GVI paper that for α-Renyi divergence GVI improves its performance for $\alpha > 1$ and worsens it for $\alpha \in (0, 1)$ (see the α-grid-search). Although standard VI (KL) works quite well here αAR, JS and F also seem to be solid choices for this setting as they are either equally good with KL or even better than it. RKL seems to have the worst performance of all. This should be reasonable and it is mainly because of its mathematical expression. More precisely, in contrast with KL divergence that tries to find and fit a single mode of the variational posterior, RKL tries to match something more spread out of the posterior. Also, interesting here is to observe the existence of a lot of outliers in some datasets. In the boxplots the nice performanc of JS and F is even more clear.

- For the **two hidden layers** case: Firstly, we can observe that αAR increases its variances and thus returns more uncertain predictions. Also, F and TVL (especially for the NLL) outperform KL almost everywhere. Additionally, JS seems to work pretty nicely here, especially for the RMSEs. We could also observe that A divergence improves its performance compared to the one hidden layer case. Another interesting observation is that TVL works better than TVU, and we could understand the benefits that we would have if we perform a "sandwich" technique here to approximate the actual TV value. Overall, KL remains a solid choice but it's

clear that GVI outperforms it in many cases especially for F and JS. From the boxplots we could conclude a few more things: First, for the NLLs most of the times KL has the worst performance. It also becomes clear that JS outperforms the rest of the divergences, as it has not only the best median value but also presents a high concentration around it. On the other hand, F seems to have great median values but its values are more spreaded around it. The instance of "Concrete" dataset is a clear evidence that GVI can outperform VI. In this example, almost all the divergences work better than the KL one.

- For the **three hidden layers** case: The predictions here seem to contain higher uncertainty than in the previous cases, mainly due to the increase of the outliers in the different runs. Just by seeing the boxplots we can validate this. Also, JS and F remain the top performers amongst the different divergences. In addition, TVL and TVU improve both their RMSE and NLL values and outperform KL in 3 out of 4 datasets.

We also provide the error bar plots at the Appendix (Fig. 7.1) to highlight the issue that we mentioned before.

Next, we are going to make a straight comparison of some of the "best" performing divergences with:
(a) The KL divergence (Standard Variational Inference) and
(b) The three approximate inference methods that we introduced before (Monte-Carlo Dropout, Deep Ensemble and Stochastic Gradient Langevin Dynamics). Here, we didn't have the issues that were mentioned before and thus we return to the error bar representation. The results can be seen in Figure 5.6. We can observe the following:

- For the **one hidden layer** case: The GVI methods seem to outperform MC-Dropout and Deep Ensembles and be equally good with the SGLD method especially for the RMSE values. MC-Dropout seems to be the worst performer here.

- For the **two hidden layers** case: Here, all the approximate inference methods seem to improve and come closer to the GVI RMSE values in most of the cases. However, we see that Ensemble and SGLD outperform all the other methods when it comes to NLL values.

- For the **three hidden layers** case: We can notice that the uncertainty value (variance) increases in all the methods. GVI remains the best performer and in some cases alongside SGLD and also improves its performance in the NLL scores.

Overall, we can conclude that GVI outperforms the standard Approximate Inference methods in most of the cases but this (and all the experiments) is also strictly dependent to the hyperparameters of optimisation and training.

