DENSITY RATIO ESTIMATION IN VARIATIONAL BAYESIAN MACHINE LEARNING

Alexander Lam

Supervisors: Professor Scott Sisson and Doctor Edwin Bonilla

School of Mathematics and Statistics
UNSW Sydney

October 2018

Submitted in partial fulfillment of the requirements of the degree of
Bachelor of Science with Honours

Plagiarism statement

I declare that this thesis is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere.

I acknowledge that the assessor of this thesis may, for the purpose of assessing it:

- Reproduce it and provide a copy to another member of the University; and/or,
- Communicate a copy of it to a plagiarism checking service (which may then retain a copy of it on its database for the purpose of future plagiarism checking).

I certify that I have read and understood the University Rules in respect of Student Academic Misconduct, and am aware of any potential plagiarism penalties which may apply.

By signing this declaration I am agreeing to the statements and conditions above.

Signed: _____ Date: _____

Acknowledgements

By far my greatest thanks must go to my supervisors Scott Sisson and Edwin Bonilla for the care and support they provided throughout this year. I am very pleased that they have guided me through such a unique topic which spans the fields of applied mathematics, statistics and computer science. I would also like to thank Edwin's PhD student Louis Tiao for helping me with some crucial areas in this thesis.

My second greatest thanks goes to my senior high school mathematics teacher Trevor Trotman. His passion and excellent teaching was crucial to helping me discover my enjoyment of mathematics. If not for him, I would have likely taken a more dull path in life.

It is imperative that I express gratitude to the inhabitants of Red Center Room 3084 for accompanying and distracting me throughout this year. In particular, I would like to thank the senior inhabitants Prosha, Ramanan and Jeffrey for their constructive criticisms of my writing and presentation.

Shout-out to the Discord fellas for entertaining me and encouraging me to pursue a career as a professional DJ. Extra shout-out to Davey cause he asked for one. Also shout-out to Kong for letting me leech off his gym pass.

I would also like to acknowledge the help of Matthew for setting up my github, Sunchit for his Python tech support, and Allan for showing me that the Euler-Lagrange equation can be employed to find the second derivatives of my loss functionals.

저를 돌보고 나를 응원 해 준 한채린누나에게 감사 드리고 싶습니다. 올해 가장 행복한 순간 중 일부는 그녀와 함께 보냈습니다. 저는 그런 친절하고 예쁜 누나를 가지고있어서 운이 좋습니다.

Lastly, I would like to thank Jin, Eliza and Brian for encouraging me to be ambitious in all aspects of my future aspirations.

Lammy, 25 October 2018.

Abstract

In variational Bayesian machine learning, deep neural networks can be used to efficiently model posterior densities associated with high dimensional data or large datasets. This can be furthered to an autoencoder model which simultaneously expresses data points as a lower dimensional latent representation, and reconstructs said data given a latent variable input. Due to the implicit nature of the involved densities, density ratio estimation techniques are used to approximate the intractable KL divergence term in the posterior network optimisation problem.

In this thesis, we generalise the estimator loss function selection to a choice of f-divergence lower bound and estimator parametrisation. We then demonstrate that, of the tested parametrisations, the class probability estimator is the optimal choice as it is feasible with large density ratios and experiences the quickest convergence. Additionally, the high accuracy yet instability of the reverse KL divergence lower bound is experimentally shown, in contrast to the stable yet inaccurate GAN divergence, implying a trade-off between estimator stability and accuracy in the choice of f-divergence.

Contents

Chapter 1

Introduction

1.1 Problem Context

In any Bayesian statistics problem, the quintessential objective is to evaluate the posterior density $p(z|x)$ [Gelman et al., 2004]. In cases where algebraic posterior evaluation is difficult, Markov Chain Monte Carlo (MCMC) methods have been used to approximate the posterior density, but they tend to have slow convergence when applied to high dimensional data or large datasets [Blei et al., 2017]. Variational inference methods overcome this obstacle by approximating the posterior density with a different, optimized density parametrized by ϕ: $q_\phi(z)$, dubbed the 'variational density'. This can be achieved by minimising its reverse KL divergence with the true posterior density, or equivalently minimising an expression equal to the negative of the evidence lower bound, $NELBO(q)$ [Blei et al., 2017]:

$$\min_{\phi}\{\underbrace{-\mathbb{E}_{q_\phi(z)}[\log p(x|z)] + KL(q_\phi(z)\|p(z))}_{=NELBO(q)}\}.$$

In traditional 'mean-field variational inference', the variational density factorises over the latent variables, taking the form $q_\phi(z) = \prod_i q_{\phi_i}(z_i)$. This algorithm is inefficient in large datasets, as a new variational density would have to be trained for each data observation. The recent increased popularity of neural networks, brought forward by improvements in computational power, has led to a new type of variational inference, amortized variational inference [Zhang et al., 2017]. This method uses the universal function approximating capability of the neural network model to condition the variational posterior on the observation x, producing a flexible model that can be generalised over large datasets. To optimise this density $q_\phi(z|x)$, we minimize $NELBO(q)$ with respect to the density of the dataset $q^*(x)$:

$$\min_{\phi}\left\{\mathbb{E}_{q^*(x)}\left[-\mathbb{E}_{q_\phi(z|x)}[\log p(x|z)] + KL(q_\phi(z|x)\|p(z))\right]\right\}.$$

Typically, the variational posterior is explicitly parametrised as a multivariate Gaussian density with means and variances specified by the posterior network output [Kingma and Welling, 2013]. However, "Adversarial Variational Bayes" by Mescheder et al. [2017] introduces a more flexible model by adding additional random noise inputs to the posterior network, which is configured to output density samples z. We label this posterior as 'implicit' as it is difficult to numerically evaluate its explicit representation. Consequently, the optimisation problem is intractable as we are unable to calculate the KL divergence term:

$$KL(q_\phi(z|x)\|p(z)) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p(z)}\right].$$

We therefore resort to density ratio estimation techniques, using another neural network to estimate this term [Mohamed and Lakshminarayanan, 2016; Sugiyama et al., 2012]. There are two main methods of optimising this network: class probability estimation, which trains the network to distinguish between samples from the numerator and denominator densities [Goodfellow et al., 2014], and divergence minimisation, which formulates an estimator loss function with a global minimum equal to the reverse KL divergence $KL(q_\phi(z|x)\|p(z))$ [Nguyen et al., 2010].

Since density ratio estimation only requires samples from the densities, we are additionally able to use this algorithm when the prior $p(z)$ is implicit. Following the terminology in "Variational Inference using Implicit Distributions" by Huszár [2017], we refer to this as the 'prior-contrastive' formulation. When the likelihood density is additionally implicit, we use a 'joint-contrastive' formulation, minimizing the reverse KL divergence between the joint densities [Tran et al., 2017]:

$$\min_\phi \left\{\mathbb{E}_{q^*(x)q_\phi(z|x)}\left[KL(q(z,x)\|p(z,x))\right]\right\}.$$

The uses of amortized variational inference are not limited to posterior inference; it can additionally be used to train an autoencoder, a model consisting of an encoder that represents a data point x as a lower dimensional latent variable z according to the variational posterior $q_\phi(z|x)$, and a decoder that reconstructs data $\tilde{x}$ from latent z according to the likelihood $p_\theta(x|z)$ [Kingma and Welling, 2013].

1.2 Aims

In the context of variational Bayesian machine learning, the density ratio term $q(\cdot)/p(\cdot)$ is often intractable due to the implicit nature of one or more densities involved. By applying density ratio estimation methods, we are able to estimate this

term or a transformation of it by using a neural network. The primary goal of this thesis is to analyse and compare different loss functions used to train these estimator networks. We accomplish this by comparing the convergence of the variational posterior networks, reflective of the density ratio estimation accuracy. We begin by generalising the loss function formulation to two selections. The first selection is an estimator parametrisation representing the density ratio transformation:

- Class Probability Estimator: $D_\alpha(\cdot) \simeq q(\cdot)/(q(\cdot) + p(\cdot))$,
- Direct Ratio Estimator: $r_\alpha(\cdot) \simeq q(\cdot)/p(\cdot)$,
- Direct Log Ratio Estimator: $T_\alpha(\cdot) \simeq \log(q(\cdot)/p(\cdot))$,

where $\simeq$ denotes equality when the estimator function is optimal. The second selection is an f-divergence used to formulate the estimator loss function:

- Reverse KL Divergence: $KL(q(\cdot)\|p(\cdot))$,
- GAN Divergence: $2JS(p(\cdot)\|q(\cdot)) - \log 4$,

where $JS(\cdot\|\cdot)$ denotes the Jensen-Shannon divergence. We then compare all combinations in varying conditions on a basic inference problem and an autoencoder experiment to determine the optimal density ratio estimation algorithm. Throughout this thesis, we assume that the probability density function always exists for all elements in the parameter and sample spaces.

1.3 Results

After generalising the density ratio estimation algorithms, we show that all three estimator parametrisations have similar accuracies when optimised properly. We then compare the effectiveness of the estimators when under-trained, showing that the class probability estimator is the most accurate, followed by the direct ratio estimator and the direct log ratio estimator. This is likely because faster estimator convergence is correlated with increased accuracy. This experiment also finds that the estimators trained with the reverse KL divergence are more accurate, but also demonstrate greater initial instability. The under-trained estimators are compared again in the optimisation of an autoencoder, for both low-dimensional and high-dimensional latent spaces. Similar results are shown in the low-dimensional setting. On the other hand, the high-dimensional latent space leads to density ratios too large to be represented by the loss functions associated with both the direct ratio log ratio estimators. The class probability estimator is the only feasible estimator and is therefore superior in that regard. In this experiment, we also find that the estimators trained with the reverse KL divergence fail to stabilise, and therefore have inferior accuracy.

Overall, in this thesis, we:

- generalise the derivation of a density ratio estimator loss function to a choice of f-divergence and estimator parametrisation,
- show that the class probability estimator $D_\alpha(\cdot) \simeq q(\cdot)/(q(\cdot) + p(\cdot))$ surpasses the direct ratio estimator $r_\alpha(\cdot) \simeq q(\cdot)/p(\cdot)$ and the direct log ratio estimator $T_\alpha(\cdot) \simeq \log(q(\cdot)/(p(\cdot))$ in both feasibility and accuracy,
- show experimentally that estimators trained with the reverse KL divergence may be unstable, but have accurate density ratio estimation when stable,
- show experimentally that estimators trained with the GAN divergence are much more stable, but estimate the density ratio less accurately.

1.4 Thesis Structure

The remainder of this thesis is broken up into the following chapters:

- **Chapter 2** provides a background on neural networks, the model used to represent our estimators and variational densities.
- **Chapter 3** explains variational inference, describing the traditional mean-field variational inference and then introducing amortized variational inference, discussing the problems it faces with implicit densities.
- **Chapter 4** proposes two major algorithms used for density ratio estimation: class probability estimation and divergence minimisation.
- **Chapter 5** generalises density ratio estimation algorithms to a selection of f-divergence used to formulate the estimator loss function, and a parametrisation of the estimator.
- **Chapter 6** introduces a basic posterior inference problem and shows that the different estimator parametrisations have similar density ratio estimation accuracies when optimally trained. This chapter also compares the choice of f-divergence used as a lower bound for the estimator loss.
- **Chapter 7** provides further experimental insights into the choice of f-divergence and differentiates between the estimator parametrisations by comparing their accuracies when under-trained.
- **Chapter 8** reinforces the conclusions of Chapter 7, again comparing under-trained estimators in the context of data auto-encoding. The experiment was repeated for both low dimensional and high dimensional latent spaces.
- **Chapter 9** summarizes the experimental results and proposes additional experiment settings, potential improvements and unanswered questions for future research.

Chapter 2

Background on Neural Networks

In this chapter we give a general overview of a common model used in deep learning: neural networks. We first explain the motivation and intuition behind the model, then move to describe the structure of an individual node. We then expand to the overall neural network structure. After describing the network initialisation method, we conclude the chapter by demonstrating gradient descent training of neural networks and how back-propagation is used to find the required partial derivatives.

2.1 Motivation

Originally, neural networks were an attempt to create an algorithm that mimics the human brain's method of solving problems. The first machines using a neural network structure were created in the 1950s, and they were used widely from the 1980s onwards, as computers became sufficiently powerful for network training [Goodfellow et al., 2016].

One key feature of the brain structure is the capability of the neurons to adapt to suit different purposes [Zilles, 1992]. Neuroscientists have conducted experiments on animals where they rewired the optic nerve from the eye to the auditory cortex. They found that the auditory cortex eventually adapted to process the visual signals, and the animals were able to perform tasks requiring sight. This experiment can be repeated for almost any input sensor and the neurons will adjust accordingly to process the signals in a useful manner. They deduced that each neuron has a similar structure regardless of its location in the brain. Within each neuron, electrical signal inputs are transformed and outputted to other neurons. Overall, the network of neurons was able to process an arbitrary input signal to suit a given purpose [Zilles, 1992].

Let f^* be some function from $\mathbb{R}$ to $\mathbb{R}$. The primary goal of a neural network is to approximate f^* using a mapping with parameters $\boldsymbol{\Theta}$ from input $\boldsymbol{x}$ to output $\boldsymbol{y}$, that is, $\boldsymbol{y} = \boldsymbol{f}_{\boldsymbol{\Theta}}(\boldsymbol{x})$ [Goodfellow et al., 2016]. In fact, the universal approximation theorem states that neural networks can approximate any function in a finite-dimensional space with any desired non-zero amount of error, provided they are complex enough [Cybenko, 1989; Hornik, 1991]. For example, a typical regression problem of estimating housing prices would have the network inputting the values of certain predictors such as size (continuous) and building type (categorical), and outputting the expected price. Another example is the classification problem of recognising handwritten digits (0-9) in a grayscale image [Simard et al., 2003]. There are many inputs corresponding to the value of each pixel, and the network would have 10 outputs corresponding to the probability of each digit. The digit with the highest probability is then selected.

2.2 Individual Node Structure

Before discussing the overall structure of the neural network, we first describe the structure of an individual node. A typical node takes inputs from either the external input, or the outputs from other nodes, in addition to a *bias node*, which has the same purpose as the intercept term in a regression problem. The nodal inputs $\boldsymbol{x}$ are multiplied by weights $\boldsymbol{\theta}$ and then passed through an *activation function* $g(\boldsymbol{x})$. The individual node function is therefore

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\sum_{i=0}^{n} \theta_i x_i\right),$$

where n is the number of inputs excluding the bias node, which always has constant value $x_0 = 1$ [Cheng and Titterington, 1994]. An example of this is given in Figure 2.1.

The objective of the activation function is to restrict the output range and determine the level of input signal required for the output to become asymptotically large [Haykin, 1998]. An example of ranges used in practice are $(0, 1)$ or $\mathbb{R}$. A list of common activation functions is given in Section 2.3.

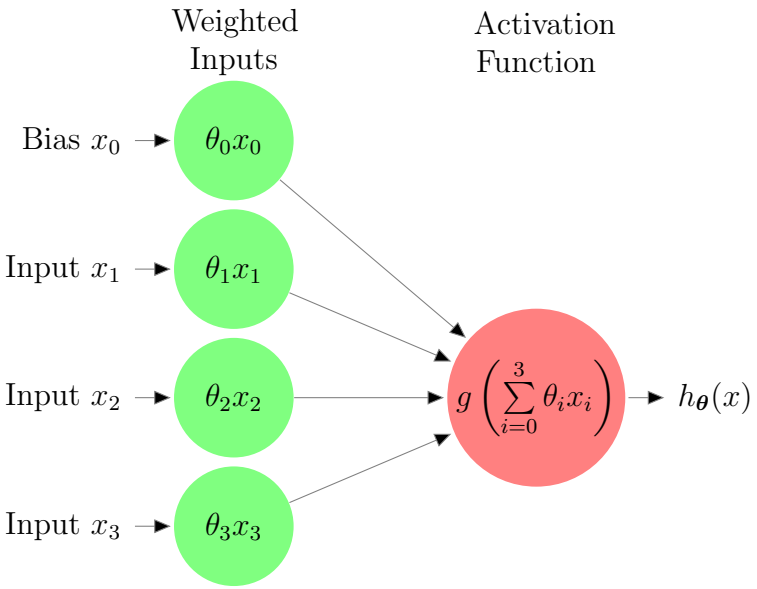


Figure 2.1: Example structure of an individual node function with 3 inputs, labelled as $\boldsymbol{x} = [x_0 \quad x_1 \quad x_2 \quad x_3]^\top$, with x_0 corresponding to the bias node. The weights are denoted as $\boldsymbol{\theta} = [\theta_0 \quad \theta_1 \quad \theta_2 \quad \theta_3]$.

2.3 Activation Functions

Some common activation functions [Goodfellow et al., 2016] are:

- The rectified linear unit (ReLU):

$$\begin{aligned} g &: \mathbb{R} \to [0, \infty) \\ g(x) &= \max\{0, x\}, \end{aligned}$$

- The sigmoid or logistic activation function:

$$\begin{aligned} g &: \mathbb{R} \to (0, 1) \\ g(x) &= (1 + \exp(-x))^{-1}, \end{aligned}$$

- The hyperbolic tangent function:

$$\begin{aligned} g &: \mathbb{R} \to (-1, 1) \\ g(x) &= \tanh(x), \end{aligned}$$

- The linear activation function:

$$\begin{aligned} g &: \mathbb{R} \to \mathbb{R} \\ g(x) &= x. \end{aligned}$$

Their plots are shown in Figure 2.2.

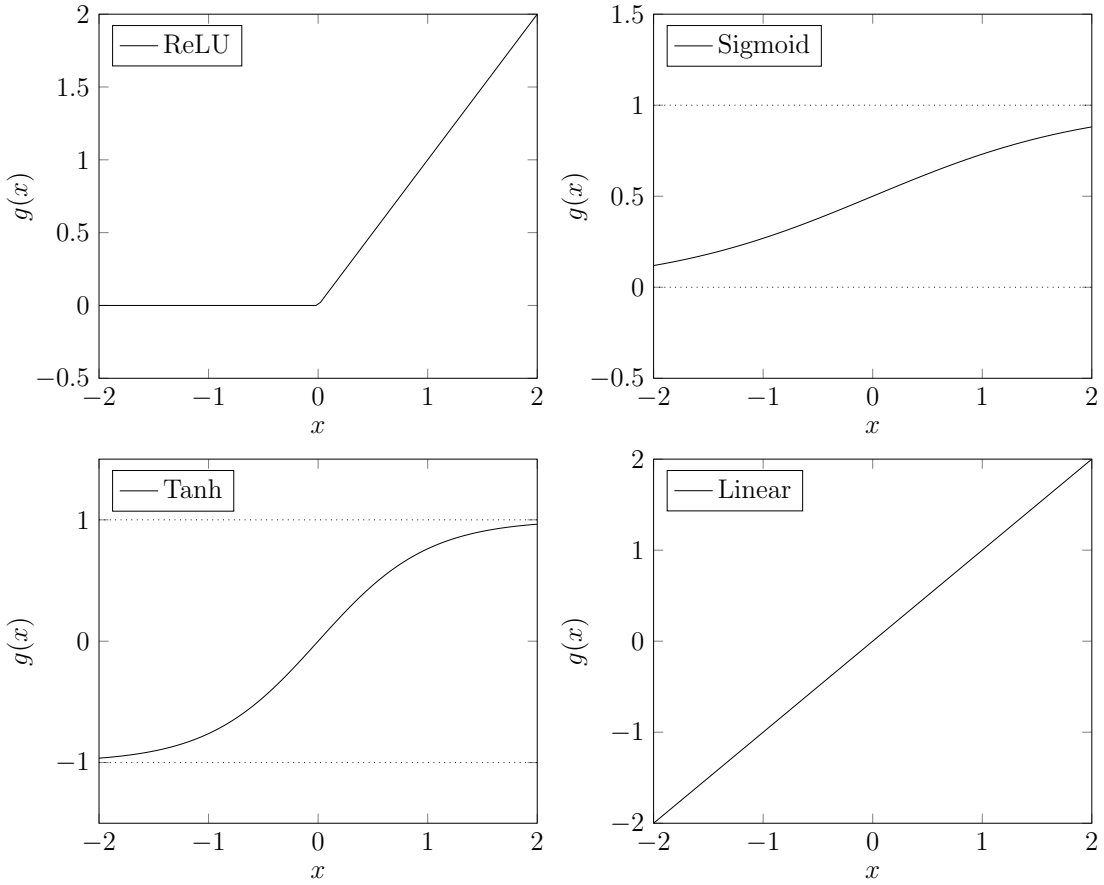


Figure 2.2: Activation Function Plots

The choice of activation function is dependent on several factors:

- The node's location in the network,
- Desired node output range,
- Continuous differentiability is ideal for gradient-based optimisation methods [Snyman, 2005],
- Monotonic activation functions ensure that the error is convex [Wu, 2009],
- In cases where the output range is restricted, the level of input signal required for the activation function output to be asymptotically close to its limit.

