Dropout as a Bayesian Approximation: Insights and Applications

Yarin Gal YG279@CAM.AC.UK
Zoubin Ghahramani ZG201@CAM.AC.UK
University of Cambridge

Abstract

Deep learning techniques are used more and more often, but they lack the ability to reason about uncertainty over the features. Features extracted from a dataset are given as point estimates, and do not capture how much the model is confident in its estimation. This is in contrast to probabilistic Bayesian models, which allow reasoning about model confidence, but often with the price of diminished performance.

We show that a multilayer perceptron (MLP) with arbitrary depth and non-linearities, with dropout applied after 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. Our analysis suggests straightforward generalisations of dropout for future research which should improve on current techniques.

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 allow us to capture how much the model is confident in its estimation. On the other hand, probabilistic Bayesian models such as the Gaussian process (Rasmussen & 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 & 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 multilayer perceptron (MLP) with arbitrary depth and non-linearities, with dropout applied after every weight layer, is mathematically equivalent to an approximation to the probabilistic deep Gaussian process model (Damianou & 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 MLPs 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.

This paper is short version of "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning" by Gal & Ghahramani (2015).

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 denote vectors, bold upper case letters denote matrices, and standard weight letters denote scalar quantities. We use subscripts to denote either entire rows / columns (with bold letters), or specific elements. 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 MLP model (Hinton et al., 2012; Srivastava et al., 2014) quickly for the case of a *single hidden layer* MLP. This is done for ease of notation, and 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 transforming 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 MLP 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 $\mathtt{b}_1, \mathtt{b}_2$ 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 $\mathtt{b}_{1,q} \sim \text{Bernoulli}(p_1)$ for $q = 1, ..., Q$, and $\mathtt{b}_{2,k} \sim \text{Bernoulli}(p_2)$ for $k = 1, ..., K$. Given an input $\mathbf{x}$, $1 - p_1$ proportion of the elements of the input are set to zero: $\mathbf{x} \circ \mathtt{b}_1$ where $\circ$ signifies the Hadamard product. The output of the first layer is given by $\sigma((\mathbf{x} \circ \mathtt{b}_1)\mathbf{W}_1 + \mathbf{b}) \circ \mathtt{b}_2$, which is linearly transformed to give the dropout model's output $\widehat{\mathbf{y}} = \big((\sigma((\mathbf{x} \circ \mathtt{b}_1)\mathbf{W}_1 + \mathbf{b})) \circ \mathtt{b}_2\big)\mathbf{W}_2$. This is equivalent to multiplying the weight matrices by the binary vectors to zero out entire rows:

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

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

To use the MLP model for regression we might use the euclidean loss,

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

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 as $1, ..., D$, we pass the output of the model $\widehat{\mathbf{y}}$ through an element-wise softmax function to obtain normalised scores: $\hat{p}_{nd} = \exp(\widehat{y}_{nd}) / \big(\sum_{d'} \exp(\widehat{y}_{nd'})\big)$. Taking the log of this function results in a *softmax* loss,

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

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

During optimisation, this term is scaled by the learning rate r_1 and a regularisation term is added. We often use L_2 regularisation weighted by some weight decay r_2 (alternatively, the derivatives might be scaled), resulting in a minimisation objective (often referred to as cost),

$$\mathcal{L}_{\text{dropout}} := r_1 E + r_2\big(||\mathbf{W}_1||_2^2 + ||\mathbf{W}_2||_2^2 + ||\mathbf{b}||_2^2\big). \qquad (3)$$

We sample new realisations for the binary vectors $\mathtt{b}_i$ 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 $\mathtt{b}_1\mathbf{W}_1$ and $\mathtt{b}_2\mathbf{W}_2$ 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. Note that the probabilities p_i can be optimised.

We will show that equations (1) to (3) arise in Gaussian process approximation as well. Next 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 & Williams, 2006; Titsias & Lawrence, 2010; Gal

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

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

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 $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and their corresponding outputs $\{\mathbf{y}_1, \dots, \mathbf{y}_N\}$, 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 $\tau\mathbf{Y}$ through an element-wise softmax,

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

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 $\mathbf{K}(\mathbf{X}, \mathbf{Y})$ 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) MLPs.

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:

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

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),

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

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 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 & Turner, 2015), which results in a model mathematically identical to the use of dropout in arbitrarily structured MLPs with arbitrary non-linearities.

3. Dropout as a Bayesian Approximation

We show that MLPs with dropout applied after every weight layer are mathematically equivalent to approximate variational inference in the deep Gaussian process. For this we build on previous work (Gal & 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 MLP 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 σ 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 MLP 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 $\mathbf{w}_d \sim \mathcal{N}(0, \mathbf{I}_K)$,

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

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 introduced 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] This is equivalent to the weighted basis function interpretation of the Gaussian process (Rasmussen & Williams, 2006).

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 $q(\mathbf{W}_1)$ 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 $\mathbf{M}_1 = [\mathbf{m}_q]_{q=1}^{Q}$, $\mathbf{M}_2 = [\mathbf{m}_k]_{k=1}^{K}$, and $\mathbf{m}$, to maximise Eq. (7). The task of classification and an extension to multiple layers is discussed in (Gal & Ghahramani, 2015).