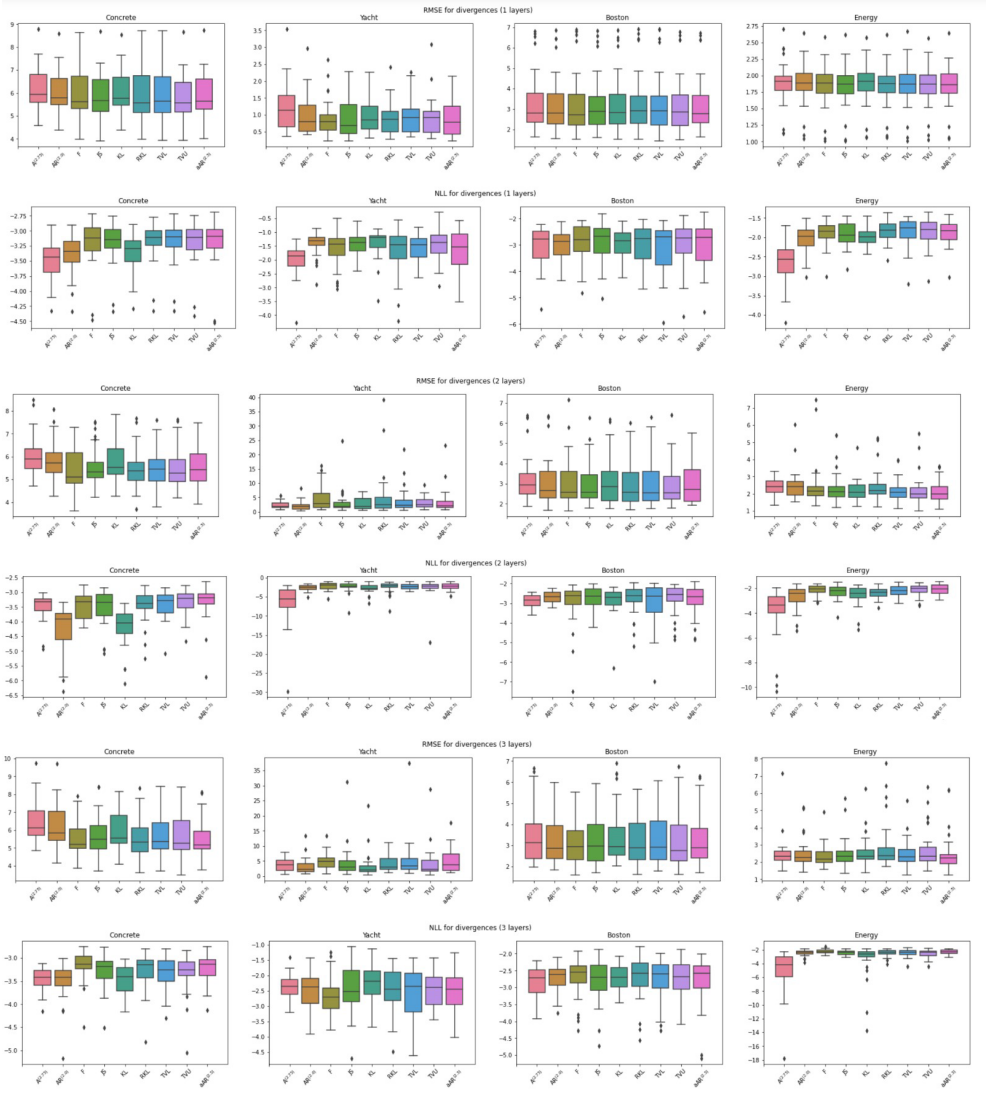

RMSE for divergences (1 layers)
Concrete
Yacht
Boston
Energy
NLL for divergences (1 layers)
Concrete
Yacht
Boston
Energy
RMSE for divergences (2 layers)
Concrete
Yacht
Boston
Energy
NLL for divergences (2 layers)
Concrete
Yacht
Boston
Energy
RMSE for divergences (3 layers)
Concrete
Yacht
Boston
Energy
NLL for divergences (3 layers)
Concrete
Yacht
Boston
Energy

Figure 5.5

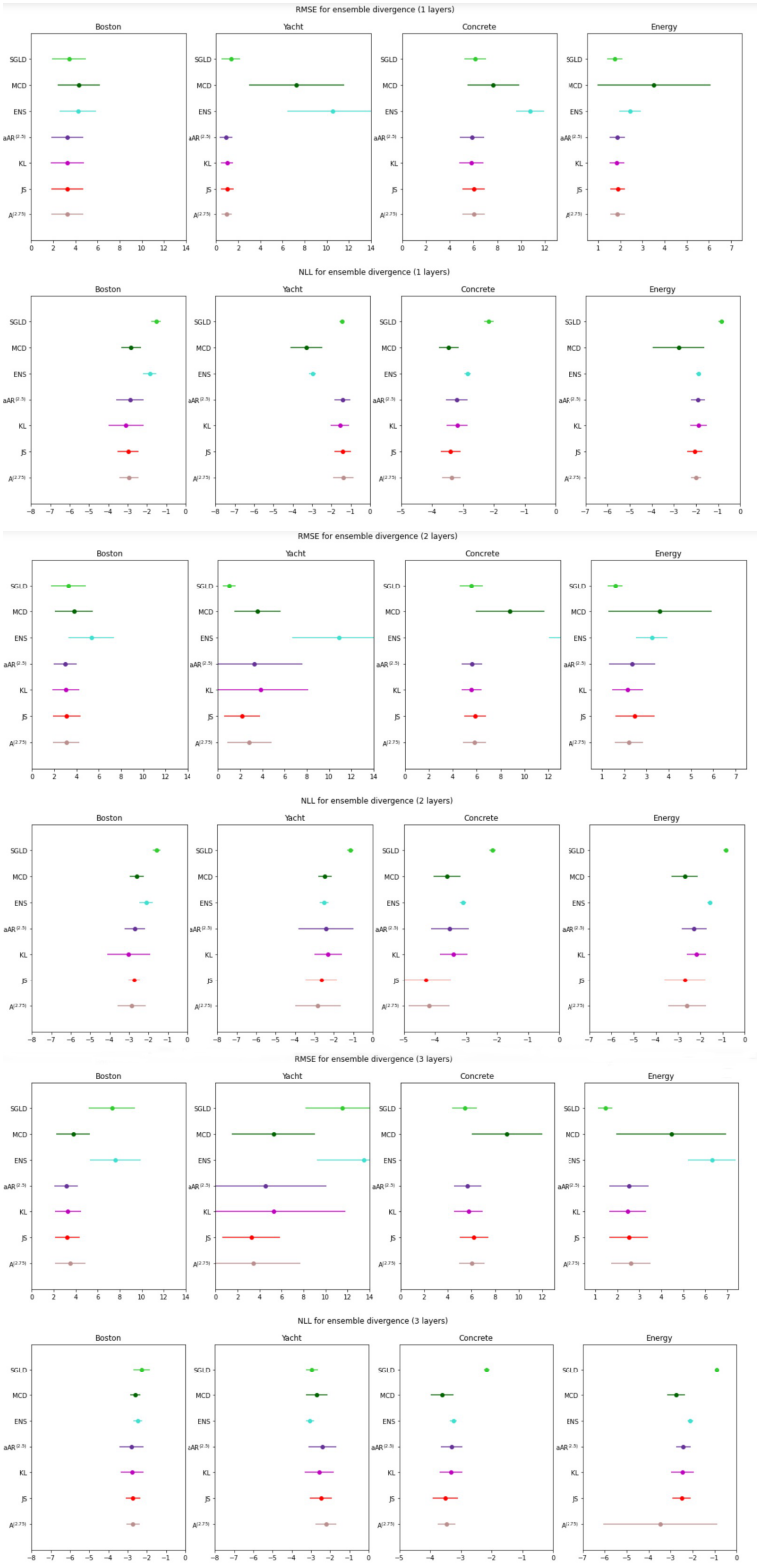

RMSE for ensemble divergence (1 layers)
NLL for ensemble divergence (1 layers)
RMSE for ensemble divergence (2 layers)
NLL for ensemble divergence (2 layers)
RMSE for ensemble divergence (3 layers)
NLL for ensemble divergence (3 layers)
Boston
Yacht
Concrete
Energy
SGLD
MCD
ENS
KL
JS

Figure 5.6

5.0.2 Regression on Gaussian Process ground truth

Experiment description

In the second part of our experiments we take inspiration from the code presented in [60] and we perform a regression on a "toy" data set which was generated with a Gaussian Process as the ground truth. We seek to evaluate aleatoric and epistemic uncertainty for Bayes by backprop algorithm while performing GVI across different discrepancy measures and neural network depths. Specifically, the heteroscedastic data was generated by a Gaussian Process with an RBF kernel ($l = 1, \sigma_n = 0.3|x + 2|$). Also a Multi-Layer Perceptron was used as the regressor with [100] ReLU hidden units for the one hidden layer case, [100, 200] ReLU hidden units for the two hidden layer case and [100, 200, 100] ReLU hidden units for the three hidden layer case and in all cases it was trained for 500 epochs. We compute epistemic and aleatoric uncertainty and we also investigate the behaviour of all the models across different divergences and different depths by not only visual inspection but also some model selection and information criteria that we are going to see below. In Figure 5.7 we can see the visualisation of the Gaussian Process ground truth.