For example, if the node's output is the estimate of a probability (ranging in $(0, 1)$), then the sigmoid function would be used [Cybenko, 1989]. In addition to its ideal output range, the sigmoid function is continuously differentiable and requires significant input to output a value asymptotically close to 0 or 1. This allows the probability to be estimated with greater precision than with an activation function that approaches its limits very quickly, such as the hyperbolic tangent function.

On the other hand, if the node's output is the network's estimation of a non-negative quantity, such as price or time, then the ReLU activation function may be ideal, as it is bound by 0 and ∞, and its linearity makes the overall node operation similar to linear regression.

We now describe the overall neural network structure, as the choice of activation function is dependent on the node's relative location on the network.

2.4 Neural Network Structure

A typical neural network is made up of layers of interconnected nodes [Cheng and Titterington, 1994]. The first layer, called the input layer, does not have an activation function or weights, rather it simply acts as an input interface for the network. The outputs from the nodes can only be sent to nodes in succeeding layers, with the exception of the final output layer; it's result is simply the output of the network. The layers of nodes between the input and output layer are called the "hidden" layers, as their outputs are generally not interpreted by the user. Hidden layers can have an arbitrary number of nodes, whilst the nodes in the input and output layers are restricted to the desired number of inputs and outputs of the program. Example 2.4.1 explains the arithmetic operations within a neural network, and is illustrated in Figure 2.3.

The choice of activation function for a node in a neural network is typically dependent on its layer. The input layer can be described as having a linear activation function, as it has no activation function. Rectified linear units are the default choice for the hidden layers for their many advantages [Goodfellow et al., 2016]:

- Sparsity: since negative ReLU inputs result in a zero output, not all of the units are "active" (non-zero output) during the network's runtime. Sparsity is preferred in neural networks as it reduces overfitting and makes the model more robust to insignificant changes in input [Glorot et al., 2011].
- Faster computation: the $\max\{0, x\}$ function is computed much faster than the exponential or hyperbolic tangent function.
- Better gradient propagation: weight training in a neural network (discussed in Sections 2.6 and 2.7) involves 'back-propagating' a loss value through the network, which calculates the partial derivatives of the loss function with respect to the weights. The weights subsequently receive a change proportional to their partial derivative. Back-propagation uses the chain rule, so activation functions such as sigmoid or tanh that have a low gradient near their asymptotes

may experience the 'vanishing gradient problem', in which the calculated partial derivatives become increasingly small as the loss value propagates through the network [Kolen and Kremer, 2001]. This causes the front layers to train very slowly. The ReLU activation function does not experience this issue as it its gradient is either linear or 0 [Glorot et al., 2011].

The types of activation function used in the output layer have been explained in Section 2.3.

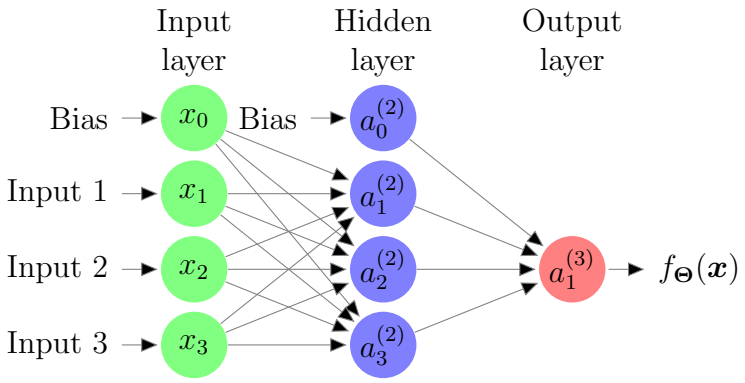


Figure 2.3: Example of a neural network structure with 3 external inputs, 1 hidden layer (with 3 nodes), 1 bias node per non-output layer and 1 output node. The variable inside each node denotes it's output as calculated in Section 2.2: the output of node i in layer j is denoted as $a_i^{(j)}$. $\boldsymbol{\Theta}$ denotes the weights of the network.

Example 2.4.1. In this example, we follow Figure 2.3, describing a neural network with 3 inputs, 1 hidden layer with 3 nodes and 1 output node. We denote the activation function as g, the output of node i in layer j as $a_i^{(j)}$, and the matrix of weights from layer j to $j+1$ as $\Theta^{(j)}$. We also use the subscript $\Theta_{m,n}^{(j)}$ where m is the row of the matrix corresponding to the node m in layer $j+1$, and n is the column of the matrix relating to node n in layer j.

Denoting the weights outputting to unit i in layer $j+1$ as $\boldsymbol{\theta}_i^{(j)}$, we have the following relation between the node weights (as described in Section 2.2) and these weight matrices: $\boldsymbol{\theta}_i^{(j)} = \begin{bmatrix}\Theta_{i,0}^{(j)} & \Theta_{i,1}^{(j)} \cdots \Theta_{i,k}^{(j)}\end{bmatrix}^\top$, where $k+1$ is the number of inputs.

Individually, the outputs in the hidden nodes and the output node are:

$$x_0 = 1, \qquad a_0^{(2)} = 1,$$

$$a_1^{(2)} = g\left(\Theta_{1,0}^{(1)}x_0 + \Theta_{1,1}^{(1)}x_1 + \Theta_{1,2}^{(1)}x_2 + \Theta_{1,3}^{(1)}x_3\right) = g\left((\boldsymbol{\theta}_1^{(1)})^\top \boldsymbol{x}\right),$$

$$a_2^{(2)} = g\left(\Theta_{2,0}^{(1)}x_0 + \Theta_{2,1}^{(1)}x_1 + \Theta_{2,2}^{(1)}x_2 + \Theta_{2,3}^{(1)}x_3\right) = g\left((\boldsymbol{\theta}_2^{(1)})^\top \boldsymbol{x}\right),$$

$$a_3^{(2)} = g\left(\Theta_{3,0}^{(1)}x_0 + \Theta_{3,1}^{(1)}x_1 + \Theta_{3,2}^{(1)}x_2 + \Theta_{3,3}^{(1)}x_3\right) = g\left((\boldsymbol{\theta}_3^{(1)})^\top \boldsymbol{x}\right),$$

$$f_\Theta(\boldsymbol{x}) = a_1^{(3)} = g\left(\Theta_{1,0}^{(2)}a_0^{(2)} + \Theta_{1,1}^{(2)}a_1^{(2)} + \Theta_{1,2}^{(2)}a_2^{(2)} + \Theta_{1,3}^{(2)}a_3^{(2)}\right) = g\left((\boldsymbol{\theta}_1^{(2)})^\top \boldsymbol{a}^{(2)}\right),$$

where $\boldsymbol{a}^{(2)} = \begin{bmatrix} a_0^{(2)} & a_1^{(2)} & a_2^{(2)} & a_3^{(3)} \end{bmatrix}^\top$.

2.5 Weight Initialisation

Proper initialisation of the weights $\boldsymbol{\Theta}$ is ideal to improve network training (discussed in Sections 2.6 and 2.7), as if the weights are too small, then the nodal outputs will continually decrease through the layers and become very small, resulting in a significant loss value which requires many iterations of training to fix. A similar scenario occurs when the initial weights are too high [Bishop, 1995]. In this section we discuss *Xavier Initialization* [Glorot and Bengio, 2010], a common initialisation method used in deep learning which aims to keep the signal variance constant throughout the network. To derive this algorithm, first consider a single node with $n+1$ inputs, and let z denote the weighted sum of the inputs $\boldsymbol{\theta}^\top \boldsymbol{x}$ before it is passed through the activation function. This is written as

$$z = \theta_0 + \sum_{i=1}^{n} \theta_i x_i.$$

Here, θ_0 is constant with respect to the external input, so $\text{Var}(\theta_0) = 0$. Now without any prior knowledge of the inputs and weights, we assume that they are independent and have 0 mean. We can then find the variance of the other terms by using the formula for the product of independent variables [Goodman, 1960]:

$$\begin{aligned}\text{Var}(\theta_i x_i) &= \mathbb{E}[x_i]^2\text{Var}(\theta_i) + \mathbb{E}[\theta_i]^2\text{Var}(x_i) + \text{Var}(\theta_i)\text{Var}(x_i) \\ &= \text{Var}(\theta_i)\text{Var}(x_i).\end{aligned}$$

Assuming that the weights and inputs are also identically distributed, we have

$$\text{Var}(z) = n\text{Var}(\theta_i)\text{Var}(x_i).$$

Since we want constant variance of the signals throughout the network, we set $\text{Var}(z) = \text{Var}(x_i)$ and the result follows:

$$\text{Var}(\theta_i) = \frac{1}{n}.$$

However, this result only considers forward propagation of the signal. A variation of this result accounts for back propagation by averaging the number of input and output nodes:

$$\text{Var}(\theta_i) = \frac{2}{n_{in} + n_{out}}.$$

Thus, to enforce constant signal variance throughout the network, the Xavier initialization of weights samples from a density, typically Gaussian, with 0 mean and $\frac{2}{n_{in}+n_{out}}$ variance:

$$\theta_i \sim \mathcal{N}\left(0, \frac{2}{n_{in} + n_{out}}\right).$$

2.6 Optimisation

The goal of optimising the network is to train the weights of the network such that a loss function, which we will denote as L, is minimized. A common loss function is the squared error between the batch of network outputs and the actual results:

$$L(\Theta) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{f}_\Theta(\boldsymbol{x}))^\top(\boldsymbol{y} - \boldsymbol{f}_\Theta(\boldsymbol{x})).$$

The $\frac{1}{2}$ factor is included to eliminate the factor of 2 in the derivative, simplifying the derivations. The derivative is multiplied by an arbitrary training rate during optimization so there is no significant impact of including that factor [Goodfellow et al., 2016].

Back-propagation (Section 2.7) is used to calculate the partial derivative of the loss function with respect to each individual weight. These partial derivatives are used in the gradient-based optimisation of the weights. There are many variations of neural network optimisation algorithms, but they are mostly based off *gradient descent*, which we will cover in this section [Ruder, 2016].

Definition 2.6.1. At point $\boldsymbol{x}^{(n)}$, $\boldsymbol{s}^{(n)}$ is a *descent direction* if $\nabla f\left(\boldsymbol{x}^{(n)}\right)^\top \boldsymbol{s}^{(n)} < 0$.

Gradient descent [Boyd and Vandenberghe, 2004] is an algorithm used to find the minimizer $\boldsymbol{x}^*$ of a function f by iterating on an arbitrary point $\boldsymbol{x}^{(n)}$, taking steps proportional to a descent direction $\boldsymbol{s}^{(n)}$:

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \alpha \boldsymbol{s}^{(n)}, \qquad \alpha > 0.$$

Proposition 2.6.2. *[Beck, 2014] If $\boldsymbol{s}^{(n)}$ is a descent direction for $x^{(n)}$, then for sufficiently small $\alpha > 0$,*

$$f\left(\boldsymbol{x}^{(n)} + \alpha \boldsymbol{s}^{(n)}\right) < f\left(\boldsymbol{x}^{(n)}\right).$$

Proof. A proof of this proposition can be found in Appendix A.1. □

A common choice of descent direction is the negative of the gradient, that is, $-\nabla f\left(\boldsymbol{x}^{(n)}\right)$. It is clearly a descent direction as

$$\begin{aligned}\nabla f(\boldsymbol{x}^{(n)})^{\top}\nabla f\left(\boldsymbol{x}^{(n)}\right) &= -\|\nabla f\left(\boldsymbol{x}^{(n)}\right)\|^2 \\ &< 0,\end{aligned}$$

where $\|\cdot\|$ denotes the Euclidean norm.

By nature, gradient descent is guaranteed to converge to a local minimum, which is problematic if the function has local minima which differ from the global minima. This is not an issue in this thesis, as each loss functions that we use is convex, so therefore any local minima are also global minima. Those interested in global optimization can refer to "Deterministic Global Optimization" by Floudas [2005].

Typically, the entire batch of data is used in each iteration to calculate the loss function and gradient values required for gradient descent. This method of *batch gradient descent* is very slow for large datasets. *Stochastic gradient descent* is defined by the use of only one observation per iteration, so the latent randomness associated with each observation effectively leads to noise added to each step, but the optimization is much faster. In practice, a compromise between stochastic and batch gradient descent is typically used; *mini-batch gradient descent* involves using several observations per iteration, reducing the gradient variance [Li et al., 2014; Ruder, 2016]. The size of the batch depends on the size and nature of the data set. For small data sets, the batch would typically be the entire data set as this would be computationally feasible. On the other hand, online datasets are stochastic in nature and generally very large, so a small batch size is ideal [Bengio, 2012].

Gradient descent convergence can be improved by using an *adaptive learning rate*, defined by the adjustment of the value of α over the iterations. This is because a low learning rate leads to slow convergence, whilst a high learning rate can cause the algorithm to oscillate around the minima [Ruder, 2016].

In this thesis, we use the *Adam algorithm*, which incorporates mini-batch gradient descent and an adaptive training rate to form an effective optimization algorithm that is commonly applied to neural networks. It uses the first and second moments of the gradient decay rate to adapt the weight training rate. More details can be found in "Adam: A Method for Stochastic Optimization" by Kingma and Ba [2014].

2.7 Back-Propagation

In the back-propagation algorithm, the goal is to find the partial derivative of the loss function with respect to the individual weights

$$\frac{\partial}{\partial \Theta_{m,n}^{(j)}} L(\Theta),$$

so that gradient descent can be performed to optimize the weights [E. Rumelhart et al., 1986]. For each training sample $\left(\boldsymbol{x}^{(I)}, \boldsymbol{y}^{(I)}\right)$, $I = 1, \ldots, N$, the input signal is propagated forward throughout the network to calculate $\boldsymbol{a}^{(j)}$ for $j = 2, \ldots, J$, where J is the total number of layers. The difference $\boldsymbol{\delta}^{(J)}$ between the network output and the ideal result is calculated with

$$\boldsymbol{\delta}^{(J)} = \boldsymbol{a}^{(J)} - \boldsymbol{y}^{(I)},$$

and this error is propagated backwards through the network to find $\boldsymbol{\delta}^{(J-1)}, \ldots, \boldsymbol{\delta}^{(2)}$ by using the formula

$$\boldsymbol{\delta}^{(j)} = \left(\left(\Theta^{(j)}\right)^{\top} \boldsymbol{\delta}^{(j+1)}\right) .* g'\left(\Theta^{(j)^{\top}} \boldsymbol{a}^{(j)}\right),$$

where $.*$ denotes element-wise multiplication and g' is the derivative of the activation function. Note that $\boldsymbol{\delta}^{(1)}$ does not need to be calculated as the input layer is not weighted.

The errors for each layer are multiplied by each of the preceding layer's activation outputs to form the estimated partial derivative for the training sample. This result is added to an accumulator matrix, so that the average partial derivative from all the training samples can be computed:

$$\Delta_{m,n}^{(j)} := \Delta_{m,n}^{(j)} + a_n^{(j)} \delta_m^{(j+1)}.$$

Finally, we divide the accumulator matrix entries by the number of training samples to find the average partial derivative of the cost function with respect to the weights:

$$\frac{\partial}{\partial \Theta_{m,n}^{(j)}} L(\Theta) = \frac{1}{N} \Delta_{m,n}^{(j)}.$$

Pseudocode for back-propagation is shown in Appendix B.1.

Chapter 3

Variational Inference

In this chapter, we explain *variational inference*, a method in Bayesian statistics that uses a specific functional form to approximate posterior densities. These are called 'variational densities', from the use of variational calculus to derive certain expressions. We first describe the Bayesian framework and problems associated with computational intractability. We then explain, with examples, two types of variational inference: mean-field variational inference and amortized inference. Finally, the chapter concludes with a description of issues that arise when one or more of the involved are densities implicit, that is, samples can be readily drawn from them but the density function is difficult to numerically evaluate.

3.1 Context

A fundamental problem in Bayesian statistics is to evaluate, or estimate posterior densities, so that inference on unknown parameters can be performed [Gelman et al., 2004]. Consider the set of latent and known variables $\boldsymbol{z} = (z_1, \ldots, z_M) \in \mathbb{R}^M$ and $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{R}^N$, respectively, with joint density $p(\boldsymbol{z}, \boldsymbol{x})$. The posterior density $p(\boldsymbol{z}|\boldsymbol{x})$ is the density of the latent parameters $\boldsymbol{z}$ conditioned on the known variables $\boldsymbol{x}$. Applying Bayes' theorem, it can be written as:

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{z}, \boldsymbol{x})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})}{\int_{\mathbb{R}} p(\boldsymbol{z}, \boldsymbol{x})\, \mathrm{d}\boldsymbol{z}},$$

where

- $p(\boldsymbol{z})$ is the prior density: the initial density of $\boldsymbol{z}$ before the data $\boldsymbol{x}$ is observed. This can be initialised to represent our subjective beliefs, or it can be an uninformative prior that implies objectivity.
- $p(\boldsymbol{x}|\boldsymbol{z})$ is the likelihood: the density of data $\boldsymbol{x}$ conditioned on latent $\boldsymbol{z}$.
- $p(\boldsymbol{x}) = \int_{\boldsymbol{z}} p(\boldsymbol{z}, \boldsymbol{x}) d\boldsymbol{z}$ is the marginal likelihood, or the evidence: the density of the data averaged across all possible parameter values.

In simple cases, the posterior can typically be calculated algebraically by using the proportionality $p(\boldsymbol{z}|\boldsymbol{x}) \propto p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})$ and normalising over the constant $p(\boldsymbol{x})$. As the model increases in complexity, the calculations required can become increasingly difficult, so traditional Markov Chain Monte Carlo (MCMC) methods overcome this obstacle by sampling from a Markov chain that converges to the stationary density $p(\boldsymbol{z}|\boldsymbol{x})$. However, these methods tend to have slow convergence for large datasets or high dimensional data. When faced with these issues or when desiring a faster computation, one may instead apply variational inference, an alternative approach to density estimation. Variational inference can be much faster than MCMC as it replaces sampling with optimisation, but it is known to underestimate the true posterior variance [Blei et al., 2017].

3.2 The KL Divergence

We first define the *f-divergence*: a measure of how much two probability densities differ.

Definition 3.2.1. The f-divergence of continuous probability density $q(x)$ from $p(x)$ is

$$D_f(p(x)\|q(x)) = \mathbb{E}_{p(x)}\left[f\left(\frac{q(x)}{p(x)}\right)\right],$$

where f is a convex function such that $f(1) = 0$.

Setting $f(x) = -\log x$ leads to the derivation of the KL (Kullback-Leibler) divergence [Kullback, 1959], a commonly used f-divergence in variational inference [Blei et al., 2017].

Definition 3.2.2. The KL divergence is the expected logarithmic difference between two densities $p(x)$ and $q(x)$ with respect to $p(x)$:

$$KL(p(x)\|q(x)) = \int_{-\infty}^{\infty} p(x)\log\left(\frac{p(x)}{q(x)}\right)\,\mathrm{d}x = \mathbb{E}_{p(x)}\left[\log\left(\frac{p(x)}{q(x)}\right)\right].$$

Remark 3.2.3. *The KL divergence is not symmetric:*

$$KL(p(x)||q(x)) \neq KL(q(x)||p(x)) \text{ for } p(x) \neq q(x).$$

In variational inference, $KL(p(x)\|q(x))$ is known as the *forward KL divergence*, whilst $KL(q(x)\|p(x))$ is the *reverse KL divergence*.

Lemma 3.2.4. *The reverse KL divergence is formulated when $f(x) = x\log x$ in an f-divergence.*

Proof. Proof of this lemma can be found in Appendix A.2. □

Note that the forward and reverse KL divergences mainly differ in the density that is used to take the expectation.

Lemma 3.2.5. *The KL divergence is non-negative, and it is equal to zero if and only if $p(x)$ and $q(x)$ are equivalent:*

$$KL(q(x)||p(x)) \geq 0.$$

Proof. Proof of this lemma can be found in Appendix A.3. □

3.3 Introduction to Variational Inference

Variational inference approximates the true posterior density $p(\boldsymbol{z}|\boldsymbol{x})$ with a different density $q(\boldsymbol{z})$, taken from a tractable family of approximate densities $\mathbb{Q}$, and then minimizes the f-divergence between the two densities in an optimization problem [Blei et al., 2017]:

$$q^*(\boldsymbol{z}) = \arg\min_{q(\boldsymbol{z})\in\mathbb{Q}} \{D_f(q(\boldsymbol{z})\|p(\boldsymbol{z}|\boldsymbol{x}))\}. \tag{3.3.1}$$

This produces an analytic approximation to the posterior density. The most common f-divergence used in variational inference is the reverse KL divergence, used instead of the forward KL divergence as we are unable to sample from our true posterior $p(\boldsymbol{z}|\boldsymbol{x})$. Equation (3.3.1) can therefore be written as:

$$q^*(\boldsymbol{z}) = \arg\min_{q(\boldsymbol{z})\in\mathbb{Q}} \{KL(q(\boldsymbol{z})\|p(\boldsymbol{z}|\boldsymbol{x}))\}. \tag{3.3.2}$$

From Lemma 3.2.5, it can be deduced that $KL(q(\boldsymbol{z})\|p(\boldsymbol{z}|\boldsymbol{x}))$ attains a minimal value of 0 when $q(\boldsymbol{z})$ and $p(\boldsymbol{z}|\boldsymbol{x})$ are equivalent.

