CHALMERS
UNIVERSITY OF TECHNOLOGY

Probabilistic deep learning with variational inference

Uncertainty quantification using variational inference for deep neural networks modelling oil and gas production

Master's thesis in Engineering Mathematics and Computational Science

EVA HEGNAR

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Probabilistic deep learning with variational inference

Uncertainty quantification using variational inference for deep neural networks modelling oil and gas production

EVA HEGNAR

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Mathematical Statistics
Chalmers University of Technology
Gothenburg, Sweden 2020

Probabilistic deep learning with variational inference
Uncertainty quantification using variational inference for
deep neural networks modelling oil and gas production
EVA HEGNAR

Supervisor: Moritz Schauer, Department of Mathematical Sciences
Co-supervisor: Bjarne Grimstad, Solution Seeker
Examiner: Umberto Picchini, Department of Mathematical Sciences

Master's Thesis 2020
Department of Mathematical Sciences
Division of Mathematical Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers
Gothenburg, Sweden 2020

Probabilistic deep learning with variational inference
Uncertainty quantification using variational inference for
deep neural networks modelling oil and gas production
EVA HEGNAR
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Deep neural networks are used in the petroleum industry to model gas and oil rate. To optimise the production, the uncertainty of the network predictions is desirable. The neural network weights are equipped with prior distributions to be able to quantify the uncertainty of the model predictions within the Bayesian paradigm. To obtain a numerically feasible procedure two different approaches of variational inference are used and compared; black box variational inference and variational inference using the reparameterisation trick. Both approaches are applied to real measurements of gas and oil rate, which were given by Solution Seeker, a company providing production optimisation to the petroleum industry. The results show a more stable convergence using the reparameterisation trick. The uncertainty in predictions is possible to be quantified using variational inference but setting a proper prior distribution is difficult.

Keywords: deep neural network, Bayesian inference, variational inference, black box variational inference, reparameterisation trick, probabilistic modelling, production optimisation, flow rate estimation

Acknowledgements

First, I would like to thank my supervisor at Chalmers Moritz Schauer for all the support on my project. He has put a lot of time in educating me on probability theory, how to write mathematics correctly and proofreading the report. Our meetings have been enlightening and helped me a lot during this project.

I would like to thank Bjarne Grimstad at Solution Seeker for finding this project for me and supporting me. His way of explaining complicated methods in an understandable way has been a major help throughout. I would like to thank him and the rest of the people at Solution Seeker for making me feel like home whenever I was working at their office.

Lastly, I would like to thank Adam Eriksson for proofreading the report, even with the lack of mathematical background. He has been giving me valuable tips for improving the flow of the writing and fix grammatical errors.

Eva Hegnar, Gothenburg, August 2020

Contents

List of Figures

List of Tables

1

Introduction

There has been a tremendous development in the machine learning field in recent years. In particular, research on artificial neural networks has advanced the understanding and practice of predictive modelling. The wide range of applications of these models has driven academia and industry to develop software for predictive modelling of big datasets. This development coincides with increased availability of computing power and sensor data in the petroleum industry. The allure of predictive modelling lies in that it may simplify and speed up the modelling efforts for complex systems compared to traditional mechanistic modelling.

The petroleum industry is interested in researching the applicability of data-driven predictive modelling for decision support and optimisation of oil and gas production systems. Machine learning techniques, especially deep learning, excel at finding good predictors. In practice what is often needed is a good prediction together with reliable uncertainty quantification. Solution Seeker develops artificial intelligence for optimisation of the upstream production of petroleum and is interested in obtaining uncertainty measurements for its models. A point estimation of the production can be misleading when the uncertainty is not included since a high uncertainty for the estimate could be crucial in the decision making.

A Bayesian approach can handle both systematic and random uncertainty in a principled fashion. Both types of uncertainty are present in the problem of modelling oil and gas production. Most of the sources of random uncertainty in the input can be quantified. A challenging aspect of the modelling problem is that the flow rate measurements are uncertain and difficult to quantify because the systematic error is assumed to be heteroscedastic.

Within the Bayesian approach the uncertainty of inference and prediction is quantified through the posterior distribution. Uncertainty quantification comes at the price of additional requirements in computing power and requires more modelling because priors need to be specified. Variational inference addresses this problem by finding an approximation of the intractable posterior distribution within a rich but tractable class of distributions by minimising a suitable distance to the actual posterior [1].

Progress in prediction uncertainty is made by embedding deep neural networks in a Bayesian framework. As neural networks form a very rich class of non-linear maps, obtaining the exact posterior distribution will in general be intractable. Therefore

it is natural to consider variational inference [2] as a possible remedy of Bayesian inference.

This project will investigate the use of uncertainty quantification in probabilistic models using variational inference in deep learning approaches for modelling oil and gas production systems. Two methods for implementing variational inference will be presented and compared: the black box variational inference method [3] and variational inference using the reparameterisation trick [4].

2

Theory

2.1 Probabilistic background

One aspect of probability theory is to describe uncertainty. Probabilistic models are an important tool in scientific problem solving and are useful when drawing conclusions from data and making predictions. A probabilistic model incorporates uncertainty in a model, and the model assigns probability distributions to the output instead of a single number. In this way, an understanding of the uncertainty in the model can be achieved. A probabilistic model is therefore a model where *random (stochastic) variables* and *probability distributions* are integrated in the model. The random variables represent the potential outcome of an uncertain event. The probability distribution is used to assign probabilities to the potential outcomes. By recognising that there is uncertainty in the input and model, a more realistic prediction can be achieved.

Bayesian inference can be used for probabilistic modelling and is based on probability theory. Variational inference is an approximation of Bayesian inference. Before deriving methods for training a model using variational inference, some probabilistic background is needed. Relevant fundamentals of probability theory are provided below and will be expanded later on. The theory section is based on Bishop's *Pattern recognition and machine learning* [5].

Consider a random variable X taking values in a measurable space $E = (\mathbb{R}^n, \mathcal{B})$ where E is a Euclidean space over n-dimensional real number and the σ-algebra $\mathcal{B}$ is the set of events.

In probability theory, the probability *distribution* is the probability of the occurrence of different outcomes of an experiment. The probability distribution P_X measures the probability of a random variable X belonging to some set A, $P(X \in A) = P_X(A)$. If X has a *density* function f with respect to a reference measure ν, then the distribution is

$$P_X(A) = \int_A f(x)\, d\nu(x). \tag{2.1}$$

P_X is a probability measure defined over $E = (\mathbb{R}^n, \mathcal{B})$.

The *expectation* of a random variable X with probability density function f is

$$\mathbf{E}[X] = \int x f(x)\, d\nu(x). \tag{2.2}$$

Capital letters are used for random variables. This will be used throughout, also when Greek letters are used as a notation for random variables.

Most of the time $d\nu(x)$ is the Lebesgue measure, but sometimes it is useful to consider for example the distribution of a random variable Y as reference measure, e.g. $\nu = P_Y$.

Let a measurable space $\mathcal{M} = (X, \Sigma)$ and let μ and ν be σ-finite on $\mathcal{M}$. The *Radon–Nikodym theorem* states that if $\mu \gg \nu$, then there is a measurable function $f : X \to [0, \infty)$, such that for any measurable set $A \subseteq X$

$$\nu(A) = \int_A f \, d\mu. \tag{2.3}$$

The function f is called the *Radon–Nikodym derivative* and is denoted by $\frac{d\nu}{d\mu}$.

To *change the reference measure* consider random variables $X \sim P_X$ and $Y \sim P_Y$ where P_X and P_Y are defined on the space $E = (\mathbb{R}^n, \mathcal{B})$, and their probability distributions (2.1) are given by the respective densities p and q

$$P_X(A) = \int_A p(x) \, d\nu(x) \quad \text{and} \quad P_Y(A) = \int_A q(x) \, d\nu(x). \tag{2.4}$$

This can be written in short

$$\frac{dP_X}{dP_Y} = \frac{dP_X/d\nu}{dP_Y/d\nu}. \tag{2.5}$$

Changing the reference measure for the probability of $X \in A$ implies

$$P_X(A) = \int_A p(x) \, d\nu(x) = \int_A p(x) \frac{q(x)}{q(x)} \, d\nu(x) = \int_A \frac{p(x)}{q(x)} \, dP_Y(x), \tag{2.6}$$

where $q(x) \, d\nu(x) = dP_Y(A)$ from equation (2.4) is used. Taking the expectation (2.2) of a probability density f and using the relation from the equation above implies

$$\mathbf{E}[f(X)] = \int_A f(x)p(x) \, d\nu(x) = \int_A f(x) \frac{p(x)}{q(x)} \, dP_Y(x) = \mathbf{E}\left[f(Y) \frac{p(Y)}{q(Y)} \right]. \tag{2.7}$$

In equation (2.7), the expectation is expressed as an integral over the measure P_Y, therefore as an expectation with respect to random variable Y.

The *joint probability* is the probability of two or more events happening together. It is the probability of an event when this event is the intersection of several events from sub-experiments. For random variables, X taking values in E_1, Y taking values in E_2, defined on a probability space, the joint probability distribution $P_{X,Y}$ is the distribution of the pair (X, Y) (seen as a random variable). The joint probability density function $f_{X,Y}$ is the density of that random variable with respect to a suitable reference measure μ on $E_1 \times E_2$. For example, if X and Y are independent, then $f_{X,Y}(x, y) = f_X(x) f_Y(y)$ is the joint density with respect to the product measure $\nu = dxdy$.

The *marginal probability* is the probability of an event taking place without the knowledge of the probability of the other events. Considering two continuous random

variables X and Y, whose joint distribution $f_{X,Y}(x, y)$ is known. The marginal probability density function of the random variable X is the integral over Y of the joint distribution

$$f_X(x) = \int_y f_{X,Y}(x, y)\, dy. \tag{2.8}$$

Conditional probability is the probability of an event given another event. For continuous random variables X and Y the conditional probability density function of Y given the occurrence of x from X is

$$f_{Y|X}(y \mid x) = \frac{f_{X,Y}(x, y)}{f_X(x)}, \tag{2.9}$$

where $f_{X,Y}(x, y)$ is the joint density of X and Y, and $f_X(x)$ is the marginal density for X.

The marginal (2.8) can be rewritten using the conditional distribution (2.9)

$$f_X(x) = \int_y f_{X|Y}(x \mid y) f_Y(y)\, dy, \tag{2.10}$$

where $f_{X|Y}(x \mid y)$ is the conditional density of X given $Y = y$ and $f_Y(y)$ is the marginal distribution. The marginal distribution can be interpreted as the conditional probability of X, given a value of Y and averaged over the distribution of all values of Y.

Bayes' rule is the probability of an event based on prior knowledge and observations of that event. For events A and B, Bayes' rule is

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}, \tag{2.11}$$

where $P(B) \neq 0$. Here $P(A \mid B)$ is the *posterior* probability; given the observed event B how probable is the event A? $P(B \mid A)$ is the *likelihood*; given that event A happens, how probable is the event B? $P(A)$ is the *prior* probability; how probable was the event A before observing event B? Lastly, $P(B)$ is the marginal probability or *evidence*; how probable is the observed event B under all possible events? The power in Bayes' rule is that it links the posterior probability that can often not directly be computed to something that can be calculated, the likelihood.

When working with probability distributions, there are several known distributions to explain how the values of a variable are distributed. The most common distribution is the *normal distribution* as it fits many natural phenomena. The normal distribution is symmetric and bell shaped around the mean showing that data close to the mean is more frequent than data far from it. The further away a value is from the mean, the less likely it is to occur. For a random variable X the notation of normal distribution is

$$X \sim \mathcal{N}(\mu, \sigma^2), \tag{2.12}$$

where $\mu \in \mathbb{R}$ is the mean and with σ^2 being the variance. The probability density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{2.13}$$

where σ is the standard deviation. When $\mu = 0$ and $\sigma = 1$ it is the simplest form of normal distribution called the *standard normal distribution* and is notated as

$$X \sim \mathcal{N}(0, 1). \tag{2.14}$$

2.2 Bayesian inference

Bayesian inference is an approach to statistical inference where uncertainty is quantified using probability distributions [5]. These probability distributions capture *prior* information on the model (or the parameters) before a statistical experiment is conducted. This makes it possible to solve the statistical problem of inferring parameters from data using the laws of probability. In particular Bayes' rule can be used to infer the model parameters by updating the prior probability for the parameters with the data obtained during the statistical experiment.

Deep learning [5] is algorithms trying to model abstractions in data using complex structures. Deep learning algorithms are often thought of as "black boxes"; it is not as crucial to know how they work as long as they perform and produce a satisfying prediction. A drawback of black box machine learning methods is that they do not provide uncertainty of the predictions. A solution to this is using the statistical method of Bayesian inference. In Bayesian inference, prior knowledge is included in the model and updated using data. For deep learning models, inference involves propagating parameter uncertainty through the network layers to produce a distribution for the model output. The distribution allows for a prediction of the uncertainty for the model and can provide a posterior predictive distribution. The posterior predictive is the distribution over the possible unobserved values based on the data already observed and can be used to predict new datapoints. Implementing Bayesian inference has traditionally required specialised experience but modern probabilistic programming such as Pytorch makes it achievable.

Bayesian inference establishes a model and parameters where the model formulates observed events, and the parameters are factors in the model that affects the observed data. The model needs a likelihood function and a prior distribution. Assume a model generating data x from a distribution with unknown parameter θ (the uncertainty in input data). There is prior knowledge about the parameter θ in the parameter space Θ that can be expressed through a prior distribution Π with density $\pi(\theta)$. The prior can be updated using the likelihood $p(x \mid \theta)$ and the evidence $p(x)$ of the model. Using the formula from Bayes' rule (2.11), the posterior distribution $\Pi(\cdot \mid x)$ on Θ is given with density

$$\pi(\theta \mid x) = \frac{p(x \mid \theta)\pi(\theta)}{p(x)} = \frac{p(x \mid \theta)\pi(\theta)}{\int_\theta p(x \mid \theta)\pi(\theta)\, d\nu(\theta)}. \tag{2.15}$$

Here, the evidence is rewritten using marginal density (2.10) and ν is the reference measure. In short, Bayes' rule computes the posterior from the likelihood, prior and evidence, see figure 2.1. The likelihood and prior can often easily be found as they are part of the assumed model. The evidence $p(x)$ works as a normalising

factor but is difficult to compute. In lower dimensions, it is possible to compute the evidence by numerical integration, but in higher dimensions this becomes intractable. Combinatorial explosion can be a problem if the variables are discrete. Therefore approximation methods are needed, and variational inference is such an approximation method.

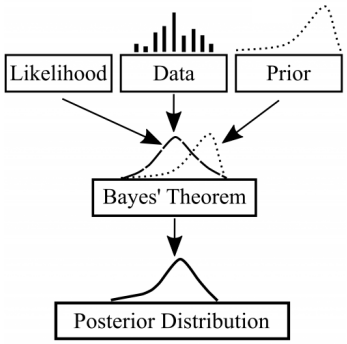

Figure 2.1: Flow chart of Bayesian inference [6].

2.2.1 Bayesian regression

To discuss Bayesian regression regular *regression* needs to be defined. Regression in modelling is a statistical process for describing the relationship between a dependent output variable y and one or more independent input variables x. This relationship is described using a function $f(x, \theta)$ where θ is the unknown parameters

$$y = f(x, \theta) + \epsilon. \tag{2.16}$$

The error term ϵ describes an unobserved noise. It is often described as independent and identically distributed normal random variables with zero mean and constant variance σ^2

$$\epsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2). \tag{2.17}$$

The goal of regression is to find the function $f(x, \theta)$ best fitted to describe the data.