3.3. Evaluating the Log Evidence Lower Bound for Regression

We need to evaluate the log evidence lower bound:

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

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

We re-parametrise the integrand 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 $\boldsymbol{\epsilon}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{Q \times K})$ and $\mathtt{b}_{1,q} \sim \text{Bernoulli}(p_1)$ for $q = 1, ..., Q$, and $\boldsymbol{\epsilon}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{K \times D})$ and $\mathtt{b}_{2,k} \sim \text{Bernoulli}(p_2)$ for $k = 1, ..., K$. Finally let $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_K)$. We write

$$\begin{aligned}\mathbf{W}_1 &= \mathtt{b}_1(\mathbf{M}_1 + \boldsymbol{\sigma}\boldsymbol{\epsilon}_1) + (1 - \mathtt{b}_1)\boldsymbol{\sigma}\boldsymbol{\epsilon}_1 \\ \mathbf{W}_2 &= \mathtt{b}_2(\mathbf{M}_2 + \boldsymbol{\sigma}\boldsymbol{\epsilon}_2) + (1 - \mathtt{b}_2)\boldsymbol{\sigma}\boldsymbol{\epsilon}_2 \\ \mathbf{b} &= \mathbf{m} + \boldsymbol{\sigma}\boldsymbol{\epsilon},\end{aligned} \tag{13}$$

allowing us to re-write the integral in the above equation as

$$\begin{aligned}&\int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \mathrm{d}\mathbf{W}_1 \mathrm{d}\mathbf{W}_2 \mathrm{d}\mathbf{b} \\ &\quad = \int q(\mathtt{b}_1, \boldsymbol{\epsilon}_1, \mathtt{b}_2, \boldsymbol{\epsilon}_2, \boldsymbol{\epsilon}) \\ &\qquad \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1(\mathtt{b}_1, \boldsymbol{\epsilon}_1), \mathbf{W}_2(\mathtt{b}_2, \boldsymbol{\epsilon}_2), \mathbf{b}(\boldsymbol{\epsilon})) \\ &\qquad \mathrm{d}\boldsymbol{\epsilon}_1 \mathrm{d}\mathtt{b}1 \mathrm{d}\boldsymbol{\epsilon}_2 \mathrm{d}\mathtt{b}_2 \mathrm{d}\boldsymbol{\epsilon}.\end{aligned}$$

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

$$\begin{aligned}\mathcal{L}_{\text{GP-MC}} := & \log p(\mathbf{Y}|\mathbf{X}, \widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \widehat{\mathbf{b}}) \\ & - \text{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) || p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}))\end{aligned}$$

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

For the task of regression we have

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

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

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

[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.

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

$$\begin{aligned}\mathrm{KL}(q(\mathbf{W}_1)||p(\mathbf{W}_1)) &\approx QK(\boldsymbol{\sigma}^2 - \log(\boldsymbol{\sigma}^2) - 1) \\ &\qquad + \frac{p_1}{2}\sum_{q=1}^{Q} \mathbf{m}_q^T \mathbf{m}_q.\end{aligned}$$

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

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

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

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

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 Gaussian distribution with such a small standard deviation will be represented on a computer, in effect, as zero. Thus the random variable realisations $\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \widehat{\mathbf{b}}$ can be approximated as

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

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

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

Scaling the optimisation objective by a positive constant γ doesn't change the parameter values at its optimum. We thus scale the objective to get

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

$$- \frac{\gamma p_1}{2}||\mathbf{M}_1||_2^2 - \frac{\gamma p_2}{2}||\mathbf{M}_2||_2^2 - \frac{\gamma}{2}||\mathbf{m}||_2^2 \tag{15}$$

and we recovered equation (1) for an appropriate setting of γ and model precision τ. Maximising eq. (14) results in the same optimal parameters as minimising eq. (3). Note that eq. (14) 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 $\widehat{\mathtt{b}}_1, \widehat{\mathtt{b}}_2$ 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, as is commonly known, the ratio between the constant scaling the likelihood term in the dorpout objective (the first term, usually referred to as the learning rate) and that of the regularisation terms (the rest of the terms, usually referred to as the weight-decays) gives us the model precision: $\frac{r_1}{r_2} = \frac{\gamma\tau/2}{\gamma/2} = \tau$. 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 MLP 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 MLP. 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 overfitting. Empirically it seems that the Bernoulli approximating distribution is sufficient to considerably prevent overfitting.

We have presented the derivation for a single hidden layer MLP. An extension of the derivation to multiple layers is given below.

4. 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.

4.1. 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 & Williams, 2006). Our interpretation offers an explanation to dropout's ability to avoid over-fitting.

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 $q(\mathbf{w}_k) = \mathbf{m}_k + \mathbf{m}_k \boldsymbol{\epsilon}$ 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 MLP 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.

4.2. Applications

Our derivation suggests an estimate for dropout models using T forward passes through the network and averaging the results (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) (Szepesvári, 2010) as well. With this information an agent can decide when to exploit and when to explore its environment. Recent advances in RL have made use of MLPs 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 the 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 separate work.