We cannot evaluate the reverse KL divergence term in Equation (3.3.2) directly as $p(\boldsymbol{z}|\boldsymbol{x})$ is unknown. Instead, we apply Bayes' law to the equation and rearrange the terms to formulate a tractable expression that can be optimized.

3.4 Derivation of the ELBO

In this section, we formulate the *evidence lower bound* ($ELBO$) of our posterior inference problem. Maximisation of this term is equivalent to solving Equation

(3.3.2). We begin by applying Bayes' law to the problem and expanding the terms:

$$\begin{aligned}
q^*(\boldsymbol{z}) &= \arg\min_{q(\boldsymbol{z})\in\mathbb{Q}} \{KL(q(\boldsymbol{z})\|p(\boldsymbol{z}|\boldsymbol{x}))\} \\
&= \arg\min_{q(\boldsymbol{z})\in\mathbb{Q}} \left\{\mathbb{E}_{q(\boldsymbol{z})}[\log q(\boldsymbol{z}) - \log p(\boldsymbol{z}|\boldsymbol{x})]\right\} \\
&= \arg\min_{q(\boldsymbol{z})\in\mathbb{Q}} \left\{\mathbb{E}_{q(\boldsymbol{z})}\left[\log q(\boldsymbol{z}) - \log \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})}\right]\right\} \\
&= \arg\min_{q(\boldsymbol{z})\in\mathbb{Q}} \left\{\mathbb{E}_{q(\boldsymbol{z})}[\log q(\boldsymbol{z}) - \log p(\boldsymbol{x}|\boldsymbol{z}) - \log p(\boldsymbol{z})] + \log p(\boldsymbol{x})\right\}.
\end{aligned}$$

Note that in the last line $\mathbb{E}_{q(\boldsymbol{z})}[p(\boldsymbol{x})] = p(\boldsymbol{x})$ as $p(\boldsymbol{x})$ is independent of $q(\boldsymbol{z})$. Since our issues with Equation (3.3.2) result from the intractability of $p(\boldsymbol{x})$, we rearrange the KL divergence expression as follows:

$$\begin{aligned}
KL(q(\boldsymbol{z})\|p(\boldsymbol{z}|\boldsymbol{x})) &= \mathbb{E}_{q(\boldsymbol{z})}[\log q(\boldsymbol{z}) - \log p(\boldsymbol{x}|\boldsymbol{z}) - \log p(\boldsymbol{z})] + \log p(\boldsymbol{x}) \\
\log p(\boldsymbol{x}) - KL(q(\boldsymbol{z})\|p(\boldsymbol{z}|\boldsymbol{x})) &= -\mathbb{E}_{q(\boldsymbol{z})}[\log q(\boldsymbol{z}) - \log p(\boldsymbol{x}|\boldsymbol{z}) - \log p(\boldsymbol{z})] \\
&= \mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{E}_{q(\boldsymbol{z})}[\log q(\boldsymbol{z}) - \log p(\boldsymbol{z})] \\
&= \mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - KL(q(\boldsymbol{z})\|p(\boldsymbol{z})) \qquad (3.4.1) \\
\log p(\boldsymbol{x}) &\geq \mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - KL(q(\boldsymbol{z})\|p(\boldsymbol{z})). \qquad (3.4.2)
\end{aligned}$$

We refer to $\mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - KL(q(\boldsymbol{z})\|p(\boldsymbol{z}))$ as $ELBO(q)$, as it is the lower bound of the evidence density. Our minimization problem in Equation (3.3.2) is now equivalent to maximizing $ELBO(q)$, so it can be written as:

$$\begin{aligned}
q^*(\boldsymbol{z}) &= \arg\min_{q(\boldsymbol{z})\in\mathbb{Q}} \{KL(q(\boldsymbol{z})\|p(\boldsymbol{z}|\boldsymbol{x}))\} \\
&= \arg\max_{q(\boldsymbol{z})\in\mathbb{Q}} \{ELBO(q)\} \\
&= \arg\max_{q(\boldsymbol{z})\in\mathbb{Q}} \left\{\mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - KL(q(\boldsymbol{z})\|p(\boldsymbol{z}))\right\}. \qquad (3.4.3)
\end{aligned}$$

3.5 Mean-Field Variational Family

The family of variational densities $\mathbb{Q}$ is typically a '*mean-field variational family*', in which the density $q(\boldsymbol{z})$ factorizes over the latent variables $\{z_i\}_{i=1}^M$, each with an individual set of parameters $\{\phi_i\}_{i=1}^M$ [Blei et al., 2017]:

$$q(\boldsymbol{z}) = \prod_{i=1}^{M} q_{\phi_i}(z_i).$$

The individual factors $q_{\phi_i}(z_i)$ can take any form, but they are typically assumed to be independent, simplifying derivations but reducing the estimation accuracy when the latent variables in the true posterior density $p(\boldsymbol{z}|\boldsymbol{x})$ exhibit dependence. Fixing the forms of the individual factors, we want to choose the parameters ϕ_i such that $ELBO(q)$ is maximized. A derivation of the optimal factor $q^*_{\phi_i}(z_i)$ can be found in Appendix C, and leads to the following equation:

$$q^*_{\phi_i}(z_i) \propto \exp\left(\mathbb{E}_{\boldsymbol{z}_{-i}}[\log p(z_i|\boldsymbol{z}_{-i}, \boldsymbol{x})]\right).$$

This expression can be used in an *expectation-maximization algorithm*, in which the $q^*_{\phi_i}(z_i)$ is evaluated and iterated from $i = 1, \ldots, M$ until $ELBO(q)$ converges. (We can say convergence occurs when there is little variation in $ELBO(q)$ over the iterations.) This particular algorithm is called *coordinate ascent variational inference* (CAVI); pseudocode can be found in Algorithm 6 in Appendix B.2. Appendix D exemplifies mean-field variational inference, closely following the "Bayesian Mixture of Gaussians" example from "Variational Inference: A Review for Statisticians" by Blei et al. [2017].

3.6 Amortized Inference

Now consider the case where we have K data points, each with dimensionality N. We denote the set of data points as $\boldsymbol{X} = \left[\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(K)}\right]^\top$, where $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbb{R}^N$, $i = 1, \ldots, K$. One disadvantage of mean field variational inference is that a specific set of variational parameters needs to be derived and optimized for each of these data points. This is computationally expensive for large datasets, and is because the parametrisation of the posterior $p(\boldsymbol{z}|\boldsymbol{x})$ changes as the data point $\boldsymbol{x}$ changes. We therefore denote our set of latent variables as $\boldsymbol{Z} = (\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(K)})^\top$, where $\boldsymbol{z}^{(i)} = (z_1^{(i)}, \ldots, z_M^{(i)}) \in \mathbb{R}^M$, $i = 1, \ldots, K$. In mean-field variational inference, each latent variable $z_j^{(i)}$ has its own individual set of parameters $\phi_j^{(i)}$, so overall there are $M \times N$ sets of parameters.

Amortized inference resolves this issue by using a single, constant set of parameters for all data points, adding the data point itself as an input to the variational density [Zhang et al., 2017]. This practice is defined as *amortizing* the density. Our variational density therefore conditions on the observation, taking the form

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}).$$

Figures 3.1 and 3.2 highlight the difference between mean-field and amortized variational inference.

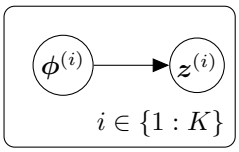


Figure 3.1: This DAG represents Mean-Field Variational Inference. For each of the K datasets, a set of parameter sets $\boldsymbol{\phi}^{(\boldsymbol{i})}$ corresponding to each latent variable point $\boldsymbol{z}^{(i)}$ has to be found. $\boldsymbol{\phi}^{(i)}$ is M-dimensional, so there is a total of $M \times K$ sets of parameters.

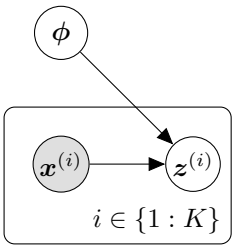


Figure 3.2: This DAG represents Amortized Variational Inference. Here, there is only one set of variational parameters $\boldsymbol{\phi}$, and each data point $\boldsymbol{x}^{(i)}$ is used as an input in the variational posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to find $\boldsymbol{z}^{(i)}$.

A very complex model is required for such a structure, so it typically takes the form of a neural network with $\boldsymbol{x}$ as an input. This method is often used in deep learning due to the significant amount of data required to train such a network.

Recall that $p(\boldsymbol{x})$ represents the 'true' density of the data. This is typically unavailable, so we instead represent it with the density of a sample dataset, which we denote as $q^*(\boldsymbol{x})$. Since our variational posterior is conditioned on data point $\boldsymbol{x}$, we now aim to minimize the expected KL divergence with respect to $q^*(\boldsymbol{x})$:

$$\begin{aligned}
\phi &= \arg\min_{\phi} \left\{ \mathbb{E}_{q^*(\boldsymbol{x})} \left[KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z}|\boldsymbol{x})) \right] \right\} \\
&= \arg\min_{\phi} \left\{ \mathbb{E}_{q^*(\boldsymbol{x}) q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z}|\boldsymbol{x}) \right] \right\} \\
&= \arg\min_{\phi} \left\{ \mathbb{E}_{q^*(\boldsymbol{x}) q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log \frac{p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z})}{p(\boldsymbol{x})} \right] \right\} \\
&= \arg\min_{\phi} \left\{ \mathbb{E}_{q^*(\boldsymbol{x}) q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{x}|\boldsymbol{z}) - \log p(\boldsymbol{z}) \right] + \log p(\boldsymbol{x}) \right\}.
\end{aligned}$$

Again, we cannot evaluate this expression as $\log p(\boldsymbol{x})$ is intractable, so we rearrange the terms to form the evidence lower bound, which we aim to maximise to minimise

the KL divergence.

$$\begin{aligned} ELBO(q) &= \mathbb{E}_{q^*(\boldsymbol{x})}[\log p(\boldsymbol{x}) - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}|\boldsymbol{x}))] \\ &= -\mathbb{E}_{q^*(\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{x}|\boldsymbol{z}) - \log p(\boldsymbol{z})\right] \\ &= \mathbb{E}_{q^*(\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z}) + \log p(\boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})\right] \\ &= \mathbb{E}_{q^*(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}))\right]. \end{aligned}$$

We take the negative of $ELBO(q)$ to form the *negative evidence lower bound* $NELBO(q)$:

$$NELBO(q) \coloneqq \mathbb{E}_{q^*(\boldsymbol{x})}\left[-\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] + KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}))\right],$$

which we aim to minimize with respect to the variational posterior weights ϕ:

$$\min_\phi\{NELBO(q)\}. \tag{3.6.1}$$

Since ϕ represents the parameters of a neural network, it is unnecessary to find the specific weight values during back-propagation: we are more interested in training the network to optimality, achieved by minimizing the objective function. Hence, it is more appropriate to write the optimization problem with min than with arg min.

In deep learning, the likelihood term $p(\boldsymbol{x}|\boldsymbol{z})$ is often represented as a neural network parametrized by θ: $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. This network is optimized alongside the variational density, in a formation known as the variational autoencoder.

3.7 Example: Variational Autoencoder

A *variational autoencoder* (Figure 3.3) is a model consisting of two simultaneously trained neural networks: an encoder representing the posterior density $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ that "compresses" a data point $\boldsymbol{x} = (x_1, \ldots, x_N)$ into a lower dimensional latent representation $\boldsymbol{z}$, and a decoder representing the likelihood density $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ that "reconstructs" the data point from the latent variable [Kingma and Welling, 2013]. Typically, the prior $p(\boldsymbol{z})$ is simply a standard multivariate normal density with dimensionality equal to that of the latent variable point $\boldsymbol{z} = (z_1, \ldots, z_M)$: $\mathcal{N}(0, I_{M\times M})$. The autoencoder has two main purposes: lower-dimensional representation learning of data and data generation. We now reiterate our optimization problem in

Equation (3.6.1), this time including the optimization of the decoder parameters θ:

$$\min_{\phi,\theta} \left\{ \mathbb{E}_{q^*(\boldsymbol{x})} \left[-\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] + KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})) \right] \right\}. \tag{3.7.1}$$

The first term is the negative likelihood, analogous to the error between the original and reconstructed data, which we aim to minimize. The KL divergence between the variational posterior and the prior density acts as a regularizer term. Without it, the encoder would learn to segregate distinct data types in separate regions of the Euclidean plane, which runs contrary to the randomness of a probability density. Due to this regularizer term, we can generate new data $\boldsymbol{x}$ by sampling $\boldsymbol{z} \sim p(\boldsymbol{z})$ and feeding it through the decoder [Doersch, 2016]. This is illustrated in Figure 3.4.

In our current formulation, we are trying to represent the variational posterior density with a deterministic neural network, so for any arbitrary fixed data point $\boldsymbol{x}$, the value of the encoder's latent output $\boldsymbol{z}$ is always the same. The solution is to add a noise density to the model to make it probabilistic. Instead of the encoder outputting the posterior sample $\boldsymbol{z}$ directly, we configure it to output a mean vector $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_M)^\top$ and variance vector $\boldsymbol{\sigma}^2 = (\sigma_1^2, \sigma_2^2, \cdots, \sigma_M^2)^\top$, each with dimensions equal to that of latent variable $\boldsymbol{z}$. We then define $\boldsymbol{z}$ as an output from a multivariate normal density with means and variances as specified by the encoder output:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2 I_{M\times M}).$$

Often in practice, this is achieved by sampling random standard normal noise ϵ, multiplying it by the variance and adding the result to the mean:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I_{M\times M}), \qquad \boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \cdot \boldsymbol{\sigma}^2.$$

A similar process of adding random noise is used for the decoder network representing the likelihood density $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, but it's parametrisation is chosen depending on the nature of the data. For example, a multivariate normal parametrisation similar to our variational posterior can be used for most continuous data. For binary data, a sigmoid output layer is specified in the neural network and the likelihood is expressed as a Bernoulli density with probabilities given by the network output.

Lemma 3.7.1. *Using the explicit form of the multivariate normal $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$ densities, the KL divergence term can be calculated through the equation:*

$$KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})) = \frac{1}{2} \sum_{i=1}^{M} \left(\sigma_i^2 + \mu_i^2 - \log(\sigma_i^2) - 1 \right).$$

Proof. Proof of this lemma can be found in Appendix A.4. □

We can evaluate and minimise $NELBO(q)$ in Equation (3.7.1) as we have an explicit parametrisation of all of the terms.

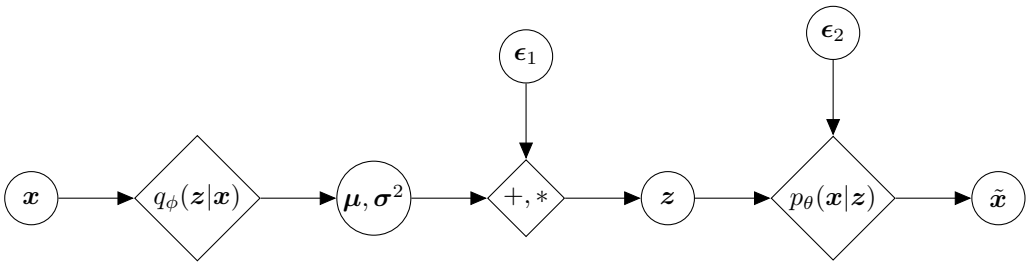


Figure 3.3: A simple DAG representing a Variational Autoencoder. An arbitrary data point $\boldsymbol{x}$ is passed through the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to produce a mean vector $\boldsymbol{\mu}$ and a variance vector $\boldsymbol{\sigma}^2$. Random noise $\boldsymbol{\epsilon}_1$ is sampled from $\mathcal{N}(0, I_{M\times M})$ and transformed to generate $\boldsymbol{z}$: $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \cdot \boldsymbol{\sigma}^2$. This latent variable $\boldsymbol{z}$ is passed through the decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ to reconstruct the data point as $\tilde{\boldsymbol{x}}$.

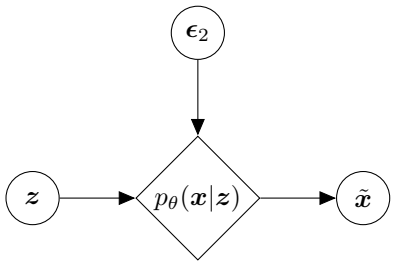


Figure 3.4: To generate new data $\tilde{\boldsymbol{x}}$ similar to existing data $\boldsymbol{x}$, we sample latent variable $\boldsymbol{z}$ from the prior density $p(\boldsymbol{z})$ and pass it through the decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$.

3.8 Problems with Implicit Densities

The above sections assume that the prior $p(\boldsymbol{z})$, likelihood ($p(\boldsymbol{x}|\boldsymbol{z})$ or $p_\theta(\boldsymbol{x}|\boldsymbol{z})$) and variational posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ have an explicit form. However, there are two main scenarios in which at least one of these densities is implicit, that is, the parametrisation of their density is difficult to numerically evaluate but we are able to easily generate samples from them. This poses problems with optimization as the objective function becomes difficult to compute.

3.8.1 Implicit Prior and/or Variational Posterior

In the first scenario, at least one of the prior $p(\boldsymbol{z})$ or variational posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ densities is *implicit*, but the likelihood is known. An implicit prior is rare but may

occur in posterior inference. The implicit variational posterior case occurs more often (in both posterior inference and data generation), and is detailed in "Adversarial Variational Bayes" by Mescheder et al. [2017].

In amortized inference, the variational posterior sample $\boldsymbol{z}$ is typically sampled from an independent multivariate normal density with means and variances defined by the variational network. However, this representation lacks dependencies between the latent variables, and therefore fails to effectively model multi-modal or flexible densities. In his paper, Mescheder adds random noise $\epsilon \sim \pi(\epsilon)$ as additional inputs to the variational network, training the network to directly output $\boldsymbol{z}$. $\pi(\epsilon)$ is a typical noise density, such as $\mathcal{N}(0, I_{p\times p})$ where p is the desired number of noise inputs. This added noise allows the probabilistic nature of a random density to be represented by a deterministic neural network. However, note that the neural network does not output the probability density $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, rather it outputs a sample $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$. Retaining the network parameters ϕ, we denote the generator of posterior samples as $\mathcal{G}_\phi(\epsilon; \boldsymbol{x})$. A diagram illustrating this is shown in Figure 3.5.

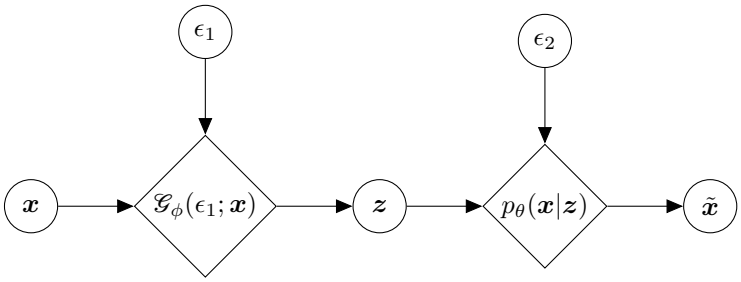


Figure 3.5: This diagram depicts the "Adversarial Variational Bayes" formulation of the variational autoencoder. Rather than transforming random noise ϵ_1 according to the mean vector $\boldsymbol{\mu}$ and variance vector $\boldsymbol{\sigma}$ output of the variational posterior $q_\phi(z|x)$, we add the noise to the encoder network $\mathcal{G}_\phi(\epsilon_1; \boldsymbol{x})$ directly as an additional input. The likelihood density $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ has the same explicit representation as in Figures 3.3 and 3.4.

Due to the complex nature of the neural network, it is difficult to numerically compute the explicit form of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, but we are able to easily generate samples by feeding data and noise through the network, hence its implicit nature.

Now recall our optimization problem from Equation (3.7.1):

$$\min_{\phi,\theta} \left\{ \mathbb{E}_{q^*(\boldsymbol{x})} \left[-\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] + KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})) \right] \right\}.$$

When either the prior or variational posterior is implicit, this expression is difficult to evaluate as we are unable to calculate the KL divergence term:

$$KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\right].$$

We therefore resort to *density ratio estimation* techniques (Chapter 4) to approximate $q_\phi(\boldsymbol{z}|\boldsymbol{x})/p(\boldsymbol{z})$, using only samples from the two densities.