Bayesian regression is an approach to regression using the assumptions from Bayesian inference for the statistics in the model. In Bayesian regression, probability distributions are used instead of point estimations. Instead of estimating y as a single value it is from a probability distribution. A posterior distribution is achieved for the model parameters if the error is normally distributed, and a prior distribution can be stated. When a normal distribution is assumed the Bayesian regression model is

$$y \sim \mathcal{N}(f(x, \theta), \sigma^2), \tag{2.18}$$

where y is the output variable, x is the predictive variables, θ is the weight parameter and σ is the standard deviation. Now the output is generated from a normal distribution determined by the mean and variance. The posterior distribution $\Pi(\cdot \mid y, x)$

on the parameter space Θ for the model is described using Bayes' rule for Bayesian regression (2.15) with density

$$\pi(\theta \mid y, x) = \frac{p(y \mid x, \theta)\pi(\theta)}{p(y \mid x)}. \tag{2.19}$$

Here $p(y \mid x, \theta)$ is the likelihood for the data, $\pi(\theta)$ is the prior density for the parameters and $p(y \mid x)$ is the evidence. If there is domain knowledge about the parameters in the distribution, this can be included through the prior. The posterior provides uncertainty about the model. For example, if there are few data point, this will result in a larger spread in the distribution [5].

2.3 Kullback–Leibler divergence

Variational inference is a method based on approximation of Bayesian inference. The distance of the approximation to the posterior is measured by the *Kullback-Leibler (KL) divergence*. An understanding of the Kullback–Leibler divergence is necessary to derive variational inference, but some essential information theory is necessary before obtaining the KL divergence. Information theory is concerned with quantifying the "information content" of a piece of information.

Consider a discrete space where an outcome $x \in E$ has been observed. The event $\{x\}$ has probability $P(\{x\}) = p(x)$, where p is the probability mass function. A way to define the *information* associated with observing x is setting

$$I(x) = -\log p(x). \tag{2.20}$$

In the same fashion, the information of an event $I(A)$, $A \subset E$ is defined. In this instance log is the natural logarithm but binary logarithms are also considered in information theory. Note that the higher the probability, the lower the amount of information. If an event A has a high probability it is expected to happen and A happening indeed provides little new information.

The *entropy* $H(P)$ is the expected information. In the discrete case, the expected information is a sum of the information contents of different outcomes weighted with their probabilities

$$H(P) = -\sum_{x \in E} p(x) \log p(x). \tag{2.21}$$

To better understand why this notion quantifies information the three fundamental properties of the information function I can be observed:

1. $I(A) \geq 0$: information is a non-negative quantity.
2. $I(E) = 0$: events that always occur do not provide information.
3. $I(A_1 \cup A_2) = I(A_1) + I(A_2)$: information due to independent events A_1 and A_2 is additive.

The expected information associated with observation of independent random variables is additive and therefore independent random variables provide new information. On the contrary, perfectly correlated random variables $\rho \in \{1, -1\}$ provide no information already known.

Using the definition of a distribution (2.1) the entropy can also be defined for continuous random variables (or their distribution P) with a density $p(x)$ with respect to a reference measure ν as

$$H(P) = -\int p(x) \log p(x)\, d\nu(x). \tag{2.22}$$

The KL divergence is a way of measuring the distance between probability distributions to see how similar they are. This method uses information theory as the measure for information provided from the distributions.

For two probability measures P and Q over the same space, if $Q \gg P$, the Radon – Nikodym derivative (2.3) exist $\frac{dP}{dQ}$, then a general notation for the *Kullback-Leibler divergence* is

$$\mathrm{KL}(P \parallel Q) = \int_A \log \frac{dP}{dQ}\, dP. \tag{2.23}$$

If P and Q have densities p and q with respect to the same reference measure ν, the Kullback-Leibler divergence from Q to P is defined as

$$\mathrm{KL}(P \parallel Q) = \int_A p(x) \log \frac{p(x)}{q(x)}\, d\nu(x). \tag{2.24}$$

If the KL integral diverges, $\mathrm{KL}(P \parallel Q) = \infty$. With a random variable $X \sim P$ the Kullback-Leibler divergence can be expressed as expectation using (2.2)

$$\mathrm{KL}(P \parallel Q) = \mathbf{E}\left[\log \frac{p(X)}{q(X)}\right] \tag{2.25}$$

or

$$\mathrm{KL}(P \parallel Q) = -\int p(x) \log q(x)\, d\nu(x) - \left(-\int p(x) \log p(x)\, d\nu(x)\right). \tag{2.26}$$

The KL divergence is therefore the difference in expectation of the information $-\log p(x)$ and the cross-information $-\log q(x)$, also called the "cross entropy". The expectation is taken of the random variable X with density p.

Some important properties of KL divergence are

$$\begin{aligned} &\mathrm{KL}(P \parallel P) = 0 \\ &\mathrm{KL}(P \parallel Q) \neq \mathrm{KL}(Q \parallel P) \quad \text{(in general)} \\ &\mathrm{KL}(P \parallel Q) \geq 0. \end{aligned} \tag{2.27}$$

The first property, that the divergence of the same probabilities are equal to zero, is because of

$$\log \frac{p(x)}{p(x)} = \log 1 = 0. \tag{2.28}$$

This means when using KL divergence to find the similarity between two distributions the value of KL should be as close to zero as possible.

To illustrate the second property of inequality consider a random variable $Y \sim Q$, expand equation (2.24) and use the change of reference measure (2.7)

$$\begin{aligned} \mathrm{KL}(P \parallel Q) &= \int_A p(x) \log\left(\frac{p(x)}{q(x)}\right) d\nu(x) \\ &= \int_A \frac{p(x)}{q(x)} \log\left(\frac{p(x)}{q(x)}\right) q(x)\, d\nu(x) \\ &= \mathbf{E}\left[\frac{p(Y)}{q(Y)} \log\left(\frac{p(Y)}{q(Y)}\right)\right]. \end{aligned} \tag{2.29}$$

Compare the result to

$$\mathrm{KL}(Q \parallel P) = \mathbf{E}\left[\log \frac{q(Y)}{p(Y)}\right]. \tag{2.30}$$

From the expectations it is obvious that $\mathrm{KL}(P \parallel Q) \neq \mathrm{KL}(Q \parallel P)$ in general.

The last property of non-negativity will not be derived because of its complexity.

If P and Q have densities p, q respective, the notation $\mathrm{KL}(p \parallel q)$ is used for $\mathrm{KL}(P \parallel Q)$.

2.3.1 Example of Kullback-Leibler divergence

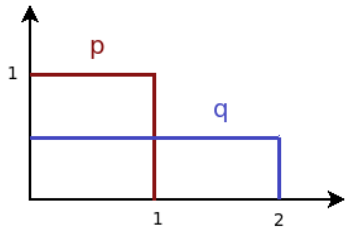

Figure 2.2: The uniform probability distributions $P_X = U(0,1)$ and $P_Y = U(0,2)$ defined on $E = \mathbb{R}^1$.

For two uniform probability distributions $P_X = U(0,1)$ and $P_Y = U(0,2)$ and $E = \mathbb{R}^1$ (the real line) there are two random variables $X \sim P_X$ and $Y \sim P_Y$, see figure 2.2. The probability densities (with respect to the Lebesgue measure) are

$$p(x) = \begin{cases} 1, & \text{if } x \in [0,1] \\ 0, & \text{else} \end{cases} \tag{2.31}$$

$$q(x) = \begin{cases} 0.5, & \text{if } x \in [0,2] \\ 0, & \text{else,} \end{cases} \tag{2.32}$$

Using (2.24) with respect to P_X is

$$\mathrm{KL}(P_X \parallel P_Y) = \int_0^1 \log \frac{1}{0.5}\, dx = \log 2. \tag{2.33}$$

Taking the KL divergence with respect to P_Y gives

$$\begin{aligned} \mathrm{KL}(P_Y \parallel P_X) &= \int_0^1 0.5 \log \frac{0.5}{1}\, dx + \int_1^2 0.5 \log \frac{0.5}{0}\, dx \\ &= 0.5 \log 0.5 + \infty = \infty. \end{aligned} \tag{2.34}$$

This example shows the importance of choosing $q(x)$ to be defined with a larger support than $p(x)$.

2.4 Variational inference

A central task in probabilistic modelling (see section 2.2 about Bayesian inference) is to find the posterior distribution given some data, a model and a prior distribution. This could be in the form of the posterior density $\pi(\theta \mid x)$ and evaluating the expectations with respect to this density, where θ is the unknown parameter or a latent variable and x is an observed quantity. For simple models this is possible but when the structures are more complex, like in a neural network the inference is intractable.

Variational inference (VI) is a form of approximating Bayesian inference where the distance between the approximation and the exact posterior is measured using Kullback-Leibler divergence. Finding a good approximation is then an optimisation problem. The idea behind variational inference is to make an approximation of Bayesian inference by finding a probability distribution Q among a parameterised family of distributions $\mathcal{Q}$ close to the actual posterior probability Π. Kullback-Leibler divergence is measuring the closeness and is the measure of how alike the two distributions are. If the approximation is perfect, then KL divergence is equal to 0. The goal of VI is in other words to minimise the KL divergence. Later it will be shown that minimising the KL divergence is equivalent to maximising what is called the evidence lower bound (ELBO). The distributions in $\mathcal{Q}$ can be of any type but in this thesis normal approximations are used.

The goal of probabilistic modelling is to find the exact posterior distribution for the model $\Pi(\cdot \mid X = x)$ with random variable X observed to be x with a distribution P_X depending on θ and density $\theta \mapsto \pi(\theta \mid x)$, $\theta \in \Theta$. When this is not possible to extract, a density $\theta \mapsto q(\theta)$ from a family of distributions $\mathcal{Q}$ is used to estimate the true posterior. Each distribution in $\mathcal{Q}$ is a possible best approximation of the exact posterior distribution. This distribution is called the *variational distribution* and should be as close to the true posterior as possible. To measure the quality of the estimation, the KL divergence (2.24) from Π to $Q \in \mathcal{Q}$ is used

$$\mathrm{KL}(Q \parallel \Pi(\cdot \mid x)) = \int q(\theta) \log \frac{q(\theta)}{\pi(\theta \mid x)}\, d\nu(\theta). \tag{2.35}$$

Remembering Bayes' rule for Bayesian inference (2.15) the true posterior $\pi(\theta \mid x)$

can be rewritten as

$$\begin{aligned}\mathrm{KL}(Q \parallel \Pi(\cdot \mid x)) &= \int q(\theta) \log\left(\frac{q(\theta)p(x)}{p(x \mid \theta)\pi(\theta)}\right) d\nu(\theta) \\ &= \int q(\theta)\left(\log\frac{q(\theta)}{p(x \mid \theta)\pi(\theta)} + \log p(x)\right) d\nu(\theta) \\ &= \int q(\theta) \log\frac{q(\theta)}{p(x \mid \theta)\pi(\theta)}\, d\nu(\theta) + \log p(x) \int q(\theta)\, d\nu(\theta) \\ &= -\int q(\theta) \log\left(\frac{p(x \mid \theta)\pi(\theta)}{q(\theta)}\right) d\nu(\theta) + \log p(x).\end{aligned} \tag{2.36}$$

Here θ is the latent variable governing the distribution of the data, x are the observations, $q(\theta)$ is the variational density, $p(x \mid \theta)$ is the likelihood function, $\pi(\theta)$ is the prior density and $p(x)$ is the evidence. This can be rewritten as

$$\mathrm{KL}(Q \parallel \Pi(\cdot \mid x)) = -\mathcal{L}(q) + \log p(x). \tag{2.37}$$

$\mathcal{L}$ is called the evidence lower bound (ELBO) and can be rewritten as

$$\begin{aligned}\mathcal{L}(q) &= \int q(\theta) \log\left(\frac{p(x \mid \theta)\pi(\theta)}{q(\theta)}\right) d\nu(\theta) \\ &= \int \log(p(x \mid \theta)) q(\theta)\, d\nu(\theta) - \int q(\theta) \log\left(\frac{q(\theta)}{\pi(\theta)}\right) d\nu(\theta) \\ &= \mathbf{E}\left[\log p(x \mid \vartheta)\right] - \mathrm{KL}(Q \parallel \Pi),\end{aligned} \tag{2.38}$$

where $\vartheta \sim Q$. The interpretation of the ELBO is that the first term will push for optimisation for the expected log likelihood of the unknown parameters that explain the observed data. The second term is pushing for the variational distribution to be close to the prior. These two terms will balance each other. A small KL implies a close estimation $q(\theta)$ of $\pi(\theta \mid x)$, see figure 2.3. Using Kullback-Leibler as defined is difficult because the evidence $p(x)$ is intractable. The ELBO can instead control the KL divergence as the likelihood $p(x \mid \theta)$ is easier to compute. Therefore, instead of optimising on the Kullback-Leibler divergence from Π to Q, it is the ELBO (2.38) that is optimised. Minimising the KL is the same as maximising the ELBO when noting the sign in front of $\mathcal{L}(q)$ in (2.37). The ELBO provides a lower bound for the evidence because the KL divergence is always non-negative, see properties (2.27),

$$\begin{aligned}\mathrm{KL}(Q \parallel \Pi(\cdot \mid x)) + \mathcal{L}(q) &= \log p(x) \\ \mathcal{L}(q) &\leq \log p(x).\end{aligned} \tag{2.39}$$

Later the variational distributions will be considered $Q(\lambda) \in \mathcal{Q}$ indexed by a variational parameter $\lambda \in \Lambda$, $\mathcal{Q} = \{Q(\lambda), \lambda \in \Lambda\}$. For the density of $Q(\lambda)$ write suggestively $q(\theta \mid \lambda)$ [5].

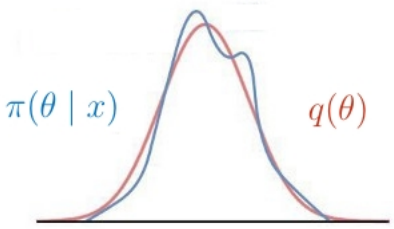

Figure 2.3: Variational inference where the variational density $q(\theta)$ estimates the posterior density $\pi(\theta \mid x)$.

2.4.1 Example of variational inference

The reader is referred to Appendix A.2 for complete calculation of all steps in the following example.

Consider a sample of independent random variables that are all normally distributed with a mean μ and variance σ_x^2, so $x_1, ..., x_n \sim N(\mu, \sigma_x^2)$. Assume that $x = (x_1, \ldots, x_n)$ was observed.

