Dropout as a Bayesian Approximation: Appendix

Yarin Gal **Zoubin Ghahramani**
University of Cambridge
{yg279,zg201}@cam.ac.uk

Abstract

We show that a neural network with arbitrary depth and non-linearities, with dropout applied before every weight layer, is mathematically equivalent to an approximation to a well known Bayesian model. This interpretation offers an explanation to some of dropout's key properties, such as its robustness to over-fitting. Our interpretation allows us to reason about uncertainty in deep learning, and allows the introduction of the Bayesian machinery into existing deep learning frameworks in a principled way.

This document is an appendix for the main paper "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning" by Gal and Ghahramani, 2015.

1 Introduction

Deep learning works very well in practice for many tasks, ranging from image processing [Krizhevsky et al., 2012] to language modelling [Bengio et al., 2006]. However the framework has some major limitations as well. Our inability to reason about uncertainty over the features is an example of such – the features extracted from a dataset are often given as point estimates. These do not capture how much the model is confident in its estimation. On the other hand, probabilistic Bayesian models such as the Gaussian process [Rasmussen and Williams, 2006] offer us the ability to reason about our confidence. But these often come with a price of lessened performance.

Another major obstacle with deep learning techniques is over-fitting. This problem has been largely answered with the introduction of dropout [Hinton et al., 2012; Srivastava et al., 2014]. Indeed many modern models use dropout to avoid over-fitting in practice. Over the last several years many have tried to explain why dropout helps in avoiding over-fitting, a property which is not often observed in *Bayesian models*. Papers such as [Wager et al., 2013; Baldi and Sadowski, 2013] have suggested that dropout performs stochastic gradient descent on a regularised error function, or is equivalent to an L_2 regulariser applied after scaling the features by some estimate.

Here we show that a neural network (NN) with arbitrary depth and non-linearities, with dropout applied before every weight layer, is mathematically equivalent to an approximation to the probabilistic deep Gaussian process model [Damianou and Lawrence, 2013]. We would like to stress that no simplifying assumptions are made on the use of dropout in the literature, and that the results derived are applicable to any network architecture that makes use of dropout exactly as it appears in practical applications. We show that the dropout objective, in effect, minimises the Kullback–Leibler divergence between an approximate model and the deep Gaussian process.

We survey possible applications of this new interpretation, and discuss insights shedding light on dropout's properties. This interpretation of dropout as a Bayesian model offers an explanation to some of its properties, such as its ability to avoid over-fitting. Further, our insights allow us to treat NNs with dropout as fully Bayesian models, and obtain uncertainty estimates over their features. In practice, this allows the introduction of Bayesian machinery into existing deep learning frameworks in a principled way. Lastly, our analysis suggests straightforward generalisations of dropout for future research which should improve on current techniques.

The work presented here is an extensive theoretical treatment of the above, with applications studied separately. We next review the required background, namely dropout, Gaussian processes, and variational inference. We then derive the main results of the paper. We finish with insights and applications, and discuss how various dropout variants fit within our framework.

2 Background

We review dropout, and survey the Gaussian process model[1] and approximate variational inference quickly. These tools will be used in the following section to derive the main results of this work. We use the following notation throughout the paper. Bold lower case letters ($\mathbf{x}$) denote vectors, bold upper case letters ($\mathbf{X}$) denote matrices, and standard weight letters (x) denote scalar quantities. We use subscripts to denote either entire rows / columns (with bold letters, $\mathbf{x}_i$), or specific elements (x_{ij}). We use subscripts to denote variables as well (such as $\mathbf{W}_1 : Q \times K, \mathbf{W}_2 : K \times D$), with corresponding lower case indices to refer to specific rows / columns ($\mathbf{w}_q, \mathbf{w}_k$ for the first variable and $\mathbf{w}_k, \mathbf{w}_d$ for the second). We use a second subscript to denote the element index of a specific variable: $w_{1,qk}$ denotes the element at row q column k of the variable $\mathbf{W}_1$.

2.1 Dropout

We review the dropout NN model [Hinton et al., 2012; Srivastava et al., 2014] quickly for the case of a *single hidden layer* NN. This is done for ease of notation, and the generalisation to multiple layers is straightforward. Denote by $\mathbf{W}_1, \mathbf{W}_2$ the weight matrices connecting the first layer to the hidden layer and connecting the hidden layer to the output layer respectively. These linearly transform the layers' inputs before applying some element-wise non-linearity $\sigma(\cdot)$. Denote by $\mathbf{b}$ the biases by which we shift the input of the non-linearity. We assume the model to output D dimensional vectors while its input is Q dimensional vectors, with K hidden units. Thus $\mathbf{W}_1$ is a $Q \times K$ matrix, $\mathbf{W}_2$ is a $K \times D$ matrix, and $\mathbf{b}$ is a K dimensional vector. A standard NN model would output $\widehat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2$ given some input $\mathbf{x}$.[2]

Dropout is applied by sampling two binary vectors $\mathbf{z}_1, \mathbf{z}_2$ of dimensions Q and K respectively. The elements of the vectors are distributed according to a Bernoulli distribution with some parameter $p_i \in [0,1]$ for $i = 1, 2$. Thus $\mathbf{z}_{1,q} \sim \text{Bernoulli}(p_1)$ for $q = 1, ..., Q$, and $\mathbf{z}_{2,k} \sim \text{Bernoulli}(p_2)$ for $k = 1, ..., K$. Given an input $\mathbf{x}$, $1 - p_1$ proportion of the elements of the input are set to zero: $\mathbf{x} \circ \mathbf{z}_1$ where $\circ$ signifies the Hadamard product. The output of the first layer is given by $\sigma((\mathbf{x} \circ \mathbf{z}_1)\mathbf{W}_1 + \mathbf{b}) \circ \mathbf{z}_2$, which is linearly transformed to give the dropout model's output $\widehat{\mathbf{y}} = \big((\sigma((\mathbf{x} \circ \mathbf{z}_1)\mathbf{W}_1 + \mathbf{b})) \circ \mathbf{z}_2\big)\mathbf{W}_2$. This is equivalent to multiplying the weight matrices by the binary vectors to zero out entire rows:

$$\widehat{\mathbf{y}} = \sigma(\mathbf{x}(\mathbf{z}_1\mathbf{W}_1) + \mathbf{b})(\mathbf{z}_2\mathbf{W}_2).$$

The process is repeated for multiple layers. Note that to keep notation clean we will write $\mathbf{z}_1$ when we mean diag($\mathbf{z}_1$) with the diag($\cdot$) operator mapping a vector to a diagonal matrix whose diagonal is the elements of the vector.

To use the NN model for regression we might use the Euclidean loss (also known as "square loss"),

$$E = \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 \tag{1}$$

where $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ are N observed outputs, and $\{\widehat{\mathbf{y}}_1, \ldots, \widehat{\mathbf{y}}_N\}$ being the outputs of the model with corresponding observed inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.

To use the model for classification, predicting the probability of $\mathbf{x}$ being classified with label $1, ..., D$, we pass the output of the model $\widehat{\mathbf{y}}$ through an element-wise softmax function to obtain normalised scores: $\widehat{p}_{nd} = \exp(\widehat{y}_{nd}) / \left(\sum_{d'} \exp(\widehat{y}_{nd'})\right)$. Taking the log of this function results in a *softmax* loss,

$$E = -\frac{1}{N} \sum_{n=1}^{N} \log(\widehat{p}_{n,c_n}) \tag{2}$$

[1] For a full treatment of Gaussian Processes, see Rasmussen and Williams [2006].

[2] Note that we omit the outer-most bias term as this is equivalent to centring the output.

where $c_n \in [1, 2, ..., D]$ is the observed class for input n.

During optimisation a regularisation term is often added. We often use L_2 regularisation weighted by some weight decay λ (alternatively, the derivatives might be scaled), resulting in a minimisation objective (often referred to as cost),

$$\mathcal{L}_{\text{dropout}} := E + \lambda\big(||\mathbf{W}_1||_2^2 + ||\mathbf{W}_2||_2^2 + ||\mathbf{b}||_2^2\big). \tag{3}$$

We sample new realisations for the binary vectors $\mathbf{z}_i$ for every input point and every forward pass thorough the model (evaluating the model's output), and use the same values in the backward pass (propagating the derivatives to the parameters).

The dropped weights $\mathbf{z}_1\mathbf{W}_1$ and $\mathbf{z}_2\mathbf{W}_2$ are often scaled by $\frac{1}{p_i}$ to maintain constant output magnitude. At test time no sampling takes place. This is equivalent to initialising the weights $\mathbf{W}_i$ with scale $\frac{1}{p_i}$ with no further scaling at training time, and at test time scaling the weights $\mathbf{W}_i$ by p_i.

We will show that equations (1) to (3) arise in Gaussian process approximation as well. But first, we introduce the Gaussian process model.

2.2 Gaussian Processes

The Gaussian process (GP) is a powerful tool in statistics that allows us to model distributions over functions. It has been applied in both the supervised and unsupervised domains, for both regression and classification tasks [Rasmussen and Williams, 2006; Titsias and Lawrence, 2010; Gal et al., 2015]. The Gaussian process offers desirable properties such as uncertainty estimates over the function values, robustness to over-fitting, and principled ways for hyper-parameter tuning. The use of *approximate variational inference* for the model allows us to scale it to large data via stochastic and distributed inference [Hensman et al., 2013; Gal et al., 2014].

Given a training dataset consisting of N inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and their corresponding outputs $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, we would like to estimate a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ that is likely to have generated our observations. We denote the inputs $\mathbf{X} \in \mathbb{R}^{N \times Q}$ and the outputs $\mathbf{Y} \in \mathbb{R}^{N \times D}$.

What is a function that is likely to have generated our data? Following the Bayesian approach we would put some *prior* distribution over the space of functions $p(\mathbf{f})$. This distribution represents our prior belief as to which functions are more likely and which are less likely to have generated our data. We then look for the *posterior* distribution over the space of functions given our dataset $(\mathbf{X}, \mathbf{Y})$:

$$p(\mathbf{f}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \mathbf{f})p(\mathbf{f}).$$

This distribution captures the most likely functions given our observed data.