3.8.2 Implicit Likelihood

When we apply amortized variational inference to an implicit likelihood density, the optimization problem is intractable even with density ratio estimation, as it is difficult to evaluate the likelihood term $-\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]$. This can occur in posterior inference or in some autoencoder variations where an implicit generative function $G(\epsilon; \boldsymbol{x})$ is used to represent the likelihood density $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ [Dumoulin et al., 2016]. An example of the latter case is illustrated in Figure 3.6.

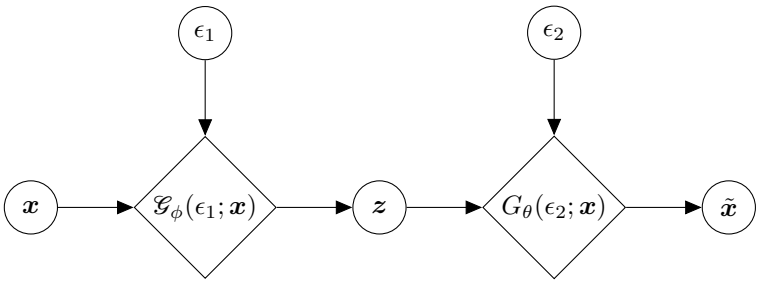


Figure 3.6: This diagram depicts a variation of "Adversarial Variational Bayes", in which noise is additionally added to the input of the generator of likelihood density samples $G_\theta(\epsilon_2; \boldsymbol{x})$. The likelihood density $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is therefore implicit and consequently, its corresponding term in the optimisation problem cannot be evaluated.

We rephrase the optimisation problem, instead minimising the reverse KL divergence between the two joint densities [Tran et al., 2017]:

$$KL(q(\boldsymbol{z}, \boldsymbol{x})\|p(\boldsymbol{z}, \boldsymbol{x})) = \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}, \boldsymbol{x})}{p(\boldsymbol{z}, \boldsymbol{x})}\right].$$

We derive this by applying Bayes' theorem to the reverse KL divergence between the true and variational posterior densities, this time formulating the joint densities:

$$\begin{aligned}
\mathbb{E}_{q^*(\boldsymbol{x})}\left[KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}|\boldsymbol{x}))\right] &= \mathbb{E}_{q^*(\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}\right] \\
&= \mathbb{E}_{q^*(\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}|\boldsymbol{x})q^*(\boldsymbol{x})}{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}\right] + \mathbb{E}_{q^*(\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x})}{q^*(\boldsymbol{x})}\right]
\end{aligned}$$

$$= \mathbb{E}_{q^*(\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z},\boldsymbol{x})}{p(\boldsymbol{z},\boldsymbol{x})}\right] - KL(q^*(\boldsymbol{x})\|p(\boldsymbol{x})).$$

Recall that $p(\boldsymbol{x})$ is typically unavailable, so we are unable to evaluate $KL(q^*(\boldsymbol{x})\|p(\boldsymbol{x}))$ even with density ratio estimation techniques. Since it is constant, we add it to both sides of the equation, leading to the following expression for $NELBO(q)$:

$$\begin{aligned} NELBO(q) &= KL(q^*(\boldsymbol{x})\|p(\boldsymbol{x})) + \mathbb{E}_{q^*(\boldsymbol{x})}\left[KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}|\boldsymbol{x}))\right] \\ &= \mathbb{E}_{q^*(\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z},\boldsymbol{x})}{p(\boldsymbol{z},\boldsymbol{x})}\right]. \end{aligned}$$

This expression attains a minimum at $KL(q^*(\boldsymbol{x})\|p(\boldsymbol{x}))$ when

$$\mathbb{E}_{q^*(\boldsymbol{x})}\left[KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}|\boldsymbol{x}))\right] = 0,$$

that is, the true and variational posteriors are equivalent with respect to the density of the data set. We therefore have the following lower bound:

$$\mathbb{E}_{q^*(\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z},\boldsymbol{x})}{p(\boldsymbol{z},\boldsymbol{x})}\right] \geq KL(q^*(\boldsymbol{x})\|p(\boldsymbol{x})).$$

Our objective now is to minimize $NELBO(q)$ with respect to our variational parameters ϕ:

$$\min_\phi \left\{\mathbb{E}_{q^*(\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z},\boldsymbol{x})}{p(\boldsymbol{z},\boldsymbol{x})}\right]\right\}. \tag{3.8.1}$$

Again, this term is approximated with density ratio estimation techniques. For consistency, we retain the notation of our variational posterior loss function as $NELBO(q)$. Note that the objective problem no longer has an explicit likelihood term, and can therefore be used with an implicit likelihood density. Since density ratio estimation only requires samples from the two densities, both the prior and variational posterior densities can also be implicit, hence the "Adversarial Variational Bayes" representation (Figure 3.5) of the variational network can be used.

In the autoencoder representation (Figure 3.6), the forward KL divergence is additionally minimized with respect to the likelihood parameters [Tiao et al., 2018]:

$$\min_\theta \left\{\mathbb{E}_{p(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{z},\boldsymbol{x})}{q(\boldsymbol{z},\boldsymbol{x})}\right]\right\}. \tag{3.8.2}$$

In this scenario, separate density ratio estimators are typically used for Equations (3.8.1) and (3.8.2).

Chapter 4

Density Ratio Estimation

In this chapter, we derive two common methods of density ratio estimation used with implicit models [Mohamed and Lakshminarayanan, 2016; Sugiyama et al., 2012]: *class probability estimation* and *divergence minimisation*, applying them to the objective functions in both the implicit prior/posterior and implicit likelihood scenarios to formulate bi-level optimization problems. In this thesis, we use Huszar's terminology, referring to the implicit prior/posterior objective as "*prior-contrastive*" and the implicit likelihood objective as "*joint-contrastive*" [Huszár, 2017].

4.1 Class Probability Estimation

4.1.1 Derivation

First, consider the probability density functions $p(u)$ and $q(u)$ defined on $U \subseteq \mathbb{R}^S$. We aim to estimate the density ratio $q(u)/p(u)$, given only samples from the two densities. In this thesis, we assume that for all $u \in U$, $p(u) > 0$ and $q(u) > 0$, avoiding the division of zero in density ratio estimation.

We take m samples from $p(u)$: $U_p = \{u_1^{(p)}, \ldots, u_m^{(p)}\}$ and label them with $y = 0$, then we take n samples from $q(u)$: $U_q = \{u_1^{(q)}, \ldots, u_n^{(q)}\}$ and label them with $y = 1$. Therefore, $p(u) = P(u|y = 0)$ and $q(u) = P(u|y = 1)$. By applying Bayes' theorem, we derive an expression for the density ratio:

$$\begin{aligned}
\frac{q(u)}{p(u)} &= \frac{P(u|y=1)}{P(u|y=0)} \\
&= \frac{P(y=1|u)P(u)}{P(y=1)} \Big/ \frac{P(y=0|u)P(u)}{P(y=0)} \\
&= \frac{P(y=1|u)}{P(y=0|u)} \times \frac{P(y=0)}{P(y=1)} \\
&= \frac{P(y=1|u)}{P(y=0|u)} \times \frac{n}{m}.
\end{aligned}$$

Often in practice, $m = n$, simplifying the density ratio to:

$$\frac{q(u)}{p(u)} = \frac{P(y = 1|u)}{P(y = 0|u)}$$

which is the ratio of the probability that an arbitrary sample u was taken from the density $q(u)$ to the probability that it was taken from $p(u)$. If we define a *discriminator* function $D(u) \simeq P(y = 1|u)$ that estimates these probabilities, then our density ratio can be expressed in terms of this function:

$$\frac{q(u)}{p(u)} \simeq \frac{D(u)}{1 - D(u)}.$$

At optimality, the discriminator provides an exact estimate of the probability that the sample was taken from $q(u)$, so we have $D^*(u) = P(y = 1|u)$ and therefore,

$$\frac{q(u)}{p(u)} = \frac{D^*(u)}{1 - D^*(u)}.$$

The discriminator function typically takes the form of a neural network with parameters α trained by minimizing the *Bernoulli loss* (the negative log-likelihood of the Bernoulli density), which we denote as L_D [Sugiyama et al., 2012]:

$$L_D \coloneqq -\mathbb{E}_{q(u)}[\log D_\alpha(u)] - \mathbb{E}_{p(u)}[\log(1 - D_\alpha(u))].$$

Lemma 4.1.1. *The discriminator reaches optimality at $D^*_\alpha(u) = q(u)/(q(u)+p(u))$, minimizing its Bernoulli loss [Goodfellow et al., 2014].*

Proof. Proof of this lemma can be found in Appendix A.5. □

Remark 4.1.2. *$D^*_\alpha(u)$ is bound in $(0, 1)$, so typically a sigmoid activation function is used for its output layer. Note that $D^*_\alpha(u) \neq 0$ and $D^*_\alpha(u) \neq 1$ as we have assumed $p(u) > 0$ and $q(u) > 0$ for all $u \in U$.*

Remark 4.1.3. *When $q(u)$ and $p(u)$ are equivalent, that is, $q(u) = p(u)$, the discriminator cannot distinguish between samples from the two densities, outputting a probability of $\frac{1}{2}$: $D^*_\alpha(u) = \frac{1}{2}$. Substituting this value into the Bernoulli loss function, we obtain its minimal value when the two densities are equivalent and the discriminator is at its optimal parametrisation: $L_{D^*} = \log 4$.*

In the context of amortized variational inference, the optimal discriminative loss can be used to check posterior convergence, that is, when the variational density $q_\phi(z|x)$ is equivalent to the true posterior density $p(z|x)$.

4.1.2 Prior-Contrastive Algorithm

We now turn to the prior-contrastive problem in Expression (3.6.1) of minimizing the $NELBO$:

$$\begin{aligned}\min_{\phi}\{NELBO(q)\} &= \min_{\phi}\left\{\mathbb{E}_{q^*(x)}\left[-\mathbb{E}_{q_\phi(z|x)}[\log p(x|z)] + KL(q_\phi(z|x)\|p(z))\right]\right\} \\ &= \min_{\phi}\left\{-\mathbb{E}_{q^*(x)q_\phi(z|x)}[\log p(x|z)] + \mathbb{E}_{q^*(x)q_\phi(z|x)}\left[\log\frac{q_\phi(z|x)}{p(z)}\right]\right\},\end{aligned} \tag{4.1.1}$$

which requires us to find the density ratio between $q_\phi(z|x)$ and $p_\theta(z)$.
If we label samples from $q_\phi(z|x)$ with $y = 1$ and samples from $p(z)$ with $y = 0$, we have the density ratio expression:

$$\frac{q_\phi(z|x)}{p(z)} = \frac{P(y=1|z)}{P(y=0|z)}.$$

We now define a discriminator function with parameters α that calculates the probability that an arbitrary sample z belongs to the variational posterior $q_\phi(z|x)$. Since the posterior changes depending on the observation x, we also amortize the discriminator by taking an additional input from the dataset density $q^*(x)$, as opposed to training multiple discriminators for each observation:

$$D_\alpha(z, x) \simeq P(y = 1|z).$$

As a binary classifier, this function can be trained by inputting an equal number of samples from both densities and minimizing its Bernoulli loss:

$$\min_{\alpha}\left\{-\mathbb{E}_{q^*(x)q_\phi(z|x)}[\log D_\alpha(z, x)] - \mathbb{E}_{q^*(x)p_\theta(z)}[\log(1 - D_\alpha(z, x))]\right\}. \tag{4.1.2}$$

We can now express the expected log ratio in terms of the probability that a posterior sample is correctly classified by the discriminator:

$$\begin{aligned}\mathbb{E}_{q^*(x)q_\phi(z|x)}\left[\log\frac{q_\phi(z|x)}{p(z)}\right] &= \mathbb{E}_{q^*(x)q_\phi(z|x)}\left[\log\frac{P(y=1|z)}{P(y=0|z)}\right] \\ &\simeq \mathbb{E}_{q^*(x)q_\phi(z|x)}\left[\log\frac{D_\alpha(z, x)}{1 - D_\alpha(z, x)}\right].\end{aligned}$$

Our $NELBO$ optimization objective in Expression (4.1.1) can now be written as:

$$\min_{\phi} \left\{ -\mathbb{E}_{q^*(x)q_\phi(z|x)}[\log p(x|z)] + \mathbb{E}_{q^*(x)q_\phi(z|x)} \left[\log \frac{D_\alpha(z,x)}{1 - D_\alpha(z,x)} \right] \right\}. \tag{4.1.3}$$

Recall from Remark 4.1.2 that a sigmoid activation function is typically used for the discriminator's output layer, so theoretically, the discriminator should never output its asymptotic limits of 0 or 1. However, numbers arbitrarily close to these limits may underflow or overflow to them when computationally stored in a floating point representation. Thus, when implementing this algorithm, we introduce a small constant $c > 0$ to the numerator and denominator of the estimated density ratio:

$$\min_{\phi} \left\{ -\mathbb{E}_{q^*(x)q_\phi(z|x)}[\log p(x|z)] + \mathbb{E}_{q^*(x)q_\phi(z|x)} \left[\log \frac{D_\alpha(z,x) + c}{1 - D_\alpha(z,x) - c} \right] \right\}.$$

Since our variational posterior density is represented as a generative neural network function, our optimization objectives in Expressions (4.1.2) and (4.1.3) become:

$$\min_{\alpha} \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log D_\alpha(\mathscr{G}_\phi(\epsilon;x),x) - \mathbb{E}_{p_\theta(z)q^*(x)}[\log(1 - D_\alpha(z,x))] \right\}, \tag{4.1.4}$$

$$\min_{\phi} \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log p(x|\mathscr{G}_\phi(\epsilon;x))] + \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{D_\alpha(\mathscr{G}_\phi(\epsilon;x),x)}{1 - D_\alpha(\mathscr{G}_\phi(\epsilon;x),x)} \right] \right\}. \tag{4.1.5}$$

In practice, the algorithm cycles between taking multiple gradient descent steps of the discriminative loss in Expression (4.1.4) with respect to α (with the generator parameters fixed) and taking one gradient descent step of $NELBO(q)$, as stated in Expression (4.1.5) (with the discriminator parameters fixed). If we apply this algorithm to an autoencoder, we would additionally parametrize the likelihood with θ and optimize $NELBO(q)$ with respect to those parameters.

Also note that from Equation (3.4.1) that $NELBO(q)$ has a lower bound of $-p(x)$, which is unknown. It is therefore inadequate to assess convergence using the value of $NELBO(q)$, so we would additionally check that the discriminative loss is arbitrarily close to $\log 4$, using Remark 4.1.3.

Pseudocode for this algorithm is shown in Algorithm 1.

Input: Dataset density $q^*(x)$, (implicit) prior $p(z)$, likelihood $p(x|z)$, noise density $\pi(\epsilon)$

Result: Optimized posterior generator $\mathscr{G}_\phi(\epsilon; x)$

for $j = 1$ **to** J **do**

  for $k = 1$ **to** K **do**

    Sample $\{\epsilon^{(i,k)}\}_{i=1}^B \sim \pi(\epsilon)$;

    Sample $\{x^{(i,k)}\}_{i=1}^B \sim q^*(x)$;

    foreach $\epsilon^{(i,k)}, x^{(i,k)}$ **do**

      Sample $z_q^{(i,k)} = \mathscr{G}(\epsilon^{(i,k)}; x^{(i,k)})$;

    end

    Sample $\{z_p^{(i,k)}\}_{i=1}^B \sim p(z)$;

    update α *by optimization step on*

    $\min_\alpha \left\{ \frac{1}{B} \sum_{i=1}^B \left[-\log D_\alpha \left(z_q^{(i,k)}, x^{(i,k)} \right) - \log \left(1 - D_\alpha \left(z_p^{(i,k)}, x^{(i,k)} \right) \right) \right] \right\}$

  end

  Sample $\{\epsilon^{(i)}\}_{i=1}^B \sim \pi(\epsilon)$;

  Sample $\{x^{(i)}\}_{i=1}^B \sim q^*(x)$;

  foreach $\epsilon^{(i)}, x^{(i)}$ **do**

    Sample $z_q^{(i)} = \mathscr{G}(\epsilon^{(i)}; x^{(i)})$;

  end

  update ϕ *by optimization step on*

  $\min_\phi \left\{ \frac{1}{B} \sum_{i=1}^B \left[-\log p \left(x^{(i)} | z_q^{(i)} \right) + \log \frac{D_\alpha \left(z_q^{(i)}, x^{(i)} \right)}{1 - D_\alpha \left(z_q^{(i)}, x^{(i)} \right)} \right] \right\}$;

end

Algorithm 1: Prior-Contrastive Class Probability Estimation

4.1.3 Joint-Contrastive Algorithm

We now restate from Expression (3.8.1) our optimization objective when the likelihood density is implicit:

$$\min_\phi \left\{ \mathbb{E}_{q^*(x) q_\phi(z|x)} \left[\log \frac{q(z,x)}{p(z,x)} \right] \right\},$$

or

$$\min_\phi \left\{ \mathbb{E}_{q^*(x) q_\phi(z|x)} \left[\log \frac{q_\phi(z|x) q^*(x)}{p(x|z) p(z)} \right] \right\}.$$

Similar to the prior-contrastive case, we can label samples from $q(z, x)$ with $y = 1$ and samples from $p(z, x)$ with $y = 0$, leading to the density ratio expression:

$$\frac{q(z,x)}{p(z,x)} = \frac{P(y=1|z,x)}{P(y=0|z,x)}.$$

Again, we use a discriminator neural network $D_\alpha(z, x) \simeq P(y = 1|z, x)$ to estimate the probability that samples (z, x) came from the joint variational density $q(z, x)$. Using this discriminator function, our density ratio expression becomes:

$$\frac{q(z,x)}{p(z,x)} \simeq \frac{D_\alpha(z,x)}{1 - D_\alpha(z,x)}.$$

The class probability algorithm again cycles between Bernoulli loss minimisation of the discriminator:

$$\min_\alpha \left\{ -\mathbb{E}_{q^*(x)q_\phi(z|x)}[\log D_\alpha(z,x)] - \mathbb{E}_{p(z)p(x|z)}[\log(1 - D_\alpha(z,x))] \right\} \tag{4.1.6}$$

and optimization of the variational posterior:

$$\min_\phi \left\{ \mathbb{E}_{q^*(x)q_\phi(z|x)} \left[\log \frac{D_\alpha(z,x)}{1 - D_\alpha(z,x)} \right] \right\}. \tag{4.1.7}$$

Using the posterior generator parametrization $\mathscr{G}_\phi(\epsilon; x)$, the optimization objectives in Expressions (4.1.6) and (4.1.7) become:

$$\min_\alpha \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log D_\alpha(\mathscr{G}_\phi(\epsilon;x),x)] - \mathbb{E}_{p(z)p(x|z)}[\log(1 - D_\alpha(z,x))] \right\}$$

$$\min_\phi \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{D_\alpha(\mathscr{G}_\phi(\epsilon;x),x)}{1 - D_\alpha(\mathscr{G}_\phi(\epsilon;x),x)} \right] \right\}.$$

Using Remark 4.1.3, it can be seen that when $q(z, x) = p(z, x)$, the optimal generative loss is 0. Pseudocode for the joint-contrastive class probability estimation algorithm is shown in Algorithm 2.

Input: Dataset density $q^*(x)$, (implicit) prior $p(z)$, (implicit) likelihood $p(x|z)$, noise density $\pi(\epsilon)$

Result: Optimized posterior generator $\mathscr{G}_\phi(\epsilon; x)$

for $j = 1$ **to** J **do**

  for $k = 1$ **to** K **do**

    Sample $\{\epsilon^{(i,k)}\}_{i=1}^B \sim \pi(\epsilon)$;

    Sample $\{x_q^{(i,k)}\}_{i=1}^B \sim q^*(x)$;

    foreach $\epsilon^{(i,k)}, x^{(i,k)}$ **do**

      Sample $z_q^{(i,k)} = \mathscr{G}(\epsilon^{(i,k)}; x_q^{(i,k)})$;

    end

    Sample $\{z_p^{(i,k)}\}_{i=1}^B \sim p(z)$;

    foreach $z_p^{(i,k)}$ **do**

      Sample $x_p^{(i,k)} \sim p(x|z_p^{(i,k)})$;

    end

    update α *by optimization step on*
    $\min_\alpha \left\{ \frac{1}{B} \sum_{i=1}^B \left[-\log D_\alpha \left(z_q^{(i,k)}, x_q^{(i,k)} \right) - \log \left(1 - D_\alpha \left(z_p^{(i,k)}, x_p^{(i,k)} \right) \right) \right] \right\}$

  end

  Sample $\{\epsilon^{(i)}\}_{i=1}^B \sim \pi(\epsilon)$;

  Sample $\{x_q^{(i)}\}_{i=1}^B \sim q^*(x)$;

  foreach $\epsilon^{(i)}, x^{(i)}$ **do**

    Sample $z_q^{(i)} = \mathscr{G}(\epsilon^{(i)}; x_q^{(i)})$;

  end

  update ϕ *by optimization step on*
  $\left\{ \min_\phi \frac{1}{B} \sum_{i=1}^B \left[\log \frac{D_\alpha \left(z_q^{(i)}, x_q^{(i)} \right)}{1 - D_\alpha \left(z_q^{(i)}, x_q^{(i)} \right)} \right] \right\}$;

end

Algorithm 2: Joint-Contrastive Class Probability Estimation

4.2 Divergence Minimisation

In this section, we derive the divergence minimisation density ratio estimation algorithm and apply it to our prior-contrastive and joint-contrastive optimisation problems.