The parameter mean μ in the parameter space M of the distribution for this model $\Pi(\cdot \mid X)$ with density $\pi(\mu \mid x)$ is found using variational inference (2.36). For this model a simple prior distribution $\Pi = N(0, \sigma_0^2)$ is assumed with density $\pi(\mu)$ and two variational distributions $Q_1 = N(0, \sigma^2)$ with density $q_1(\mu)$ and $Q_2 = N(1, \sigma^2)$ with density $q_2(\mu)$. To simplify the calculations assume that $\sigma_x = \sigma_0 = \sigma$. Using the probability density function for the normal distribution (2.13) the different densities for the likelihood, prior and the two different variational densities are found:

the likelihood

$$\begin{aligned} p(x \mid \mu) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x_i - \mu)^2}{2\sigma^2}\right) \\ &= (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2\right), \end{aligned} \tag{2.40}$$

the prior

$$\pi(\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right), \tag{2.41}$$

the first variational

$$q_1(\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) \tag{2.42}$$

and the second variational

$$q_2(\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mu - 1)^2}{2\sigma^2}\right). \tag{2.43}$$

To find the best distribution for approximating the posterior distribution, the KL divergence from the posterior distribution $\Pi(\cdot \mid X)$ to the variational distributions Q_1 and Q_2, are to be minimised. This is done by instead maximising the evidence

lower bound (ELBO) (2.38). To simplify the integration of the likelihood times the prior the distribution of the two combined can be used instead. Using conjugate prior the hyperparameters of the posterior can be found from (A.2)

$$\begin{aligned} p(x \mid \mu)\pi(\mu) &\propto N\left(\frac{1}{\frac{1}{\sigma^2}+\frac{n}{\sigma^2}}\left(0+\frac{\sum_{i=1}^{n} x_i}{\sigma^2}\right), \left(\frac{1}{\sigma^2}+\frac{n}{\sigma^2}\right)^{-1}\right) \\ &= N\left(\frac{\sum_{i=1}^{n} x_i}{n+1}, \frac{\sigma^2}{n+1}\right). \end{aligned} \tag{2.44}$$

The ELBO is calculated for each variational density at a time and the result compared. First for $q_1(\mu)$

$$\begin{aligned} \mathcal{L}(q_1) &= \int q_1(\mu) \log\left(\frac{p(x \mid \mu)\pi(\mu)}{q_1(\mu)}\right) d\mu \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) \left(-\frac{1}{2}\log\frac{2\pi\sigma^2}{n+1} - \frac{(n+1)\left(\mu - \frac{\sum_{i=1}^{n} x_i}{n+1}\right)^2}{2\sigma^2} \right. \\ &\quad \left. + \frac{1}{2}\log 2\pi\sigma^2 + \frac{\mu^2}{2\sigma^2}\right) d\mu. \end{aligned} \tag{2.45}$$

The Gaussian integrals (see A.9) are needed for the integration. Expanding the brackets and applying the Gaussian integrals implies

$$\mathcal{L}(q_1) = \frac{1}{2}\log(n+1) - \frac{1}{2\sigma^2}\left((n+1)\sigma^2 + \frac{1}{n+1}\left(\sum_{i=1}^{n} x_i\right)^2 - \sigma^2\right). \tag{2.46}$$

For $q_2(\mu)$ the ELBO is

$$\begin{aligned} \mathcal{L}(q_2) &= \int q_2(\mu) \log\left(\frac{p(x \mid \mu)\pi(\mu)}{q_2(\mu)}\right) d\mu \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{(\mu-1)^2}{2\sigma^2}\right) \left(-\frac{1}{2}\log\frac{2\pi\sigma^2}{n+1} - \frac{(n+1)\left(\mu - \frac{\sum_{i=1}^{n} x_i}{n+1}\right)^2}{2\sigma^2} \right. \\ &\quad \left. + \frac{1}{2}\log 2\pi\sigma^2 + \frac{(\mu-1)^2}{2\sigma^2}\right) d\mu. \end{aligned} \tag{2.47}$$

Expanding the brackets and applying the Gaussian integrals (A.9) implies

$$\mathcal{L}(q_2) = \frac{1}{2}\log(n+1) - \frac{1}{2\sigma^2}\left((n+1)(1+\sigma^2) - 2\sum_{i=1}^{n} x_i + \frac{1}{n+1}\left(\sum_{i=1}^{n} x_i\right)^2 - \sigma^2\right). \tag{2.48}$$

When comparing the results of the ELBO for the two different variational densities to see which gives the largest result, it can be noted that all terms in the result for

$\mathcal{L}(q_1)$ (2.46) are found in the result for $\mathcal{L}(q_2)$ (2.48). In this way,

$$\begin{aligned} & \mathcal{L}(q_1) \leq \mathcal{L}(q_2) \\ \Leftrightarrow \quad & 0 \leq -\frac{1}{2\sigma^2}\left(-2\sum_{i=1}^{n} x_i + (n+1)\right) \\ \Leftrightarrow \quad & \frac{1}{2} \leq \frac{1}{n+1}\sum_{i=1}^{n} x_i. \end{aligned} \tag{2.49}$$

For a sample of random variables the mean is given by $\frac{1}{n}\sum_{i=1}^{n} x_i$. Because $q_1(\mu)$ has the mean $\mu_1 = 0$ and $q_2(\mu)$ the mean $\mu_2 = 1$, if the sample mean is above $\frac{1}{2}$, $q_2(\mu)$ should give the best approximation. The result above has the term of $\frac{1}{n+1}$ instead of $\frac{1}{n}$. This can be interpreted as that more observations need to be closer to the value 1 for $q_2(\mu)$ being the best approximation. The random variables need to compensate for the prior distribution having a mean value equal to 0 because the prior is also an observation at 0. Note that when the number of observations is large the impact of this decreases.

2.4.2 Black box variational inference

While a method for computing the variational inference has been established, an optimisation method is also necessary. Variational inference is an optimisation problem where the variational parameter is updated trying to be as similar to the true posterior as possible. Knowledge about the model is required to efficiently derive an algorithm using variational inference but this is often not available. Using gradient descent is a standard method for optimisation and is an iterative optimisation algorithm for finding a local minimum for a differentiable function. The problem when using Bayesian inference for modelling is that the parameters have distributions and using regular gradient descent is therefore problematic. Two methods for optimisation while getting around this problem will be presented. First, the *black box variational inference* method is presented based on Ranganath, Gerrish and Blei 2014 [3]. Later, *variational inference using the reparameterisation trick* based on Kingma and Welling 2013 [4] will be shown. Last, these two optimisation methods will be compared.

Black box variational inference is a stochastic optimisation method computing the gradient of the variational objective using Monte Carlo sampling from the variational distribution. This method is quite general, meaning it can be applied to almost any model with little effort because of the rewriting of the gradient to only be of the variational distribution and not the whole objective function. The rewriting can be done even though the likelihood and prior is dependent on the variational objective. This makes the black box method easy to use to any model because the likelihood is the model-specific probability.

The variational distribution is from now on parameterised by λ so the variational distribution Q have a density $q = q(\cdot \mid \lambda)$ depending on λ. The objective function, the evidence lower bound (ELBO) (2.38), to optimise is

$$\mathcal{L}(q) = \int q(\theta \mid \lambda) \log \left(\frac{p(x \mid \theta)\pi(\theta)}{q(\theta \mid \lambda)} \right) d\theta, \tag{2.50}$$

where $q(\theta \mid \lambda)$ is the variational density with latent parameters θ and free parameters of the variational distribution λ. The variational parameter λ will later represent the mean value μ and standard deviation σ of the variational distribution because it is assumed in this thesis that $q(\cdot \mid \lambda)$ is Gaussian. As stated earlier, optimising the ELBO implies minimising the Kullback-Leibler divergence which in turn searches for the parameters in the variational distribution closest to the conditional distribution that is not known.

The black box method uses stochastic optimisation to optimise the ELBO. Stochastic optimisation is a method for optimising an objective function with randomness present, so noisy estimates of the gradient are used to maximise the objective function [7]. The estimation is done by sampling from the variational distribution, computing the gradient using these samples, multiplying them with the ELBO and updating the parameters with the result. Stochastic optimisation for the variable λ at iteration t will be

$$\lambda^{(t+1)} = \lambda^{(t)} + \rho^{(t)} \nabla_\lambda \mathcal{L}^{(t)}. \tag{2.51}$$

λ is updated by the gradient of the objective function $\mathcal{L}(q)$ with respect to λ. This converges to a local maximum of $\mathcal{L}(q)$. ρ is the step size, also called the learning rate. The learning rate can follow the Robbins-Monro conditions [7]

$$\sum_{t=1}^{\infty} \rho_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \rho_t^2 < \infty. \tag{2.52}$$

A step size that meets these conditions and does not decrease too fast is for example

$$\rho = \rho_0 \frac{1}{\sqrt{t}}, \tag{2.53}$$

where ρ_0 is a parameter selected small.

The gradient of the ELBO is estimated by Monte Carlo samples from the variational distribution. By using randomness, Monte Carlo methods give approximate solutions to deterministic problems that are hard to solve. When deriving the gradient note that the integral in the ELBO is the same as the expected value dependent on the variational distribution Q with density $q(\theta \mid \lambda)$. Start by taking the gradient of the ELBO (2.50) with respect to λ

$$\begin{aligned} \nabla_\lambda \mathcal{L}(q) &= \nabla_\lambda \int q(\theta \mid \lambda) \log \left(\frac{p(x \mid \theta)\pi(\theta)}{q(\theta \mid \lambda)} \right) d\theta \\ &= \int \nabla_\lambda \left[q(\theta \mid \lambda) \left(\log p(x \mid \theta) + \log \pi(\theta) - \log q(\theta \mid \lambda) \right) \right] d\theta, \end{aligned} \tag{2.54}$$

assuming the gradient and integral can be interchanged. Using the product rule for

the gradient and cancelling out the terms not dependent on λ implies

$$\begin{aligned}\nabla_\lambda \mathcal{L}(q) &= \int \nabla_\lambda [q(\theta \mid \lambda)](\log p(x \mid \theta) + \log \pi(\theta) - \log q(\theta \mid \lambda))\, d\theta \\ &\quad + \int \nabla_\lambda [\log p(x \mid \theta) + \log \pi(\theta) - \log q(\theta \mid \lambda)] q(\theta \mid \lambda)\, d\theta \\ &= \int \nabla_\lambda [q(\theta \mid \lambda)](\log p(x \mid \theta) + \log \pi(\theta) - \log q(\theta \mid \lambda))\, d\theta \\ &\quad - \mathbf{E}[\nabla_\lambda \log q(\vartheta \mid \lambda)],\end{aligned} \tag{2.55}$$

where $\vartheta \sim q(\cdot \mid \lambda)$. The expectation is zero and is realised by using the chain rule on the logarithm and integrating over the whole probability space which is equal to one,

$$\mathbf{E}[\nabla_\lambda \log q(\vartheta \mid \lambda)] = \mathbf{E}\left[\frac{\nabla_\lambda q(\vartheta \mid \lambda)}{q(\vartheta \mid \lambda)}\right] = \nabla_\lambda \int q(\theta \mid \lambda)\, d\theta = \nabla_\lambda 1 = 0. \tag{2.56}$$

Using the same rewriting for the first term in (2.55), so that the gradient of the logarithm using the chain rule is $\nabla_\lambda [q(\theta \mid \lambda)] = \nabla_\lambda [\log q(\theta \mid \lambda)] q(\theta \mid \lambda)$, implies that

$$\begin{aligned}\nabla_\lambda \mathcal{L}(q) &= \int \nabla_\lambda [\log q(\theta \mid \lambda)](\log p(x \mid \theta) + \log \pi(\theta) - \log q(\theta \mid \lambda)) q(\theta \mid \lambda)\, d\theta \\ &= \mathbf{E}[\nabla_\lambda [\log q(\vartheta \mid \lambda)](\log p(x \mid \vartheta) + \log \pi(\vartheta) - \log q(\vartheta \mid \lambda))].\end{aligned} \tag{2.57}$$

In the end the gradient has only to be taken on the logarithm of the variational distribution and not on the underlying model. The expectation is with respect to the variational distribution. Therefore samples θ can be drawn from the variational density $q(\theta \mid \lambda)$ to approximate the gradient of the ELBO.

To compute this gradient of the ELBO a Monte Carlo method where samples drawn from the variational distribution are used to approximate the expected value

$$\nabla_\lambda \mathcal{L}(q) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\lambda [\log q(\vartheta_s \mid \lambda)](\log p(x \mid \vartheta_s) + \log \pi(\vartheta_s) - \log q(\vartheta_s \mid \lambda)), \tag{2.58}$$

where $\vartheta_s \overset{\text{iid}}{\sim} q(\cdot \mid \lambda)$ and S is the number of samples drawn from the variational distribution. The approximation of the gradient of the ELBO, $\nabla_\lambda \mathcal{L}(q)$, is used in the stochastic optimisation (2.51) and moves λ in direction of promising values for the distribution.

2.4.3 Variational inference using the reparameterisation trick

The black box method rewrites the gradient of the ELBO and avoids the need to take the gradient of the likelihood with respect to the parameters. On the other hand if this gradient is available using it can speed up the optimisation. The problem is that taking the gradient of a parameterised distribution is problematic. When using a neural network and backpropagating through a random node there can be a problem with high variance. The *reparameterisation trick* [4] s a way of getting around this

by backpropagating through a deterministic node instead. Monte Carlo sampling is still used to approximate the gradient of the ELBO but shifting the randomness to a parameter-free distribution so the gradient of the stochastic variables can be computed.

To illustrate the reparameterisation trick consider the variational density $q(\theta \mid \lambda)$ parameterised by $\lambda = (\mu, \sigma)$ where μ is the mean and σ is the standard deviation of the normal distribution (2.12), ϑ is sampled from $q(\theta \mid \lambda)$ and

$$\vartheta \sim \mathcal{N}(\mu, \sigma^2). \tag{2.59}$$

This can be shifted to a deterministic linear function of μ and σ and a standard normal distribution (2.14) random variable Ξ

$$\vartheta = \mu + \sigma\Xi, \quad \Xi \sim \mathcal{N}(0, 1). \tag{2.60}$$

The randomness in ϑ is shifted to a much simpler parameter-free distribution. Ξ will be sampled the same number of times that ϑ is earlier sampled from the variational distribution in the Monte Carlo approximation. Then ϑ is generated from this reparameterisation function $g(\lambda, \xi)$ where

$$g(\lambda, \xi) = \mu + \sigma\xi. \tag{2.61}$$

Even though ϑ is rewritten using $g(\lambda, \xi)$ it still follows a normal distribution with parameters μ and σ. Recall that $Y = a + bX$, where $X \sim N(\mu_X, \sigma_X^2)$ is normally distributed with $Y \sim N(a + b\mu_X, b^2\sigma_X^2)$. Thus with $\Xi \sim \mathcal{N}(0, 1)$,

$$g(\lambda, \Xi) \sim N(\mu, \sigma^2).$$

Finding the gradient descent will require some rewriting using the reparameterisation function $g(\lambda, \xi)$. Recall the stochastic optimisation of variable λ (2.51) at iteration t is updated using the gradient descent of $\mathcal{L}(q)$ by

$$\lambda^{(t+1)} = \lambda^{(t)} + \rho^{(t)} \nabla_\lambda \mathcal{L}^{(t)}, \tag{2.62}$$

where ρ is the step size, and the gradient of the ELBO (2.38) is

$$\nabla_\lambda \mathcal{L}(q) = \nabla_\lambda \int q(\theta \mid \lambda) \log \left(\frac{p(x \mid \theta)\pi(\theta)}{q(\theta \mid \lambda)} \right) d\theta. \tag{2.63}$$

Splitting the integral into two terms makes the derivation of the gradient using the reparameterisation trick easier:

$$\begin{aligned} \nabla_\lambda \mathcal{L}(q) = &\int \nabla_\lambda \left[\log p(x \mid \theta) q(\theta \mid \lambda) \right] d\theta \\ &- \int \nabla_\lambda \left[(\log q(\theta \mid \lambda) - \log \pi(\theta)) q(\theta \mid \lambda) \right] d\theta. \end{aligned} \tag{2.64}$$

For the first term in the equation above, since the randomness in ϑ is shifted to Ξ using $g(\lambda, \xi)$ (2.61), a trick of shifting the reference measure (2.7) to rewriting the integral in terms of ξ can be used. Because of the choice of g it holds that

$q(\theta \mid \lambda)\, d\theta = q(\xi)\, d\xi$, compare with Proposition 1 in [8]. Also θ in the likelihood is rewritten using the function $g(\lambda, \xi)$ so using this with $p(x \mid \theta) = p(x \mid g(\lambda \mid \xi))$ and the chain rule implies

$$\begin{aligned}\nabla_\lambda \int \log p(x \mid \theta) q(\theta \mid \lambda)\, d\theta &= \nabla_\lambda \int \log p(x \mid g(\lambda, \xi)) q(\xi)\, d\xi \\ &= \int \frac{\partial}{\partial \lambda} g(\lambda, \xi) \cdot \nabla_\theta [\log p(x \mid g(\lambda, \xi))] q(\xi)\, d\xi \\ &= \mathbf{E}\left[\frac{\partial}{\partial \lambda} g(\lambda, \Xi) \cdot \nabla_\vartheta \log p(x \mid \vartheta)\right],\end{aligned} \tag{2.65}$$

where $\vartheta \sim q(\cdot \mid \lambda)$, $\Xi \sim \mathcal{N}(0, 1)$ and $\frac{\partial}{\partial x} f = \left(\frac{\partial}{\partial x_i} f_j\right)_{i,j}$ is the Jacobian matrix of $f(x)$ in x.