By modelling our distribution over the space of functions with a Gaussian process we can analytically evaluate its corresponding posterior in regression tasks, and estimate the posterior in classification tasks. In practice what this means is that for regression we place a joint Gaussian distribution over all function values,

$$\begin{aligned}\mathbf{F} \mid \mathbf{X} &\sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X})) \\ \mathbf{Y} \mid \mathbf{F} &\sim \mathcal{N}(\mathbf{F}, \tau^{-1}\mathbf{I}_N)\end{aligned} \tag{4}$$

with some precision hyper-parameter τ and where $\mathbf{I}_N$ is the identity matrix with dimensions $N \times N$. For classification we sample from a categorical distribution with probabilities given by passing $\mathbf{Y}$ through an element-wise softmax,

$$\begin{aligned}&\mathbf{F} \mid \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X})) \\ &\mathbf{Y} \mid \mathbf{F} \sim \mathcal{N}(\mathbf{F}, 0 \cdot \mathbf{I}_N) \\ &c_n \mid \mathbf{Y} \sim \text{Categorical}\left(\exp(y_{nd}) \Big/ \left(\sum_{d'} \exp(y_{nd'})\right)\right)\end{aligned} \tag{5}$$

for $n = 1, ..., N$ with observed class label c_n. Note that we did not simply write $\mathbf{Y} = \mathbf{F}$ because of notational convenience that will allow us to treat regression and classification together.

To model the data we have to choose a covariance function $\mathbf{K}(\mathbf{X}_1, \mathbf{X}_2)$ for the Gaussian distribution. This function defines the (scalar) similarity between every pair of input points $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$. Given a finite dataset of size N this function induces an $N \times N$ covariance matrix which we will denote $\mathbf{K} := \mathbf{K}(\mathbf{X}, \mathbf{X})$. For example we may choose a stationary squared exponential covariance function.

We will see below that certain non-stationary covariance functions correspond to *TanH* (hyperbolic tangent) or *ReLU* (rectified linear) NNs.

Evaluating the Gaussian distribution above involves an inversion of an N by N matrix, an operation that requires $\mathcal{O}(N^3)$ time complexity. Many approximations to the Gaussian process result in a manageable time complexity. Variational inference can be used for such, and will be explained next.

2.3 Variational Inference

To approximate the model above we could condition the model on a finite set of random variables $\boldsymbol{\omega}$. We make a modelling assumption and assume that the model depends on these variables alone, making them into sufficient statistics in our approximate model.

The predictive distribution for a new input point $\mathbf{x}^*$ is then given by

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) \, \mathrm{d}\boldsymbol{\omega},$$

with $\mathbf{y}^* \in \mathbb{R}^D$. The distribution $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ cannot usually be evaluated analytically. Instead we define an approximating *variational* distribution $q(\boldsymbol{\omega})$, whose structure is easy to evaluate.

We would like our approximating distribution to be as close as possible to the posterior distribution obtained from the full Gaussian process. We thus minimise the Kullback–Leibler (KL) divergence, intuitively a measure of similarity between two distributions:

$$\mathrm{KL}(q(\boldsymbol{\omega}) \mid p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})),$$

resulting in the approximate predictive distribution

$$q(\mathbf{y}^*|\mathbf{x}^*) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) q(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}. \tag{6}$$

Minimising the Kullback–Leibler divergence is equivalent to maximising the *log evidence lower bound* [Bishop, 2006],

$$\mathcal{L}_{\text{VI}} := \int q(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} - \mathrm{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})) \tag{7}$$

with respect to the variational parameters defining $q(\boldsymbol{\omega})$. Note that the KL divergence in the last equation is between the approximate posterior and the *prior* over $\boldsymbol{\omega}$. Maximising this objective will result in a variational distribution $q(\boldsymbol{\omega})$ that explains the data well (as obtained from the first term—the log likelihood) while still being close to prior—preventing the model from over-fitting.

We next present a variational approximation to the Gaussian process extending on [Gal and Turner, 2015], which results in a model mathematically identical to the use of dropout in arbitrarily structured NNs with arbitrary non-linearities.

3 Dropout as a Bayesian Approximation

We show that NNs with dropout applied before every weight layer are mathematically equivalent to approximate variational inference in the deep Gaussian process. For this we build on previous work [Gal and Turner, 2015] that applied variational inference in the *sparse spectrum* Gaussian process approximation [Lázaro-Gredilla et al., 2010]. Starting with the full Gaussian process we will develop an approximation that will be shown to be equivalent to the NN optimisation objective with dropout (eq. (3)) with either the Euclidean loss (eq. (1)) in the case of regression or softmax loss (eq. (2)) in the case of classification. This view of dropout will allow us to derive new probabilistic results in deep learning.

3.1 A Gaussian Process Approximation

We begin by defining our covariance function. Let $\sigma(\cdot 1)$ be some non-linear function such as the rectified linear (ReLU) or the hyperbolic tangent function (TanH). We define $\mathbf{K}(\mathbf{x}, \mathbf{y})$ to be

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = \int p(\mathbf{w}) p(b) \sigma(\mathbf{w}^T \mathbf{x} + b) \sigma(\mathbf{w}^T \mathbf{y} + b) \mathrm{d}\mathbf{w} \mathrm{d}b$$

with $p(\mathbf{w})$ a standard multivariate normal distribution of dimensionality Q and some distribution $p(b)$. It is trivial to show that this defines a valid covariance function following [Tsuda et al., 2002].

We use Monte Carlo integration with K terms to approximate the integral above. This results in

$$\widehat{\mathbf{K}}(\mathbf{x}, \mathbf{y}) = \frac{1}{K} \sum_{k=1}^{K} \sigma(\mathbf{w}_k^T \mathbf{x} + b_k) \sigma(\mathbf{w}_k^T \mathbf{y} + b_k)$$

with $\mathbf{w}_k \sim p(\mathbf{w})$ and $b_k \sim p(b)$. K will be the number of hidden units in our single hidden layer NN approximation.

Using $\widehat{\mathbf{K}}$ instead of $\mathbf{K}$ as the covariance function of the Gaussian process yields the following generative model:

$$\begin{aligned}
\mathbf{w}_k &\sim p(\mathbf{w}),\ b_k \sim p(b), \\
\mathbf{W}_1 &= [\mathbf{w}_k]_{k=1}^K, \mathbf{b} = [b_k]_{k=1}^K \\
\widehat{\mathbf{K}}(\mathbf{x}, \mathbf{y}) &= \frac{1}{K} \sum_{k=1}^{K} \sigma(\mathbf{w}_k^T \mathbf{x} + b_k) \sigma(\mathbf{w}_k^T \mathbf{y} + b_k) \\
\mathbf{F} \mid \mathbf{X}, \mathbf{W}_1, \mathbf{b} &\sim \mathcal{N}(\mathbf{0}, \widehat{\mathbf{K}}(\mathbf{X}, \mathbf{X})) \\
\mathbf{Y} \mid \mathbf{F} &\sim \mathcal{N}(\mathbf{F}, \tau^{-1} \mathbf{I}_N),
\end{aligned} \tag{8}$$

with $\mathbf{W}_1$ a $Q \times K$ matrix.

This results in the following predictive distribution:

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{F}) p(\mathbf{F}|\mathbf{W}_1, \mathbf{b}, \mathbf{X}) p(\mathbf{W}_1) p(\mathbf{b})$$

where the integration is with respect to $\mathbf{F}, \mathbf{W}_1$, and $\mathbf{b}$.

Denoting the $1 \times K$ row vector

$$\boldsymbol{\phi}(\mathbf{x}, \mathbf{W}_1, \mathbf{b}) = \sqrt{\frac{1}{K}} \sigma(\mathbf{W}_1^T \mathbf{x} + \mathbf{b})$$

and the $N \times K$ feature matrix $\Phi = [\boldsymbol{\phi}(\mathbf{x}_n, \mathbf{W}_1, \mathbf{b})]_{n=1}^N$, we have $\widehat{\mathbf{K}}(\mathbf{X}, \mathbf{X}) = \Phi \Phi^T$. We rewrite $p(\mathbf{Y}|\mathbf{X})$ as

$$p(\mathbf{Y}|\mathbf{X}) = \int \mathcal{N}(\mathbf{Y}; \mathbf{0}, \Phi\Phi^T + \tau^{-1} \mathbf{I}_N) p(\mathbf{W}_1) p(\mathbf{b}) \mathrm{d}\mathbf{W}_1 \mathrm{d}\mathbf{b},$$

analytically integrating with respect to $\mathbf{F}$.

The normal distribution of $\mathbf{Y}$ inside the integral above can be written as a joint normal distribution over $\mathbf{y}_d$, the d'th columns of the $N \times D$ matrix $\mathbf{Y}$, for $d = 1, ..., D$. For each term in the joint distribution, following identity [Bishop, 2006, page 93], we introduce a $K \times 1$ auxiliary random variable $\mathbf{w}_d \sim \mathcal{N}(0, \mathbf{I}_K)$,

$$\mathcal{N}(\mathbf{y}_d; 0, \Phi\Phi^T + \tau^{-1} \mathbf{I}_N) = \int \mathcal{N}(\mathbf{y}_d; \Phi \mathbf{w}_d, \tau^{-1} \mathbf{I}_N) \mathcal{N}(\mathbf{w}_d; 0, \mathbf{I}_K) \mathrm{d}\mathbf{w}_d.$$

Writing $\mathbf{W}_2 = [\mathbf{w}_d]_{d=1}^D$ a $K \times D$ matrix, the above is equivalent to[3]

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) p(\mathbf{W}_1) p(\mathbf{W}_2) p(\mathbf{b})$$

where the integration is with respect to $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{b}$.

We have re-parametrised the GP model and marginalised over the additional auxiliary random variables $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{b}$. We next approximate the posterior over these variables with appropriate approximating variational distributions.

3.2 Variational Inference in the Approximate Model

Our sufficient statistics are $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{b}$. To perform variational inference in our approximate model we need to define a variational distribution $q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) := q(\mathbf{W}_1) q(\mathbf{W}_2) q(\mathbf{b})$. We define

[3] This is equivalent to the weighted basis function interpretation of the Gaussian process [Rasmussen and Williams, 2006] where the various quantities are analytically integrated over.