4.2.1 Derivation

We first define the *convex conjugate*.

Definition 4.2.1. In $\mathbb{R}^S$, the convex conjugate $f^*(x^*)$ of a function $f(x)$ is:

$$f^*(x^*) \coloneqq \sup\left\{\langle x^*, x\rangle - f(x) | x \in \mathbb{R}^S\right\},$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product.

Now recall from Definition 3.2.1 that an f-divergence is defined by a convex function f such that $f(1) = 0$. A lower bound for the f-divergence in terms of a *direct ratio estimator* $r(u) \simeq q(u)/p(u)$ can be found using the following theorem.

Theorem 4.2.2. *[Nguyen et al., 2010] If f is a convex function with derivative f' and convex conjugate f^*, and $\mathscr{R}$ is a class of functions with codomains equal to the domain of f', then we have the lower bound for the f-divergence between densities $p(u)$ and $q(u)$ in terms of a density ratio estimator $r(u) \in \mathscr{R}$:*

$$D_f(p(u)\|q(u)) \geq \sup_{r\in\mathscr{R}}\{\mathbb{E}_{q(u)}[f'(r(u))] - \mathbb{E}_{p(u)}[f^*(f'(r(u)))]\},$$

with equality when $r(u) = q(u)/p(u)$.

Proof. The proof of this theorem can be found in "Estimating divergence functionals and the likelihood ratio by convex risk minimization" by Nguyen et al. [2010]. □

The derivative and convex conjugate of $f(u) = u \log u$ are

$$f'(u) = 1 + \log u, \qquad f^*(u) = \exp(u - 1),$$

so the convex conjugate of the derivative is simply $f^*(f'(u)) = u$.
Using Theorem 4.2.2, we derive the following reverse KL divergence lower bound:

$$\begin{aligned} D_{RKL}(p(x)\|q(x)) &\coloneqq KL(q(u)\|p(u)) \\ &\geq \sup_{r\in\mathscr{R}}\{\mathbb{E}_{q(u)}[1 + \log r(u)] - \mathbb{E}_{p(u)}[r(u)]\}. \end{aligned} \tag{4.2.1}$$

Fixing the variational density $q(u)$, $KL(q(u)\|p(u))$ is constant, so recalling that the bound reaches equality when $r(u) = q(u)/p(u)$, we can represent the direct ratio estimator as a neural network parametrised by α and minimize the negative of Expression (4.2.1), which we define as L_r, with respect to α:

$$L_r \coloneqq -\mathbb{E}_{q(u)}[\log r_\alpha(u)] + \mathbb{E}_{p(u)}[r_\alpha(u)]. \tag{4.2.2}$$

Note we have removed the $+1$ term as it is independent of α.

Lemma 4.2.3. *The direct ratio estimator reaches optimality at* $r^*_\alpha(u) = q(u)/p(u)$*, minimizing its ratio loss.*

Proof. A proof using the first and second functional derivatives of the ratio loss can be found in Appendix A.6. □

Remark 4.2.4. $r^*_\alpha(u)$ *is bound in* $(0, \infty)$*. Note that* $r^*_\alpha(u) \neq 0$ *as we have assumed that* $p(u) > 0$ *and* $q(u) > 0$ *for all* $u \in U$*.*

Remark 4.2.5. *When the probability densities are equivalent, that is,* $q(u) = p(u)$*, the ratio estimator has a constant output of 1:* $r^*_\alpha(u) = 1$*. This corresponds to a ratio estimator loss of* $L_{r^*} = 1$*.*

Using Remark 4.2.5, we can check for posterior convergence by observing when the ratio estimator loss becomes arbitrarily close to 1.

4.2.2 Prior-Contrastive Algorithm

We begin by restating the optimization objective from Expression (4.1.1):

$$\min_\phi \left\{ -\mathbb{E}_{q^*(x)q_\phi(z|x)}[\log p(x|z)] + \mathbb{E}_{q^*(x)q_\phi(z|x)} \left[\log \frac{q_\phi(z|x)}{p(z)} \right] \right\}.$$

Applying Theorem 4.2.2, we have the lower bound for the reverse KL divergence:

$$KL(q_\phi(z|x) \| p(z)) \geq \sup_{\hat{r} \in \mathscr{R}} \{ \mathbb{E}_{q_\phi(z|x)}[1 + \log r(z)] - \mathbb{E}_{p(z)}[r(z)] \}.$$

Now we let our direct ratio estimator be a neural network parametrized by α, and since $q_\phi(z|x)$ is dependent on the input $x \sim q^*(x)$, we add x as an input and amortize the KL divergence across $q^*(x)$:

$$\mathbb{E}_{q^*(x)} \left[KL(q_\phi(z|x) \| p(z)) \right] \geq \sup_\alpha \{ \mathbb{E}_{q^*(x)q_\phi(z|x)}[1 + \log r_\alpha(z, x)] - \mathbb{E}_{q^*(x)p(z)}[r_\alpha(z, x)] \}.$$

By taking the negative of this lower bound, we formulate the minimisation problem for the direct ratio estimator:

$$\min_\alpha \left\{ -\mathbb{E}_{q^*(x)q_\phi(z|x)}[\log r_\alpha(z, x)] + \mathbb{E}_{q^*(x)p(z)}[r_\alpha(z, x)] \right\}. \tag{4.2.3}$$

Since $r_\alpha(z, x) \simeq \frac{q_\phi(z|x)}{p(z)}$, we write our $NELBO$ optimization problem in terms of the direct ratio estimator:

$$\min_\phi \left\{ -\mathbb{E}_{q^*(x)q_\phi(z|x)} \left[\log p(x|z) \right] + E_{q^*(x)q_\phi(z|x)}[\log r_\alpha(z, x)] \right\}. \tag{4.2.4}$$

Substituting the generator form of the posterior into Expressions (4.2.3) and (4.2.4), we have the bi-level optimization problem:

$$\min_{\alpha} \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log r_\alpha(\mathscr{G}_\phi(\epsilon; x), x)] + \mathbb{E}_{q^*(x)p(z)}[r_\alpha(z, x)] \right\}$$

$$\min_{\phi} \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log p(x|\mathscr{G}_\phi(\epsilon; x)\right] + E_{q^*(x)\pi(\epsilon)}[\log r_\alpha(\mathscr{G}(\epsilon; x), x)] \right\}.$$

Similar to class probability estimation, the algorithm involves cycling between multiple ratio estimator optimisation steps and a single step of $NELBO$ minimisation, as shown in Algorithm 3.

Input: Dataset density $q^*(x)$, (implicit) prior $p(z)$, likelihood $p(x|z)$, noise density $\pi(\epsilon)$

Result: Optimized posterior generator $\mathscr{G}_\phi(\epsilon; x)$

for $j = 1$ **to** J **do**

  for $k = 1$ **to** K **do**

    Sample $\{\epsilon^{(i,k)}\}_{i=1}^B \sim \pi(\epsilon)$;

    Sample $\{x^{(i,k)}\}_{i=1}^B \sim q^*(x)$;

    foreach $\epsilon^{(i,k)}, x^{(i,k)}$ **do**

      Sample $z_q^{(i,k)} = \mathscr{G}(\epsilon^{(i,k)}; x^{(i,k)})$;

    end

    Sample $\{z_p^{(i,k)}\}_{i=1}^B \sim p(z)$;

    update α *by optimization step on*

    $\min_\alpha \left\{ \frac{1}{B} \sum_{i=1}^B \left[-\log \left(r_\alpha \left(z_q^{(i,k)}, x^{(i,k)} \right) \right) + r_\alpha \left(z_p^{(i,k)}, x^{(i,k)} \right) \right] \right\}$;

  end

  Sample $\{\epsilon^{(i)}\}_{i=1}^B \sim \pi(\epsilon)$;

  Sample $\{x^{(i)}\}_{i=1}^B \sim q^*(x)$;

  foreach $\epsilon^{(i)}, x^{(i)}$ **do**

    Sample $z_q^{(i)} = \mathscr{G}(\epsilon^{(i)}; x^{(i)})$;

  end

  update ϕ *by optimization step on*

  $\min_\phi \left\{ \frac{1}{B} \sum_{i=1}^B \left[-\log p \left(x^{(i)} | z_q^{(i)} \right) + \log r_\alpha \left(z_q^{(i)}, x^{(i)} \right) \right] \right\}$;

end

Algorithm 3: Prior-Contrastive Divergence Minimisation

4.2.3 Joint-Contrastive Algorithm

First, we restate the problem from Expression (3.8.1) of minimizing the reverse KL divergence between the joint densities:

$$\min_{\phi} \left\{ \mathbb{E}_{q^*(x)q_\phi(z|x)} \left[\log \frac{q(z,x)}{p(z,x)} \right] \right\}.$$

Applying Theorem 4.2.2, a lower bound for our KL divergence is

$$KL(q(z,x)\|p(z,x)) \geq \sup_{r \in \mathscr{R}} \{ \mathbb{E}_{q(z,x)}[1 + \log r(z,x)] - \mathbb{E}_{p(z,x)}[r(z,x)] \}.$$

Note here we do not have to amortize our ratio estimator as the joint probability density already includes an input from the dataset. Again we set our ratio estimator to take the form of a neural network parametrized by α:

$$r_\alpha(z,x) \simeq \frac{q(z,x)}{p(z,x)},$$

so that the lower bound becomes

$$KL(q(z,x)\|p(z,x)) \geq \sup_{\alpha} \{ \mathbb{E}_{q(z,x)}[1 + \log r_\alpha(z,x)] - \mathbb{E}_{p(z,x)}[r_\alpha(z,x)] \}.$$

Again taking the negative of this lower bound, we formulate the following minimisation problem for the direct ratio estimator:

$$\min_{\alpha} \left\{ -\mathbb{E}_{q^*(x)q_\phi(z|x)}[\log r_\alpha(z,x)] + \mathbb{E}_{p(z)p(x|z)}[r_\alpha(z,x)] \right\}. \tag{4.2.5}$$

The $NELBO$ minimization problem in terms of the direct ratio estimator is:

$$\min_{\phi} \left\{ \mathbb{E}_{q^*(x)q_\phi(z|x)}[\log r_\alpha(z,x)] \right\}. \tag{4.2.6}$$

Finally, we derive the bi-level optimization problem by writing the variational posterior term in Expressions (4.2.5) and (4.2.6) in terms of their generator form:

$$\min_{\alpha} \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log r_\alpha(\mathscr{G}(\epsilon;x),x)] + \mathbb{E}_{p(z)p(x|z)}[r_\alpha(z,x)] \right\}$$

$$\min_{\phi} \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)}[\log r_\alpha(\mathscr{G}(\epsilon;x),x)] \right\}.$$

Again this process, shown in Algorithm 4, involves cycling between multiple steps of optimizing the ratio estimator and taking a single optimization step of the $NELBO$.

Input: Dataset density $q^*(x)$, (implicit) prior $p(z)$, (implicit) likelihood $p(x|z)$, noise density $\pi(\epsilon)$

Result: Optimized posterior generator $\mathscr{G}_\phi(\epsilon; x)$

for $j = 1$ **to** J **do**

  for $k = 1$ **to** K **do**

    Sample $\{\epsilon^{(i,k)}\}_{i=1}^B \sim \pi(\epsilon)$;

    Sample $\{x_q^{(i,k)}\}_{i=1}^B \sim q^*(x)$;

    foreach $\epsilon^{(i,k)}, x_q^{(i,k)}$ **do**

      Sample $z_q^{(i,k)} = \mathscr{G}(\epsilon^{(i,k)}; x_q^{(i,k)})$;

    end

    Sample $\{z_p^{(i,k)}\}_{i=1}^B \sim p(z)$;

    foreach $z_p^{(i,k)}$ **do**

      Sample $x_p^{(i,k)} \sim p(x|z_p^{(i,k)})$;

    end

    update α *by optimization step on*

    $\min_\alpha \left\{ \frac{1}{B} \sum_{i=1}^B \left[-\log\left(r_\alpha\left(z_q^{(i,k)}, x_q^{(i,k)} \right) \right) + r_\alpha\left(z_p^{(i,k)}, x_p^{(i,k)} \right) \right] \right\}$;

  end

  Sample $\{\epsilon^{(i)}\}_{i=1}^B \sim \pi(\epsilon)$;

  Sample $\{x_q^{(i)}\}_{i=1}^B \sim q^*(x)$;

  foreach $\epsilon^{(i)}, x_q^{(i)}$ **do**

    Sample $z_q^{(i)} = \mathscr{G}(\epsilon^{(i)}; x_q^{(i)})$;

  end

  update ϕ *by optimization step on*

  $\min_\phi \left\{ \frac{1}{B} \sum_{i=1}^B \left[\log r_\alpha\left(z_q^{(i)}, x_q^{(i)} \right) \right] \right\}$;

end

Algorithm 4: Joint-Contrastive Divergence Minimisation

4.2.4 Alternative Derivation of Class Probability Estimation

If we apply Theorem 4.2.2 with the function $f(u) = u \log u - (u+1)\log(u+1)$, the resulting bi-level optimization problem is equivalent to that of class probability estimation [Tiao et al., 2018]. Before deriving the corresponding f-divergence, we define the *Jenson-Shannon* divergence which will be used in the expression.

Definition 4.2.6. The Jensen-Shannon divergence is a symmetric metric used to measure the discrepancy between two probability densities [Fuglede and Topsoe, 2004]:

$$JS(p(u)\|q(u)) = \frac{1}{2}KL(p(u)\|m(u)) + \frac{1}{2}KL(q(u)\|m(u)),$$

where $m(u) = (q(u) + p(u))/2$.

We denote this f-divergence as D_{GAN} after the *generative adversarial network* models which typically use the class probability estimation optimisation problem [Goodfellow et al., 2014].

$$\begin{aligned}
D_{GAN}(p(u)\|q(u)) &= \mathbb{E}_{p(u)}\left[\frac{q(u)}{p(u)}\log\left(\frac{q(u)}{p(u)}\right) - \left(\frac{q(u)}{p(u)}+1\right)\log\left(\frac{q(u)}{p(u)}+1\right)\right] \\
&= \int_U q(u)\log\left(\frac{q(u)}{p(u)}\right) - (q(u)+p(u))\log\left(\frac{q(u)+p(u)}{p(u)}\right)\,\mathrm{d}u \\
&= \int_U q(u)(\log q(u) - \log p(u)) - (q(u)+p(u))\log(q(u)+p(u)) \\
&\qquad + (q(u)+p(u))\log p(u)\;\;\mathrm{d}u \\
&= \int_U q(u)\log\left(\frac{q(u)}{q(u)+p(u)}\right) + p(u)\log\left(\frac{p(u)}{q(u)+p(u)}\right)\,\mathrm{d}u \\
&= \int_U p(u)\log\left(\frac{2p(u)}{q(u)+p(u)}\right) - p(u)\log 2 + q(u)\log\left(\frac{2q(u)}{q(u)+p(u)}\right) \\
&\qquad - q(u)\log 2\;\;\mathrm{d}u \\
&= \int_U p(u)\left(\log\left(\frac{p(u)}{m(u)}\right) - \log 2\right) + q(u)\left(\log\left(\frac{q(u)}{m(u)}\right) - \log 2\right)\,\mathrm{d}u \\
&\text{where } m(u) = \frac{q(u)+p(u)}{2} \\
&= \mathbb{E}_{p(u)}\left[\log\left(\frac{p(u)}{m(u)}\right) - \log 2\right] + \mathbb{E}_{q(u)}\left[\log\left(\frac{q(u)}{m(u)}\right) - \log 2\right] \\
&= KL(p(u)\|m(u)) + KL(q(u)\|m(u)) - \log 4 \\
&= 2JS(p(u)\|q(u)) - \log 4.
\end{aligned}$$

Recall one of the conditions of f to form an f-divergence is $f(1) = 0$, but in this case, $f(1) = -\log 4$, hence the constant term in our GAN divergence.
The derivative and convex conjugate of $f(u) = u\log u - (u+1)\log(u+1)$ are

$$f'(u) = \log\left(\frac{u}{u+1}\right), \qquad f^*(u) = -\log(1-\exp u),$$

so the convex conjugate of the derivative is

$$f^*(f'(u)) = \log(u+1).$$

Using Theorem 4.2.2, we derive the following lower bound for the Jensen-Shannon divergence:

$$\begin{aligned} D_{GAN}(p(u)\|q(u)) &\coloneqq 2JS(p(u)\|q(u)) - \log 4 \\ &\geq \sup_{r\in\mathscr{R}} \left\{ \mathbb{E}_{q(u)} \left[\log\left(\frac{r(u)}{r(u)+1} \right) \right] - \mathbb{E}_{p(u)}[\log(r(u)+1)] \right\} \\ &= \sup_{D} \{ \mathbb{E}_{q(u)}[\log D(u)] + \mathbb{E}_{p(u)}[\log(1-D(u))] \}, \end{aligned}$$

where $D(u) = r(u)/(r(u)+1)$. Maximisation of the lower bound, equivalent to minimising its negative, follows the optimization problem:

$$\min_{\alpha} \left\{ -\mathbb{E}_{q(u)}[\log D(u)] - \mathbb{E}_{p(u)}[\log(1-D(u))] \right\},$$

which is the discriminative loss of the class probability estimation approach. Equality of the bound is accomplished at the optimal function of

$$\begin{aligned} D^*(u) &= \frac{\frac{q(u)}{p(u)}}{\frac{q(u)}{p(u)}+1} \\ &= \frac{q(u)}{q(u)+p(u)}, \end{aligned}$$

which is the optimal discriminator of class probability estimation. Thus, the density ratio in terms of this discriminator function is

$$\frac{q(u)}{p(u)} \simeq \frac{D(u)}{1-D(u)}.$$

Therefore, our previous formulation of class probability estimation by minimizing the Bernoulli loss of a discriminator function is equivalent to maximizing the lower bound of the "GAN" divergence between the two densities.

Chapter 5

Algorithm Generalisation

In this chapter, we present the two density ratio estimation techniques in a comparable form, that is, in terms of the relevant f-divergence's lower bound as derived in Section 4.2 using Theorem 4.2.2. From this, we generalise the density ratio algorithms to a selection of an f-divergence to formulate the lower bound and a parametrisation of the density ratio estimator. We also suggest a third estimator parametrisation: the *direct log density ratio estimator* $T_\alpha(u) \simeq \log(q(u)/p(u))$. The formal generalisation of density ratio estimation algorithms is the first contribution of this thesis.

5.1 Introduction

First, recall that within each of the prior-contrastive and joint-contrastive formulations, both class probability estimation and divergence minimisation methods use the same $NELBO$ loss function to optimise the posterior weights. It is therefore evident that the only differences between the two density ratio estimation techniques are the estimator loss functions used. As shown in Subsections 4.2.1 and 4.2.4, these estimator losses can be derived using Theorem 4.2.2, hence the only differences between them lie within their f-divergence bound and the parametrization of the estimator. We reiterate these derivations below.

For divergence minimisation, the following lower bound is achieved using the reverse KL divergence $D_{RKL}(p(u)\|q(u)) = KL(q(u)\|p(u))$:

$$D_{RKL}(p(u)\|q(u)) \geq \sup_{\alpha}\{\mathbb{E}_{q(u)}[1 + \log r_\alpha(u)] - \mathbb{E}_{p(u)}[r_\alpha(u)]\}, \tag{5.1.1}$$

and class probability estimation follows the GAN divergence $D_{GAN}(p(u)\|q(u)) = 2JS(p(u)\|q(u)) - \log 4$:

$$D_{GAN}(p(u)\|q(u)) \geq \sup_{\alpha}\{\mathbb{E}_{q(u)}\left[\log \frac{r_\alpha(u)}{r_\alpha(u)+1}\right] - \mathbb{E}_{p(u)}[\log(r_\alpha(u)+1)]\}, \tag{5.1.2}$$

with equality of both equations at $r_\alpha(u) = q(u)/p(u)$.
To convert the Equation (5.1.2) to the class probability estimation loss function, we have used the estimator transformation $r_\alpha(u) = \frac{D_\alpha(u)}{1-D_\alpha(u)} \iff D_\alpha(u) = \frac{r_\alpha(u)}{r_\alpha(u)+1}$. Since this mapping is bijective and monotonically increasing, the equality of the bound is retained at equivalent estimator parametrisations. We can therefore also use this transformation on Equation (5.1.1) deriving its respective loss function as a function of $D_\alpha(u) \simeq q(u)/(q(u) + p(u))$ instead of $r_\alpha(u) \simeq q(u)/p(u)$.