Following our interpretation, one should apply dropout after each weight layer and not only after 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 after a subset of the layers corresponds to interleaving MAP estimates and fully Bayesian estimates. The application of dropout after 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 after every weight layer results in much better empirical performance on some MLP structures. One can also interpret the approximation above as approximate variational inference in Bayesian neural networks (NNs). Thus, dropout applied after 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 a 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 layers 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 & 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.

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.

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.

5. Conclusions

We have shown that a multilayer perceptron with arbitrary depth and non-linearities and with dropout applied after 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, Mr Nilesh Tripuraneni, Prof Yoshua Bengio, and Prof Phil Blunsom for helpful comments. Yarin Gal is supported by the Google European Fellowship in Machine Learning.

References

Baldi, Pierre and Sadowski, Peter J. Understanding dropout. In *Advances in Neural Information Processing Systems*, pp. 2814–2822, 2013.

Bengio, Yoshua, Schwenk, Holger, Senécal, Jean-Sébastien, Morin, Fréderic, and Gauvain, Jean-Luc. Neural probabilistic language models. In *Innovations in Machine Learning*, pp. 137–186. Springer, 2006.

Bishop, Christopher M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

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

Damianou, Andreas and Lawrence, Neil. Deep Gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pp. 207–215, 2013.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*, 2015.

Gal, Yarin and Turner, Richard. 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)*, 2015.

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

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

Hensman, James, Fusi, Nicolo, and Lawrence, Neil D. Gaussian processes for big data. In Nicholson, Ann and Smyth, Padhraic (eds.), *UAI*. AUAI Press, 2013.

Hinton, Geoffrey E, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

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

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

Kingma, Diederik P and Welling, Max. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lázaro-Gredilla, Miguel, Quiñonero-Candela, Joaquin, Rasmussen, Carl Edward, and Figueiras-Vidal, Aníbal R. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Rasmussen, Carl Edward and Williams, Christopher K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006. ISBN 026218253X.

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

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

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

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

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

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

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

Wager, Stefan, Wang, Sida, and Liang, Percy S. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pp. 351–359, 2013.

Wang, Jack, Hertzmann, Aaron, and Blei, David M. Gaussian process dynamical models. In *Advances in neural information processing systems*, pp. 1441–1448, 2005.

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:*

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

for large enough K.

Proof. We have

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

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

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

for some $T > 0$ with $\mathbf{L}_i \mathbf{L}_i^T = \boldsymbol{\Sigma}_i$ and $\boldsymbol{\epsilon}_{it} \sim \mathcal{N}(0, \mathbf{I})$.

Now, the term inside the logarithm can be written as

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

where $|| \cdot ||_{\boldsymbol{\Sigma}}$ is the Mahalanobis distance. Since $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$ are assumed to be normally distributed, the quantity $\boldsymbol{\mu}_j - \boldsymbol{\mu}_i -$

$\mathbf{L}_i \boldsymbol{\epsilon}_{it}$ 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 $||\boldsymbol{\mu}_j - \boldsymbol{\mu}_i - \mathbf{L}_i \boldsymbol{\epsilon}_{it}||^2_{\boldsymbol{\Sigma}_j} >> 0$ for $i \neq j$. Finally, we have for $i = j$ that $||\boldsymbol{\mu}_i - \boldsymbol{\mu}_i - \mathbf{L}_i \boldsymbol{\epsilon}_{it}||^2_{\boldsymbol{\Sigma}_i} = \boldsymbol{\epsilon}_{it}^T \mathbf{L}_i^T \mathbf{L}_i^{-T} \mathbf{L}_i^{-1} \mathbf{L}_i \boldsymbol{\epsilon}_{it} = \boldsymbol{\epsilon}_{it}^T \boldsymbol{\epsilon}_{it}$. Therefore the last equation can be written as

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

This gives us

$$\begin{aligned} &\mathcal{H}(q(\mathbf{x})) \\ &\approx -\sum_{i=1}^L p_i \log \left(\frac{1}{T} \sum_{t=1}^T p_i (2\pi)^{-K/2} |\boldsymbol{\Sigma}_i|^{-1/2} \exp\big\{ -\frac{1}{2} \boldsymbol{\epsilon}_{it}^T \boldsymbol{\epsilon}_{it} \big\} \right) \\ &= \sum_{i=1}^L \frac{p_i}{2} \big(\log |\boldsymbol{\Sigma}_i| + \frac{1}{T} \sum_{t=1}^T \boldsymbol{\epsilon}_{it}^T \boldsymbol{\epsilon}_{it} \big) + C \end{aligned}$$

where $C = -\sum_{i=1}^L p_i \Bigg(\log p_i - \frac{K}{2} \log(2\pi) \Bigg)$. Since $\boldsymbol{\epsilon}_{it}^T \boldsymbol{\epsilon}_{it}$ distributes according to a χ^2 distribution, it's expectation is K, and the last term can be approximated as

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

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

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

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

Finally, we get

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

□