$q(\mathbf{W}_1)$ to be a Gaussian mixture distribution with two components, factorised over Q:[4]

$$q(\mathbf{W}_1) = \prod_{q=1}^{Q} q(\mathbf{w}_q), \tag{9}$$
$$q(\mathbf{w}_q) = p_1 \mathcal{N}(\mathbf{m}_q, \boldsymbol{\sigma}^2 \mathbf{I}_K) + (1 - p_1)\mathcal{N}(0, \boldsymbol{\sigma}^2 \mathbf{I}_K)$$

with some probability $p_1 \in [0, 1]$, scalar $\boldsymbol{\sigma} > 0$ and $\mathbf{m}_q \in \mathbb{R}^K$. We put a similar approximating distribution over $\mathbf{W}_2$:

$$q(\mathbf{W}_2) = \prod_{k=1}^{K} q(\mathbf{w}_k), \tag{10}$$
$$q(\mathbf{w}_k) = p_2 \mathcal{N}(\mathbf{m}_k, \boldsymbol{\sigma}^2 \mathbf{I}_D) + (1 - p_2)\mathcal{N}(0, \boldsymbol{\sigma}^2 \mathbf{I}_D)$$

with some probability $p_2 \in [0, 1]$.

We put a simple Gaussian approximating distribution over $\mathbf{b}$:

$$q(\mathbf{b}) = \mathcal{N}(\mathbf{m}, \boldsymbol{\sigma}^2 \mathbf{I}_K). \tag{11}$$

Next we evaluate the log evidence lower bound for the task of regression, for which we optimise over the variational parameters $\mathbf{M}_1 = [\mathbf{m}_q]_{q=1}^{Q}$, $\mathbf{M}_2 = [\mathbf{m}_k]_{k=1}^{K}$, and $\mathbf{m}$, to maximise Eq. (7). The task of classification is discussed later.

3.3 Evaluating the Log Evidence Lower Bound for Regression

We need to evaluate the log evidence lower bound:

$$\mathcal{L}_{\text{GP-VI}} := \int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) - \text{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})||p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})), \tag{12}$$

where the integration is with respect to $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{b}$.

For the task of regression we can rewrite the integrand as a sum:

$$\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = \sum_{d=1}^{D} \log \mathcal{N}(\mathbf{y}_d; \Phi \mathbf{w}_d, \tau^{-1} \mathbf{I}_N)$$
$$= -\frac{ND}{2} \log(2\pi) + \frac{ND}{2} \log(\tau) - \sum_{d=1}^{D} \frac{\tau}{2} ||\mathbf{y}_d - \Phi \mathbf{w}_d||_2^2,$$

as the output dimensions of a multi-output Gaussian process are assumed to be independent. Denote $\widehat{\mathbf{Y}} = \Phi \mathbf{W}_2$. We can then sum over the rows instead of the columns of $\widehat{\mathbf{Y}}$ and write

$$\sum_{d=1}^{D} \frac{\tau}{2} ||\mathbf{y}_d - \widehat{\mathbf{y}}_d||_2^2 = \sum_{n=1}^{N} \frac{\tau}{2} ||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2.$$

Here $\widehat{\mathbf{y}}_n = \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{W}_1, \mathbf{b})\mathbf{W}_2 = \sqrt{\frac{1}{K}} \sigma(\mathbf{x}_n \mathbf{W}_1 + \mathbf{b})\mathbf{W}_2$, resulting in the integrand

$$\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = \sum_{n=1}^{N} \log \mathcal{N}(\mathbf{y}_n; \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{W}_1, \mathbf{b})\mathbf{W}_2, \tau^{-1} \mathbf{I}_D).$$

This allows us to write the log evidence lower bound as

$$\sum_{n=1}^{N} \int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \log p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) - \text{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})||p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})).$$

We re-parametrise the integrands in the sum to not depend on $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{b}$ directly, but instead on the standard normal distribution and the Bernoulli distribution. Let $q(\boldsymbol{\epsilon}_1) = \mathcal{N}(\mathbf{0}, \mathbf{I}_{Q \times K})$ and $q(\mathbf{z}_{1,q}) = \text{Bernoulli}(p_1)$ for $q = 1, ..., Q$, and $q(\boldsymbol{\epsilon}_2) = \mathcal{N}(\mathbf{0}, \mathbf{I}_{K \times D})$ and $q(\mathbf{z}_{2,k}) = \text{Bernoulli}(p_2)$ for $k = 1, ..., K$. Finally let $q(\boldsymbol{\epsilon}) = \mathcal{N}(0, \mathbf{I}_K)$. We write

$$\mathbf{W}_1 = \mathbf{z}_1(\mathbf{M}_1 + \boldsymbol{\sigma}\boldsymbol{\epsilon}_1) + (1 - \mathbf{z}_1)\boldsymbol{\sigma}\boldsymbol{\epsilon}_1,$$

[4] Note that this is a bi-modal distribution defined over each output dimensionality; as a result the joint distribution over $\mathbf{W}_1$ is highly multi-modal.

$$\mathbf{W}_2 = \mathbf{z}_2(\mathbf{M}_2 + \boldsymbol{\sigma}\boldsymbol{\epsilon}_2) + (1 - \mathbf{z}_2)\boldsymbol{\sigma}\boldsymbol{\epsilon}_2,$$
$$\mathbf{b} = \mathbf{m} + \boldsymbol{\sigma}\boldsymbol{\epsilon}, \tag{13}$$

allowing us to re-write the sum over the integrals in the above equation as

$$\sum_{n=1}^{N} \int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \log p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \mathrm{d}\mathbf{W}_1 \mathrm{d}\mathbf{W}_2 \mathrm{d}\mathbf{b}$$
$$= \sum_{n=1}^{N} \int q(\mathbf{z}_1, \boldsymbol{\epsilon}_1, \mathbf{z}_2, \boldsymbol{\epsilon}_2, \boldsymbol{\epsilon}) \log p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}_1(\mathbf{z}_1, \boldsymbol{\epsilon}_1), \mathbf{W}_2(\mathbf{z}_2, \boldsymbol{\epsilon}_2), \mathbf{b}(\boldsymbol{\epsilon}))$$

where each integration is over $\boldsymbol{\epsilon}_1, \mathbf{z}_1, \boldsymbol{\epsilon}_2, \mathbf{z}_2$, and $\boldsymbol{\epsilon}$.

We estimate each integral using Monte Carlo integration with a distinct single sample to obtain:

$$\mathcal{L}_{\text{GP-MC}} := \sum_{n=1}^{N} \log p(\mathbf{y}_n | \mathbf{x}_n, \widehat{\mathbf{W}}_1^n, \widehat{\mathbf{W}}_2^n, \widehat{\mathbf{b}}^n) - \text{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) || p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}))$$

with realisations $\widehat{\mathbf{W}}_1^n, \widehat{\mathbf{W}}_2^n, \widehat{\mathbf{b}}^n$ defined following eq. (13) with $\widehat{\boldsymbol{\epsilon}}_1^n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{Q \times K})$, $\widehat{\mathbf{z}}_{1,q}^n \sim \text{Bernoulli}(p_1)$, $\widehat{\boldsymbol{\epsilon}}_2^n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{K \times D})$, and $\widehat{\mathbf{z}}_{2,k}^n \sim \text{Bernoulli}(p_2)$. Following [Blei et al., 2012; Hoffman et al., 2013; Kingma and Welling, 2013; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014], optimising the *stochastic* objective $\mathcal{L}_{\text{GP-MC}}$ we would converge to the same limit as $\mathcal{L}_{\text{GP-VI}}$.

We can't evaluate the KL divergence term between a mixture of Gaussians and a single Gaussian analytically. However we can perform Monte Carlo integration like in the above. A further approximation for large K (number of hidden units) and small $\boldsymbol{\sigma}^2$ yields a weighted sum of KL divergences between the mixture components and the single Gaussian (proposition 1 in the appendix). Intuitively, this is because the entropy of a mixture of Gaussians with a large enough dimensionality and randomly distributed means tends towards the sum of the Gaussians' volumes. Following the proposition, for large enough K we can approximate the KL divergence term as

$$\text{KL}(q(\mathbf{W}_1) || p(\mathbf{W}_1)) \approx QK(\boldsymbol{\sigma}^2 - \log(\boldsymbol{\sigma}^2) - 1) + \frac{p_1}{2} \sum_{q=1}^{Q} \mathbf{m}_q^T \mathbf{m}_q + C$$

with constant C, and similarly for $\text{KL}(q(\mathbf{W}_2) || p(\mathbf{W}_2))$. The term $\text{KL}(q(\mathbf{b}) || p(\mathbf{b}))$ can be evaluated analytically as

$$\text{KL}(q(\mathbf{b}) || p(\mathbf{b})) = \frac{1}{2} \big(\mathbf{m}^T \mathbf{m} + K(\boldsymbol{\sigma}^2 - \log(\boldsymbol{\sigma}^2) - 1) \big) + C$$

with constant C. We drop the constants for brevity.

Next we explain the relation between the above equations and the equations brought in section 2.1.

3.4 Log Evidence Lower Bound Optimisation

Ignoring the constant terms $\tau, \boldsymbol{\sigma}$ we obtain the maximisation objective

$$\mathcal{L}_{\text{GP-MC}} \propto -\frac{\tau}{2} \sum_{n=1}^{N} ||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 - \frac{p_1}{2} ||\mathbf{M}_1||_2^2 - \frac{p_2}{2} ||\mathbf{M}_2||_2^2 - \frac{1}{2} ||\mathbf{m}||_2^2. \tag{14}$$

Note that in the Gaussian processes literature the terms $\tau, \boldsymbol{\sigma}$ will often be optimised as well.