The second term in equation (2.64) can be rewritten using the same method as earlier in the black box version of variational inference. The product rule for the gradient is first used

$$\begin{aligned}\int \nabla_\lambda [(\log q(\theta \mid \lambda) - \log \pi(\theta)) q(\theta \mid \lambda)]\, d\theta = \\ \int \nabla_\lambda [\log q(\theta \mid \lambda) - \log \pi(\theta)] q(\theta \mid \lambda)\, d\theta \\ + \int \nabla_\lambda [q(\theta \mid \lambda)](\log q(\theta \mid \lambda) - \log \pi(\theta))\, d\theta.\end{aligned} \tag{2.66}$$

The first term is equal to zero,

$$\int \nabla_\lambda [\log q(\theta \mid \lambda)] q(\theta \mid \lambda)\, d\theta - \int \nabla_\lambda [\log \pi(\theta)] q(\theta \mid \lambda)\, d\theta = 0, \tag{2.67}$$

because the integral on the whole probability space of q is equal to 1 and the gradient of 1 is equal to zero (same reasoning as in equation (2.56)). The gradient of the prior distribution is not dependent on λ and therefore this term is also equal to zero.

The second term can be rewritten using the same trick as from black box, see equation (2.57),

$$\begin{aligned}&\int \nabla_\lambda [q(\theta \mid \lambda)](\log q(\theta \mid \lambda) - \log \pi(\theta))\, d\theta \\ &= \int \nabla_\lambda [\log q(\theta \mid \lambda)](\log q(\theta \mid \lambda) - \log \pi(\theta)) q(\theta \mid \lambda)\, d\theta \\ &= \mathbf{E}[\nabla_\lambda [\log q(\vartheta \mid \lambda)](\log q(\vartheta \mid \lambda) - \log \pi(\vartheta))].\end{aligned} \tag{2.68}$$

where $\vartheta \sim q(\cdot \mid \lambda)$. The important difference from the black box method is that ϑ is *not* sampled directly from the variational distribution $\mathcal{N}(\mu, \sigma^2)$. Instead Ξ is sampled from the standard normal distribution $\mathcal{N}(0, 1)$ and ϑ is obtained from $g(\lambda, \xi) = \mu + \sigma\xi$. The Monte Carlo approximation of the gradient used for the reparameterisation trick is

$$\begin{aligned}\nabla_\lambda \mathcal{L}(q) \approx \frac{1}{S} \sum_{s=1}^{S} \Bigg[&\frac{\partial}{\partial \lambda} g(\lambda, \Xi_s) \cdot \nabla_{\vartheta_s} \log p(x \mid \vartheta_s) \\ &- \nabla_\lambda [\log q(\vartheta_s \mid \lambda)](\log q(\vartheta_s \mid \lambda) - \log \pi(\vartheta_s))\Bigg].\end{aligned} \tag{2.69}$$

2.5 Neural network

Machine learning is a way for achieving artificial intelligence through a process of automatically teaching and updating a program without explicitly programming it. By learning the structure of the input data, machine learning models can be used for predicting upcoming events. A neural network is a machine learning method loosely based on the structure of the biological brain, consisting of neurons and the connections between them. The neural network is a deep learning technique that is skilled at learning complex patterns in large datasets and are popular modelling tools thanks to their ability to learn complicated non-linear functions.

A *feedforward neural network* [5] is a simple deep neural network. Mathematically, it can be represented as a function $x \mapsto f(x, w)$ mapping an input x to an output y depending on weight parameters w. A feedforward network consists of one or more hidden layers with several connected nodes (neurons) that send information through the network. These nodes and connections emulate the neurons in the human brain. The feedforward network moves information forward from the input nodes, through the hidden nodes and to the output nodes. The feedforward network is often used in supervised learning. Supervised learning is when a function is determined from labelled data with input-output pairs, meaning that in the training an input value will lead to a known output. The goal of the feedforward network is to learn the function mapping input values to the output values so that $\hat{y} = f(x, w) \approx y$.

A feedforward network is separated into *layers*. It consists of an input layer, one or more hidden layers and an output layer. Each layer consists of several *nodes* that have connections (edges) to the nodes in the previous layer, see figure 2.4. In a fully connected network, all the nodes in one layer are connected with all nodes in the next layer. All of the edges have *weights* associated with them. Each neuron takes input and multiplies it with a weight. The weight will determine how much to emphasise or ignore the inputs. A *bias* is then added and transformed using an *activation function*, and this is the output. The bias helps to shift the activation function to fit the prediction better, the same way as the intercept is added in a linear function. A simple structure of a feedforward network with two hidden layers can be

$$x \to \underbrace{\phi\left(w^{(1)} \cdot x + b^{(1)}\right)}_{z^{(1)}} \to \underbrace{\phi\left(w^{(2)} \cdot z^{(1)} + b^{(2)}\right)}_{z^{(2)}} \to \hat{y}. \tag{2.70}$$

Here x is the input data, $w^{(i)}$ is the weights of the l-th layer, $b^{(i)}$ is the biases, and ϕ is the activation function. The activation function adds non-linearity to the model. A very common activation function is the ReLu function that transform negative values to zero and keep the rest

$$\phi(z) = \max(0, z). \tag{2.71}$$

A neural network needs to be *trained* to learn how to classify the input-output pairs correctly. When training, the weights and biases are adjusted so that the estimated output $\hat{y}$ is as close to the true value y as possible. A *cost function* of choice is used

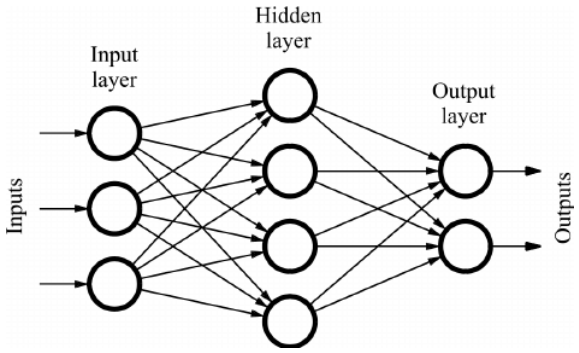

Figure 2.4: An example of the structure of a feedforward neural network [9]

.

to keep track of how correct the prediction is. A common choice is the *mean squared error* MSE for a set of N input-output pairs $X = \{(x_1, y_1), \ldots, (x_N, y_N)\}$

$$MSE(X) = \frac{1}{2N} \sum_{i=1}^{N} \| \hat{y}_i - y_i \|^2 . \tag{2.72}$$

When the predictions are identical $\hat{y} = y$ for all input-output pairs, $MSE(X) = 0$. Therefore, the objective is to minimise the mean square error with respect to the weight w and bias b. *Backpropagation* is when the error from the cost function is propagated back through each layer to update the weights using the gradient of the loss function with respect to the weights.

In order to minimise the objective function, *gradient descent* can be used. Gradient descent is using the direction of the negative gradient of the cost function (2.72) to adjust the weights and biases. It is an iterative process that converges the cost function to a local minimum. For the weights and biases, the update at iteration i when using gradient descent is

$$\begin{aligned} w^{(i+1)} &= w^{(i)} - \alpha \frac{\partial MSE(X)}{\partial w^{(i)}} \\ b^{(i+1)} &= b^{(i)} - \alpha \frac{\partial MSE(X)}{\partial b^{(i)}}. \end{aligned} \tag{2.73}$$

$\alpha > 0$ is the *learning rate* which determines the step size the gradient takes in each iteration. The learning rate is a *hyperparameter* often chosen to be a small number. Hyperparameters in machine learning are the parameters set before training and often need to be adjusted. A too small learning rate will produce a slow convergence while a too large learning rate will fail to converge at all.

Often a stochastic gradient which uses a randomised approximation of the gradient is used. A popular optimiser is the adaptive learning rate optimisation algorithm, *Adam* [10]. Adam is often used when training deep neural networks and is an extension of stochastic gradient descent. It computes individual learning rates for the different parameters and is adapted during training. The learning rate in Adam is scaled by the squared gradients and use a moving average of the gradient to create momentum.

When modelling using a neural network, the dataset is often split into a *training* dataset and a *testing* dataset. The training dataset is used to train the model where the weights and biases are learned. The testing dataset evaluates the model once when finished training. In this way, the model can be tested on data it has never seen before.

The *batch size* is another "hyperparameter of gradient descent that controls the number of training samples to work through before the model's internal parameters are updated"[11]. The number of *epochs* is a hyperparameter "that controls the number of complete passes through the training dataset"[11].

2.5.1 Bayesian neural networks

Neural networks have become popular tools for prediction but with regular networks there may be an overconfidence in the results. For example a small or imbalanced dataset can lead to overfitting. Bayesian inference for neural networks is useful when the data is sparse or noisy, prior knowledge about the unknown parameters exist or details about the uncertainty in the results are wanted.

As for Bayesian regression in section 2.2.1 consider a dataset $\mathcal{D} = (x, y)$ with input variables x and output variables y. A Bayesian neural network is basically a regression model where the functional relationship is given by a neural network. Assuming a normal distribution error a Bayesian neural network model can be expressed as

$$y \sim \mathcal{N}(f(x, \theta), \sigma^2), \tag{2.74}$$

where θ are the weight parameters and σ is the standard deviation. The function to be determined is $f(x, \theta)$ and is describing the relationship in the data. A likelihood function can be constructed for the data $p(y \mid x, \theta)$. Imposing a prior belief on the weights and Bayesian inference can be used for prediction.

Bayesian neural networks include a measure of the uncertainty in the prediction by providing a posterior distribution. Where regular neural networks give point estimations, Bayesian neural networks provide uncertainty measurements for the model. The model parameters in the neural network are assigned priors, so both the weights and the biases will have probability distributions. Specification on which priors used in the case study section 3.4. The variational parameter θ will therefore obtain all unknown parameters of the model so on this case the mean and standard deviation of the weights and biases so

$$\theta = \{\text{collection of parameters for each weight and bias}\}. \tag{2.75}$$

The posterior of the prediction is quantified using the uncertainty of the weights propagated through the system. Therefore the output y will have uncertainty as well. This uncertainty is quantified by the posterior predictive distribution. It determines a model's confidence at new datapoints and the range of which the new data is most likely to be observed.

Bayesian inference for deep learning works similarly to regular deep learning so it is useful to compare the different methods. The following three steps are common in deep learning: first a model is defined, then a dataset is chosen and last the algorithm is run which will learn the unknown parameters of the model. When using a Bayesian neural network, these three steps are adjusted in the following way. First, when defining the model, the parameters get distributions with a prior belief of which values these distributions can take. Second, the data is viewed as observations from these distributions. Last, as the learning algorithm runs, the knowledge of these parameters are used to update the model. In the end, a probability distribution is provided. The wider the spread of the distribution, the larger the uncertainty in the model.

By applying Bayes' rule for Bayesian regression (2.19) on a neural network the probability distribution for the weights given the data can be found. Using Bayesian inference for neural networks will provide a full distribution for the parameter θ instead of the single most likely value for θ. The Bayesian network can also be used to find the distribution over output y and in model selection by learning the appropriate model size and type.

Optimisation in neural networks is to minimise the cost function, for example, the mean squared error function. For Bayesian networks, this is the same as finding a maximum of the likelihood function $p(y \mid x, \theta^*)$, meaning finding the weights and biases θ^* that maximises the probability of the data given that parameter.

The challenge with a Bayesian neural network is that it is computationally expensive when there are large datasets and complex models (many parameters). Bayesian inference involves integrals that are intractable except in trivial cases. There are several strategies to get around this, for example making algorithms that handle large datasets better, using several computers or approximating the answer by using variational inference. The VI as an approximation for Bayesian neural networks works the same way as described in section 2.4. Specifications of how VI is considered for the neural network in details will be described in the case study section 3.3.

To summarise, a family of distributions Q is used to approximate the posterior distribution. A minimisation of the Kullback-Leibler divergence is used to get a satisfying approximation which is equivalent to maximising the ELBO (2.38). A Monte Carlo sampling is used, as explained in 2.4.2 Black box and 2.4.3 Reparameterisation trick, to approximate the intractable integrals. These two VI methods also ensure that the sampling is from the simpler distribution Q which is also used for approximating the posterior.

3

Case study

3.1 Flow rate prediction

A model structure is needed for problem-solving using Bayesian inference. The modelling process is separated into four steps: *modelling, approximating, optimising* and *the goal*, see the flowchart in figure 3.1. First, the model is stated with stochastic variables and a prior distribution. A Bayesian approach is used to find a posterior distribution for the model. Bayesian inference is intractable for complex models so approximations is used. The posterior distribution is approximated using a family of densities $q \in Q$. The objective function is the ELBO (2.38) from the Kullback-Leibler divergence. Third, optimising through either black box variational inference (2.58) or variational inference using the reparameterisation trick (2.69). Last, the goal is to get a posterior distribution explaining the distribution of the model.

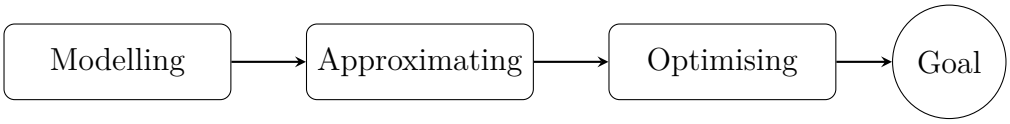

Figure 3.1: Flowchart of the modelling process.

There are two methods presented for optimising using variational inference, the black box- and the reparameterisation trick method. These two methods will be compared to identify which performs best. A regression model is used instead of a neural network for the comparison. The data used for the regression model is generated from a simple function and not flow rate data. When the best-suited optimisation method is identified, this method will be used on a neural network plate model with flow rate data provided by Solution Seeker.

The dataset used for training the neural network is comprised of samples of steady-state production data from a petroleum field on the Norwegian continental shelf, recorded between January 2014 and January 2018. The production data consists of 11 wells producing oil, gas and water through two risers. Solution Seeker generated this data using a steady-state classification algorithm that tracks periods of steady-state production. Steady-state production is when all variables measured are approximately the same during a specific time period. One data point is therefore a period when no variables change outside certain given restrictions. The variables in the dataset have a mean value and a standard error measurement. In this paper only

the mean value measurement of the variables are used. The dataset is anonymised and shared under a non-disclosure agreement with Solution Seeker. The dataset will be explained in detail in section 3.5.

3.2 Plate model

A plate model can be used to illustrate the model for a Bayesian network. The different shapes in the plate model in figure 3.2 have the following meanings:

- *Circle:* distribution of random variables.
  - *Coloured:* observed random variable.
  - *White:* unknown (latent) variable.
- *Arrow:* relationship between random variables. The arrow pointing from x to y means that y depends on x.
- *Square:* the compact figure for N samples.

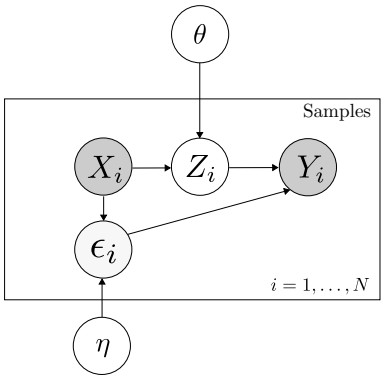

Figure 3.2: Plate model for flow rate.

The plate model for the Bayesian network modelling flow rate is shown in figure 3.2. The different variables are:

- X_i – Explanatory variables such as pressure, temperature, choke and flow rates. These are observed values and not used as output. These variables could be random, but for example the choke is known to have minimal randomness. The distributions of pressure and temperature are known and often considered as normally distributed.
- Z_i – Latent random variables and the true flow rate. The true flow rate is not measured.
- Y_i – The measurements of the flow rate. These are dependent on a noise ϵ_i. Note that Y_i is directly dependent on Z_i and not X_i. This mean that if Z_i is known X_i does not provide any new information for Y_i.

- θ – Parameters of the conditional probability $\pi(\theta \mid x)$, given data x. These can be thought of as the usual parameters used for training a network without the heteroscedastic noise.
- ϵ_i – Heteroscedastic noise. These are dependent on variables X_i but could also be noise to Z_i.
- η – Parameters describing the conditional probability of ϵ_i.