Example 5.1.1. In this example, we represent the estimator loss function formulated by the reverse KL divergence in terms of the class probability estimator $D_\alpha(u)$:

$$\begin{aligned} D_{RKL}(p(u)\|q(u)) &\geq \sup_\alpha \{\mathbb{E}_{q(u)}[1 + \log r_\alpha(u)] - \mathbb{E}_{p(u)}[r_\alpha(u)]\} \\ &\geq \sup_\alpha \left\{ \mathbb{E}_{q(u)} \left[1 + \log \frac{D_\alpha(u)}{1 - D_\alpha(u)} \right] - \mathbb{E}_{p(u)} \left[\frac{D_\alpha(u)}{1 - D_\alpha(u)} \right] \right\}. \end{aligned} \tag{5.1.3}$$

By taking the negative of Expression (5.1.3) and eliminating the constant +1 term, we form the following minimisation problem:

$$\min_\alpha \left\{ \mathbb{E}_{q(u)} \left[\log \frac{1 - D_\alpha(u)}{D_\alpha(u)} \right] + \mathbb{E}_{p(u)} \left[\frac{D_\alpha(u)}{1 - D_\alpha(u)} \right] \right\}.$$

We also propose a third estimator parametrisation, the direct log density ratio estimator:

$$T_\alpha(u) \simeq \log \frac{q(u)}{p(u)}.$$

This is derived by simply using the transformation $T_\alpha(u) = \log r_\alpha(u)$.

5.2 Algorithm Generalisation

For each f-divergence, we can derive a loss function for each of our three estimator parametrisations.

5.2.1 Reverse KL Divergence

Direct Ratio Estimator $r_\alpha(u)$:

$$\min_\alpha \left\{ -\mathbb{E}_{q(u)}[\log r_\alpha(u)] + \mathbb{E}_{p(u)}[r_\alpha(u)] \right\}.$$

Class Probability Estimator/Discriminator $D_\alpha(u)$:

$$\min_\alpha \left\{ \mathbb{E}_{q(u)} \left[\log \frac{1 - D_\alpha(u)}{D_\alpha(u)} \right] + \mathbb{E}_{p(u)} \left[\frac{D_\alpha(u)}{1 - D_\alpha(u)} \right] \right\}.$$

Direct Log Ratio Estimator $T_\alpha(u)$:

$$\min_\alpha \left\{ -\mathbb{E}_{q(u)}[T_\alpha(u)] + \mathbb{E}_{p(u)} \left[e^{T_\alpha(u)} \right] \right\}.$$

5.2.2 GAN Divergence

Direct Ratio Estimator $r_\alpha(u)$:

$$\min_\alpha \left\{ \mathbb{E}_{q(u)} \left[\log \frac{r_\alpha(u) + 1}{r_\alpha(u)} \right] + \mathbb{E}_{p(u)} [\log(r_\alpha(u) + 1)] \right\}.$$

Class Probability Estimator/Discriminator $D_\alpha(u)$:

$$\min_\alpha \left\{ -\mathbb{E}_{q(u)}[\log D_\alpha(u)] - \mathbb{E}_{p(u)}[\log(1 - D_\alpha(u))] \right\}.$$

Direct Log Ratio Estimator $T_\alpha(u)$:

$$\min_\alpha \left\{ \mathbb{E}_{q(u)} \left[\log \frac{e^{T_\alpha(u)} + 1}{e^{T_\alpha(u)}} \right] + \mathbb{E}_{p(u)} \left[\log \left(e^{T_\alpha(u)} + 1 \right) \right] \right\}.$$

We have therefore generalised the choice of algorithm for training density ratio estimators to a choice of f-divergence:

- Reverse KL Divergence: $D_{KL}(p(u)\|q(u)) = \mathbb{E}_{q(u)} \left[1 + \log \frac{q(u)}{p(u)} \right] - \mathbb{E}_{p(u)} \left[\frac{q(u)}{p(u)} \right]$,
- GAN Divergence: $D_{GAN}(p(u)\|q(u)) = \mathbb{E}_{q(u)} \left[\log \frac{q(u)}{q(u)+p(u)} \right] + \mathbb{E}_{p(u)} \left[\log \frac{p(u)}{q(u)+p(u)} \right]$,

and a choice of estimator parametrization:

- Direct ratio estimator: $r_\alpha(u) \simeq \frac{q(u)}{p(u)}$,
- Class probability estimator/Discriminator: $D_\alpha(u) \simeq \frac{q(u)}{q(u)+p(u)}$,
- Direct log ratio estimator: $T_\alpha(u) \simeq \log \frac{q(u)}{p(u)}$.

The original "KL divergence minimization" approach simply chooses the reverse KL Divergence and the direct ratio estimator, whilst "class probability estimation" uses the GAN Divergence and the class probability estimator. These are just two variations of the six available algorithms.

5.3 Optimization Algorithms

The ratio estimator forms in the posterior loss functions also have to be transformed accordingly. In the following subsections we give the specific prior-contrastive and joint-contrastive $NELBO$s as functions of each estimator parametrisation. For completeness, we also list the two different estimator loss functions for each case, corresponding to the two f-divergences that we have discussed.

5.3.1 Prior-Contrastive Loss Functions

Direct Ratio Estimator $r_\alpha(u)$:

$$\text{Reverse KL: } \min_\alpha \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log r_\alpha(\mathscr{G}(\epsilon; x), x)] + \mathbb{E}_{p(z)q^*(x)}[r_\alpha(z, x)] \right\}.$$

$$\text{GAN: } \min_\alpha \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{r_\alpha(\mathscr{G}(\epsilon; x), x) + 1}{r_\alpha(\mathscr{G}(\epsilon; x), x)} \right] + \mathbb{E}_{p(z)q^*(x)}[\log(r_\alpha(z, x) + 1)] \right\}.$$

$$\min_\phi \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log p(x|\mathscr{G}_\phi(\epsilon; x))] + \mathbb{E}_{q^*(x)\pi(\epsilon)}[\log r_\alpha(\mathscr{G}(\epsilon; x), x)] \right\}.$$

Class Probability Estimator/Discriminator $D_\alpha(u)$:

$$\text{Reverse KL: } \min_\alpha \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{1 - D_\alpha(\mathscr{G}(\epsilon; x), x)}{D_\alpha(\mathscr{G}(\epsilon; x), x)} \right] + \mathbb{E}_{p(z)q^*(x)} \left[\frac{D_\alpha(\mathscr{G}(\epsilon; x), x)}{1 - D_\alpha(\mathscr{G}(\epsilon; x), x)} \right] \right\}.$$

$$\text{GAN: } \min_\alpha \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log D_\alpha(\mathscr{G}(\epsilon; x), x)] - \mathbb{E}_{p(z)q^*(x)}[\log(1 - D_\alpha(z, x))] \right\}.$$

$$\min_\phi \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log p(x|\mathscr{G}_\phi(\epsilon; x))] + \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{D_\alpha(\mathscr{G}(\epsilon; x), x)}{1 - D_\alpha(\mathscr{G}(\epsilon; x), x)} \right] \right\}.$$

Direct Log Ratio Estimator $T_\alpha(u)$:

$$\text{Reverse KL: } \min_\alpha \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[T_\alpha(\mathscr{G}(\epsilon; x), x)] + \mathbb{E}_{p(z)q^*(x)} \left[e^{T_\alpha(z,x)} \right] \right\}.$$

$$\text{GAN: } \min_\alpha \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{e^{T_\alpha(\mathscr{G}(\epsilon;x),x)} + 1}{e^{T_\alpha(\mathscr{G}(\epsilon;x),x)}} \right] + \mathbb{E}_{p(z)q^*(x)}[\log(e^{T_\alpha(z,x)} + 1)] \right\}.$$

$$\min_\phi \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log p(x|\mathscr{G}_\phi(\epsilon; x))] + \mathbb{E}_{q^*(x)\pi(\epsilon)}[T_\alpha(\mathscr{G}(\epsilon; x), x)] \right\}.$$

5.3.2 Joint-Contrastive Loss Functions

Direct Ratio Estimator $r_\alpha(u)$:

$$\text{Reverse KL: } \min_\alpha \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)}[\log r_\alpha(\mathscr{G}(\epsilon; x), x)] + \mathbb{E}_{p(z)p(x|z)}[r_\alpha(z, x)] \right\}.$$

$$\text{GAN: } \min_\alpha \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{r_\alpha(\mathscr{G}(\epsilon; x), x) + 1}{r_\alpha(\mathscr{G}(\epsilon; x), x)} \right] + \mathbb{E}_{p(z)p(x|z)}[\log(r_\alpha(z, x) + 1)] \right\}.$$

$$\min_{\phi} \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log r_\alpha(\mathscr{G}_\phi(\epsilon; x), x) \right] \right\}.$$

Class Probability Estimator/Discriminator $D_\alpha(u)$:

$$\text{Reverse KL: } \min_{\alpha} \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{1 - D_\alpha(\mathscr{G}(\epsilon; x), x)}{D_\alpha(\mathscr{G}(\epsilon; x), x)} \right] + \mathbb{E}_{p(z)p(x|z)} \left[\frac{D_\alpha(\mathscr{G}(\epsilon; x), x)}{1 - D_\alpha(\mathscr{G}(\epsilon; x), x)} \right] \right\}.$$

$$\text{GAN: } \min_{\alpha} \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)} [\log D_\alpha(\mathscr{G}(\epsilon; x), x)] - \mathbb{E}_{p(z)p(x|z)} [\log(1 - D_\alpha(z, x))] \right\}.$$

$$\min_{\phi} \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{D_\alpha(\mathscr{G}_\phi(\epsilon; x), x)}{1 - D_\alpha(\mathscr{G}_\phi(\epsilon; x), x)} \right] \right\}.$$

Direct Log Ratio Estimator $T_\alpha(u)$:

$$\text{Reverse KL: } \min_{\alpha} \left\{ -\mathbb{E}_{q^*(x)\pi(\epsilon)} [T_\alpha(\mathscr{G}(\epsilon; x), x)] + \mathbb{E}_{p(z)p(x|z)} \left[e^{T_\alpha(z,x)} \right] \right\}.$$

$$\text{GAN: } \min_{\alpha} \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[\log \frac{e^{T_\alpha(\mathscr{G}(\epsilon;x),x)} + 1}{e^{T_\alpha(\mathscr{G}(\epsilon;x),x)}} \right] + \mathbb{E}_{p(z)p(x|z)} \left[\log \left(e^{T_\alpha(z,x)} + 1 \right) \right] \right\}.$$

$$\min_{\phi} \left\{ \mathbb{E}_{q^*(x)\pi(\epsilon)} \left[T_\alpha(\mathscr{G}_\phi(\epsilon; x), x) \right] \right\}.$$

Chapter 6

Comparing Optimal Estimators

In this chapter, we verify that for a fixed f-divergence, each estimator parametrisation leads to similar levels of convergence when optimally trained. In this experiment, we also determine whether the f-divergence used to derive the estimator loss function has any effect on posterior convergence.

6.1 Theory

It is intuitive that the rate of posterior convergence improves with the accuracy of the $NELBO$ estimation, dependent on how well the density ratio estimator is trained. Recall from Section 5.1 that the estimator transformation preserves the equality of the bound in Theorem 4.2.2. Therefore each parametrisation should reach optimality when optimized sufficiently, estimating the $NELBO$ with minimal error. Consequently, there should be no notable difference in posterior convergence.

Theorem 4.2.2 states that equality of the bound is attained at $r^*_\alpha(u) = \frac{q(u)}{p(u)}$ regardless of the f-divergence used, so we also expect similar levels of convergence between the f-divergences. However, "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization" by Nowozin et al. [2016] shows experimentally that the fastest posterior convergence is attained when the f-divergence used to optimize the variational posterior is also used to derive the lower bound. This implies that in our context, estimator loss functions formulated by the reverse KL divergence correspond to superior posterior convergence.

6.2 Problem Context

We use the basic "Continuous Sprinkler" posterior inference experiment as described in "Variational Inference using Implicit Densities" by Huszár [2017]. This experiment involves the hypothetical scenario of grass wetness as a result of two independent sources: rainfall and a sprinkler. The problem exhibits "*explaining away*" [Wellman and Henrion, 1993], a pattern of reasoning in which the confirmation of one cause of an effect reduces the need to discuss alternative causes. In this example,

despite the two possible sources of wetness being independent, they demonstrate dependence when we condition on the observation of wet grass. This is because grass wetness is correlated with the significance of either water source.

The latent variables $\boldsymbol{z} = (z_1, z_2)$ represent the sprinkler and the rain respectively, and the observation x represents the wetness of the grass. We assume that the levels of incoming water are normally distributed, and that the grass wetness increases with these values. The prior and likelihood distributions are as follows:

$$(z_1, z_2) \sim \mathcal{N}(0, \sigma^2 I_{2\times 2}),$$

$$x|\boldsymbol{z} \sim Exp(3 + \max\{0, z_1\}^3 + \max\{0, z_2\}^3).$$

The aim of this experiment is to compare the density ratio estimation algorithms in the use of amortized variational inference to estimate the posterior distribution. For increased reliability, we perform this experiment in both prior-contrastive and joint-contrastive contexts. The prior-contrastive algorithm requires the likelihood function:

$$p(x|\boldsymbol{z}) = -(3+\max\{0, z_1\}^3+\max\{0, z_2\}^3)-\exp\left(\frac{x}{3 + \max\{0, z_1\}^3 + \max\{0, z_2\}^3}\right).$$

This explicit form is not directly used for the joint-contrastive algorithms; it is only used to generate samples from the likelihood density.
The unnormalised true posterior density function is derived as shown below.
Posterior Calculation:

$$\begin{aligned} p(z_1, z_2|x) &\propto \exp(\log p(z_1, z_2) + \log p(x|z_1, z_2)) \\ &\propto \exp\left(\frac{1}{2}\boldsymbol{z}^T(\sigma^{-2}I_{2\times 2})\boldsymbol{z} - \log(3 + \max\{0, z_1\}^3 + \max\{0, z_2\}^3)\right. \\ &\quad \left. - \frac{x}{3 + \max\{0, z_1\}^3 + \max\{0, z_2\}^3}\right) \\ &\propto \exp\left(\frac{1}{2\sigma^2}(z_1^2 + z_2^2) - \log(3 + \max\{0, z_1\}^3 + \max\{0, z_2\}^3)\right. \\ &\quad \left. - \frac{x}{3 + \max\{0, z_1\}^3 + \max\{0, z_2\}^3}\right). \end{aligned} \tag{6.2.1}$$

In the experiment, we let $\sigma^2 = 2$, and assume that each of the observation values $x = 0, 5, 8, 12, 50$ have the same frequency. Our data density is therefore:

$$q^*(x) = \frac{1}{5}\left(\mathbb{1}_{\{0\}}(x) + \mathbb{1}_{\{5\}}(x) + \mathbb{1}_{\{8\}}(x) + \mathbb{1}_{\{12\}}(x) + \mathbb{1}_{\{50\}}(x)\right).$$

In Figure 6.1, we plot the true (unnormalised) posterior $p(\boldsymbol{z}|x)$ over a grid of equally spaced points from $(z_1, z_2) = \{(-5, -5), \ldots, (5, 5)\}$. We are not concerned with the lack of normalisation as the purpose of the plots is to depict the shape and relative density of the posterior density.

Figure 6.1: This figure depicts the true (unnormalised) posterior plots for the "Continuous Sprinkler" experiment. As x increases, the posteriors become increasingly bi-modal and unusually shaped.

6.3 Program Structure

Recall from Subsection 3.8.1 that our generator $\mathcal{G}_\phi(x, \epsilon)$ is a neural network that takes in noise $\epsilon \sim \pi(\epsilon)$ along with a data sample $x \sim q^*(x)$, outputting a sample from the variational posterior density $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|x)$. In this experiment, the generator has 3 noise inputs $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \epsilon_3)^\top \sim \mathcal{N}(0, I_{3\times3})$ along with 1 data input x, and 2 posterior outputs z_1, z_2. The structure of this network is as depicted in Figure 6.2. In the figure, the number inside the node indicates how many nodes the layer has, and the text above the node describes the activation function. Recall that the input layer does not have an activation function, and that Rectified Linear Units (ReLU) are used for most of the hidden layers due to their many advantages. In the Concat. layer, the two input vectors are concatenated and there is no activation function.

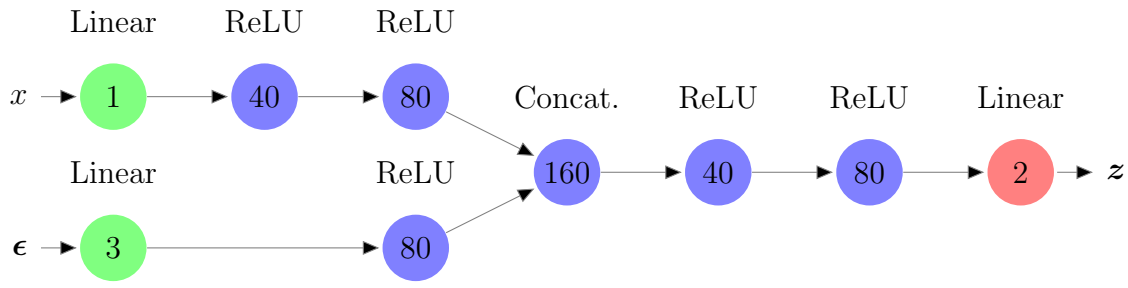


Figure 6.2: This figure illustrates the structure of the generator network $\mathcal{G}(x, \boldsymbol{\epsilon})$ used in the "Continuous Sprinkler" experiment. We do not use an activation function for the output layer as $\boldsymbol{z} \in \mathbb{R}^2$.

The structure of the estimator (discriminator $D_\alpha(\boldsymbol{z}, x)$, direct ratio estimator $r_\alpha(\boldsymbol{z}, x)$ or direct log ratio estimator $T_\alpha(\boldsymbol{z}, x)$) is similar to that of the generator,

with an additional hidden layer associated with the $\boldsymbol{z}$ input and a different output activation function corresponding to the estimator's identity. When the estimator takes the parametrisation of a discriminator $D_\alpha(z, x)$, a sigmoid output layer is used [Goodfellow et al., 2014]. An exponential activation function $g(x) = \mathrm{e}^x$ is used for the output layer when the estimator output is the direct density ratio $r_\alpha(z, x)$, and a linear output is used for the direct log density ratio estimator $T_\alpha(z, x)$ [Nowozin et al., 2016]. The estimator structure is shown in Figure 6.3.

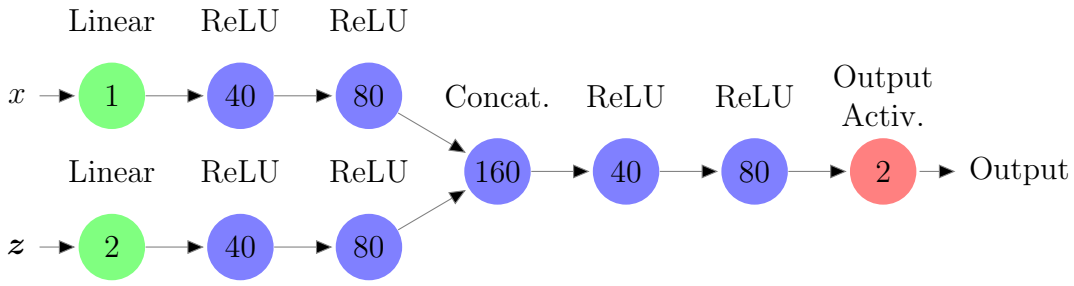


Figure 6.3: This figure depicts the structure of the estimator network for the "Continuous Sprinkler Experiment". The notation is the same as in Figure 6.2. Note that the only differences between the estimators are the activation function of their output layers and the loss functions used to train them.

The network weights are initialized with Xavier initialization (as explained in Section 2.5), and trained using the Adam optimizer (described in Section 2.6) with learning rate 0.0001 for the prior-contrastive setting and 0.00001 for the joint-contrastive algorithms. These values were determined via trial and error to yield consistent results. Recall from the nature of our dataset density $q^*(x)$ that we only train the network on data values $x = 0, 5, 8, 12, 50$. In each training iteration, 200 samples are taken for each x-value, corresponding to a batch size of 1000. This large batch size makes training more consistent, as the latent variables $\boldsymbol{z}$ are probabilistic in nature. Due to the similar network structure in each algorithm, they have near-identical runtime, as the majority of the training time is spent in back-propagation [Goodfellow et al., 2016].