Letting $\boldsymbol{\sigma}$ tend to zero, we get that the KL divergence blows-up and tends to infinity. However, in real-world scenarios setting $\boldsymbol{\sigma}$ to be machine epsilon (10^{-33} for example in quadruple precision decimal systems) results in a constant value $\log \boldsymbol{\sigma} = -76$. With high probability samples from a standard Gaussian distribution with such a small standard deviation will be represented on a computer, in effect, as zero. Thus the random variable realisations $\widehat{\mathbf{W}}_1^n, \widehat{\mathbf{W}}_2^n, \widehat{\mathbf{b}}^n$ can be approximated as

$$\widehat{\mathbf{W}}_1^n \approx \widehat{\mathbf{z}}_1^n \mathbf{M}_1, \;\; \widehat{\mathbf{W}}_2^n \approx \widehat{\mathbf{z}}_2^n \mathbf{M}_2, \;\; \widehat{\mathbf{b}}^n \approx \mathbf{m}.$$

Note that $\widehat{\mathbf{W}}_1^n$ are not maximum a posteriori (MAP) estimates, but random variables realisations. This gives us

$$\widehat{\mathbf{y}}_n \approx \sqrt{\frac{1}{K}} \sigma(\mathbf{x}_n(\widehat{\mathbf{z}}_1^n \mathbf{M}_1) + \mathbf{m})(\widehat{\mathbf{z}}_2^n \mathbf{M}_2).$$

Scaling the optimisation objective by a positive constant $\frac{1}{\tau N}$ doesn't change the parameter values at its optimum (as long as we don't optimise with respect to τ). We thus scale the objective to get

$$\mathcal{L}_{\text{GP-MC}} \propto -\frac{1}{2N}\sum_{n=1}^{N}||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 - \frac{p_1}{2\tau N}||\mathbf{M}_1||_2^2 - \frac{p_2}{2\tau N}||\mathbf{M}_2||_2^2 - \frac{1}{2\tau N}||\mathbf{m}||_2^2 \tag{15}$$

and we recovered equation (1) for an appropriate setting of τ. Maximising eq. (15) results in the same optimal parameters as the minimisation of eq. (3). Note that eq. (15) is a scaled unbiased estimator of eq. (12). With correct stochastic optimisation scheduling both will converge to the same limit.

The optimisation of $\mathcal{L}_{\text{GP-MC}}$ proceeds as follows. We sample realisations $\widehat{\mathbf{z}}_1^n, \widehat{\mathbf{z}}_2^n$ to evaluate the lower-bound and its derivatives. We perform a single optimisation step (for example a single gradient descent step), and repeat, sampling new realisations.

We can make several interesting observations at this point. First, we can find the model precision from the identity $\lambda = \frac{p}{2\tau N}$ which gives $\tau = \frac{p}{2\lambda N}$. Second, it seems that the weight-decay for the dropped-out weights should be scaled by the probability of the weights to not be dropped. This might explain why doubling the learning rate of the bias during NN optimisation works well in practice in dropout networks with $p = 0.5$. Lastly, it is known that setting the dropout probability to zero ($p_1 = p_2 = 1$) results in a standard NN. Following the derivation above, this would result in delta function approximating distributions on the weights (replacing eqs. (9)-(11)). As was discussed in [Lázaro-Gredilla et al., 2010] this leads to model over-fitting. Empirically it seems that the Bernoulli approximating distribution is sufficient to considerably prevent over-fitting.

Note that even though our approximating distribution is, in effect, made of a sum of two point masses, each point mass with zero variance, the mixture does not have zero variance. It has the variance of a Bernoulli random variable, which is transformed through the network. This choice of approximating distribution results in the dropout model.

4 Extensions

We have presented the derivation for a single hidden layer NN in the task of regression. The derivation above extends to tasks of classification, flexible priors, mini-batch optimisation, deep models, and much more. This will be studied next.

4.1 Evaluating the Log Evidence Lower Bound for Classification

For classification we have an additional step in the generative model in eq. (5) compared to eq. (4), sampling class assignment c_n given weight $\mathbf{y}_n$. We can write this generative model using the auxiliary random variables introduced in section 3.1 for the regression case by

$$\begin{aligned} p(\mathbf{c}|\mathbf{X}) &= \int p(\mathbf{c}|\mathbf{Y})p(\mathbf{Y}|\mathbf{X})\mathrm{d}\mathbf{Y} \\ &= \int p(\mathbf{c}|\mathbf{Y})\Big(\int p(\mathbf{Y}|\mathbf{X},\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})p(\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})\mathrm{d}\mathbf{W}_1\mathrm{d}\mathbf{W}_2\mathrm{d}\mathbf{b}\Big)\mathrm{d}\mathbf{Y} \end{aligned}$$

where $\mathbf{c}$ is an N dimensional vector of categorical values. We can write the log evidence lower bound in this case as (proposition 2 in the appendix)

$$\begin{aligned} \mathcal{L}_{\text{GP-VI}} := &\int p(\mathbf{Y}|\mathbf{X},\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})q(\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})\log p(\mathbf{c}|\mathbf{Y})\mathrm{d}\mathbf{W}_1\mathrm{d}\mathbf{W}_2\mathrm{d}\mathbf{b}\mathrm{d}\mathbf{Y} \\ &- \mathrm{KL}(q(\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})||p(\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})). \end{aligned}$$

The integrand of the first term can be re-written like before as a sum

$$\log p(\mathbf{c}|\widehat{\mathbf{Y}}) = \sum_{n=1}^{N}\log p(\mathbf{c}_n|\widehat{\mathbf{y}}_n)$$

resulting in a log evidence lower bound given by

$$\begin{aligned} \mathcal{L}_{\text{GP-VI}} := &\sum_{n=1}^{N}\int p(\mathbf{y}_n|\mathbf{x}_n,\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})q(\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})\log p(\mathbf{c}_n|\mathbf{y}_n) \\ &- \mathrm{KL}(q(\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})||p(\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})) \end{aligned}$$

where the integration of each term in the first expression is over $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}$, and $\mathbf{y}_n$.

We can re-parametrise each integrand in the sum following (13) to obtain

$$\begin{aligned}\mathbf{W}_1 &= \mathbf{z}_1(\mathbf{M}_1 + \boldsymbol{\sigma}\boldsymbol{\epsilon}_1) + (1 - \mathbf{z}_1)\boldsymbol{\sigma}\boldsymbol{\epsilon}_1,\\ \mathbf{W}_2 &= \mathbf{z}_2(\mathbf{M}_2 + \boldsymbol{\sigma}\boldsymbol{\epsilon}_2) + (1 - \mathbf{z}_2)\boldsymbol{\sigma}\boldsymbol{\epsilon}_2,\\ \mathbf{b} &= \mathbf{m} + \boldsymbol{\sigma}\boldsymbol{\epsilon},\\ \mathbf{y}_n &= \sqrt{\frac{1}{K}}\sigma(\mathbf{x}_n\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2. \end{aligned} \tag{16}$$

Like before, we estimate each integral using Monte Carlo integration with a distinct single sample to obtain:

$$\mathcal{L}_{\text{GP-MC}} := \sum_{n=1}^{N} \log p(\mathbf{c}_n | \widehat{\mathbf{y}}_n(\mathbf{x}_n, \widehat{\mathbf{W}}_1^n, \widehat{\mathbf{W}}_2^n, \widehat{\mathbf{b}}^n)) - \text{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) || p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}))$$

with realisations $\widehat{\mathbf{y}}_n, \widehat{\mathbf{W}}_1^n, \widehat{\mathbf{W}}_2^n$, and $\widehat{\mathbf{b}}^n$.

Each term in the sum in the first expression can be re-written as

$$\log p(\mathbf{c}_n | \widehat{\mathbf{y}}_n) = \widehat{y}_{nc_n} - \log\left(\sum_{d'} \exp(\widehat{y}_{nd'})\right).$$

We evaluate the second expression as before. Scaling the objective by a positive $\frac{1}{N}$ this results in the following maximisation objective,

$$\mathcal{L}_{\text{GP-MC}} \propto \frac{1}{N}\sum_{n=1}^{N} \widehat{p}_{n,c_n} - \frac{p_1}{2N}||\mathbf{M}_1||_2^2 - \frac{p_2}{2N}||\mathbf{M}_2||_2^2 - \frac{1}{2N}||\mathbf{m}||_2^2,$$

with $\widehat{p}_{n,c_n} = \log p(\mathbf{c}_n | \widehat{\mathbf{y}}_n)$, identical (up to a sign flip) to that of eqs. (2), (3) for appropriate selection of weight decay λ.

4.2 Prior Length-scale

We can define a more expressive prior than $\mathcal{N}(0, I_K)$ over the weights, that will allow us to incorporate our prior belief over the frequency of the observed data. Instead of $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; 0, I_K)$ we may use $p_l(\mathbf{w}) = \mathcal{N}(\mathbf{w}; 0, l^{-2} I_K)$ with length-scale l.

To use this more expressive prior we need to adapt proposition 1 approximating the prior term's KL divergence. It is easy to see that the KL divergence between $q(\mathbf{x})$ and $p_l(\mathbf{x})$ can be approximated as:

$$\text{KL}(q(\mathbf{x}) || p_l(\mathbf{x})) \approx \sum_{i=1}^{L} \frac{p_i}{2}\left(l^2 \boldsymbol{\mu}_i^T \boldsymbol{\mu}_i + \text{tr}(l^2 \boldsymbol{\Sigma}_i) - K - \log|\boldsymbol{\Sigma}_i| + K \log l^{-2}\right)$$

plus a constant for large enough K. This follows from the KL divergence for multivariate normal distributions. Following the previous derivations we obtain the regression objective

$$\mathcal{L}_{\text{GP-MC}} \propto -\frac{1}{2N}\sum_{n=1}^{N} ||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 - \frac{l^2 p_1}{2\tau N}||\mathbf{M}_1||_2^2 - \frac{l^2 p_2}{2\tau N}||\mathbf{M}_2||_2^2 - \frac{l^2}{2\tau N}||\mathbf{m}||_2^2, \tag{17}$$

and similarly for classification.

For high frequency data, setting l to a small value would result in a weaker regularisation over the weights. This leads to larger magnitude weights which can capture high frequency data [Gal and Turner, 2015].