3.3 Variational inference for the regression model and the neural network

To estimate the uncertainty of a probabilistic model the ELBO (2.38) is optimised. For latent parameter θ governing the distribution of the data, observed variables x and output variable y the ELBO is

$$\mathcal{L}(q) = \int q(\theta \mid \lambda) \log \left(\frac{p(y \mid x, \theta)\pi(\theta)}{q(\theta \mid \lambda)} \right) d\theta. \tag{3.1}$$

$q(\theta \mid \lambda)$ have free parameters of the variational distribution $\lambda = (\mu, \sigma)$ which is normally distributed where $\mu \in \mathbb{R}^m$ is the mean vector and $\sigma \in \mathbb{R}^{m \times m}$ are the elements of a diagonal covariance matrix. m is the number of unkown parameters in the model. Because σ are the elements of a diagonal covariance matrix the parameters θ are assumed to be independent. Remembering from section 2.4 that a parameterised family of distributions $\mathcal{Q}$ is needed to find the distribution Q to approximate the posterior distribution. The *mean-field variational family* [1] is used for creating a variational family $\mathcal{Q}$. Here each unknown parameter θ in Θ is assumed to be independent. So for m parameters of θ, the density of a distribution that belongs to the mean-field variational family can be written as

$$q(\theta \mid \lambda) = \prod_{i=1}^{m} q(\theta_i \mid \lambda_i), \tag{3.2}$$

where λ parameterises the marginal distribution.

Recall the optimisation of variable λ at iteration t with step size ρ is (2.51):

$$\lambda^{(t+1)} = \lambda^{(t)} + \rho^{(t)} \nabla_\lambda \mathcal{L}^{(t)}. \tag{3.3}$$

Using *black box VI* (2.58) to approximate the gradient of $\mathcal{L}$, the optimisation step in stochastic gradient descent is

$$\lambda^{(t+1)} = \lambda^{(t)} + \rho^{(t)} \frac{1}{S} \sum_{s=1}^{S} \nabla_{\lambda^{(t)}} [\log q(\theta_s \mid \lambda^{(t)})](\log p(y \mid x, \theta_s) + \log \pi(\theta_s) - \log q(\theta_s \mid \lambda^{(t)})). \tag{3.4}$$

VI using the reparameterisation trick (2.69) gives the optimisation

$$\lambda^{(t+1)} = \lambda^{(t)} + \rho^{(t)} \frac{1}{S} \sum_{s=1}^{S} \Bigg[\frac{\partial}{\partial \lambda^{(t)}} g(\lambda^{(t)}, \xi_s) \cdot \nabla_{\theta_s} \log p(y \mid x, \theta_s) \\ - \nabla_{\lambda^{(t)}} [\log q(\theta_s \mid \lambda^{(t)})] (\log q(\theta_s \mid \lambda^{(t)}) - \log \pi(\theta_s)) \Bigg]. \tag{3.5}$$

To calculate the gradient of the ELBO the logarithms of the different distributions are needed. The probability density function for the normal distribution (2.13) is used throughout.

The *log likelihood density* of $p(y \mid x, \theta)$ has the form $p(y \mid f(x, \theta))$ in both the regression model (2.18) and the neural network model (2.74). When $y \sim \mathcal{N}(f(x, \theta), \sigma^2)$ and n is the number of samples the logarithm of the normal density is

$$\begin{aligned} \log p(y \mid x, \theta) &= \log \prod_{i=1}^{n} p(y_i \mid f(x_i, \theta)) \\ &= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2} \left(\frac{y_i - f(x_i, \theta)}{\sigma} \right)^2 \right) \\ &= -\frac{1}{2} \sum_{i=1}^{n} \left(\log 2\pi + \log \sigma^2 + \frac{(y_i - f(x_i, \theta))^2}{\sigma^2} \right) \\ &= -\frac{1}{2} \left(n \log 2\pi + n \log \sigma^2 + \frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2 \right). \end{aligned} \tag{3.6}$$

For the *log prior density* $\pi(\theta)$, where $\theta \sim \mathcal{N}(\mu_\pi, \sigma_\pi^2)$,

$$\log \pi(\theta) = -\frac{1}{2} \left(\log 2\pi + \log \sigma_\pi^2 + \frac{(\theta - \mu_\pi)^2}{\sigma_\pi^2} \right). \tag{3.7}$$

Note that the prior distribution has a fixed mean μ_π and standard deviation σ_π that are not updated during training.

Similarly, the *log variational density* for $q(\theta \mid \lambda)$, where $\theta \sim \mathcal{N}(\mu, \sigma^2)$ being parameterised by $\lambda = (\mu, \sigma)$, is

$$\log q(\theta \mid \lambda) = -\frac{1}{2} \left(\log 2\pi + \log \sigma^2 + \frac{(\theta - \mu)^2}{\sigma^2} \right). \tag{3.8}$$

The different *gradients* of the log densities are also needed. The gradient should be able to take both positive and negative values when training the model. If the value of λ is overestimated it should be able to decrease and vice versa. σ is a parameter of λ, but σ is the standard deviation so it is assumed positive. This constraint needs to be included but it is tedious to add constraints to stochastic gradient descent optimisation. To ensure a variable is positive the *softplus* [8] transformation can be used instead

$$\sigma = \log(1 + \exp(v)). \tag{3.9}$$

The softplus function maps v from a value that can be both positive and negative to a value σ taking only positive values. Note that as v grows larger, $v \approx \sigma$. The model is therefore parameterised with v instead of σ. The softplus transformation of v is used when λ is input to a distribution because σ needs to be positive in the normal distribution.

The *gradient of the log variational density* (3.8) using the softplus transformation (3.9) on v is

$$\nabla_\lambda \log q(\theta \mid \lambda) = \nabla_\lambda \left[-\frac{1}{2} \left(\log 2\pi + \log \sigma^2 + \frac{(\theta - \mu)^2}{\sigma^2} \right) \right]. \tag{3.10}$$

Here $\sigma = \log(1 + \exp(v))$. Because $\lambda = (\mu, v)$ this leads to two gradients, one for μ and one for v

$$\begin{aligned} \nabla_\mu \log q(\theta \mid \lambda) &= \frac{\theta - \mu}{(\log(1 + \exp(v)))^2}, \\ \nabla_v \log q(\theta \mid \lambda) &= -\exp(v) \frac{-\mu^2 + 2\mu\theta - \theta^2 + (\log(1 + \exp(v)))^2}{(1 + \exp(v))(\log(1 + \exp(v)))^3}. \end{aligned} \tag{3.11}$$

To calculate the *gradient of the log likelihood* (3.6) the simple definition for the derivative can be used because θ is a single value variable.

The *partial derivatives of reparameterisation function* g (2.61) is dependent on $\lambda = (\mu, \sigma)$ so the softplus transformation (3.9) is used for v,

$$\frac{\partial}{\partial \lambda} g(\lambda, \xi) = \frac{\partial}{\partial \lambda} [\mu + \log(1 + \exp(v))\xi]. \tag{3.12}$$

The different derivatives are

$$\begin{aligned} \frac{\partial}{\partial \mu} g(\lambda, \xi) &= 1, \\ \frac{\partial}{\partial v} g(\lambda, \xi) &= \frac{\exp(v)}{1 + \exp(v)} \xi. \end{aligned} \tag{3.13}$$

For simplicity the case where $g : \mathbb{R}^1 \to \mathbb{R}^1$ is used.

3.4 Initialisation of the regression model and the neural network

The variational parameter θ is sampled from the normal distribution $\mathcal{N}(\mu, \sigma^2)$ parameterised by $\lambda = (\mu, \sigma)$. λ is initialised using the prior distribution $\mathcal{N}(\mu_\pi, \sigma_\pi^2)$. Note that this will lead to a KL-divergence close to zero in the first iteration, but θ will change as the training updates λ, and the parameters of the prior distribution remain the same.

As addressed earlier the parameter v is used instead of σ. The inverse softplus (3.9) function transforms σ in the initialisation so $\lambda = (\mu, v)$ can be used in training,

$$v = \log(\exp(\sigma) - 1). \tag{3.14}$$

For the *regression model* (2.18), the mean $\mu = 0$ and the standard deviation $\sigma = 1$ is used.

For the *neural network* (2.74), all the weights and biases have a probability distribution, meaning θ is the shape of the number of unknown parameters. The μ and σ of the weights and biases have different initialisations.

The mean of both weights and bias is initialised to zero, $\mu_w = \mu_b = 0$.

The standard deviation of the weights is initialised using *He initialisation* [12]. He initialisation depends on the size of the previous layer

$$\sigma_w = \mathcal{N}(0, 1) \cdot \sqrt{\frac{2}{\text{fan\_in}}}, \tag{3.15}$$

where "fan_in" is the number of input nodes to the weight matrix.

The standard deviation of the bias is set to a small number to avoid large output variances when the network is deep, so $\sigma_b = 0.01$. This is a hyperparameter that can be changed to see which value works best.

Note that the weights and the biases of the network is not initialised to zero even though the mean value are because of the values of the standard deviation.

The structure of the feedforward neural network consists of two hidden layers with 100 nodes in each layer. This is hyperparameters for the architecture of the network that are set to reasonable values but could be changed. The input layer has 7 variables and output has 1 variable. This is because of the structure of the dataset explained in section 3.5. The total amount of parameters in the model is $m = 22004$. The batch size is set to 100. Adam is the optimiser used with a learning rate set to $\alpha = 0.001$.

The number of Monte Carlo samples S of θ from the variational density $q(\theta \mid \lambda)$ is $S = 3$ for the neural network. In the results section 4.1 where the black box and reparameterisation trick is compared both $S = 3$ and $S = 10$ is used. It is shown that using a large number of samples is computationally heavier without achieving a significant improvement in convergence, so therefore a smaller amount of sampling is used for the neural network.

The ELBO is the objective function to be maximised and the training is done when the ELBO *converges*. The stopping criteria when training the model is either a maximum number of epochs run or that the ELBO has converged. The maximum number of epochs is set to a million. The ELBO is tested for convergence if the training has not improved the last number of iterations. The accepted number of iterations without improvement is set to 100. An averaging window called the

moving average is moved across the ELBO to smooth the result. The window size used is 100.

The model is tested after training using the test dataset. When estimating the prediction, mean and variance are sampled 1000 times to get average estimations. This gives a mean predicted value and a standard deviation that can be plotted

Implementation of the algorithm is done in Python where functions from PyTorch are used for the neural network.

3.5 Datasets

Two different datasets are used for modelling; one simple dataset generated for the regression model and Solution Seeker's flow rate dataset for the neural network.

The black box and the reparameterisation trick are compared for a simple regression model (2.16). A *regression dataset* is created using

$$y = f(x) + \sigma_{data} \cdot \mathcal{N}(0, 1), \quad \text{with} \quad f(x) = \sin(\theta_{data} x), \tag{3.16}$$

where the variational parameter to be learned is $\theta_{data} = 2.7$, the noise $\sigma_{data} = 0.3$, and x is in the range between 0 and 1. Number of data points used is $n = 1000$. See figure 3.3.

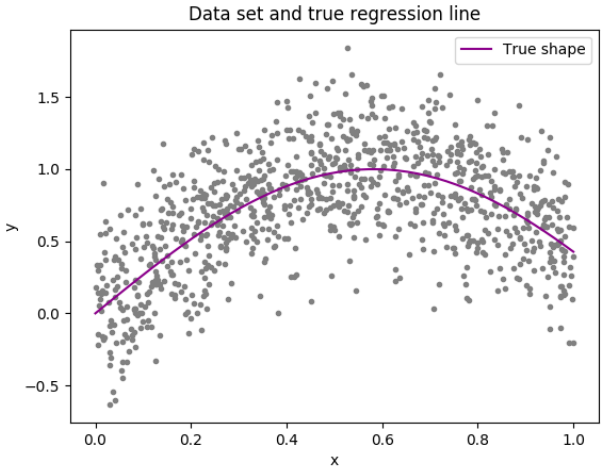

Figure 3.3: Dataset using y (3.16) where $\theta_{data} = 2.7$, $\sigma_{data} = 0.3$ and number of data points $n = 1000$.

Solution Seeker's *flow rate dataset* for the neural network consists of several variables measured by sensors during the extraction of liquid (meaning the oil, gas and water). The seven variables used as *input* when training the model is described below:

- CHK - This is the opening of a choke where all the liquid coming upstream pass. Unit in %.

- PWH - pressure measured at the well-head. Unit in bar.
- TWH - temperature measured at the well-head. Unit in ℃.
- PDC - pressure measured downstream the choke. Unit in bar.
- TDC - temperature measured downstream the choke. Unit in ℃.
- GOR - gas per oil. Value calculated from measured QGAS and QOIL which is volumetric gas and oil flow rate both measured in Sm^3/h, cubic meters per hour measured under standard conditions. GOR = $\frac{\text{QGAS}}{\text{QOIL}}$.
- WCT - water per water and oil. Value calculated from measured QWAT and QOIL which is volumetric water and oil flow rate both measured in Sm^3/h, cubic meters per hour measured under standard conditions. WCT = $\frac{\text{QWAT}}{\text{QWAT+QOIL}}$.

The *output* variable is QTOT which is the total quantity of measured oil, gas and water volumetric flow rate. QTOT = QOIL + QGAS + QWAT.

Time is also a variable saved in the dataset but is only used when plotting the result of the predictions, not in training. Only the start time of a steady-state period is used.

There are several other variables in the dataset that are not used as they do not contribute with relevant enough information beyond the variables described above.

Preprocessing of the dataset is needed. Preprocessing is important when predicting using real-life data as the datasets are rarely in good enough condition from the beginning. Error sources such as missing data points or inaccurate measurements need to be eliminated. First, the variables of GOR, WCT and QTOT need to be created from QOIL, QGAS and QWAT as they are not measured but calculated. All the columns with no measurements on some of the variables, also called NaNs, are removed. If a value is infinitely large it is also removed which could happen when GOR and WCT are created if QOIL or QWAT plus QOIL is equal to zero. The choke value CHK must be in between the value of 0 and 100 as this is measured in percentage. If the choke value is below 0, it is set to 0, and if it is above 100, it is set to 100. The flow rate measurements of oil, gas and water, QOIL, QGAS, QWAT, are set to 0 if they are negative as there can not be a negative flow. If PDC > PWH the measurement is considered inaccurate and removed from the dataset since the well-head pressure must always be larger than the choke pressure.

The dataset is split into a training and testing dataset. The training set contains 1998 data points from the period between January 2014 and January 2017. The test set has 639 data points from the period between January 2017 and January 2018. The model is developed using the training dataset and then tested on the test dataset.

Due to poor quality data from some wells they lack many datapoints after preprocessing. In general, the data from these wells are eliminated and instead wells with interesting and more complete datasets are used for modelling.

4

Results

4.1 Black box compared to reparameterisation trick using regression

The Bayesian regression model (2.18) is used on the dataset generated by the sine function (3.16). The model is used to compare the black box VI (2.58) and VI using the reparameterisation trick (2.69).

The variational parameter to be learned is $\theta = 2.7$ because this determines the shape of the dataset. The algorithm is run a maximum of $T = 5000$ iterations. Even though the ELBO may not always have converged at this time a tendency of how well the method have learned the true parameter θ is visible. Two parameters are changed when testing to clarify the difference between the two methods:

- ρ_0 used in the step size (2.53), $\rho = \rho_0/\sqrt{t}, \quad t = 1, .., T$ and T is the maximum number of iterations.

- S which is the number of samples of θ from the variational density $q(\theta \mid \lambda)$.

In figures 4.1 - 4.4 the black box and reparameterisation trick is trained for different values of ρ_0 and S. The number of samples used is $S = 3$ and $S = 10$ and the step sizes is $\rho_0 = 0.001$ and $\rho_0 = 0.0001$. In the figures the true value of $\theta = 2.7$ is plotted and the posterior mean value μ should be as close to this as possible. The posterior standard deviation σ is also included. Frequentist inference is used so there is no true σ to be learned. The posterior value for σ should instead be roughly equal to the difference between the true parameter θ and the estimated posterior μ. The value of σ will tell about how far off the estimate is.