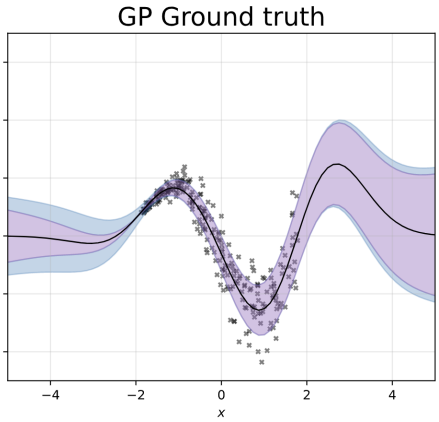


Figure 5.7

Experiment analysis

In the next plots, Epistemic and Aleatoric uncertainties were evaluated for the different divergences in GVI across different neural network depths. We recall that epistemic uncertainty has to do with the uncertainty in model parameters that best explain the observed data and the architecture of the model while aleatoric uncertainty has to do with the uncertainty arising from the noise of observations.

Uncertainty Decomposition

Here we use the results of [57] to extract and decompose uncertainty into epistemic and aleatoric components: Aleatoric uncertainty (noise uncertainty) can be quantified as:

$$\mathcal{U}_A = \mathbb{E}[\sigma^2_{\text{pred}}] \quad \text{or} \quad \mathcal{U}_A = \mathbb{E}_{q(w)}[\mathcal{U}(y'|x', w)]$$

Also, Epistemic uncertainty (model uncertainty) can be quantified as:

$$\mathcal{U}_E = Var_{q(w)}(\mu_{\text{pred}}) \quad \text{or} \quad \mathcal{U}_E = \mathcal{U}(y'|x') - \mathcal{U}_A$$

In Figures 5.8 and 5.9 we interpret the aleatoric uncertainty in purple, the epistemic one in blue and the black line represents the mean of prediction with respect to the variational posterior. Also in the Appendix (Fig. 7.2) we present the same thing for Deep Ensemble, Monte-Carlo Dropout and Stochastic Gradient Langevin Dynamics for the one hidden layer case.

Visual Inspection

Here, we observe the plots in Figure 5.8 and 5.9 and come up with some intuitions via their visual inspection. We are going to divide our remarks into three parts, one for each neural network depth (one, two, three) and divide each of them into three extra parts of the plot: observations for $x \leq 2$, $x \in (-2, 2)$ and $x \geq 2$. For each neural network depth we provide an ordering of the best models according to our judgement. A good model is one that balances uncertainty in the unseen regions with the right confidence within the interval in which it has been trained. We can notice the following:

- For the **one hidden layer** case
  - For $x \leq -2$: The majority of the models seem overconfident and epistemic uncertainty dominates over aleatoric. The latter one makes sense as there are no data points further than -2, except for A and JS which present a reasonable level of uncertainty and RKL, which although is not overconfident, seems not follow the distribution of the data. If we collect more data points and grow the interval, then we'll see that epistemic uncertainty will decrease.
  - For $x \in (-2, 2)$: Most of the models fit the data really well, except again RKL.
  - For $x \geq 2$: Most of the models seem underconfident in this domain. However this is reasonable and legitimate as we cannot be sure in areas that we do not have any data points. Especially, KL, AR, αAR and F provide nice uncertainty measures while the other are extremely underconfident. On the other hand TV (both TVL and TVU) do not provide enough uncertainty and someone could argue they are either overconfident.