The length-scale decouples the precision parameter τ from the weight-decay λ:

$$\lambda = \frac{l^2 p}{2N\tau}, \tag{18}$$

which results in

$$\tau = \frac{l^2 p}{2N\lambda}. \tag{19}$$

The length-scale is a user specified value that captures our belief over the function frequency. A short length-scale l (corresponding to high frequency data) with high precision τ (equivalently, small observation noise) results in a small weight-decay λ – encouraging the model to fit the data well.

A long length-scale with low precision results in a large weight-decay – and stronger regularisation over the weights. This trade-off between the length-scale and model precision results in different weight-decay values.

4.3 Mini-batch Optimisation

We often use mini-batches when optimising eq. (3). This is done by setting eq. (1) to

$$E = \frac{1}{2M} \sum_{n \in S} ||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2$$

with a random subset of data points S of size M, and similarly for classification.

Using recent results in stochastic variational inference [Hoffman et al., 2013] we can derive the equivalent to this in the GP approximation above. We change the likelihood term $-\frac{\tau}{2}\sum_{n=1}^N ||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2$ in eq. (14) to sum over the data points in S alone, and multiply the term by $\frac{N}{M}$ to get an unbiased estimator to the original sum. Multiplying equation (14) by $\frac{1}{\tau N}$ as before we obtain

$$\mathcal{L}_{\text{GP-MC}} \approx -\frac{1}{2M} \sum_{n \in S} ||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 - \frac{p_1}{2\tau N}||\mathbf{M}_1||_2^2 - \frac{p_2}{2\tau N}||\mathbf{M}_2||_2^2 - \frac{1}{2\tau N}||\mathbf{m}||_2^2 \tag{20}$$

recovering eq. (3) for the mini-batch optimisation case as well.

4.4 Predictive Log-likelihood

Given a dataset $\mathbf{X}, \mathbf{Y}$ and a new data point $\mathbf{x}^*$ we can calculate the probability of possible output values $\mathbf{y}^*$ using the predictive probability $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$. The log of the predictive likelihood captures how well the model fits the data, with larger values indicating better model fit.

Our predictive log-likelihood can be approximated by Monte Carlo integration of eq. (6) with T terms:

$$\begin{aligned}\log p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) &= \log \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) \mathrm{d}\boldsymbol{\omega} \\ &\approx \log \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) q(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \\ &\approx \log \left(\frac{1}{T} \sum_{t=1}^T p(y^*|\mathbf{x}^*, \boldsymbol{\omega}_t) \right)\end{aligned}$$

with $\boldsymbol{\omega}_t \sim q(\boldsymbol{\omega})$.

For regression we have

$$\log p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \text{logsumexp}\Big(-\frac{1}{2}\tau||\mathbf{y} - \widehat{\mathbf{y}}_t||^2 \Big) - \log T - \frac{1}{2}\log 2\pi - \frac{1}{2}\log \tau^{-1} \tag{21}$$

with our precision parameter τ.

Uncertainty quality can be determined from this quantity as well. Excessive uncertainty (large observation noise, or equivalently small model precision τ) results in a large penalty from the last term in the predictive log-likelihood. An over-confident model with large model precision compared to poor mean estimation results in a penalty from the first term – the distance $||\mathbf{y} - \widehat{\mathbf{y}}_t||^2$ gets amplified by τ which drives the first term to zero.

4.5 Going Deeper than a Single Hidden Layer

We will demonstrate how to extend the derivation above to two hidden layers for the case of regression. Extension to further layers and classification is trivial.

We use the deep GP model – feeding the output of one GP to the covariance of the next, in the same way the input is used in the covariance of the first GP. However, to match the dropout NN model, we have to select a different a covariance function for the GPs in the layers following the first one. For clarity, we denote here all quantities related to the first GP with subscript 1, and as a second subscript denote the element index. So $\phi_{1,nk}$ denotes the element at row n column k of the variable Φ_1, and $\boldsymbol{\phi}_{1,n}$ denotes row n of the same variable.

We next define the new covariance function $\mathbf{K}_2$. Let σ_2 be some non-linear function, not necessarily the same as the one used with the previous covariance function. We define $\mathbf{K}_2(\mathbf{x}, \mathbf{y})$ to be

$$\mathbf{K}_2(\mathbf{x}, \mathbf{y}) = \frac{1}{K_2} \int p(\mathbf{b}_2) \sigma_2(\mathbf{x} + \mathbf{b}_2)^T \sigma_2(\mathbf{y} + \mathbf{b}_2) \mathrm{d}\mathbf{b}_2$$

with some distribution $p(\mathbf{b}_2)$ over $\mathbf{b}_2 \in \mathbb{R}^{K_1}$.

We use Monte Carlo integration with one term to approximate the integral above. This results in

$$\widehat{\mathbf{K}}_2(\mathbf{x}, \mathbf{y}) = \frac{1}{K_2} \sigma(\mathbf{x} + \mathbf{b}_2)^T \sigma(\mathbf{y} + \mathbf{b}_2)$$

with $\mathbf{b}_2 \sim p(\mathbf{b}_2)$.

Using $\widehat{\mathbf{K}}_2$ instead as the covariance function of the second Gaussian process yields the following generative model. First, we sample the variables for all covariance functions:

$$\mathbf{w}_{1,k} \sim p(\mathbf{w}_1),\ b_{1,k} \sim p(b_1),\ \mathbf{b}_2 \sim p(\mathbf{b}_2)$$
$$\mathbf{W}_1 = [\mathbf{w}_{1,k}]_{k=1}^{K_1}, \mathbf{b}_1 = [b_{1,k}]_{k=1}^{K_1}$$

with $\mathbf{W}_1$ a $Q \times K_1$ matrix, $\mathbf{b}_1$ a K_1 dimensional vector, and $\mathbf{b}_2$ a K_2 dimensional vector. Given these variables, we define the covariance functions for the two GPs:

$$\widehat{\mathbf{K}}_1(\mathbf{x}, \mathbf{y}) = \frac{1}{K_1} \sum_{k=1}^{K_1} \sigma_1(\mathbf{w}_{1,k}^T \mathbf{x} + b_{1,k}) \sigma_1(\mathbf{w}_{1,k}^T \mathbf{y} + b_{1,k})$$
$$\widehat{\mathbf{K}}_2(\mathbf{x}, \mathbf{y}) = \frac{1}{K_2} \sigma(\mathbf{x} + \mathbf{b}_2)^T \sigma(\mathbf{y} + \mathbf{b}_2)$$

Conditioned on these variables, we generate the model's output:

$$\mathbf{F}_1 \mid \mathbf{X}, \mathbf{W}_1, \mathbf{b}_1 \sim \mathcal{N}(\mathbf{0}, \widehat{\mathbf{K}}_1(\mathbf{X}, \mathbf{X}))$$
$$\mathbf{F}_2 \mid \mathbf{X}, \mathbf{b}_2 \sim \mathcal{N}(\mathbf{0}, \widehat{\mathbf{K}}_2(\mathbf{F}_1, \mathbf{F}_1))$$
$$\mathbf{Y} \mid \mathbf{F}_2 \sim \mathcal{N}(\mathbf{F}_2, \tau^{-1}\mathbf{I}_N).$$

We introduce auxiliary random variables $\mathbf{W}_2$ a $K_1 \times K_2$ matrix and $\mathbf{W}_3$ a $K_2 \times D$ matrix. The columns of each matrix distribute according to $\mathcal{N}(0, \mathbf{I})$.

Like before, we write $\widehat{\mathbf{K}}_1(\mathbf{X}, \mathbf{X}) = \Phi_1 \Phi_1^T$ with Φ_1 an $N \times K_1$ matrix and $\widehat{\mathbf{K}}_2(\mathbf{X}, \mathbf{X}) = \Phi_2 \Phi_2^T$ with Φ_2 an $N \times K_2$ matrix:

$$\phi_{1,nk} = \sqrt{\frac{1}{K_1}} \sigma_1(\mathbf{w}_{1,k}^T \mathbf{x}_n + b_{1,k})$$
$$\phi_{2,nk} = \sqrt{\frac{1}{K_2}} \sigma_2(f_{1,nk} + b_{2,k}).$$

We can then write $\mathbf{F}_1 = \Phi_1 \mathbf{W}_2$, since

$$\mathbb{E}_{p(\mathbf{W}_2)}(\mathbf{F}_1) = \Phi_1 \mathbb{E}_{p(\mathbf{W}_2)}(\mathbf{W}_2) = \mathbf{0}$$

and

$$\text{Cov}_{p(\mathbf{W}_2)}(\mathbf{F}_1) = \mathbb{E}_{p(\mathbf{W}_2)}(\mathbf{F}_1 \mathbf{F}_1^T) = \Phi_1 \mathbb{E}_{p(\mathbf{W}_2)}(\mathbf{W}_2 \mathbf{W}_2^T) \Phi_1^T = \Phi_1 \Phi_1^T,$$

and similarly for $\mathbf{F}_2$. Note that $\mathbf{F}_1$ is an $N \times K_2$ matrix, and that $\mathbf{F}_2$ is an $N \times D$ matrix. Thus,

$$\phi_{2,nk} = \sqrt{\frac{1}{K_2}} \sigma_2(\mathbf{w}_{2,k}^T \boldsymbol{\phi}_{1,n} + b_{2,k}).$$

Finally, we can write

$$\mathbf{y}_n | \mathbf{X}, \mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_3 \ \sim \mathcal{N}(\mathbf{W}_3^T \boldsymbol{\phi}_{2,n}, \tau^{-1}\mathbf{I}_D).$$

The application of variational inference continues as before.

Note that an alternative approach would be to use the same covariance function in each layer. For that we would need to set $\mathbf{W}_2$ to be of dimensions $K_1 \times K_1$ normally distributed. This results in a product of two normally distributed matrices: $\mathbf{W}_2$ and the weights resulting from the Monte Carlo integration of $\widehat{\mathbf{K}}_2$ (denoted $\mathbf{W}_2'$ for convenience). Even though the composition of two linear transformations is a linear transformation, the resulting prior distribution over the weight matrix $\mathbf{W}_2 \mathbf{W}_2'$ is quite complicated.