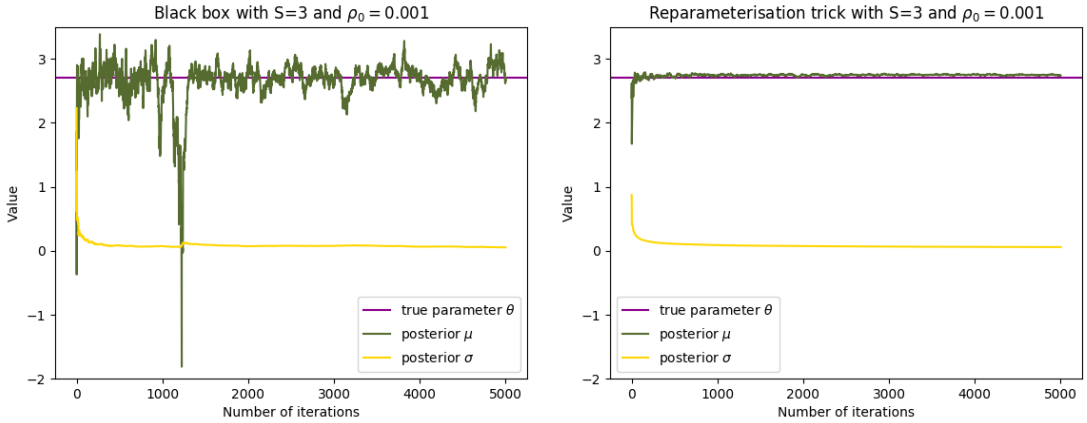

Figure 4.1: The black box and the reparameterisation trick with $S = 3$ and $\rho_0 = 0.001$.

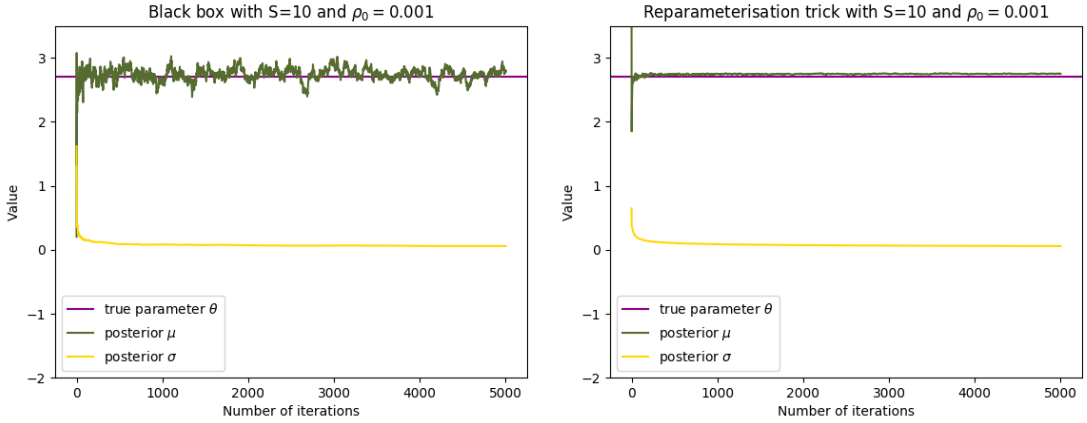

Figure 4.2: The black box and the reparameterisation trick with $S = 10$ and $\rho_0 = 0.001$.

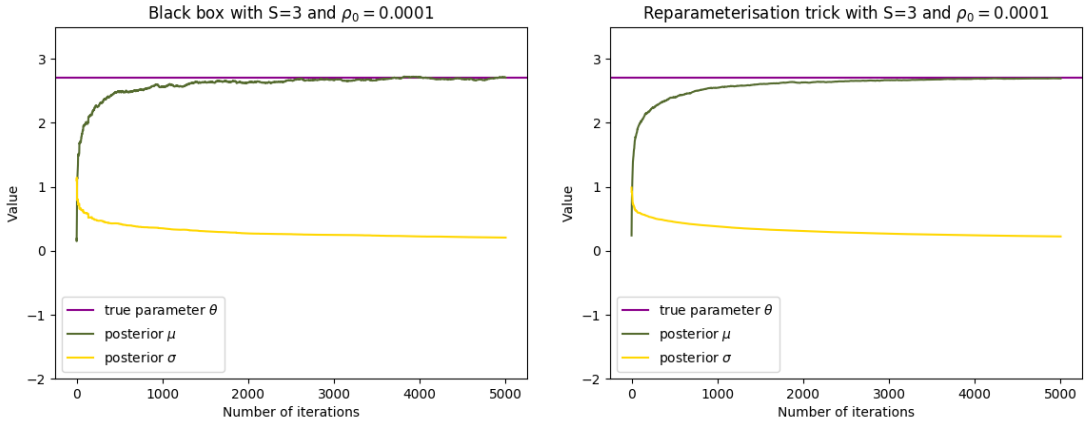

Figure 4.3: The black box and the reparameterisation trick with $S = 3$ and $\rho_0 = 0.0001$.

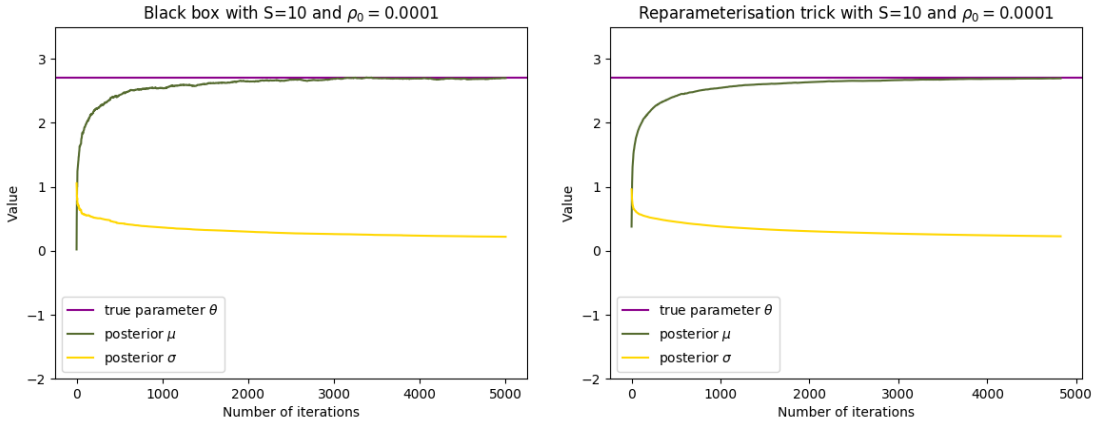

Figure 4.4: The black box and the reparameterisation trick with $S = 10$ and $\rho_0 = 0.0001$.

For $S = 3$ and $\rho_0 = 0.001$ in figure 4.1, the black box method is performing worse than the reparameterisation trick in predicting θ. The posterior mean μ is noisy and not converging for the black box. The reparameterisation trick seems to handle the small step length combined with few samples and converges almost to θ. The black box method improves when sampling more times $S = 10$ in figure 4.2 and the same step size, but the predicted μ is still noisy. The best improvement for the black box method is when smaller initial step size is used $\rho_0 = 0.0001$ in figures 4.3 and 4.4. Here the mean converges to the true value for both $S = 3$ and $S = 10$. The reparameterisation trick converges in all cases and is much more stable.

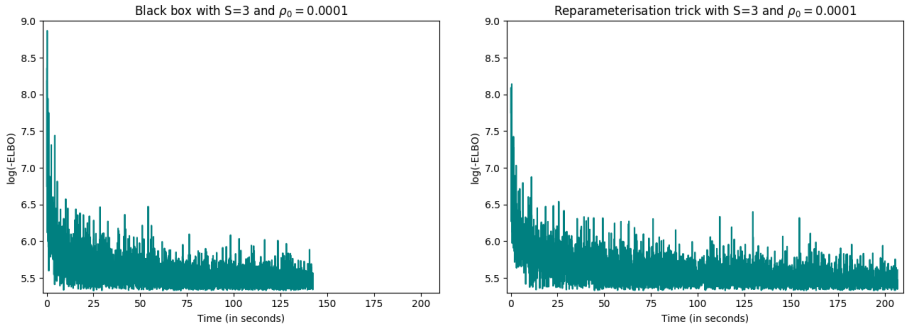

Figure 4.5: ELBO versus time for the black box left and the reparameterisation trick right for $S = 3$ and $\rho_0 = 0.0001$. $\log(-\text{ELBO})$ is used to scale the y-axis. Reparameterisation have a longer computational time.

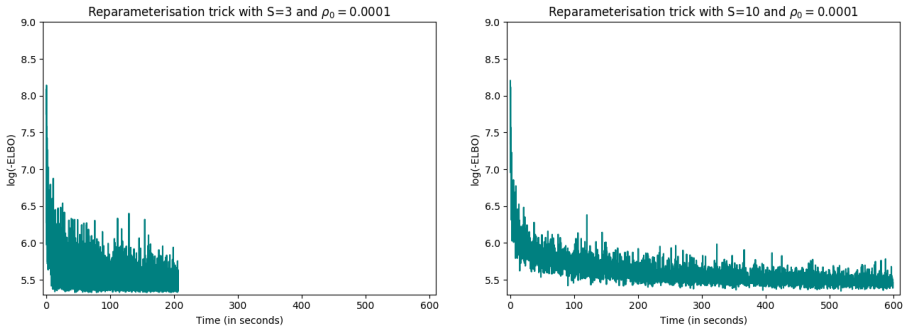

Figure 4.6: ELBO versus time using the reparameterisation trick for $\rho_0 = 0.0001$. $S = 3$ left and $S = 10$ right. $\log(-\text{ELBO})$ is used to scale the y-axis. $S = 10$ have a longer computational time.

Comparing the computational time for the black box and the reparameterisation trick for $S = 3$ and $\rho_0 = 0.0001$ in figure 4.5, the black box is more efficient than the reparameterisation trick. The computational time is probably because the gradient in the black box method is only involving the variational distribution. For the reparameterisation trick, the gradient is taken of both the likelihood and variational distribution, and this is computationally heavier. When comparing the computational time between the number of samples in figure 4.6, it is a larger difference when comparing the number of samples rather than methods. Seeing the stability of the reparameterisation trick, it makes more sense using this model with a smaller amount of samples S.

4.2 Neural network using reparameterisation trick on flow rate dataset

The results from the last section 4.1 indicate the reparameterisation trick for variational inference is a better method to use. For modelling the flow rate using a Bayesian feedforward neural network (2.74), the reparameterisation trick (2.69) is used. Solution Seeker's dataset of production data explained in section 3.5 is used when modelling. The specifics of the neural network is explained in section 3.4 about initialisation.

In figures 4.7 - 4.14 are the results of training the neural network using data from different wells. The testing data is plotted simulating the total flow rate $QTOT$ over time. The real data is the red line. The predicted mean value μ is the black line and the grey area around is the credible interval using the standard deviation $\pm 2\sigma$. There is a spread in how well the models perform. The quality of the data sets varies as the preprocessing remove datapoints. Wells with interesting results are presented here and for all predictions see appendix A.3.

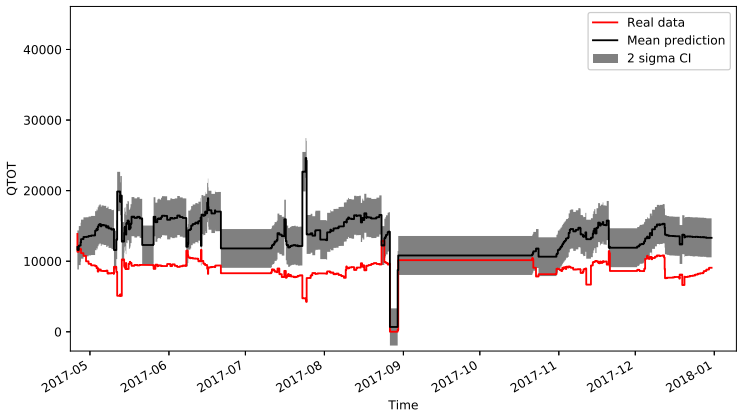

Figure 4.7: Result of modelling well 1 using test data. QTOT versus time

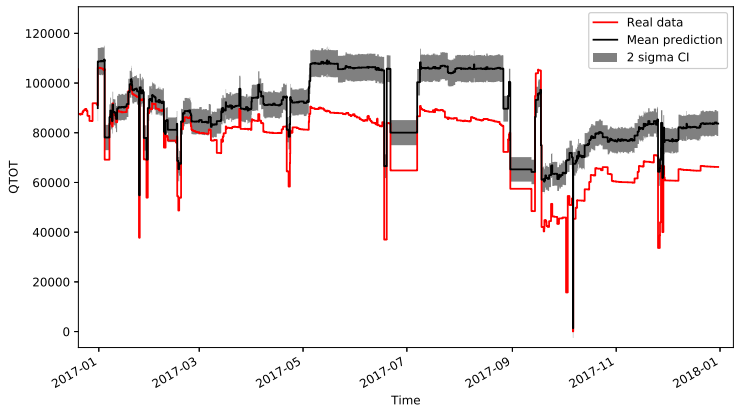

Figure 4.8: Result of modelling well 2 using test data. QTOT versus time

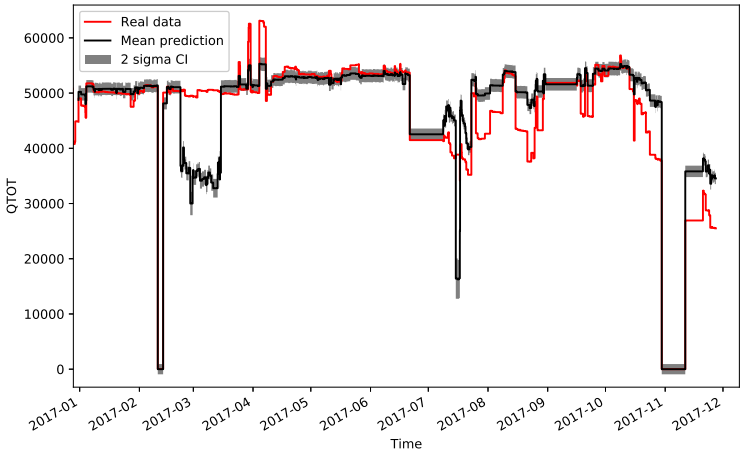

Figure 4.9: Result of modelling well 3 using test data. QTOT versus time

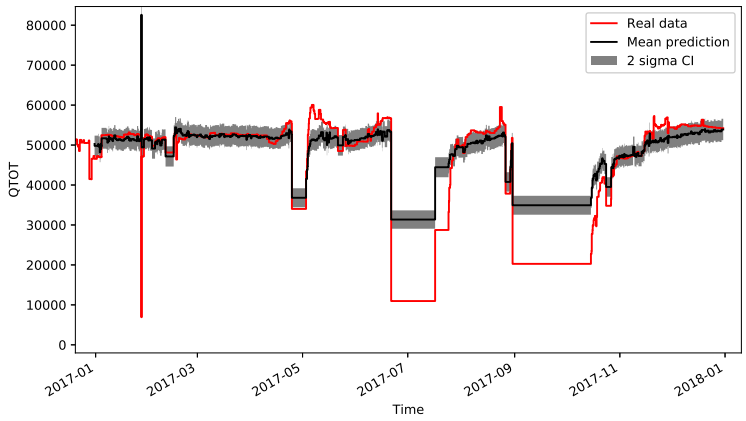

Figure 4.10: Result of modelling well 4 using test data. QTOT versus time

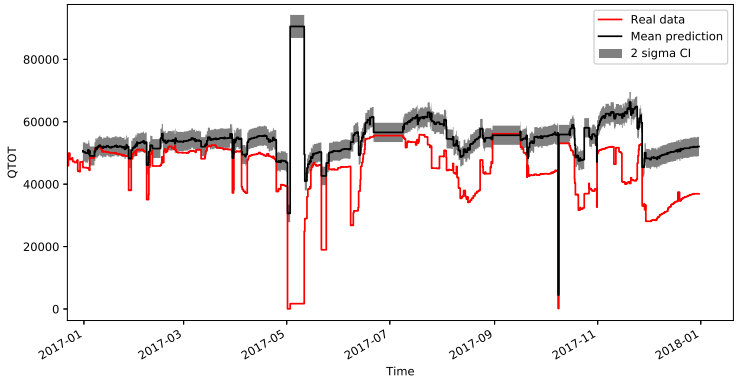

Figure 4.11: Result of modelling well 6 using test data. QTOT versus time.