    Best models: 1) JS, 2) KL, 3) A

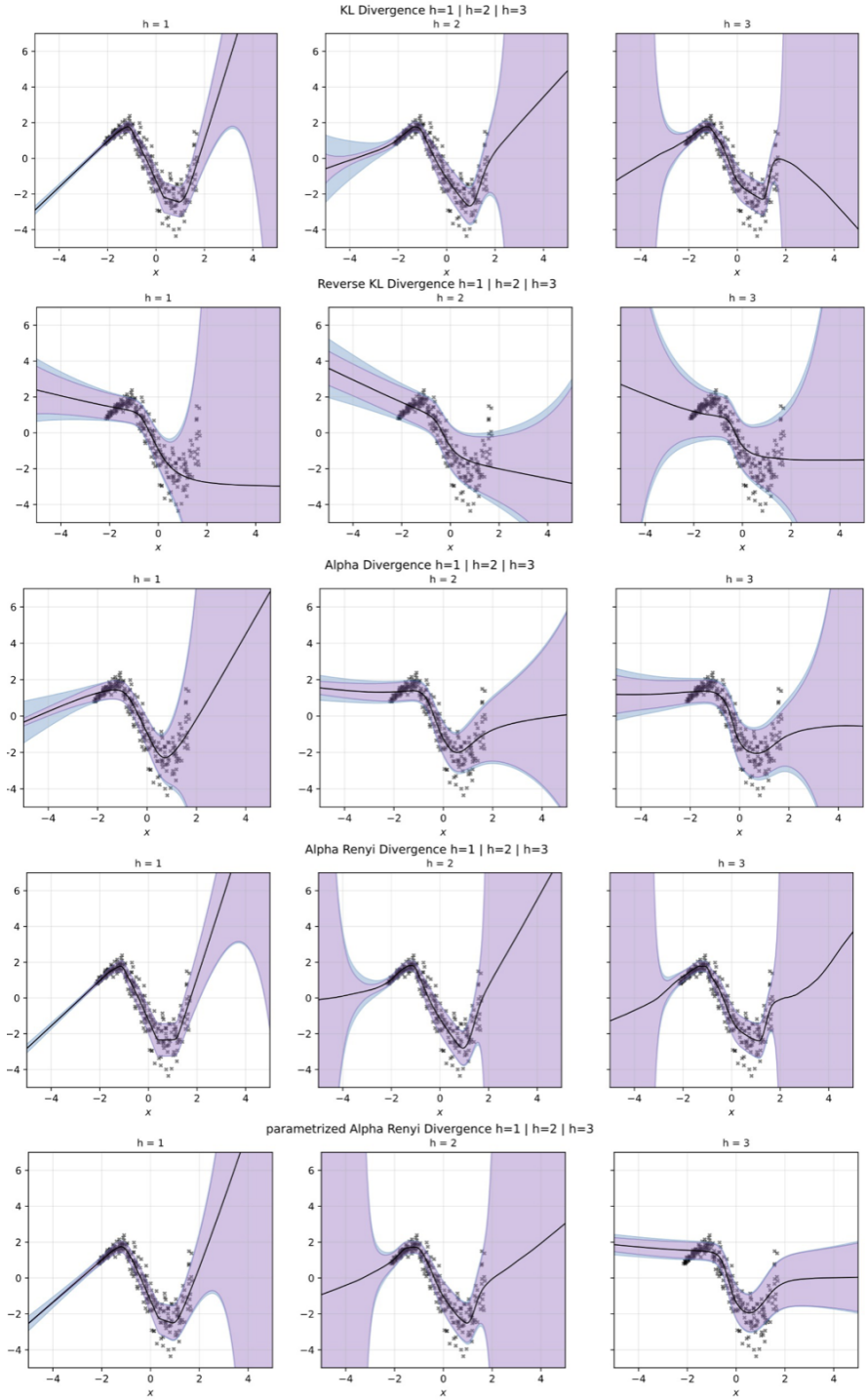

KL Divergence h=1 | h=2 | h=3
Reverse KL Divergence h=1 | h=2 | h=3
Alpha Divergence h=1 | h=2 | h=3
Alpha Renyi Divergence h=1 | h=2 | h=3
parametrized Alpha Renyi Divergence h=1 | h=2 | h=3
h = 1
h = 2
h = 3
x

Figure 5.8

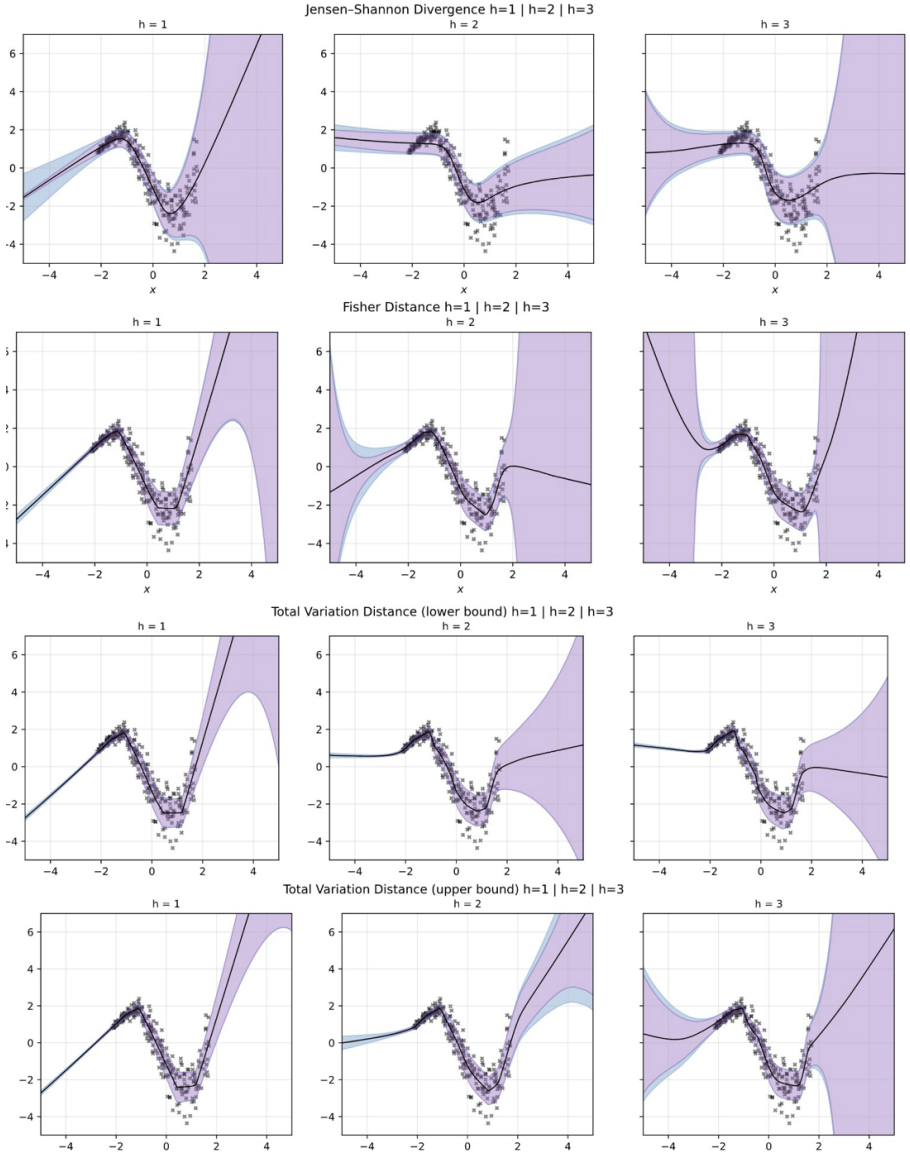

Jensen–Shannon Divergence h=1 | h=2 | h=3
Fisher Distance h=1 | h=2 | h=3
Total Variation Distance (lower bound) h=1 | h=2 | h=3
Total Variation Distance (upper bound) h=1 | h=2 | h=3
h = 1
h = 2
h = 3
x

Figure 5.9

- For the **two hidden layers** case
  - For $x \leq -2$: Here, we have more reasonable confidence levels than in the one hidden layer case from the models, especially for the KL, AR and F divergences.
  - For $x \in (-2, 2)$: Again, most of the models fit the data really well, except RKL and JS which deteriorates its performance.
  - For $x \geq 2$: Here, more models than before are extremely underconfident (A, AR, F) while others provide a better confidence measure than the one in their one hidden layer case in the same domain (KL, TVL, TVU).

    Best models: 1) KL, 2) F, 3) TVU
- For the **three hidden layers** case
  - For $x \leq -2$: The majority of the models are extremely underconfident. The only exception is A, which makes poor predictions .
  - For $x \in (-2, 2)$: Again the models fit the data really well and here they also increase the uncertainty value to a legitimate level.
  - For $x \geq 2$: The same as the $x \leq -2$ holds here.