To prevent taking $\log 0$ in the loss functions, we add a constant $c = 10^{-18}$ to the log function's input. Preliminary tests show that this sufficiently small constant does not have an apparent effect on the program performance. The estimator is pre-trained for 5000 iterations to ensure that optimization of the variational network begins with an optimal estimator. The variance of the results is also reduced as the initial parametrisation of the estimator is similar for each fixed generator initialization. Afterwards, the variational posterior is optimized for 10000 iterations for the prior-contrastive experiment, and 40000 iterations in the joint-contrastive

formulation. Again, these values were chosen through trial and error to result in sufficient posterior convergence. Between each of these iterations, the algorithm takes 100 training steps of the estimator. Pseudocode for the prior-contrastive and joint-contrastive algorithms, based off their derivations in Chapters 4 and 5, are shown in Algorithms 7 and 8 in Appendix B.3.

We evaluate the algorithms by comparing the level of posterior convergence. To measure this, we use the average 'true' KL divergence between the true and variational posteriors at the end of the program's runtime:

$$\mathbb{E}_{q^*(x)}\left[KL(q_\phi(\boldsymbol{z}|x)\|p(\boldsymbol{z}|x))\right].$$

To do so, we estimate the probability density function of the variational posterior for each of the 5 data points using a Gaussian kernel density estimator $\hat{q}(\boldsymbol{z}|x)$. We then calculate its expected KL divergence with the unnormalised true posterior derived in Equation (6.2.1). The normalisation constant $p(x)$ is independent of the variational posterior $q_\phi(\boldsymbol{z}|x)$ that we compare it with, so its omission does not affect our comparison of the algorithms. An explanation of kernel density estimation can be found in Appendix E. Due to the stochastic nature of the optimisation, it is desirable to have a large number of experiment repetitions for reliable results. However, due to limited computational time, each algorithm was only run 30 times. At the end of each posterior optimisation step, both the estimator loss and $NELBO$ estimation were stored. Additionally, the 'true' KL divergence was estimated every 100 posterior iterations. Arrays containing these values were averaged over the 30 experiment repetitions, and plotted as shown in Section 6.4.

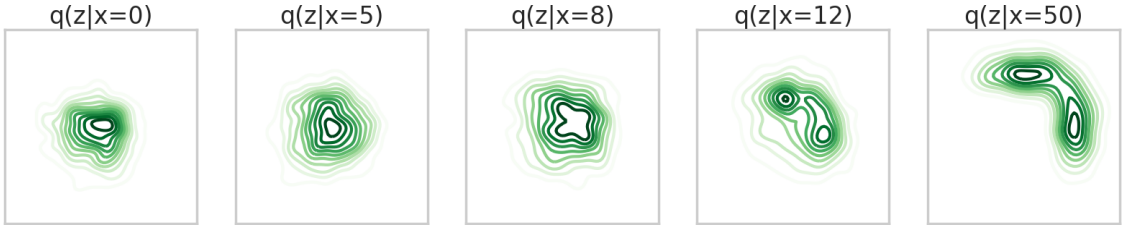
6.4 Results

Tables 6.1 and 6.2 compare the 'true' KL divergence between the 30 iterations for each algorithm variation. Figure 6.4 compares variational posterior output plots corresponding to different 'true' KL divergences. In the prior-contrastive context, there is no major difference in posterior convergence for the algorithms, and the variation in the results is very low. However, Figure 6.4 suggests that the variational posterior reaches optimality at the 'true' KL divergence of around 1.326. This was verified by running a prior-contrastive algorithm for twice as long (20000 posterior iterations) and observing that the metric did not decrease any further. We can therefore deduce that all of the prior-contrastive programs reached optimality, so we cannot draw any conclusions from those results. It is intuitive that explicit

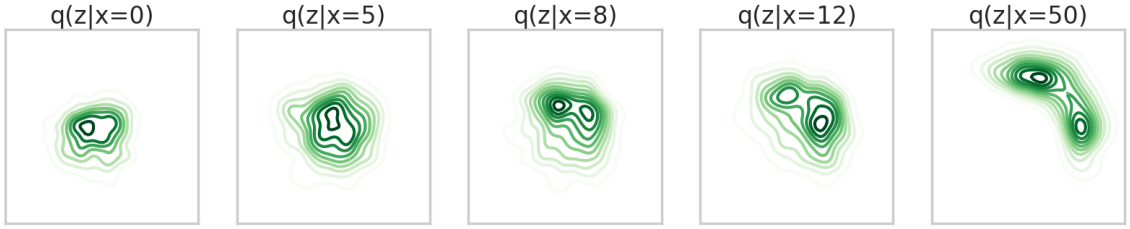
knowledge of the likelihood density in the prior-contrastive algorithm leads to faster posterior convergence than in the joint-contrastive experiments. Plots corresponding to the prior-contrastive optimal estimator experiment can be found in Appendix F.

Figures 6.5-6.7 correspond to the average KL divergence, estimator loss and estimated $NELBO$ over the runtime of the program for the joint-contrastive cases. In these figures, there is little relative difference between the estimator parametrisations. Note that the relative trend in the estimator loss plot is reflected in the corresponding $NELBO$ plot.

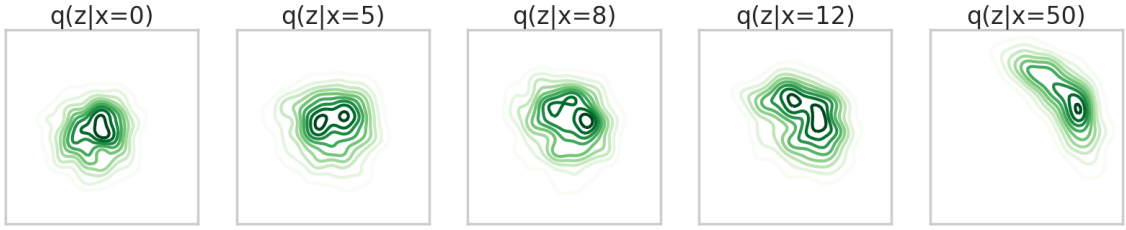
(a) True Posterior Plot



(b) Variational Posterior Plot with 'True' KL Divergence of 1.3288



(c) Variational Posterior Plot with 'True' KL Divergence of 1.3623



(d) Variational Posterior Plot with 'True' KL Divergence of 1.3963

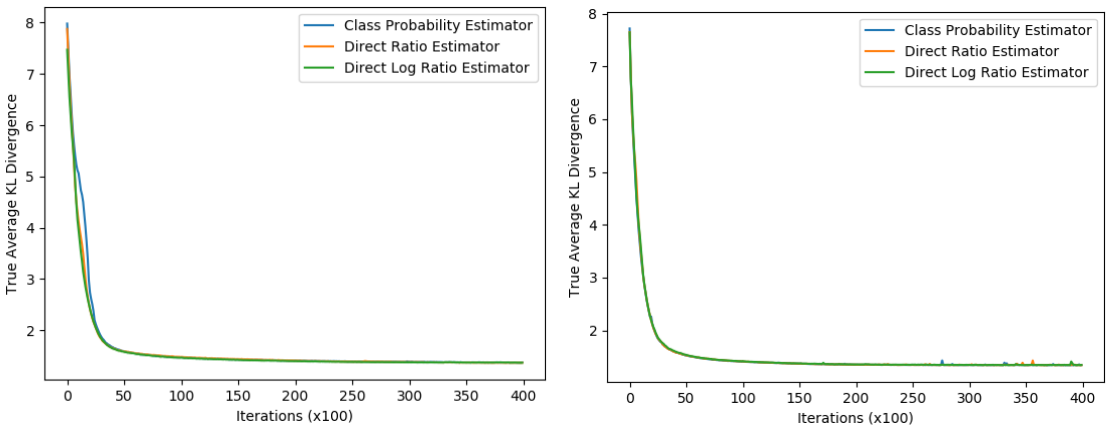
Figure 6.4: Above we compare the true (unnormalised) posterior as plotted in Figure 6.1 against variational posterior outputs corresponding to varying KL divergences. Sub-figures (b)-(d) were created by fitting a Gaussian kernel density estimation over 1000 posterior samples. Note that the variational posterior is relatively optimal at a 'true' KL divergence of 1.3288, and that the output appears less flexible as the divergence increases. In sub-figure (d), the variational posterior fails to capture the bimodality of the true posterior.

Prior-Contrastive Results			
Algorithm		Mean KL Divergence	Standard Deviation
Reverse KL	$D_\alpha(\boldsymbol{z}, x)$	1.3271	0.0041
	$r_\alpha(\boldsymbol{z}, x)$	1.3265	0.0045
	$T_\alpha(\boldsymbol{z}, x)$	1.3262	0.0041
GAN	$D_\alpha(\boldsymbol{z}, x)$	1.3267	0.0041
	$r_\alpha(\boldsymbol{z}, x)$	1.3263	0.0035
	$T_\alpha(\boldsymbol{z}, x)$	1.3258	0.0039

Table 6.1: This table presents the results from the prior-contrastive experiment. Since the programs reached optimality in this case, it is uncertain from those results whether the choice of f-divergence or estimator parametrisation impacts posterior convergence.

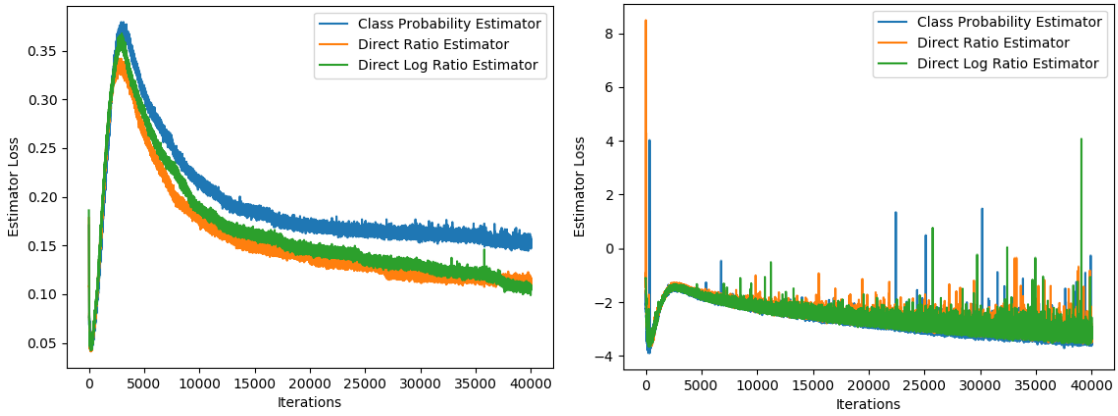
Joint-Contrastive Results			
Algorithm		Mean KL Divergence	Standard Deviation
Reverse KL	$D_\alpha(\boldsymbol{z}, x)$	**1.3416**	**0.0068**
	$r_\alpha(\boldsymbol{z}, x)$	**1.3397**	**0.0066**
	$T_\alpha(\boldsymbol{z}, x)$	**1.3446**	0.0108
GAN	$D_\alpha(\boldsymbol{z}, x)$	1.3648	0.0242
	$r_\alpha(\boldsymbol{z}, x)$	1.3657	0.0302
	$T_\alpha(\boldsymbol{z}, x)$	1.3670	0.0387

Table 6.2: From these joint-contrastive results, it is evident that for each f-divergence, similar posterior convergence is associated with the different estimator parametrisations. As "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization" by Nowozin et al. [2016] suggests, the GAN divergence demonstrates slower and more inconsistent results than the reverse KL divergence.



(a) GAN Divergence (b) Reverse KL Divergence

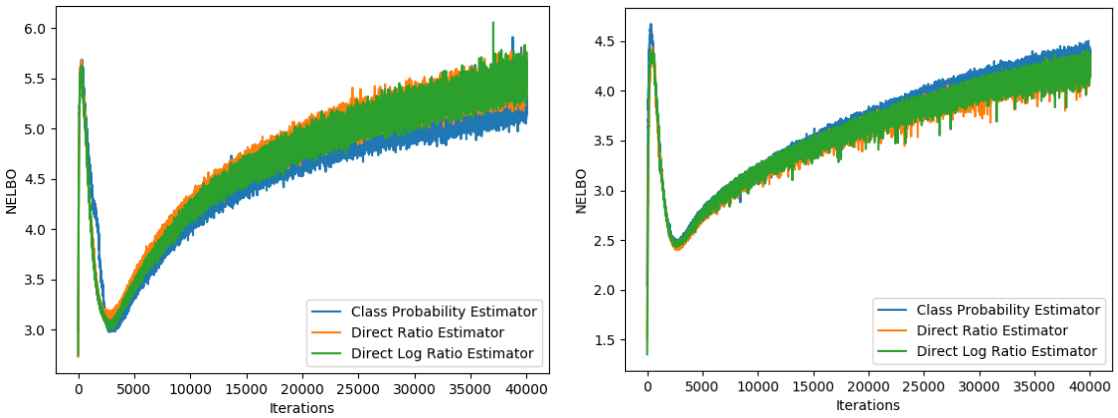
Figure 6.5: These plots of the true KL divergence in the joint-contrastive context show near identical convergence for all estimators.



(a) GAN Divergence

(b) Reverse KL Divergence

Figure 6.6: In sub-figure (a), the estimators have similar levels of stability, but the class probability estimator loss appears to be higher than the other two estimator parametrisations. This does not appear to affect posterior convergence. All three estimators formulated by the reverse KL divergence have similar, increasingly unstable losses.



(a) GAN Divergence

(b) Reverse KL Divergence

Figure 6.7: These joint-contrastive $NELBO$ plots generally mirror the estimator loss trends depicted in Figure 6.6. Lower and more stable $NELBO$ estimations are associated with the reverse KL divergence, correlating with its superior posterior convergence. It is unclear why the $NELBO$ increases over the iterations when its minimisation theoretically leads to posterior convergence.

Chapter 7

Comparing Undertrained Estimators

Since all three estimator parametrisations have similar accuracies at optimality, it does not matter which one is chosen in that context. However, insufficient estimator training iterations can lead to an inaccurate $NELBO$ estimation, reducing the speed at which the variational posterior converges. In this chapter, we theoretically compare the estimator convergence rates between the three parametrisations by analysing the bounds of the estimator outputs, the second functional derivatives of their loss functions and the displacement in the optimal estimator output with each posterior optimisation step. We then compare the performance of undertrained estimators with different parametrisations and f-divergence lower bounds.

7.1 Theory

7.1.1 Estimator Bounds

Consider the bounds on the estimator outputs. The class probability estimator $D_\alpha(u) \simeq q(u)/(q(u)+p(u))$ is bound in $(0,1)$, so from any arbitrary starting point in the same space, very few optimization steps are required to reach the global minimum. On the other hand, the direct ratio estimator $r_\alpha(u) \simeq q(u)/p(u)$ can only output values in $(0,\infty)$, and the direct log ratio estimator $T_\alpha(u) \simeq \log(q(u)/p(u))$ can take any value in $\mathbb{R}$, so optimization can take many iterations if the difference between the estimator's initial and optimal values is too large. This can be problematic if there are insufficient iterations of optimizing the estimator, particularly in the initialization stage: the estimator may not properly converge and the posterior will be trained with an inaccurate density ratio estimation. Essentially, the gradient descent convergence can be improved by using an appropriate transformation [LeCun et al., 2012].

By converting the loss functions to integral form, we are able to visualise the bounds on the loss function output by fixing $q(u)$ and $p(u)$ as arbitrary values in

$(0, 1)$, and plotting the functional inside the integral. Figure 7.1 depicts plots of estimator outputs against the loss functional where $q(u) = p(u) = 0.5$.

Example 7.1.1. In this example we formulate the loss functional plotted in Figure 7.1 (a): the class probability estimator $D_\alpha(u)$ derived from the GAN divergence. The relevant estimator loss function is:

$$\begin{aligned}-\mathbb{E}_{q(u)}[\log D_\alpha(u)] - \mathbb{E}_{p(u)}[\log(1 - D_\alpha(u))] &= \int_U -q(u) \log D_\alpha(u) \\ &\quad - p(u) \log(1 - D_\alpha(u)) \, \mathrm{d}u \\ &= \int_U -\frac{\log D_\alpha(u) + \log(1 - D_\alpha(u))}{2} \, \mathrm{d}u.\end{aligned}$$

In the derivation above we have set $q(u) = p(u) = 0.5$. Note we plot the functional inside the integral:

$$f(u) = -\frac{\log D_\alpha(u) + \log(1 - D_\alpha(u))}{2}.$$

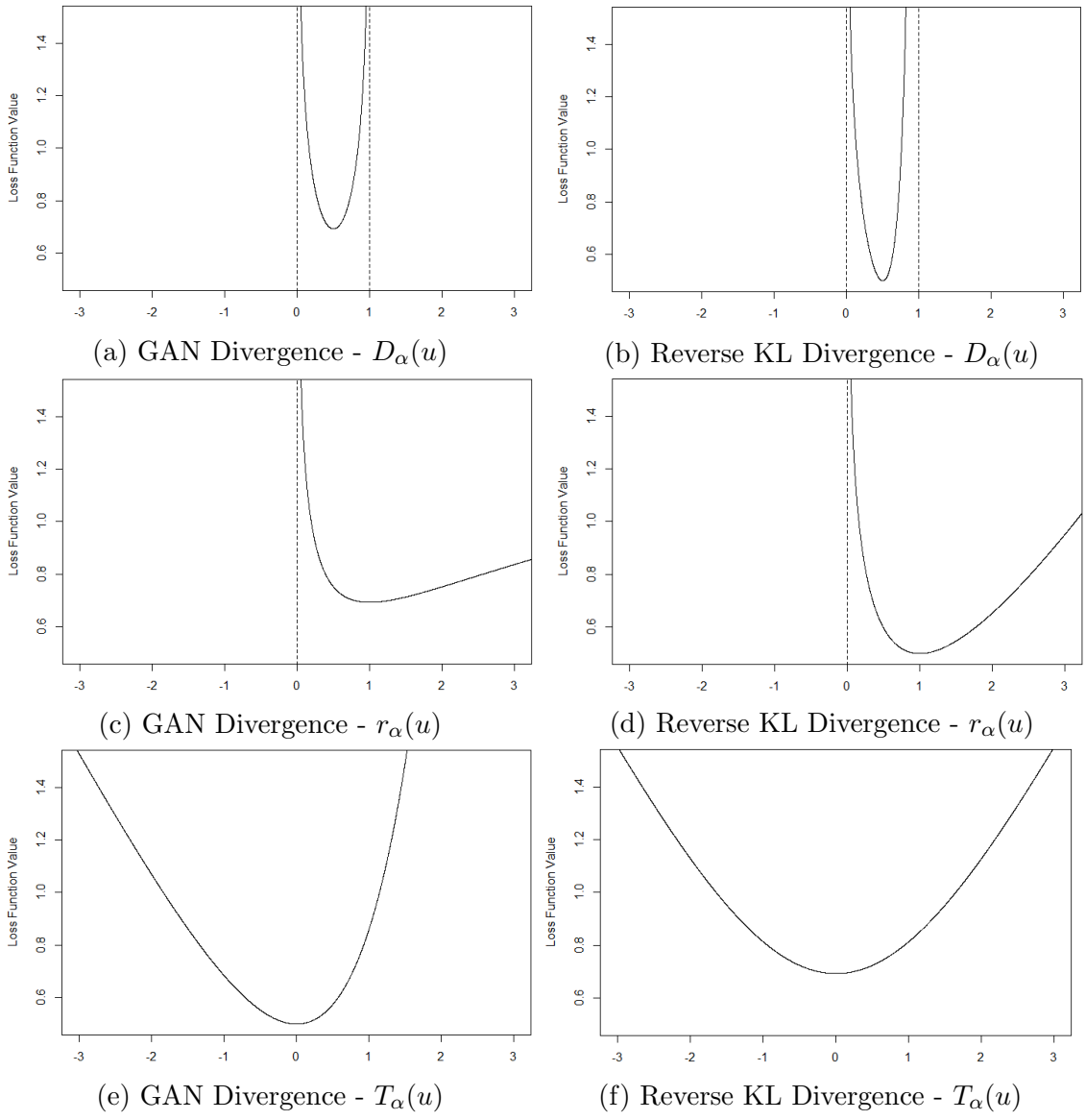
7.1.2 First and Second Derivatives of Estimator Loss Functions

It is known that the convergence rate of a gradient descent method is proportional to the size of its second derivative [LeCun et al., 2012], hence we can use it to compare our algorithms. Below we formulate the second functional derivatives of the estimator losses with respect to the estimator function. It can be shown that the conditions to take the first and second functional derivative using the Euler-Lagrange equation are met. Due to the exponential term in the direct log ratio estimator loss functions, it is difficult to compare their second derivatives, hence we omit them in this section. Their derivations can be found in Appendix G.