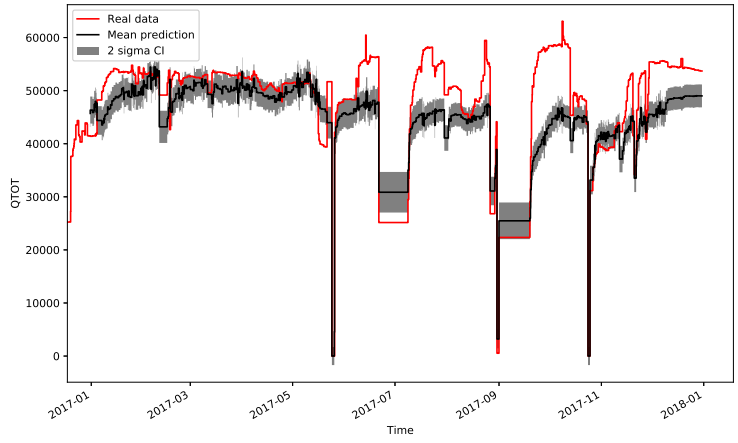

Figure 4.12: Result of modelling well 7 using test data. QTOT versus time.

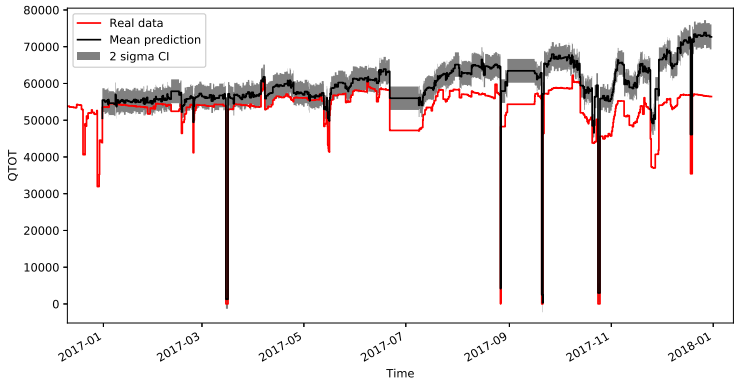

Figure 4.13: Result of modelling well 8 using test data. QTOT versus time.

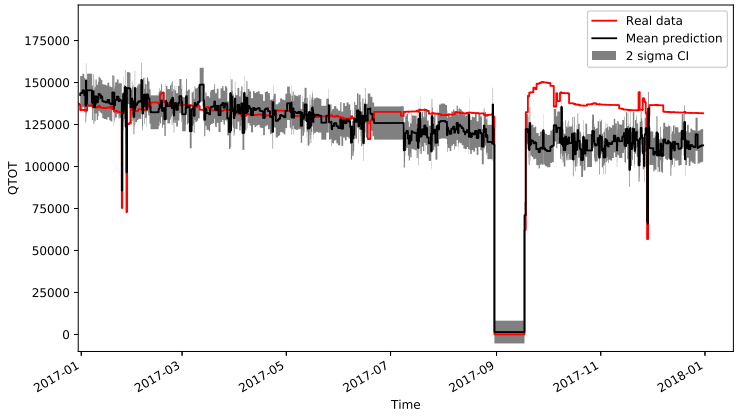

Figure 4.14: Result of modelling well 10 using test data. QTOT versus time

For well 2 in figure 4.8 the predicted mean are following the real data but then after a time overpredicting while still following the shape of the flow rate. The same tendency can be found for well 3 in figure 4.9, well 6 in figure 4.11, well 8 in figure 4.13 and well 10 in figure 4.14. In figure 4.12 there is instead an underprediction of the flow rate after some time while still following the shape. The result of modelling well 4 in figure 4.10 shows a prediction matching quite well but missing some of the outliers. The prediction of the flow rate for well 1 in figure 4.7 is never quite matching. This is despite the well having many datapoints, see table 4.1. Well 2, 7 and 10 have fewer datapoints but managing to match the shape of the real data. Having fewer datapoints might instead affect the computation time, see appendix A.3, where the ELBO is plotted versus the number of epochs. There is a difference in the credible intervals for the different predictions. The error between the predicted and the real result are not always covered.

Table 4.1: Training dataset sizes for each well. The wells had 1998 datapoints before preprocessing.

Well	1	2	3	4	6	7	8	10
Training dataset size	1109	532	1268	1318	1311	659	1301	590

For well 4 in figure 4.10 and well 6 in figure 4.11, the model is wrongly predicting when the flow rate is suddenly low and instead of predicting the flow rate to be high. This does not seem to be a problem for the other wells. A probable explanation could be that the training data does not consist of scenarios like this so the model has not "seen" this behaviour before. The training data for well 4 and 6 are plotted in figures 4.15 and 4.16. These kinds of sudden drops in the flow rate is indeed a part of the training data.

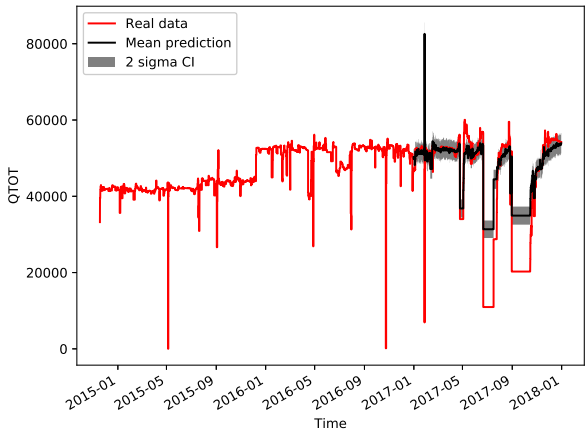

Figure 4.15: Training and test data for well 4.

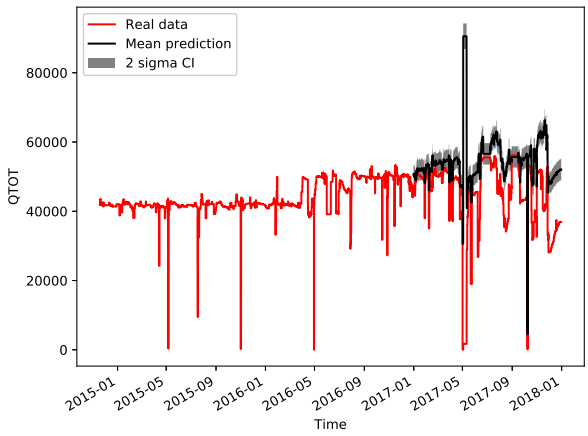

Figure 4.16: Training and test data for well 6.

5

Discussion

Based on section 4.1, variational inference using the reparameterisation trick yielded a more stable convergence than the black box variational inference. The reparameterisation trick was used when modelling using a feedforward neural network. The training of the neural network consisted of a forward pass and a backward pass. In the forward pass, a stochastic sample from the variational distribution was used to evaluate the ELBO with the aim of maximising it. The variational and prior distribution were evaluated layer-wise while the likelihood, being data-dependent, was evaluated at the end of the forward pass. In the backward pass, the gradient of λ was computed and backpropagated through the network. This was done by an optimiser so that the values were updated continuously. For the backpropagation to work, because of the stochastic sampling, the reparameterisation trick needs to be used. The reparameterisation trick conduct the sampling from a parameter-free distribution (the standard normal distribution) and transforms the samples to a deterministic function which the gradient can be taken of in the backpropagation.

The mean-field approximation (3.2) for the variational family is used, giving a diagonal covariance matrix and independent variational parameters. Having a diagonal covariance matrix makes computations easier. Another Gaussian family with non-diagonal covariance matrix could be used with the interactions of the weights dependent (they are assumed independent with the mean-field). It is not obvious how the correlation between the weights work. Neural networks are often described as ”black boxes” meaning it is hard to get the insight on how they work; what do the weights learn and why do they emphasise certain features. Picking a variational family that works with this is not obvious. One could be used where all weights in one layer are correlated, for example. As a first try, the mean-field is used, but other Gaussian approximations are feasible, but it is not clear which one that should be used.

When modelling using the flow rate dataset in section 4.2 note that the predictions over time diverge more and more. It is a worsening in the testing the further away from the training dataset the predictions are. The mean value follows the shape of the flow rate but is missing by a factor that grows over time. This can be because the wells changes over time. The training and testing data are split according to time with the training dataset from January 2014 to January 2017 and the test dataset from January 2017 to January 2018. The development of a production field changes over time as more oil and gas are being extracted. The model is on the other hand

not dependent on time as a variable and this can be the source of the error that grows. Improved modelling of the noise in the model could make the predictions more accurate as well.

The prior distribution can be hard to determine. Because the prior has set values for mean and standard deviation affecting the training of the model throughout, these must be as accurate as possible. On the other hand is it hard to guess a prior distribution. The structure of the variational inference method is that information is extracted from the data when training and subtracted with the prior times the variational distribution. If the prior is far off, this will result in poor results. This can be seen in the results in section 4.2. When the mean is not predicted correctly, the standard deviation is not on the same scale as the error between the predicted and the real result. The standard deviation should cover the actual result. The model is probably poorly calibrated because of a bad prior. As the weights of a neural network do not have an interpretation being hidden in a ”black box”, the prior distribution to weights are hard to set. A KL-factor multiplied with the variational times the prior term could reduce the effect of the prior. Another idea is that a network could be pre-trained on a dataset. The resulted posterior distribution could then be used as the the prior distribution to the same network when new data is achieved. Due to limitation in time this has not been tested.

It should be noted that variational inference might not be needed on the dataset provided here because it is not that complex or large. Methods like the computationally expensive Markov chain Monte Carlo (MCMC) that also provide a posterior distribution could be used. In future modelling for Solution Seeker, the models will be more complex and the dataset larger, motivating the desire to explore variational inference.

Kullback-Leibler divergence is one way of performing variational inference, but it does not necessarily have to be used. Other optimisation methods approximating the inference can be used such expectation propagation, belief propagation or Laplace approximation.

Bibliography

[1] David M Blei, Alp Kucukelbir and Jon D McAuliffe. 'Variational inference: A review for statisticians'. In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877.

[2] Alex Graves. 'Practical variational inference for neural networks'. In: *Advances in neural information processing systems*. 2011, pp. 2348–2356.

[3] Rajesh Ranganath, Sean Gerrish and David M Blei. 'Black Box Variational Inference.' In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. 2014.

[4] Diederik P Kingma and Max Welling. 'Auto-encoding Variational Bayes'. In: *arXiv preprint arXiv:1312.6114* (2013).

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

[6] *Bayesian statistics for data science*. URL: `https://towardsdatascience.com/bayesian-statistics-for-data-science-45397ec79c94` (visited on 01/11/2019).

[7] Léon Bottou, Frank E Curtis and Jorge Nocedal. 'Optimization methods for large-scale machine learning'. In: *Siam Review* 60.2 (2018), pp. 223–311.

[8] Charles Blundell et al. 'Weight uncertainty in neural networks'. In: *arXiv preprint arXiv:1505.05424* (2015).

[9] Chetan Warke. *Simple Feed Forward Neural Network code for digital Handwritten digit recognition*. 2019. URL: `https://medium.com/random-techpark/simple-feed-forward-neural-network-code-for-digital-handwritten-digit-recognition-a234955103d4` (visited on 11/08/2020).

[10] Diederik P Kingma and Jimmy Ba. 'Adam: A method for stochastic optimization'. In: *arXiv preprint arXiv:1412.6980* (2014).

[11] Jason Brownlee. *Difference Between a Batch and an Epoch in a Neural Network*. 2018. URL: `https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/` (visited on 07/08/2020).

[12] Kaiming He et al. 'Delving deep into rectifiers: Surpassing human-level performance on imagenet classification'. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

A

Appendix 1

A.1 Conjugate prior

When the likelihood and prior are in the same family of distributions the *conjugate prior* can be used to find the posterior hyperparameters. In Bayesian statistics the hyperparameters are the parameters that belong to the prior distribution and distinguished from the parameters of the model for the underlying distribution (likelihood). Consider the problem of extracting a distribution from the parameter θ given some data x. Because of Bayes' rule (2.11) the posterior hyperparameters can be found from

$$p(\theta \mid x) = \frac{p(x \mid \theta)p(\theta)}{p(x)} \propto p(x \mid \theta)p(\theta) \tag{A.1}$$

When the likelihood is normally distributed with known variance σ_x^2, the prior is normally distributed with hyperparameters μ_0 and σ_0^2 and the model parameter is μ, the posterior is also normally distributed as

$$N\left(\frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma_x^2}}\left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^n x_i}{\sigma_x^2}\right), \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma_x^2}\right)^{-1}\right). \tag{A.2}$$

A.2 Example of variational inference

Following is the example of variational inference from section 2.4.1 with all steps.

Consider a sample of random variables that are all normally distributed with a mean μ and variance σ_x^2 so $X = x_1, ..., x_n \sim N(\mu, \sigma_x^2)$. The parameter mean μ in the parameter space M of the distribution for this model $\Pi(\cdot \mid X)$ with density $\pi(\mu \mid x)$ is to be found using variational inference (2.36). For this model it is assumed a simple prior distribution $\Pi = N(0, \sigma_0^2)$ with density $\pi(\mu)$ and two variational distributions $Q_1 = N(0, \sigma^2)$ with density $q_1(\mu)$ and $Q_2 = N(1, \sigma^2)$ with density $q_2(\mu)$. To simplify the calculations assume that $\sigma_x = \sigma_0 = \sigma$. Using the probability density function for the normal distribution (2.13) the different densities for the likelihood, prior and

the two different variational densities are found: the likelihood,

$$\begin{aligned} p(X \mid \mu) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right) \\ &= (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right), \end{aligned} \tag{A.3}$$

the prior,

$$\pi(\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right), \tag{A.4}$$

the first variational

$$q_1(\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right), \tag{A.5}$$

the second variational,

$$q_2(\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mu-1)^2}{2\sigma^2}\right). \tag{A.6}$$

To find the best distribution for approximating the posterior distribution, the KL divergence from the posterior distribution $\Pi(\cdot \mid X)$ to the variational distributions Q_1 and Q_2, are to be minimised. This is done by instead maximising the evidence lower bound (ELBO) (2.38). To simplify the integration of the likelihood times the prior the distribution of the two combined can be used instead. Using conjugate prior the hyperparameters of the posterior can be found from (A.2),

$$\begin{aligned} p(X \mid \mu)\pi(\mu) &\propto N\left(\frac{1}{\frac{1}{\sigma^2}+\frac{n}{\sigma^2}}\left(0+\frac{\sum_{i=1}^{n} x_i}{\sigma^2}\right), \left(\frac{1}{\sigma^2}+\frac{n}{\sigma^2}\right)^{-1}\right) \\ &= N\left(\frac{\sum_{i=1}^{n} x_i}{n+1}, \frac{\sigma^2}{n+1}\right). \end{aligned} \tag{A.7}$$

The ELBO is calculated for each variational density at a time and the result compared. First for $q_1(\mu)$,

$$\begin{aligned} \mathcal{L}(q_1) &= \int q_1(\mu) \log\left(\frac{p(X \mid \mu)\pi(\mu)}{q_1(\mu)}\right) d\mu \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) \log \frac{\frac{1}{\sqrt{\frac{2\pi\sigma^2}{n+1}}} \exp\left(-\frac{-\left(\mu-\frac{\sum_{i=1}^{n} x_i}{n+1}\right)^2}{\frac{2\sigma^2}{n+1}}\right)}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right)} d\mu \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) \left(-\frac{1}{2}\log\frac{2\pi\sigma^2}{n+1} - \frac{(n+1)\left(\mu-\frac{\sum_{i=1}^{n} x_i}{n+1}\right)^2}{2\sigma^2}\right. \\ &\quad \left. + \frac{1}{2}\log 2\pi\sigma^2 + \frac{\mu^2}{2\sigma^2}\right) d\mu. \end{aligned} \tag{A.8}$$