    Best models: 1) TVU, 2) JS, 3) A

Model Calibration

In order to select the best calibrated model, we inspect their confidence at different intervals for all the divergence measures and neural network depths.
We report the predicted probability and the difference with respect to the true probability at one standard deviation interval, which should be 0.68 under the Gaussian assumption, for all the different divergence measures and neural network depths. In order to measure the predicted probability, we calculate the overlap between the predicted and the true distribution by counting how many out of the predicted points fall inside the specified sigma interval of the true distribution. We define a well-calibrated model as the one whose predicted probability differs no more than 0.1 from the true probability.
In Figures 5.10, 5.11 and 5.12 we can see the calibration curves, where each plot corresponds to a different number of hidden layers. The the x-axis the σ-interval and y-axis represents the probability inside a σ-interval. The dashed line represents the expected values under a Gaussian distribution ($1\sigma \approx 0.68$, $2\sigma \approx 0.955$ and $3\sigma \approx 0.997$). For each divergence we have a certain curve which represents how confidence varies across different sigma intervals, indicating either underconfidence if the curve is above the dashed line or overconfidence if it falls below it.

We can make the following remarks:

- Confidence increases together with the number of layers, except for the case of RKL, where the reverse can be observed which is consistent with the statement from 5.0.1 section under the experiment analysis.
- There is a general overconfidence trend on lower sigma intervals i.e. $(0, 0.7)$.

Here we enumerate the divergence measures which correspond to the two best calibrated models for each neural network depth:

- For the **one hidden layer** case: TVL and αAR
- For the **two hidden layers** case: KL, AR and A
- For the **three hidden layers** case: TVL, A and JS.

The discrepancies between the best models according to uncertainty calibration and the ones according to the visual inspection might be attributed to the fact that uncertainty calibration only provides information over the range of seen data, whereas for visual inspection we also take into consideration the uncertainty quantification in the unseen domains.

Divergence	**No. of hidden layers**	**Predicted probability**	**Difference**
KL	1	0.55	0.13
KL	2	0.81	0.13
KL	3	0.53	0.15
RKL	1	0.56	0.12
RKL	2	0.54	0.14
RKL	3	0.52	0.16
A	1	**0.58**	0.10
A	2	**0.66**	0.02
A	3	**0.75**	0.07
AR	1	0.55	0.13
AR	2	0.57	0.11
AR	3	0.84	0.16
αAR	1	0.56	0.12
αAR	2	**0.60**	0.10
αAR	3	**0.68**	0.00

Divergence	No. of hidden layers	Predicted probability	Difference
JS	1	**0.58**	0.10
JS	2	0.53	0.15
JS	3	0.52	0.16
TVL	1	**0.58**	0.10
TVL	2	0.89	0.21
TVL	3	**0.65**	0.03
TVU	1	0.57	0.11
TVU	2	0.56	0.12
TVU	3	0.82	0.14
F	1	0.55	0.13
F	2	0.56	0.12
F	3	0.83	0.15

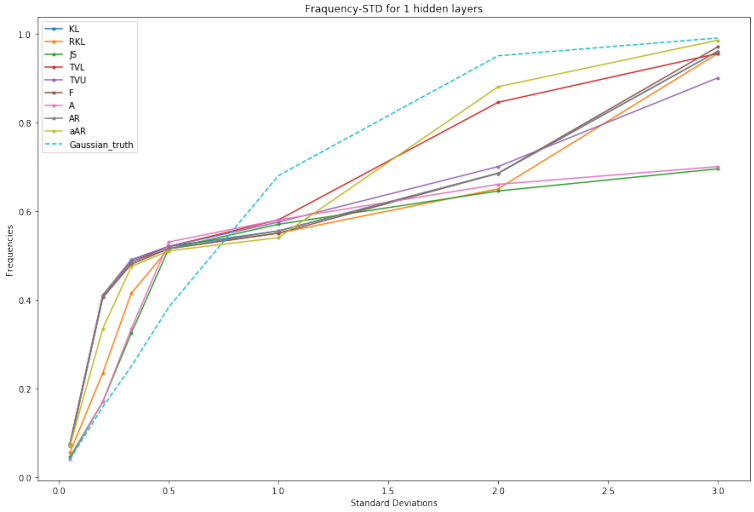


Figure 5.10

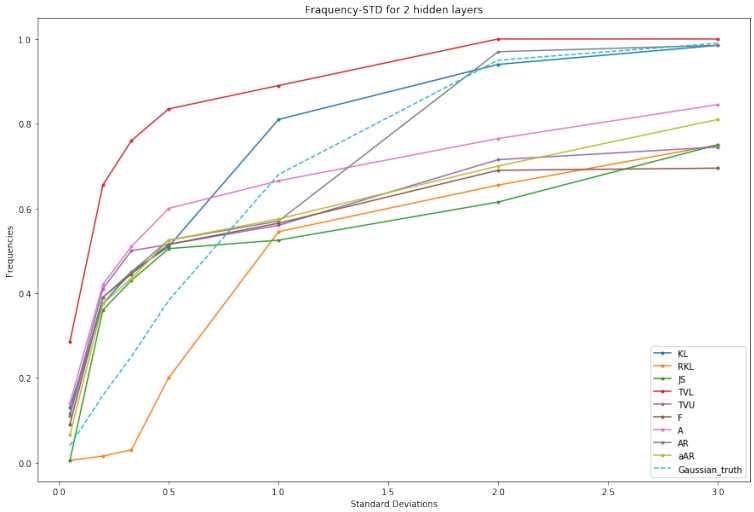


Figure 5.11

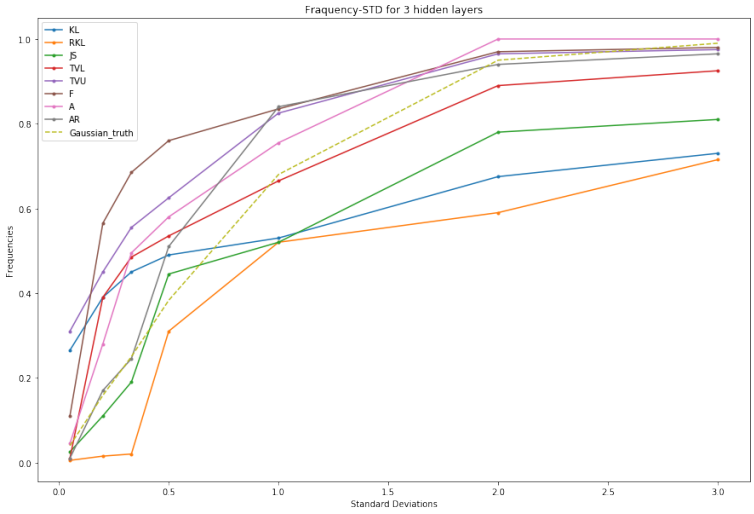


Figure 5.12

Information Criteria

Finally, one standard way to perform model selection in Probabilistic Deep Learning is to compute the values of certain criteria from the Information Theory field. Specifically here we tested three of them for each model:

1. **Bayesian Information Criterion - BIC**

$$BIC(\mathrm{M}) = p \ln(n) - 2 \ln(\hat{L})$$

2. **Akaike Information Criterion - AIC**

$$AIC(\mathrm{M}) = 2p - 2 \ln(\hat{L})$$

3. **Hannan–Quinn Information Criterion - HQC**

$$HQC(\mathrm{M}) = 2p \ln(\ln(n)) - 2\hat{L}$$

where $\hat{L}$ is the maximized value of the likelihood function of the model M, p is the number of parameters estimated by the model and n the sample size. Information Criteria (ICs) are really useful as they can measure the efficiency of a model in terms of predicting the data and also the complexity of the model. We can observe that all three of them are constituted of two terms: a complexity term, which is the one dependent to the number of model's parameters, and a data fit term. Generally, the computation of ICs is a trade-off between these two terms as they try to answer the question: Do I need more data or a more complex model? Therefore, when as a next step we computed all the values of the ICs for all the models we ended up observing that the complexity penalty dominated the IC value and hence the one hidden layer case was always the preferable choice (see table in the Appendix). Here it is important to mention a few things about the information criteria. It would be naive to conclude that the information criteria, which were used here, are useless. Instead, they serve their purpose as they penalise the model for trying to fit more parameters than the needed ones for this specific problem/data set. In the example here, where the data set was generated from an RBF kernel of a Gaussian process which is a stationary kernel, the penalty term dominates. It is not necessarily true that the IC value has to increase as long as the number of parameters grows. As we said, all the ICs perform a trade-off between model complexity and data fit. Here, the data fit component is roughly equivalent for all the different neural network depths, thus the only thing that changes is the number of parameters. Consequently, it makes perfect sense to see this monotonic deterioration. For neural networks, an advantage of having more than one hidden layers is that they can capture non stationary kernels (Neural Network kernel, Polynomial kernel, etc). So we encourage the reader to examine the same results for a non-stationary case where the ICs would be much more informative than here. Overall, it is crucial to understand that the results have to do with the specific problem, which is simple, and the fact that number of parameters is not a really

good complexity value for an MLP. Having said that, it would probably be more fruitful to compare just the differences between the log-likelihoods of the models. In the table below we present the log-likelihood value for each model and we highlight in bold the best for each divergence.

Divergence	**No. of hidden layers**	**Log-likelihood**
KL	1	352
KL	2	347
KL	3	**343**
RKL	1	**380**
RKL	2	390
RKL	3	727
A	1	**398**
A	2	413
A	3	636
AR	1	352
AR	2	347
AR	3	**343**
αAR	1	**348**
αAR	2	361
αAR	3	360
JS	1	**357**
JS	2	368
JS	3	400
TVL	1	**344**
TVL	2	345
TVL	3	347
TVU	1	**343**
TVU	2	**343**
TVU	3	344
F	1	351
F	2	349
F	3	**343**

From this table we can notice that, although some divergences achieve the best log-likelihood value at the maximum number of hidden layers, it is not always the case and it does not justify the increase of the number of model parameters. It is interesting to see that the best model based on the log-likelihood value agrees in most of the cases with our visual inspection conclusion about each layer.

Chapter 6

Conclusion

6.1 Discussion

This thesis sought to investigate whether a generalised Bayesian Inference framework in approximating posterior distributions could improve the performance of the traditional Approximate and Variational inference methods. In particular, we have compared the most common approximate inference methods with a recently proposed approach which treats Variational Inference as a $\mathcal{Q}$-constrained optimisation problem, where $\mathcal{Q}$ is a variational family. More specifically, we were interested in demonstrating this comparison on different data sets for a particular model: Bayesian Neural Networks. First, we have introduced the basics of Deep Learning and have tried to motivate the importance of having confidence measurement outputs from our models alongside the standard point estimate predictions. By introducing a prior distribution in model's parameters this made the predictions have the form of posterior distributions via the Bayes rule and thus classical Deep Learning models could return uncertainty measurements. Moreover, we highlighted the issues that standard Bayesian Inference and BNNs in particular should be expected to suffer from and built an argument on how GVI could tackle them. Variational Inference in general seeks to minimise a discrepancy measure between the true posterior distribution, which is usually intractable, and a Variational family of approximations. In the standard setting of VI this happens via the minimisation of the KL divergence and the negative log-likelihood loss function. However, over the last decades there have been a lot of proposed methods that instead of sticking with KL as the divergence and negative log-likelihood as the loss, try to generalise to other measures either of divergences or of losses. In this direction, the novel framework of GVI was built on the top of Rule of Three, a new class of posterior belief distributions, which does not rely on the traditional Bayesian assumptions. RoT is an optimisation problem with three independent arguments, each of which tackles a shortcoming of standard Bayesian Inference. It was interesting to see that standard Variational Inference could be recovered as a special case of RoT whereas

the DVI methods could not. Therefore, GVI restricts the domain of RoT to its tractable subset of posterior beliefs by the $\mathcal{Q}$ family. We have carried out two experiments in order to inspect the performance of the different methods, not only across the different data sets but also across different neural network depths. The first experiment was a regression on four UCI data sets where we compared posteriors derived from standard Approximate Inference methods with ones derived from multiple divergence measures in the GVI setting. The second experiment was a regression on an artificially-generated data set from a Gaussian Process ground truth. We have evaluated epistemic and aleatoric uncertainty for different GVI divergences and provided calibration plots of the different models and their respective objectives. In this part of the experiments we have also made an analytic discussion about model selection from the perspective of traditional information criteria and have appointed their weaknesses in such a problem where the complexity term dominates. Overall, we have showed the power and usefulness of GVI on a model architecture that perfectly encapsulates the miss-alignments between the traditional Bayesian assumptions and the reality of modern large-scale Bayesian Inference such as the BNN and provide a fruitful insight of the benefits of different divergences especially from the f-divergence family. Through our experiments we made some really useful contributions. In particular,

- We have conducted optimisation over a vast divergence landscape on the GVI setting by trying different divergence functions and finding optimal hyperparameters for existing ones (A, AR, αAR), extending the work of the original GVI paper.
- We have provided visualisations of uncertainty quantification and evaluation of epistemic and aleatoric uncertainties for a Gaussian Process ground truth.
- We have made an extensive discussion (visual inspection, information criteria, model calibration) about model selection for GVI posteriors.
- Overall, we have provided an empirical proof of the superiority of GVI over traditional approximate inference methods and mainly over standard VI, in certain cases in the framework of Bayesian Neural Networks.