5 Insights and Applications

Our derivation suggests many applications and insights, including the representation of model uncertainty in deep learning, better model regularisation, computationally efficient Bayesian convolutional neural networks, use of dropout in recurrent neural networks, and the principled development of dropout variants, to name a few. These are *briefly* discussed here, and studied more in depth in separate work.

5.1 Insights

The Gaussian process's robustness to over-fitting can be contributed to several different aspects of the model and is discussed in detail in [Rasmussen and Williams, 2006]. Our interpretation offers an explanation to dropout's ability to avoid over-fitting. Dropout can be seen as approximately integrating over the weights of the network.

Our derivation also suggests that an approximating variational distribution should be placed over the bias $\mathbf{b}$. This could be sampled jointly with the weights $\mathbf{W}$. Note that it is possible to interpret dropout as doing so when used with non-linearities with $\sigma(0) = 0$. This is because the product by the vector of Bernoulli random variables can be passed through the non-linearity in this case. However the GP interpretation changes in this case, as the inputs are randomly set to zero rather than the weights. By sampling Bernoulli variables for the bias weights as well, the model might become more robust.

In [Srivastava et al., 2014] alternative distributions to the Bernoulli are discussed. For example, it is suggested that multiplying the weights by $\mathcal{N}(1, \boldsymbol{\sigma}^2)$ results in similar results to dropout (although this becomes a more costly operation at run time). This can be seen as an alternative approximating variational distribution where we set $q(\mathbf{w}_k) = \mathbf{m}_k + \mathbf{m}_k \boldsymbol{\sigma} \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

We noted in the text that the weight-decay for the dropped-out weights should be scaled by the probability of the weights to not be dropped. This follows from the KL approximation. This might explain why doubling the learning rate of the bias during NN optimisation works well in practice in dropout networks with $p = 0.5$.

We also note that the model brought in section 2.1 does not use a bias at the output layer. This is equivalent to shifting the data by a constant amount and thus not treated in our derivation. Alternatively, using a Gaussian process mean function given by $\mu(\mathbf{x}) = \mathbf{c}$ is equivalent to setting the bias of the output layer to $\mathbf{c}$.

5.1.1 Model Calibration

We can show that the dropout model is not calibrated. This is because Gaussian processes' uncertainty is not calibrated and the model draws its properties from these. The Gaussian process's uncertainty depends on the covariance function chosen, which we showed above to be equivalent to the non-linearities and prior over the weights. The choice of a GP's covariance function follows from our assumptions about the data. If we believe, for example, that the model's uncertainty should increase far from the data we might choose the squared exponential covariance function.

For many practical applications this means that model uncertainty can increase with data magnitude or be of different scale for different datasets. To calibrate model uncertainty in regression tasks we can scale the uncertainty linearly to remove data magnitude effects, and manipulate uncertainty percentiles to compare among different datasets. This can be done by fitting a simple distribution over the training set output uncertainty, and using the cumulative distribution function to find the relative ratio of a new data point's uncertainty. This quantity can be used to compare a data point's uncertainty obtained from a model trained on one data distribution to another.

For example, if a test point has standard deviation 5, whereas almost all other data points have standard deviation ranging from 0.2 to 2, then the data point will be in the top percentile, and the model will be considered as very uncertain about the point compared to the rest of the data. However, another model might give the same point standard deviation 5 with most of the data modelled with standard deviation ranging from 10 to 15. In this model the data point will be in the lowest percentile, and the model will be considered as fairly certain about the point with respect to the rest of the data.

5.1.2 Approximation Tightness

As the approximation above is a variational one, it will result in a good approximation as long as the class of approximate posteriors contains a model which is "close enough" (in KL terms) to the true posterior. The prior term's KL approximation (proposition 1) depends on the assumption that the number of hidden units is large – and will become tighter as the number of these increases. We could then invoke the limit argument [Williams, 1997] showing that in the limit of number of units the network would converge to a Gaussian process.

5.2 Applications

Our derivation suggests an estimate for dropout models by averaging T forward passes through the network (referred to as *MC dropout*, compared to *standard dropout* with weight averaging). This result has been presented in the literature before as model averaging [Srivastava et al., 2014]. Our interpretation suggests a new look as to why MC dropout is more sensible than the current approach of averaging the weights. Furthermore, with the obtained samples we can estimate the model's confidence in its predictions and take actions accordingly. For example, in the case of classification, the model might return a result with high uncertainty, in which case we might decide to pass the input to a human to classify. Alternatively, one can use a weak and fast model to perform classification, and use a more elaborate but slower model only on inputs for which the weak model in uncertain. Uncertainty is important in reinforcement learning (RL) as well [Szepesvári, 2010]. With uncertainty information an agent can decide when to exploit and when to explore its environment. Recent advances in RL have made use of NNs to estimate agents' Q-value functions, a function that estimates the quality of different states and actions in the environment [Mnih et al., 2013]. Epsilon greedy search is often used in this setting, where an agent selects its currently estimated best action with some probability, and explores otherwise. With uncertainty estimates over the agent's Q-value function, techniques such as Thompson sampling [Thompson, 1933] can be used to train the model faster. These ideas are studied in the main paper.

Following our interpretation, one should apply dropout before each weight layer and not only before inner-product layers at the end of the model. This is to avoid parameter over-fitting on all layers as the dropout model, in effect, integrates over the parameters. The use of dropout before a subset of the layers corresponds to interleaving MAP estimates and fully Bayesian estimates. The application of dropout before every weight layer is not used in practice however, as empirical results using *standard* dropout suggest inferior performance. The use of MC dropout, however, with dropout applied before every weight layer results in much better empirical performance on some NN structures.

One can also interpret the approximation above as approximate variational inference in Bayesian neural networks (NNs). Thus, dropout applied before every weight layer is equivalent to variational inference in Bayesian NNs. This allows us to develop new Bayesian NN architectures which are not directly related to the Gaussian process, using operations such as pooling and convolutions. This leads to good, efficient, and trivial approximations to Bayesian convolutional neural networks (convnets). We discuss these ideas with empirical evaluation in separate work.

Another possible application is the adaptation of dropout to recurrent neural networks (RNNs). Currently, dropout is not used with these models as the repeated application of noise over potentially thousands of repetitions results in a very weak signal at the output. GP dynamical models [Wang et al., 2005] and recursive GPs with perfect integrators correspond to the ideas behind RNNs and long-short-term-memory (LSTM) networks [Hochreiter and Schmidhuber, 1997]. The GP models integrate over the parameters and thus avoid over-fitting. Seen as a GP approximation one would expect there to exist a suitable dropout approximation for these tasks as well. We discuss these ideas in separate work.

Model ensembles are often used in deep learning as well, where the same model is trained several times and at test time the results of all models are averaged. This is computationally very expensive as either training time is increased considerably, or many computational resources are used at the same time. One would expect that stochastically simulating forward passes through a dropout network will result in similar performance.

In future research we aim to assess model uncertainty on adversarial inputs as well, such as corrupted images that classify incorrectly with high confidence [Szegedy et al., 2014]. Adding or subtracting a single pixel from each input dimension is perceived as almost unchanged input to a human eye, but can change classification probabilities considerably. In the high dimensional input space the new corrupted image lies far from the data, and one would expect model uncertainty to increase for such inputs.

Lastly, our interpretation allows the development of principled extensions of dropout. The use of non-diminishing $\boldsymbol{\sigma}^2$ (eqs. (9) to (11)) and the use of a mixture of Gaussians with more than two components is an immediate example of such. For example the use of a low rank covariance matrix would allow us to capture complex relations between the weights. These approximations could result in alternative uncertainty estimates to the ones obtained with MC dropout. This is subject to current research.

6 Conclusions

We have shown that a neural network with arbitrary depth and non-linearities and with dropout applied before every weight layer is mathematically equivalent to an approximation to the deep Gaussian process. This interpretation offers an explanation to some of dropout's key properties. Our analysis suggests straightforward generalisations of dropout for future research which should improve on current techniques.

Acknowledgments

The authors would like to thank Mr Shane Gu, Dr Ankur Handa, Mr Nilesh Tripuraneni, Prof Yoshua Bengio, Prof Phil Blunsom, and many other reviewers for helpful comments. Yarin Gal is supported by the Google European Fellowship in Machine Learning.

References

Baldi, P. and Sadowski, P. J. (2013). Understanding dropout. In *Advances in Neural Information Processing Systems*, pages 2814–2822.

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L. (2006). Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

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

Blei, D. M., Jordan, M. I., and Paisley, J. W. (2012). Variational Bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1367–1374.

Damianou, A. and Lawrence, N. (2013). Deep Gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pages 207–215.

Gal, Y., Chen, Y., and Ghahramani, Z. (2015). Latent Gaussian processes for distribution estimation of multivariate categorical data. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*.

Gal, Y. and Turner, R. (2015). Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*.

Gal, Y., van der Wilk, M., and Rasmussen, C. (2014). Distributed variational inference in sparse Gaussian process regression and latent variable models. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 3257–3265. Curran Associates, Inc.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In Nicholson, A. and Smyth, P., editors, *UAI*. AUAI Press.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. (2010). Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1278–1286.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. *ICLR*.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294.

Titsias, M. and Lawrence, N. (2010). Bayesian Gaussian process latent variable model. *Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 6:844–851.

Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1971–1979.

Tsuda, K., Kin, T., and Asai, K. (2002). Marginalized kernels for biological sequences. *Bioinformatics*, 18(suppl 1):S268–S275.

Wager, S., Wang, S., and Liang, P. S. (2013). Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pages 351–359.

Wang, J., Hertzmann, A., and Blei, D. M. (2005). Gaussian process dynamical models. In *Advances in neural information processing systems*, pages 1441–1448.

Williams, C. K. (1997). Computing with infinite networks. *Advances in neural information processing systems*, pages 295–301.

A KL of a Mixture of Gaussians

Proposition 1. *Let*

$$q(\mathbf{x}) = \sum_{i=1}^{L} p_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

be a mixture of Gaussians with L components and $\boldsymbol{\mu}_i \in \mathbb{R}^K$ normally distributed, and let $p(\mathbf{x}) = \mathcal{N}(0, \mathbf{I}_K)$.