The Gaussian integrals are needed for the integration,

$$\begin{aligned}
&\int_{-\infty}^{\infty} \exp\left(-\frac{x^2}{a}\right) dx = \sqrt{a\pi}, \\
&\int_{-\infty}^{\infty} \exp\left(-\frac{x^2}{a}\right) x dx = 0, \\
&\int_{-\infty}^{\infty} \exp\left(-\frac{x^2}{a}\right) x^2 dx = \frac{a\sqrt{a\pi}}{2}, \\
&\int_{-\infty}^{\infty} \exp\left(-\frac{(x-b)^2}{a}\right) dx = \sqrt{a\pi}, \\
&\int_{-\infty}^{\infty} \exp\left(-\frac{(x-b)^2}{a}\right) x dx = b\sqrt{a\pi}, \\
&\int_{-\infty}^{\infty} \exp\left(-\frac{(x-b)^2}{a}\right) x^2 dx = \frac{(2b^2+a)\sqrt{a\pi}}{2}.
\end{aligned} \tag{A.9}$$

Expanding the brackets and applying the Gaussian integrals implies,

$$\begin{aligned}
\mathcal{L}(q_1) &= \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \frac{1}{2}\log(n+1)\sqrt{2\pi\sigma^2} \\
&\quad + \frac{1}{2\sigma^2\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) \left(-(n+1)\left(\mu^2 - \frac{2\mu\sum_{i=1}^n x_i}{n+1} + \left(\frac{\sum_{i=1}^n x_i}{n+1}\right)^2\right) + \mu^2\right) d\mu \\
&= \frac{1}{2}\log(n+1) - \frac{n+1}{2\sigma^2\sqrt{2\pi\sigma^2}} \left(\frac{2\sigma^2\sqrt{2\pi\sigma^2}}{2} - 0 + \left(\frac{\sum_{i=1}^n x_i}{n+1}\right)^2 \cdot \sqrt{2\pi\sigma^2}\right) \\
&\quad + \frac{1}{2\sigma^2\sqrt{2\pi\sigma^2}} \frac{2\sigma^2\sqrt{2\pi\sigma^2}}{2} \\
&= \frac{1}{2}\log(n+1) - \frac{1}{2\sigma^2}\left((n+1)\sigma^2 + \frac{1}{n+1}\left(\sum_{i=1}^n x_i\right)^2 - \sigma^2\right).
\end{aligned} \tag{A.10}$$

For $q_2(\mu)$ the ELBO is,

$$\begin{aligned}
\mathcal{L}(q_2) &= \int q_2(\mu) \log\left(\frac{p(X \mid \mu)\pi(\mu)}{q_2(\mu)}\right) d\mu \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mu-1)^2}{2\sigma^2}\right) \log \frac{\frac{1}{\sqrt{\frac{2\pi\sigma^2}{n+1}}} \exp\left(-\frac{-\left(\mu - \frac{\sum_{i=1}^n x_i}{n+1}\right)^2}{\frac{2\sigma^2}{n+1}}\right)}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mu-1)^2}{2\sigma^2}\right)} d\mu \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{(\mu-1)^2}{2\sigma^2}\right) \left(-\frac{1}{2}\log\frac{2\pi\sigma^2}{n+1} - \frac{(n+1)\left(\mu - \frac{\sum_{i=1}^n x_i}{n+1}\right)^2}{2\sigma^2}\right. \\
&\quad \left. + \frac{1}{2}\log 2\pi\sigma^2 + \frac{(\mu-1)^2}{2\sigma^2}\right) d\mu
\end{aligned} \tag{A.11}$$

Expanding the brackets and applying the Gaussian integrals (A.9) implies,

$$\begin{aligned}\mathcal{L}(q_2) = &\frac{1}{\sqrt{2\pi\sigma^2}} \cdot \frac{1}{2}\log(n+1)\sqrt{2\pi\sigma^2} \\ &+ \frac{1}{2\sigma^2\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{(\mu-1)^2}{2\sigma^2}\right)\left(-(n+1)\left(\mu^2 - \frac{2\mu\sum_{i=1}^n x_i}{n+1} + \left(\frac{\sum_{i=1}^n x_i}{n+1}\right)^2\right)\right. \\ &\left. + (\mu^2 - 2\mu + 1)\right) d\mu \\ = &\frac{1}{2}\log(n+1) - \frac{n+1}{2\sigma^2\sqrt{2\pi\sigma^2}}\left(\frac{(2+2\sigma^2)\sqrt{2\pi\sigma^2}}{2} - \frac{2\sqrt{2\pi\sigma^2}\sum_{i=1}^n x_i}{n+1} + \left(\frac{\sum_{i=1}^n x_i}{n+1}\right)^2 \cdot \sqrt{2\pi\sigma^2}\right) \\ &+ \frac{1}{2\sigma^2\sqrt{2\pi\sigma^2}}\left(\frac{(2+2\sigma^2)\sqrt{2\pi\sigma^2}}{2} - 2\sqrt{2\pi\sigma^2} + \sqrt{2\pi\sigma^2}\right) \\ = &\frac{1}{2}\log(n+1) - \frac{1}{2\sigma^2}\left((n+1)(1+\sigma^2) - 2\sum_{i=1}^n x_i + \frac{1}{n+1}\left(\sum_{i=1}^n x_i\right)^2 - \sigma^2\right).\end{aligned} \tag{A.12}$$

A.3 Neural network using reparameterisation trick on flow rate dataset

Modelling of the flow rate using a Bayesian feedforward neural network (2.74) and the reparameterisation trick (2.69). Solution Seeker's dataset of production data explained in section 3.5 is used when modelling. Predictions using all wells are presented here.

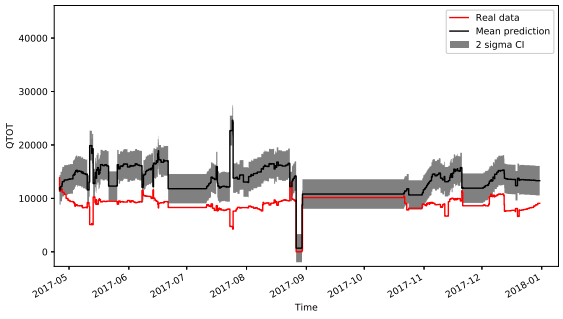

Figure A.1: Result of modelling well 1 using test data.

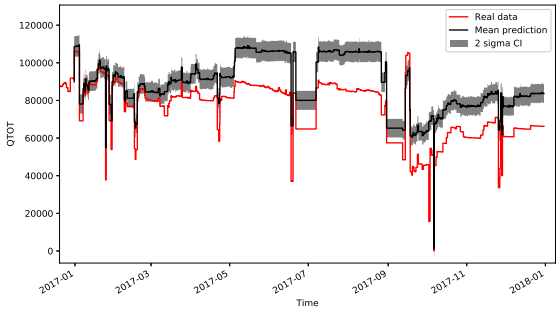

Figure A.2: Result of modelling well 2 using test data.

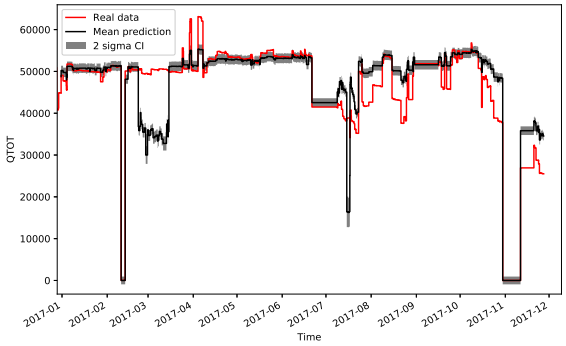

Figure A.3: Result of modelling well 3 using test data.

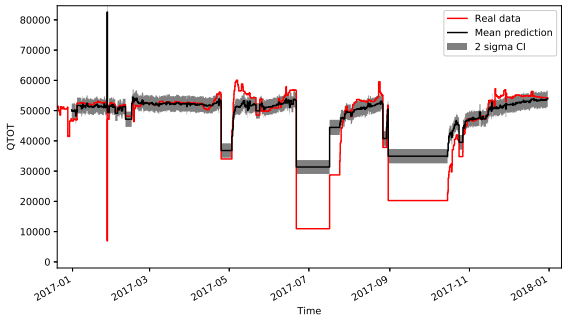

Figure A.4: Result of modelling well 4 using test data.

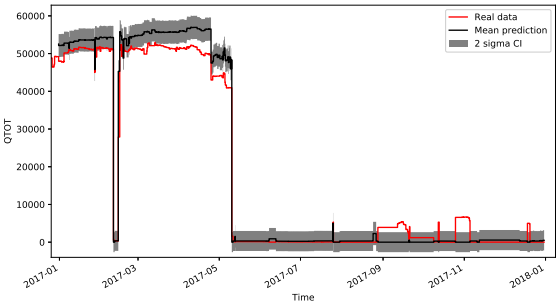

Figure A.5: Result of modelling well 5 using test data.

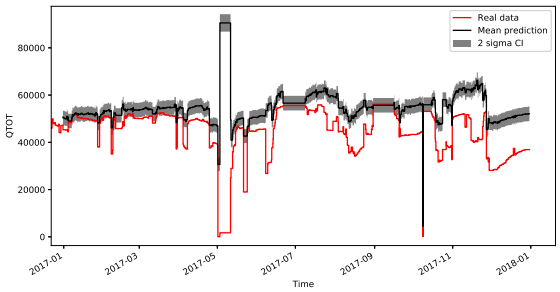

Figure A.6: Result of modelling well 6 using test data.

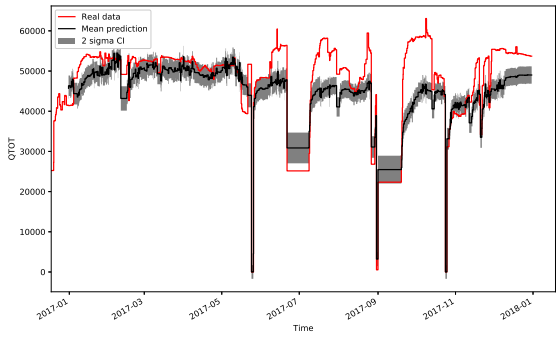

Figure A.7: Result of modelling well 7 using test data.

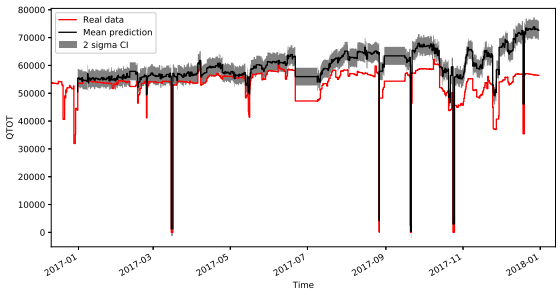

Figure A.8: Result of modelling well 8 using test data.

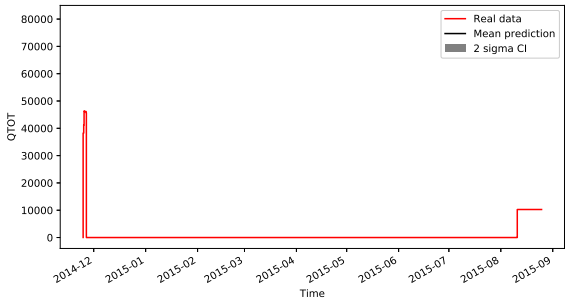

Figure A.9: Result of modelling well 9 using test data.

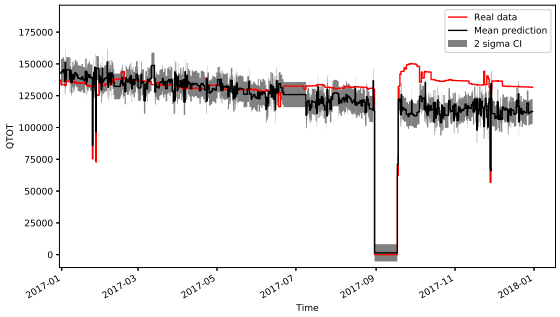

Figure A.10: Result of modelling well 10 using test data.

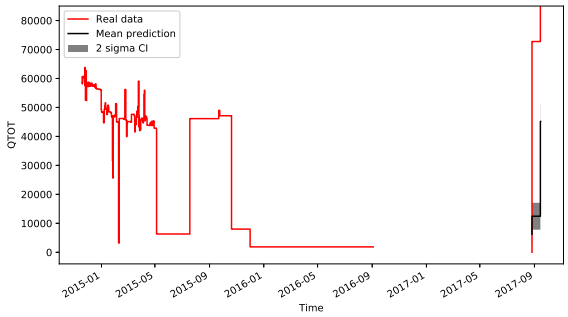

Figure A.11: Result of modelling well 11 using test data.

The convergence of the ELBO (2.38) for some of the different wells. log(−ELBO) is used to scale the y-axis.

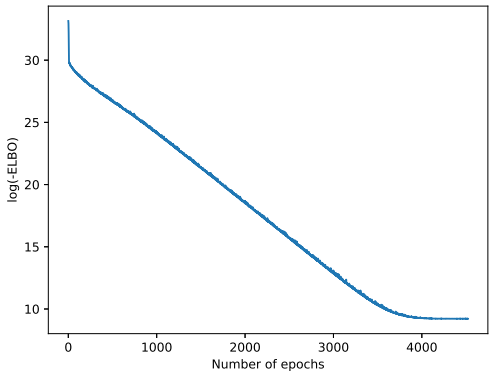

Figure A.12: ELBO versus number of epochs for well 1. log(−ELBO) is used to scale the y-axis.

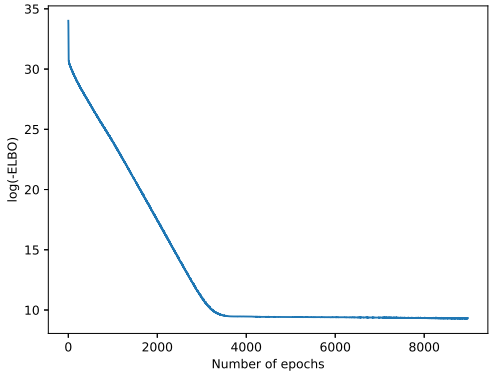

Figure A.13: ELBO versus number of epochs for well 4. log(−ELBO) is used to scale the y-axis.

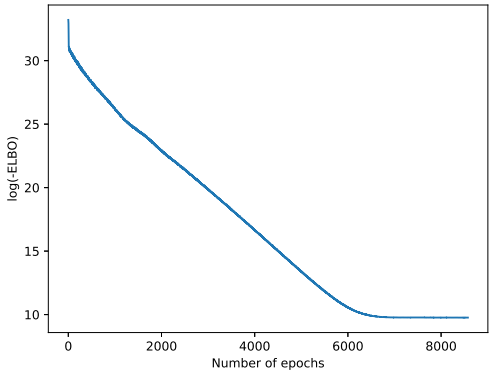

Figure A.14: ELBO versus number of epochs for well 7. log(−ELBO) is used to scale the y-axis.

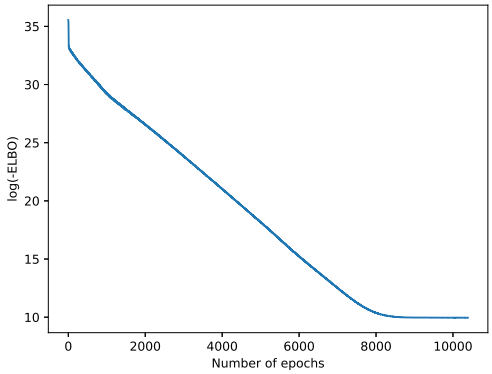

Figure A.15: ELBO versus number of epochs for well 10. log(−ELBO) is used to scale the y-axis.