6.2 Future work

Moving forward, although this has work tried to thoroughly answer the questions that aroused from the hypothesis about the benefits of switching to a generalised framework for inference in Bayesian Neural Networks via the experiments, due to the time constraint and limited resources, there is always space for improvement. Here, we investigate some of the things that might have worked better and make suggestions for further work.
The first thing that we should mention is that generally speaking, in Probabilistic Deep Learning the results are heavy dependent on the hyper-parameter

tuning. For this reason, it would be of interest to see an even more exhaustive search in this direction by trying different model architectures, train for more epochs or try different splits, or sample more data points from the variational posterior. As far as the second experiment is concerned, as we mentioned before, it is interesting to see the behaviour of the different divergences for alternative kernels and especially for non stationary ones. Also, as an extension of present work, it would be worthwhile exploring the performance for more robust divergence measures and loss functions, especially from the ones that have been proved to be robust in approximating posterior distributions. Specifically, new discrepancy measures that are worth investigating are: Wasserstein distance, β/γ divergences, Kagan's divergence etc. Additionally, as we saw in the experiments Jensen–Shannon divergence seems to work really well, thus an idea would be to try a generalisation of it, introduced by Nielsen [61] and called α-Jensen–Shannon divergence:

$$JS_{\alpha}(p||q) = \frac{1}{2}(K_{\alpha}(p||q) + K_{\alpha}(q||p))$$

where $K_{\alpha}(p||q) = KL(p||(1-\alpha)p + \alpha q)$ and we can retrieve Jensen–Shannon divergence for $\alpha = \frac{1}{2}$ and Jeffrey's divergence for $\alpha = 1$.
Furthermore, in our work we used Bayes by Backprop as our inference algorithm and it was a "black box" algorithm only with respect to the loss function. It would be an interesting idea, although definitely more challenging, to try to make the algorithm completely "black box" which would mean that we would sample to also approximate the divergences and not use closed forms as we did. Obviously, an idea would also be to try a different inference algorithm such as "Probabilistic Backpropagation" [58]. Finally, a recent work [59] seems to use GVI for online learning and specifically online variational approximations. In this work, new regret bounds were derived for GVI posteriors and were shown to have even better generalisation properties. Hence, this amplifies the argument about the power and versatility of GVI to also other models outside Bayesian Neural Networks and Gaussian Processes.

Bibliography

[1] LeCun, Y., Bengio, Y. Hinton, G. Deep learning. Nature 521, 436–444 (2015).

[2] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016.

[3] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. Brain Theory and Neural Networks, 1995.

[4] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

[5] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[6] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

[7] Krizhevsky, A., Sutskever, I. Hinton, G. ImageNet classification with deep convolutional neural networks. In Proc. Advances in Neural Information Processing Systems 25 1090–1098 (2012).

[8] Rosenblatt, Frank (1957). "The Perceptron—a perceiving and recognizing automaton". Report 85-460-1. Cornell Aeronautical Laboratory.

[9] Rumelhart, David E., and David Zipser. "Feature discovery by competitive learning." Cognitive science 9.1 (1985): 75-112.

[10] D. Rumelhart, G. Hinton, and R. Williams. "Learning Internal Representations by Error Propagation."

[11] Robbins, H. (2007). A Stochastic Approximation Method. Annals of Mathematical Statistics, 22, 400-407.

[12] Kiefer, J.; Wolfowitz, J. Stochastic Estimation of the Maximum of a Regression Function. Ann. Math. Statist. 23 (1952), no. 3, 462–466.

[13] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[14] Cybenko, G. (1989) "Approximations by superpositions of sigmoidal functions", Mathematics of Control, Signals, and Systems, 2(4), 303–314.

[15] Pinkus, Allan (January 1999). "Approximation theory of the MLP model in neural networks". Acta Numerica. 8: 143–195.

[16] Neal, Radford M. Bayesian Learning for Neural Networks. Diss. University of Toronto, 1995.

[17] "Computation with infinite neural networks" Christopher K.I Williams "Computation with infinite neural networks", Neural Computation (July 1998).

[18] MacKay, David JC. "A practical Bayesian framework for backpropagation networks." Neural computation 4.3 (1992): 448-472.

[19] Wenzel, F., Roth, K., Veeling, B. S., Swiatkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the Bayes posterior in deep neural networks really? arXiv preprint arXiv:2002.02405, 2020.

[20] Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D. (2015). Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424.

[21] Graves, A. (2011). Practical variational inference for neural networks. In Advances in neural information processing systems (pp. 2348-2356).

[22] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes."

[23] Alan Cobham (1965), The intrinsic computational difficulty of functions.

[24] Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. J. Amer. Statist. Assoc. 85 398–409.

[25] Hastings, W. (1970), "Monte Carlo Sampling Methods using Markov Chains and their Applications," Biometrika, 57, 97–109.

[26] Peterson, Carsten. "A mean field theory learning algorithm for neural networks." Complex systems 1 (1987): 995-1019.

[27] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., Saul, L. K. (1999). An introduction to variational methods for graphical models. Machine learning, 37(2), 183-233.

[28] Hinton, G. E., Van Camp, D. (1993, August). Keeping the neural networks simple by minimizing the description length of the weights. In Proceedings of the sixth annual conference on Computational learning theory (pp. 5-13).

[29] Neal, Radford Hinton, Geoffrey. (1998). A View Of The Em Algorithm That Justifies Incremental, Sparse, And Other Variants. Learning in graphical models.