The KL divergence between $q(\mathbf{x})$ and $p(\mathbf{x})$ can be approximated as:

$$KL(q(\mathbf{x})||p(\mathbf{x})) \approx \sum_{i=1}^{L} \frac{p_i}{2} \big(\boldsymbol{\mu}_i^T \boldsymbol{\mu}_i + tr(\boldsymbol{\Sigma}_i) - K - \log |\boldsymbol{\Sigma}_i| \big)$$

plus a constant for large enough K.

Proof. We have

$$\begin{aligned} \mathrm{KL}(q(\mathbf{x})||p(\mathbf{x})) &= \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} \mathrm{d}\mathbf{x} \\ &= \int q(\mathbf{x}) \log q(\mathbf{x}) \mathrm{d}\mathbf{x} - \int q(\mathbf{x}) \log p(\mathbf{x}) \mathrm{d}\mathbf{x} \\ &= -\mathcal{H}(q(\mathbf{x})) - \int q(\mathbf{x}) \log p(\mathbf{x}) \mathrm{d}\mathbf{x} \end{aligned}$$

where $\mathcal{H}(q(\mathbf{x}))$ is the entropy of $q(\mathbf{x})$. The second term in the last line can be evaluated analytically, but the entropy term has to be approximated.

We begin by approximating the entropy term. We write

$$\begin{aligned} \mathcal{H}(q(\mathbf{x})) &= -\sum_{i=1}^{L} p_i \int \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \log q(\mathbf{x}) \mathrm{d}\mathbf{x} \\ &= -\sum_{i=1}^{L} p_i \int \mathcal{N}(\boldsymbol{\epsilon}_i; 0, \mathbf{I}) \log q(\boldsymbol{\mu}_i + \mathbf{L}_i \boldsymbol{\epsilon}_i) \mathrm{d}\boldsymbol{\epsilon}_i \end{aligned}$$

with $\mathbf{L}_i \mathbf{L}_i^T = \boldsymbol{\Sigma}_i$.

Now, the term inside the logarithm can be written as

$$\begin{aligned} & q(\boldsymbol{\mu}_i + \mathbf{L}_i \boldsymbol{\epsilon}_i) \\ & = \sum_{j=1}^{L} p_i \mathcal{N}(\boldsymbol{\mu}_i + \mathbf{L}_i \boldsymbol{\epsilon}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ & = \sum_{j=1}^{L} p_i (2\pi)^{-K/2} |\boldsymbol{\Sigma}_j|^{-1/2} \exp \big\{ -\frac{1}{2} ||\boldsymbol{\mu}_j - \boldsymbol{\mu}_i - \mathbf{L}_i \boldsymbol{\epsilon}_i||^2_{\boldsymbol{\Sigma}_j} \big\}. \end{aligned}$$

where $|| \cdot ||_{\boldsymbol{\Sigma}}$ is the Mahalanobis distance. Since $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$ are assumed to be normally distributed, the quantity $\boldsymbol{\mu}_j - \boldsymbol{\mu}_i - \mathbf{L}_i \boldsymbol{\epsilon}_i$ is also normally distributed. Using the expectation of the generalised χ^2 distribution with K degrees of freedom, we have that for $K >> 0$ there exists that $||\boldsymbol{\mu}_j - \boldsymbol{\mu}_i - \mathbf{L}_i \boldsymbol{\epsilon}_i||^2_{\boldsymbol{\Sigma}_j} >> 0$ for $i \neq j$. Finally, we have for $i = j$ that $||\boldsymbol{\mu}_i - \boldsymbol{\mu}_i - \mathbf{L}_i \boldsymbol{\epsilon}_i||^2_{\boldsymbol{\Sigma}_i} = \boldsymbol{\epsilon}_i^T \mathbf{L}_i^T \mathbf{L}_i^{-T} \mathbf{L}_i^{-1} \mathbf{L}_i \boldsymbol{\epsilon}_i = \boldsymbol{\epsilon}_i^T \boldsymbol{\epsilon}_i$. Therefore the last equation can be approximated as

$$q(\boldsymbol{\mu}_i + \mathbf{L}_i \boldsymbol{\epsilon}_i) \approx p_i (2\pi)^{-K/2} |\boldsymbol{\Sigma}_i|^{-1/2} \exp \big\{ -\frac{1}{2} \boldsymbol{\epsilon}_i^T \boldsymbol{\epsilon}_i \big\}.$$

This gives us

$$\begin{aligned} & \mathcal{H}(q(\mathbf{x})) \\ & \approx -\sum_{i=1}^{L} p_i \int \mathcal{N}(\boldsymbol{\epsilon}_i; 0, \mathbf{I}) \log \Big(p_i (2\pi)^{-K/2} |\boldsymbol{\Sigma}_i|^{-1/2} \exp \big\{ -\frac{1}{2} \boldsymbol{\epsilon}_i^T \boldsymbol{\epsilon}_i \big\} \Big) \mathrm{d}\boldsymbol{\epsilon}_i \end{aligned}$$

$$= \sum_{i=1}^{L} \frac{p_i}{2} \Bigg(\log |\mathbf{\Sigma}_i| + \int \mathcal{N}(\boldsymbol{\epsilon}_i; 0, \mathbf{I}) \boldsymbol{\epsilon}_i^T \boldsymbol{\epsilon}_i \mathrm{d}\boldsymbol{\epsilon}_i \Bigg) + C$$

where $C = -\sum_{i=1}^{L} p_i \Bigg(\log p_i - \frac{K}{2} \log(2\pi) \Bigg)$. Since $\boldsymbol{\epsilon}_i^T \boldsymbol{\epsilon}_i$ distributes according to a χ^2 distribution, its expectation is K, and the entropy can be approximated as

$$\mathcal{H}(q(\mathbf{x})) \approx \sum_{i=1}^{L} \frac{p_i}{2} \big(\log |\mathbf{\Sigma}_i| + K \big) + C$$

Next, evaluating the first term of the KL divergence we get

$$\int q(\mathbf{x}) \log p(\mathbf{x}) \mathrm{d}\mathbf{x} = \sum_{i=1}^{L} p_i \int \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \mathbf{\Sigma}_i) \log p(\mathbf{x}) \mathrm{d}\mathbf{x}$$

for $p(\mathbf{x}) = \mathcal{N}(0, \mathbf{I}_K)$ it is easy to validate that this is equivalent to $-\frac{1}{2} \sum_{i=1}^{L} p_i \big(\boldsymbol{\mu}_i^T \boldsymbol{\mu}_i + \mathrm{tr}(\mathbf{\Sigma}_i) \big)$. Finally, we get

$$\mathrm{KL}(q(\mathbf{x}) || p(\mathbf{x})) \approx \sum_{i=1}^{L} \frac{p_i}{2} \big(\boldsymbol{\mu}_i^T \boldsymbol{\mu}_i + \mathrm{tr}(\mathbf{\Sigma}_i) - K - \log |\mathbf{\Sigma}_i| \big) - C.$$

□

B Log Evidence Lower Bound for Classification

Proposition 2. *Given*

$$\begin{aligned} p(\mathbf{c}|\mathbf{X}) &= \int p(\mathbf{c}|\mathbf{Y}) p(\mathbf{Y}|\mathbf{X}) d\mathbf{Y} \\ &= \int p(\mathbf{c}|\mathbf{Y}) \Bigg(\int p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \\ &\qquad \cdot p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) d\mathbf{W}_1 d\mathbf{W}_2 d\mathbf{b} \Bigg) d\mathbf{Y} \end{aligned}$$

where $\mathbf{c}$ *is an* N *dimensional vector of categorical values, one for each observation, we can write the log evidence lower bound as*

$$\begin{aligned} \mathcal{L}_{\text{GP-VI}} := \int & p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \\ & \cdot \log p(\mathbf{c}|\mathbf{Y}) d\mathbf{W}_1 d\mathbf{W}_2 d\mathbf{b} d\mathbf{Y} \\ & - \mathit{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) || p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})). \end{aligned}$$

Proof. We have

$$\begin{aligned} & \log p(\mathbf{c}|\mathbf{X}) \\ & = \log \int p(\mathbf{c}|\mathbf{Y}) p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \\ & \qquad \cdot p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \mathrm{d}\mathbf{W}_1 \mathrm{d}\mathbf{W}_2 \mathrm{d}\mathbf{b} \mathrm{d}\mathbf{Y} \\ & = \log \int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) p(\mathbf{c}|\mathbf{Y}) \\ & \qquad \cdot \frac{p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})}{q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})} \mathrm{d}\mathbf{W}_1 \mathrm{d}\mathbf{W}_2 \mathrm{d}\mathbf{b} \mathrm{d}\mathbf{Y} \\ & \geq \int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \log \Bigg(p(\mathbf{c}|\mathbf{Y}) \\ & \qquad \cdot \frac{p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})}{q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})} \Bigg) \mathrm{d}\mathbf{W}_1 \mathrm{d}\mathbf{W}_2 \mathrm{d}\mathbf{b} \mathrm{d}\mathbf{Y} \\ & = \int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \end{aligned}$$

$$
\begin{aligned}
&\cdot \log p(\mathbf{c}|\mathbf{Y})\mathrm{d}\mathbf{W}_1\mathrm{d}\mathbf{W}_2\mathrm{d}\mathbf{b}\mathrm{d}\mathbf{Y} \\
&- \mathrm{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})||p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})),
\end{aligned}
$$

as needed. □

C Predictive Mean

Proposition 3. *Given weights matrices $\mathbf{M}_i$ of dimensions $K_i \times K_{i-1}$, bias vectors $\mathbf{m}_i$ of dimensions K_i, and binary vectors $\mathbf{z}_i$ of dimensions K_{i-1} for each layer $i = 1, ..., L$, as well as the approximating variational distribution*

$$q(\mathbf{y}^*|\mathbf{x}^*) := \mathcal{N}\big(\mathbf{y}^*; \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{z}_1, ..., \mathbf{z}_L), \tau^{-1}\mathbf{I}_D\big)\mathit{Bern}(\mathbf{z}_1) \cdots \mathit{Bern}(\mathbf{z}_L)$$

for some $\tau > 0$, with

$$\widehat{\mathbf{y}}^* = \sqrt{\frac{1}{K_L}}(\mathbf{M}_L\mathbf{z}_L)\sigma\bigg(...\sqrt{\frac{1}{K_1}}(\mathbf{M}_2\mathbf{z}_2)\sigma\big((\mathbf{M}_1\mathbf{z}_1)\mathbf{x}^* + \mathbf{m}_1\big)...\bigg),$$

we have

$$\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*) \approx \frac{1}{T}\sum_{t=1}^{T} \widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{z}}_{1,t}, ..., \widehat{\mathbf{z}}_{L,t})$$

with

$$\widehat{\mathbf{z}}_{i,t} \sim \mathit{Bern}(p_i).$$

Proof.