[30] Waterhouse, Steve MacKay, David Robinson, Tony. (1995). Bayesian Methods for Mixtures of Experts, 351-357.

[31] MacKay, D. J. (1997). Ensemble learning for hidden Markov models (pp. 362-378). Technical report, Cavendish Laboratory, University of Cambridge.

[32] Barber, D., and Bishop, C. M. (1998), "Ensemble Learning in Bayesian Neural Networks," 215–237.

[33] M D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. JMLR, 14(1), 2013.

[34] L. Bottou. Large-scale machine learning with stochastic gradient descent. Springer, 2010.

[35] R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. In AISTATS, 2014

[36] Y.Z Li and r. e. Turner. Renyi divergence variational inference. In NIPS, 2016

[37] T. P. Minka. Divergence measures and message passing. In Microsoft Research Technical Report, 2005.

[38] C. Zhang, R. Bamler, M. Opper, and S. Mandt. Perturbative black box variational inference. In NIPS, 2017.

[39] R. Ranganath, D. Tran, and D. M. Blei. Hierarchical variational models. In ICML, 2016.

[40] Kullback, S.; Leibler, R.A. (1951). "On information and sufficiency". Annals of Mathematical Statistics. 22 (1): 79–86.

[41] Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.

[42] Winn, John, and Christopher M. Bishop. "Variational message passing." Journal of Machine Learning Research 6.Apr (2005): 661-694.

[43] Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe. "Variational inference: A review for statisticians."

[44] Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. 2016.

[45] Gal, Yarin. "Uncertainty in Deep Learning." (2016).

[46] Welling, Max, and Yee W. Teh. "Bayesian learning via stochastic gradient Langevin dynamics." Proceedings of the 28th international conference on machine learning (ICML-11). 2011.

[47] Knoblauch, Jeremias, Jack Jewson, and Theodoros Damoulas. "Generalised variational inference: Three arguments for deriving new posteriors."

[48] Basu, Ayanendranath, et al. "Robust and efficient estimation by minimising a density power divergence." Biometrika 85.3 (1998): 549-559.

[49] Aapo Hyvärinen. Estimation of non-normalised statistical models by score matching. Journal of Machine Learning Research, 6:695–708, 2005.

[50] Hung Hung, Zhi-Yu Jou, and Su-Yun Huang. Robust mislabel logistic regression without modeling mislabel probabilities. Biometrics, 74(1):145–154, 2018.

[51] Alessandro Barp, Francois-Xavier Briol, Andrew B. Duncan, Mark Girolami, and Lester Mackey. Minimum Stein discrepancy estimators. Advances in Neural Information Processing Systems, 2019.

[52] Csiszar, I. I-Divergence Geometry of Probability Distributions and Minimisation Problems. Ann. Probab. 3 (1975), no. 1, 146–158.

[53] Donsker, M. D., Varadhan, S. R. S. (1975). Asymptotics for the wiener sausage. Communications on Pure and Applied Mathematics, 28(4), 525-565.

[54] Arnold Zellner. Optimal information processing and Bayes's theorem. The American Statistician, 42(4):278–280, 1988.

[55] Pier Giovanni Bissiri, Chris Holmes, and Stephen Walker. A general framework for updating belief distributions. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 78(5):1103–1130, 2016.

[56] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[57] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning.

[58] Hernández-Lobato, José Miguel, and Ryan Adams. "Probabilistic backpropagation for scalable learning of bayesian neural networks." International Conference on Machine Learning. 2015.

[59] Pierre Alquier. "Non-exponentially weighted aggregation: regret bounds for unbounded loss functions."(preprint) 2020

[60] J. Antorán, X. Liu, E. Markou, X. Zheng, "Uncertainty in Bayesian Neural Networks" https://github.com/JavierAntoran/Bayesian-Neural-Networks

[61] Nielsen, Frank. "A family of statistical symmetric divergences based on Jensen's inequality." arXiv preprint arXiv:1009.4004 (2010).

Chapter 7

Appendix

Divergence	No. of hidden layers	BIC	AIC	HQC
KL	1	4670	1459	2731
KL	2	249238	83074	148871
KL	3	487837	162713	291454
RKL	1	4927	1716	2987
RKL	2	249385	83221	149018
RKL	3	488139	163015	291756
A	1	4993	1782	3054
A	2	249410	83245	149042
A	3	488141	163017	291758
AR	1	4673	1462	2733
AR	2	249238	83073	148870
AR	3	487837	162713	291454
αAR	1	4646	1435	2706
αAR	2	249284	83119	148916
αAR	3	487885	162761	291502
JS	1	4958	1747	3018
JS	2	249389	83224	149021
JS	3	488115	488115	291731
TVL	1	4678	1467	2738
TVL	2	24919	83054	148851
TVL	3	487796	162672	291413
TVU	1	4704	1493	2764
TVU	2	249247	83082	148879
TVU	3	291433	162692	291433
F	1	4660	1449	2721
F	2	249241	83076	1488723
F	3	487828	162703	291444


RMSE for divergences - (1 layers)
Concrete
Yacht
Boston
Energy
NLL for divergences - (1 layers)
Concrete
Yacht
Boston
Energy
RMSE for divergences - (2 layers)
Concrete
Yacht
Boston
Energy
NLL for divergences - (2 layers)
Concrete
Yacht
Boston
Energy
RMSE for divergences - (3 layers)
Concrete
Yacht
Boston
Energy
NLL for divergences - (3 layers)
Concrete
Yacht
Boston
Energy

Figure 7.1

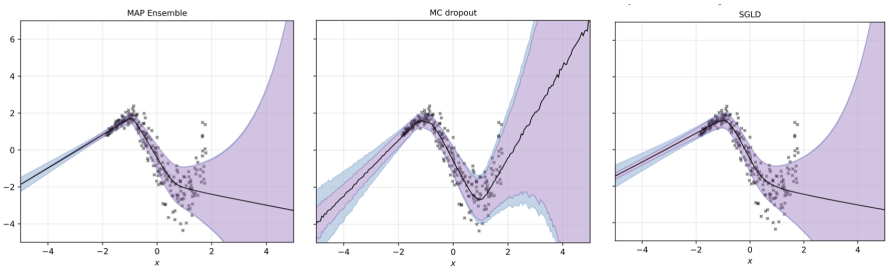


Figure 7.2