$$
\begin{aligned}
\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*) &= \int \mathbf{y}^* q(\mathbf{y}^*|\mathbf{x}^*)\mathrm{d}\mathbf{y}^* \\
&= \int \mathbf{y}^* \mathcal{N}\big(\mathbf{y}^*; \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{z}_1, ..., \mathbf{z}_L), \tau^{-1}\mathbf{I}_D\big)\mathrm{Bern}(\mathbf{z}_1) \cdots \mathrm{Bern}(\mathbf{z}_L)\mathrm{d}\mathbf{z}_1 \cdots \mathrm{d}\mathbf{z}_L\mathrm{d}\mathbf{y}^* \\
&= \int \bigg(\int \mathbf{y}^* \mathcal{N}\big(\mathbf{y}^*; \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{z}_1, ..., \mathbf{z}_L), \tau^{-1}\mathbf{I}_D\big)\mathrm{d}\mathbf{y}^*\bigg)\mathrm{Bern}(\mathbf{z}_1) \cdots \mathrm{Bern}(\mathbf{z}_L)\mathrm{d}\mathbf{z}_1 \cdots \mathrm{d}\mathbf{z}_L\mathrm{d}\mathbf{y}^* \\
&= \int \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{z}_1, ..., \mathbf{z}_L)\mathrm{Bern}(\mathbf{z}_1) \cdots \mathrm{Bern}(\mathbf{z}_L)\mathrm{d}\mathbf{z}_1 \cdots \mathrm{d}\mathbf{z}_L \\
&\approx \frac{1}{T}\sum_{t=1}^{T} \widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{z}}_{1,t}, ..., \widehat{\mathbf{z}}_{L,t}).
\end{aligned}
$$

□

D Predictive Variance

Proposition 4. *Given weights matrices $\mathbf{M}_i$ of dimensions $K_i \times K_{i-1}$, bias vectors $\mathbf{m}_i$ of dimensions K_i, and binary vectors $\mathbf{z}_i$ of dimensions K_{i-1} for each layer $i = 1, ..., L$, as well as the approximating variational distribution*

$$
\begin{aligned}
q(\mathbf{y}^*|\mathbf{x}^*) &:= p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})q(\boldsymbol{\omega}) \\
q(\boldsymbol{\omega}) &= \mathit{Bern}(\mathbf{z}_1) \cdots \mathit{Bern}(\mathbf{z}_L) \\
p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) &= \mathcal{N}\big(\mathbf{y}^*; \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{z}_1, ..., \mathbf{z}_L), \tau^{-1}\mathbf{I}_D\big)
\end{aligned}
$$

for some $\tau > 0$, with

$$\widehat{\mathbf{y}}^* = \sqrt{\frac{1}{K_L}}(\mathbf{M}_L\mathbf{z}_L)\sigma\bigg(...\sqrt{\frac{1}{K_1}}(\mathbf{M}_2\mathbf{z}_2)\sigma\big((\mathbf{M}_1\mathbf{z}_1)\mathbf{x}^* + \mathbf{m}_1\big)...\bigg),$$

we have

$$\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}\big((\mathbf{y}^*)^T(\mathbf{y}^*)\big) \approx \tau^{-1}\mathbf{I}_D + \frac{1}{T}\sum_{t=1}^{T} \widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{z}}_{1,t}, ..., \widehat{\mathbf{z}}_{L,t})^T \widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{z}}_{1,t}, ..., \widehat{\mathbf{z}}_{L,t})$$

with

$$\widehat{\mathbf{z}}_{i,t} \sim \mathit{Bern}(p_i).$$

Proof.

$$
\begin{aligned}
&\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}\big((\mathbf{y}^*)^T(\mathbf{y}^*)\big) \\
&= \int \bigg(\int (\mathbf{y}^*)^T(\mathbf{y}^*) p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) \mathrm{d}\mathbf{y}^* \bigg) q(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \\
&= \int \bigg(\mathrm{Cov}_{p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})}(\mathbf{y}^*) + \mathbb{E}_{p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})}(\mathbf{y}^*)^T \mathbb{E}_{p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})}(\mathbf{y}^*) \bigg) q(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \\
&= \int \bigg(\tau^{-1}\mathbf{I}_D + \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{z}_1, ..., \mathbf{z}_L)^T \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{z}_1, ..., \mathbf{z}_L) \bigg) \mathrm{Bern}(\mathbf{z}_1) \cdots \mathrm{Bern}(\mathbf{z}_L) \mathrm{d}\mathbf{z}_1 \cdots \mathrm{d}\mathbf{z}_L \\
&\approx \tau^{-1}\mathbf{I}_D + \frac{1}{T} \sum_{t=1}^{T} \widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{z}}_{1,t}, ..., \widehat{\mathbf{z}}_{L,t})^T \widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{z}}_{1,t}, ..., \widehat{\mathbf{z}}_{L,t})
\end{aligned}
$$

since $p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) = \mathcal{N}\big(\mathbf{y}^*; \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{z}_1, ..., \mathbf{z}_L), \tau^{-1}\mathbf{I}_D\big)$. □

E Detailed Experiment Set-up

E.1 Model Uncertainty in Regression Tasks – Extrapolation

We ran a stochastic gradient descent optimiser for 1,000,000 iterations (until convergence) with learning rate policy base-lr $*$ $(1 + \gamma * \text{iter})^{-p}$ with $\gamma = 0.0001, p = 0.25$ and momentum 0.9. We initialise the bias at 0 and initialise the weights uniformly from $[-\sqrt{3/\text{fan-in}}, \sqrt{3/\text{fan-in}}]$. We use no mini-batch optimisation as the data is fairly small and with high frequencies. The learning rates used are 0.01 with weight decay of $1e^{-06}$ for CO_2 (corresponding to a high noise precision of $1e^5$). This is to model the low observation noise in the data due to the scaling and high frequencies.

E.2 Model Uncertainty in Reinforcement Learning

For the purpose of this experiment, we used future rewards discount of 0.7, no temporal window, and an experience replay of 30,000. The network starts learning after 1,000 steps, where in the initial 5,000 steps random actions are performed. The networks consist of two ReLU hidden layers of size 50, with a learning rate and weight decay of 0.001. Stochastic gradient descent was used with no momentum and batch size of 64.

The original implementation makes use of epsilon greedy exploration with epsilon changing as

$$\epsilon = \min\left(1, \max\left(\epsilon_{min}, 1 - \frac{\text{age} - \text{burn-in}}{\text{steps-total} - \text{burn-in}}\right)\right)$$

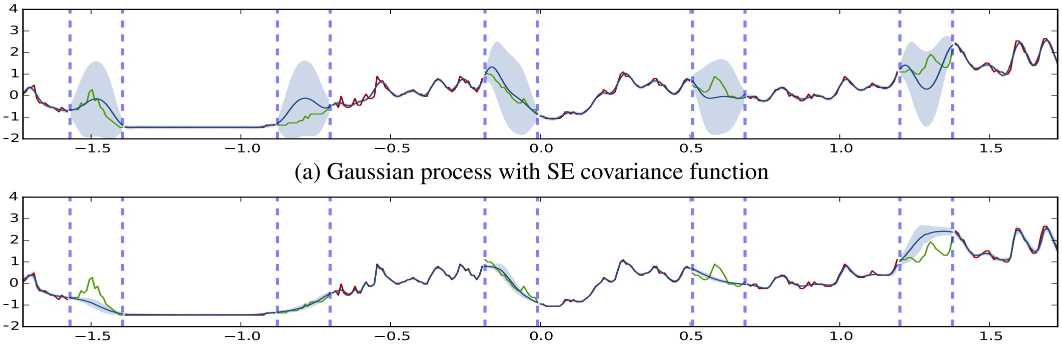
with steps-total of 200,000, burn-in of 3,000, and $\epsilon_{min} = 0.05$.

E.3 Model Uncertainty in Regression Tasks – Interpolation

For interpolation we repeat the experiment in the main paper with ReLU networks with 5 hidden layers and the same setup on a new dataset – solar irradiance. We use base learning rate of $5e^{-3}$ and weight decay of $5e^{-7}$.

Interpolation results are shown in fig. 1. Fig. 1a shows interpolation of missing sections (bounded between pairs of dashed blue lines) for the Gaussian process with squared exponential covariance function, as well the function value on the training set. In red is the observed function, in green are the missing sections, and in blue is the model predictive mean. Fig. 1b shows the same for the ReLU dropout model with 5 layers.

Both models interpolate the data well, with increased uncertainty over the missing segments. However the GP's uncertainty is larger and over-estimates its 95% confidence interval in capturing the true function. The observed uncertainty in MC dropout is similar to that of [Gal and Turner, 2015] with the model over-confident in its predictions. Note that this is often observed with variational techniques, where model uncertainty is under-estimated. This is because all variational inference techniques would under-estimate model uncertainty unless the true posterior is in the class of approximating variational distributions. Note also that the models correspond to different GP covariance functions that would result in different variance estimations.



(a) Gaussian process with SE covariance function

(b) MC dropout with ReLU non-linearities

Figure 1: **Predictive mean and uncertainties on the reconstructed solar irradiance dataset with missing segments, for the GP and MC dropout approximation.** In red is the observed function and in green are the missing segments. In blue is the predictive mean plus/minus two standard deviations of the various approximations.