***Reverse KL Divergence Bound**:*

Class Probability Estimator:

$$\begin{aligned}L_{RKL}(u) &\coloneqq -\mathbb{E}_{q(u)}\left[\log \frac{D_\alpha(u)}{1 - D_\alpha(u)}\right] + \mathbb{E}_{p(u)}\left[\frac{D_\alpha(u)}{1 - D_\alpha(u)}\right] \\ &= -\int_U q(u) \left(\log \frac{D_\alpha(u)}{1 - D_\alpha(u)}\right) \mathrm{d}u + \int_U p(u) \left(\frac{D_\alpha(u)}{1 - D_\alpha(u)}\right) \mathrm{d}u. \\ \frac{\partial L_{RKL}(u)}{\partial D_\alpha(u)} &= -\frac{q(u)}{D_\alpha(u)} - \frac{q(u)}{1 - D_\alpha(u)} + \frac{p(u)}{1 - D_\alpha(u)} + \frac{p(u) D_\alpha(u)}{(1 - D_\alpha(u))^2} \\ &= -\frac{q(u)}{D_\alpha(u)} - \frac{q(u)}{1 - D_\alpha(u)} + \frac{p(u)}{(1 - D_\alpha(u))^2}. \\ \frac{\partial^2 L_{RKL}(u)}{\partial D_\alpha^2(u)} &= \frac{q(u)}{D_\alpha^2(u)} - \frac{q(u)}{(1 - D_\alpha(u))^2} + \frac{2p(u)}{(1 - D_\alpha(u))^3}.\end{aligned}$$



(a) GAN Divergence - $D_\alpha(u)$

(b) Reverse KL Divergence - $D_\alpha(u)$

(c) GAN Divergence - $r_\alpha(u)$

(d) Reverse KL Divergence - $r_\alpha(u)$

(e) GAN Divergence - $T_\alpha(u)$

(f) Reverse KL Divergence - $T_\alpha(u)$

Figure 7.1: Above we have plotted the estimator output against its loss functional. The same x and y-axis scales are used for all plots. Due to the bounds, it is evident that the loss functionals using the class probability estimator $D_\alpha(u)$ are associated with the highest gradient values and therefore the fastest convergence. It is difficult to compare the direct ratio estimator $r_\alpha(u)$ and the direct log ratio estimator $T_\alpha(u)$, as the former estimator appears to have higher gradients when $q(u) < p(u)$, but slower convergence when $q(u) > p(u)$. Since the relative gradients of the graph can vary with the choice of $p(u)$ and $q(u)$, we cannot compare the f-divergences.

Direct Ratio Estimator:

$$\begin{aligned} L_{RKL}(u) &\coloneqq -\mathbb{E}_{q(u)}[\log r_\alpha(u)] + \mathbb{E}_{p(u)}[r_\alpha(u)] \\ &= -\int_U q(u) \log r_\alpha(u)\, \mathrm{d}u + \int_U p(u) r_\alpha(u)\, \mathrm{d}u. \end{aligned}$$

$$\frac{\partial L_{RKL}(u)}{\partial r_\alpha(u)} = -\frac{q(u)}{r_\alpha(u)} + p(u).$$
$$\frac{\partial^2 L_{RKL}(u)}{\partial r_\alpha^2(u)} = \frac{q(u)}{r_\alpha^2(u)}. \tag{7.1.1}$$

***GAN Divergence Bound**:*

Class Probability Estimator:

$$\begin{aligned} L_{GAN}(u) &\coloneqq -\mathbb{E}_{q(u)}[\log D_\alpha(u)] - \mathbb{E}_{p(u)}[\log(1 - D_\alpha(u))] \\ &= -\int_U q(u) \log D_\alpha(u)\, \mathrm{d}u - \int_U p(u) \log(1 - D_\alpha(u))\, \mathrm{d}u. \\ \frac{\partial L_{GAN}(u)}{\partial D_\alpha(u)} &= -\frac{q(u)}{D_\alpha(u)} + \frac{p(u)}{1 - D_\alpha(u)}. \\ \frac{\partial^2 L_{GAN}(u)}{\partial D_\alpha^2(u)} &= \frac{q(u)}{D_\alpha^2(u)} + \frac{p(u)}{(1 - D_\alpha(u))^2}. \end{aligned} \tag{7.1.2}$$

Direct Ratio Estimator:

$$\begin{aligned} L_{GAN}(u) &\coloneqq -\mathbb{E}_{q(u)}\left[\log \frac{r_\alpha(u)}{r_\alpha(u) + 1}\right] + \mathbb{E}_{p(u)}\left[\log(r_\alpha(u) + 1)\right] \\ &= -\int_U q(u) \left[\log \frac{r_\alpha(u)}{r_\alpha(u) + 1}\right] \mathrm{d}u + \int_U p(u) \left[\log(r_\alpha(u) + 1)\right] \mathrm{d}u. \\ \frac{\partial L_{GAN}(u)}{\partial r_\alpha(u)} &= -\frac{q(u)}{r_\alpha(u)} + \frac{q(u)}{r_\alpha(u) + 1} + \frac{p(u)}{r_\alpha(u) + 1}. \\ \frac{\partial^2 L_{GAN}(u)}{\partial r_\alpha^2(u)} &= \frac{q(u)}{r_\alpha^2(u)} - \frac{q(u)}{(r_\alpha(u) + 1)^2} - \frac{p(u)}{(r_\alpha(u) + 1)^2}. \end{aligned} \tag{7.1.3}$$

By comparing Equations (7.1.2) and (7.1.3), we find that within the GAN divergence, the second derivative of the class probability estimator is strictly superior to that of the direct ratio estimator. Also, a comparison of Equations (7.1.1) and (7.1.3) shows that the direct ratio estimator has a strictly greater second derivative when its loss function is bounded by the reverse KL divergence than with the GAN divergence. As previously stated, the effectiveness of different f-divergence upper bounds has been tested in Nowozin et al. [2016]. They found that the ideal f-divergence used for the estimator's loss function is the same f-divergence that is being minimized in the variational posterior training: in our case, it is the reverse KL divergence.

7.1.3 Displacement of Estimator Optimal Values

We can also consider the effect of the posterior density displacement from each training step: every time $q(u)$ changes, the optimal parametrisation of the estimator

also changes. Consequently, the estimator must take optimization steps to reach this new parametrisation, but again, if the displacement is too significant, then the estimator may not converge in time.

Lemma 7.1.2. *For a fixed displacement of the variational density $q(u)$, the class probability estimator's global minimum displaces less than the direct ratio estimator, that is, $|D_\alpha^{*(n+1)}(u) - D_\alpha^{*(n)}(u)| < |r_\alpha^{*(n+1)}(u) - r_\alpha^{*(n)}(u)|$:*

Proof. Letting $\varepsilon \neq 0$ be the change in $q(u)$ with an optimization step, and assuming that $|\varepsilon| < q(u)$, we have

$$\begin{aligned}
|D_\alpha^{*(n+1)}(u) - D_\alpha^{*(n)}(u)| &= \left| \frac{q(u)+\varepsilon}{q(u)+\varepsilon+p(u)} - \frac{q(u)}{q(u)+p(u)} \right| \\
&= \left| \frac{q^2(u)+q(u)p(u)+\varepsilon q(u)+\varepsilon p(u)}{(q(u)+\varepsilon+p(u))(q(u)+p(u))} \right. \\
&\quad \left. - \frac{q^2(u)+\varepsilon q(u)+q(u)p(u)}{(q(u)+\varepsilon+p(u))(q(u)+p(u))} \right| \\
&= \left| \frac{\varepsilon p(u)}{(q(u)+\varepsilon+p(u))(q(u)+p(u))} \right| \\
&= \left| \frac{\varepsilon}{(q(u)+\varepsilon+p(u))\left(\frac{q(u)}{p(u)}+1\right)} \right|, \\
|r_\alpha^{*(n+1)}(u) - r_\alpha^{*(n)}(u)| &= \left| \frac{q(u)+\varepsilon}{p(u)} - \frac{q(u)}{p(u)} \right| \\
&= \left| \frac{\varepsilon}{p(u)} \right|.
\end{aligned}$$

If $\varepsilon > 0$, then

$$|D_\alpha^{*(n+1)}(u) - D_\alpha^{*(n)}(u)| < |r_\alpha^{*(n+1)}(u) - r_\alpha^{*(n)}(u)| \text{ as } (q(u)+\varepsilon+p(u))\left(\frac{q(u)}{p(u)}+1\right) > p(u).$$

If $\varepsilon < 0$, then recalling that $|\varepsilon| < q(u) < q(u)+p(u)$,

$$\begin{aligned}
(q(u)+\varepsilon+p(u))\left(\frac{q(u)}{p(u)}+1\right) &= \frac{q^2(u)}{p(u)} + 2q(u) + p(u) + \varepsilon\left(\frac{q(u)}{p(u)}+1\right) \\
&= (q(u)+\varepsilon)\left(\frac{q(u)}{p(u)}+1\right) + q(u) + p(u) \\
&> p(u).
\end{aligned}$$

$$\Rightarrow |D_\alpha^{*(n+1)}(u) - D_\alpha^{*(n)}(u)| < |r_\alpha^{*(n+1)}(u) - r_\alpha^{*(n)}(u)|.$$

□

Remark 7.1.3. *The class probability estimator $D_\alpha(u)$ requires few optimization steps than the direct ratio estimator $r_\alpha(u)$ to adjust for a fixed change in the variational posterior density.*

For the direct log ratio estimator, we have

$$\begin{aligned} |T_\alpha^{*(n+1)}(u) - T_\alpha^{*(n)}(u)| &= \left| \log \frac{q(u)+\varepsilon}{p(u)} - \log \frac{q(u)}{p(u)} \right| \\ &= \left| \log \frac{q(u)+\varepsilon}{q(u)} \right|. \end{aligned}$$

It is difficult to make a direct comparison with the other displacement expressions.

7.2 Experiment Outline

In this experiment, we aim to confirm our theory that the estimator parametrisations have differing density ratio estimation accuracies when improperly trained. We also aim to determine the most accurate undertrained estimator by observing the variational posterior convergence.

The same "Continuous Sprinkler" experimental setup from Chapter 6 is used, but with several changes to the training parameters aimed towards reducing the relative amount of estimator training. We significantly reduce the amount of estimator training, lowering the estimator training rate to 0.00004 and the number of training steps before each posterior iteration to 11 for both contrastive settings. We also increase the posterior training rate to 0.0002 and to account for this change, the number of optimization steps of the variational density is reduced to 2000 for the prior-contrastive context and 4000 in the joint-contrastive algorithms.

7.3 Results

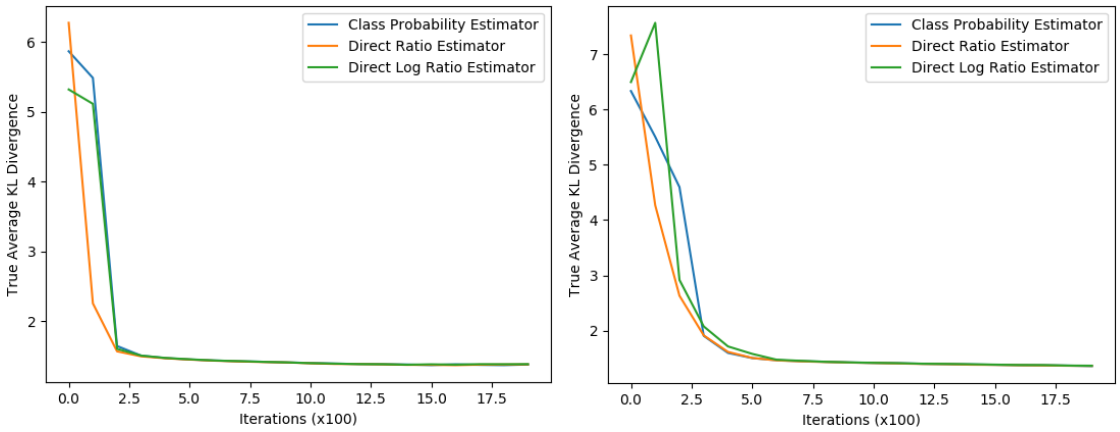
Tables 7.1 and 7.2, and Figures 7.2–7.7 present the results in a similar manner to Section 6.4. For each contrastive context and f-divergence bound, the class probability estimator $D_\alpha(\boldsymbol{z}, x)$ consistently demonstrates the lowest mean KL divergence, followed by the direct ratio estimator $r_\alpha(\boldsymbol{z}, x)$ and the direct log ratio estimator $T_\alpha(\boldsymbol{z}, x)$. The standard deviation of the results also generally increases with this trend: a lower standard deviation implies more consistent results. These results support the theory presented in Section 7.1, but their significance varies depending on the comparison. Additionally, the estimators formulated by the reverse KL divergence have consistently better results, reinforcing the results in Chapter 6.

The prior-contrastive estimators demonstrate relatively similar results. This is likely because a likelihood term $-\mathbb{E}_{q^*(x)\pi(\varepsilon)}[\log p(x|\mathscr{G}_\phi(\varepsilon; x))]$ is additionally used to

optimise the posterior weights ϕ, so convergence is less dependent on an accurate density ratio estimation. In Figures 7.2–7.4, we cannot distinguish any noticeable trends between the estimator parametrisations. However, there is an initial instability associated with the estimators formulated with the reverse KL divergence, whilst the GAN divergence corresponds to stable estimators. This observation, in spite of the superiority of the reverse KL divergence, implies a trade-off between the stability and accuracy of an estimator when choosing an f-divergence to formulate its loss function.

Prior-Contrastive Results			
Algorithm		Mean KL Divergence	Standard Deviation
Reverse KL	$D_\alpha(\boldsymbol{z}, x)$	**1.3572**	**0.0136**
	$r_\alpha(\boldsymbol{z}, x)$	1.3607	0.0199
	$T_\alpha(\boldsymbol{z}, x)$	1.3641	0.0141
GAN	$D_\alpha(\boldsymbol{z}, x)$	1.3788	0.0258
	$r_\alpha(\boldsymbol{z}, x)$	1.3811	0.0365
	$T_\alpha(\boldsymbol{z}, x)$	1.3849	0.0450

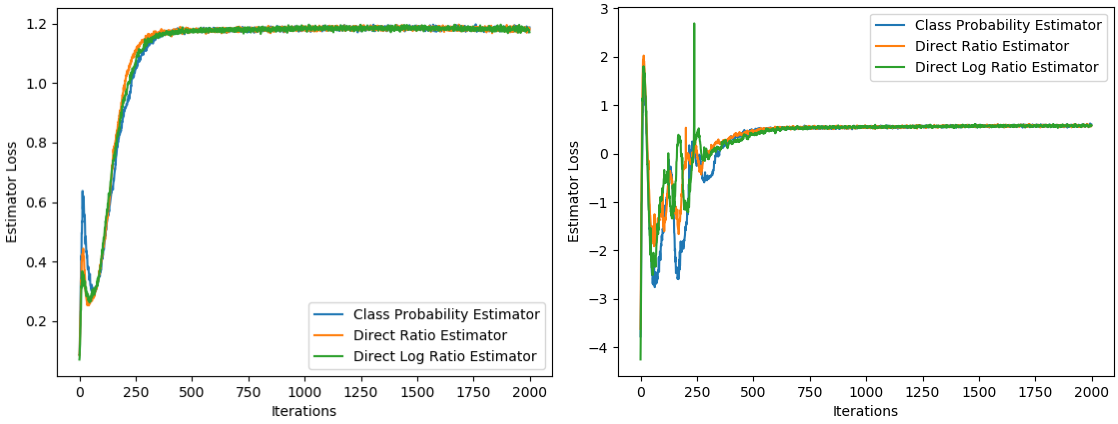
Table 7.1: In these prior-contrastive results, it is evident that using the reverse KL divergence to derive the lower bound for the estimator loss function leads to higher posterior convergence, supporting our experimental results in Chapter 6.



(a) GAN Divergence

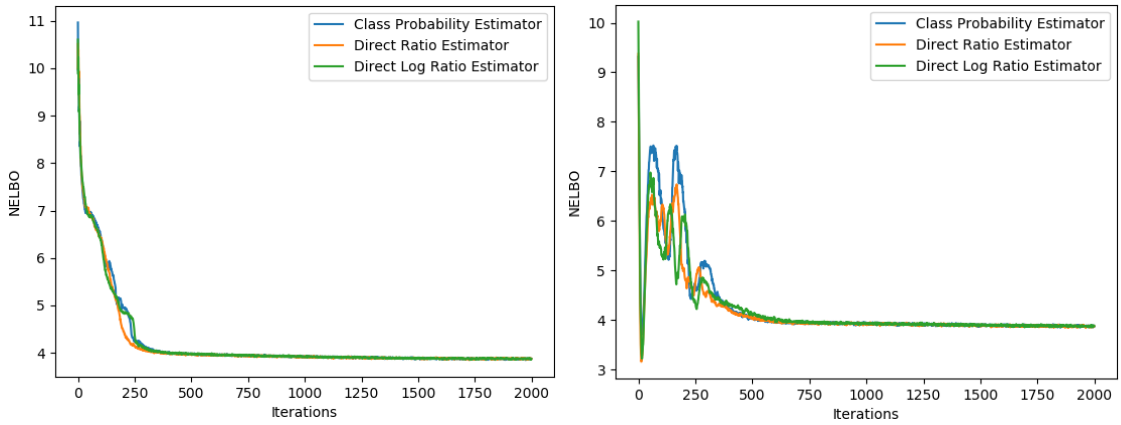
(b) Reverse KL Divergence

Figure 7.2: This figure illustrates the reduction in the true average KL divergence over the prior contrastive experiments' runtime. Although the direct ratio estimator plots initially experience a faster reduction, all three estimator plots meet at approximately the same KL divergence. It can be seen that the GAN divergence is associated with a faster initial drop than the reverse KL divergence, yet experiences lower overall posterior convergence.



(a) GAN Divergence (b) Reverse KL Divergence

Figure 7.3: In these prior-contrastive estimator loss plots, it is evident that the estimators bound by the reverse KL divergence experience initial fluctuations which eventually stabilise, whilst the GAN divergence is associated with higher stability. This is consistent with the observations made in Figure 7.2, and implies a trade-off between estimator stability and accuracy in the choice of f-divergence.



(a) GAN Divergence (b) Reverse KL Divergence

Figure 7.4: The trends shown in the $NELBO$ plots above are generally consistent with the corresponding estimator loss plots. All of the estimators converge at approximately the same estimated $NELBO$, indicating that the effects of under-training the estimator are mostly experienced in the early program iterations.

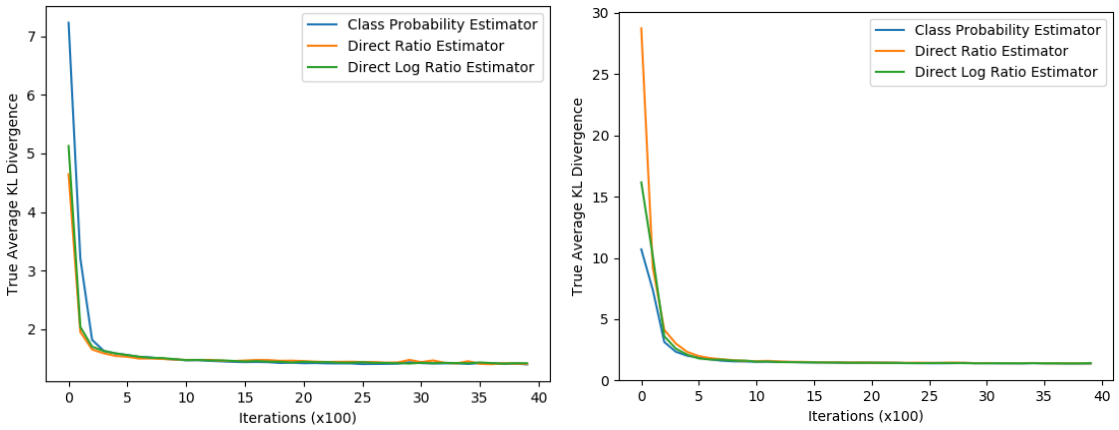
More notable results are found in the joint-contrastive experiments, tabulated in Table 7.2. This is likely because the entirety of the joint-contrastive $NELBO$ is a density ratio:

$$NELBO(q) = \mathbb{E}_{q^*(x)q_\phi(\boldsymbol{z}|x)}\left[\frac{q(\boldsymbol{z},x)}{p(\boldsymbol{z},x)}\right],$$

so posterior optimisation is more dependent on an accurate density ratio estimation. Similar to the prior-contrastive results, Figures 7.5–7.7 imply a trade-off between estimator stability and accuracy in the choice of f-divergence.

Joint-Contrastive Results			
Algorithm		Mean KL Divergence	Standard Deviation
Reverse KL	$D_\alpha(\boldsymbol{z},x)$	**1.3786**	**0.0286**
	$r_\alpha(\boldsymbol{z},x)$	1.3934	0.0410
	$T_\alpha(\boldsymbol{z},x)$	1.4133	0.0597
GAN	$D_\alpha(\boldsymbol{z},x)$	1.4017	**0.0286**
	$r_\alpha(\boldsymbol{z},x)$	1.4086	0.0555
	$T_\alpha(\boldsymbol{z},x)$	1.4214	0.0518

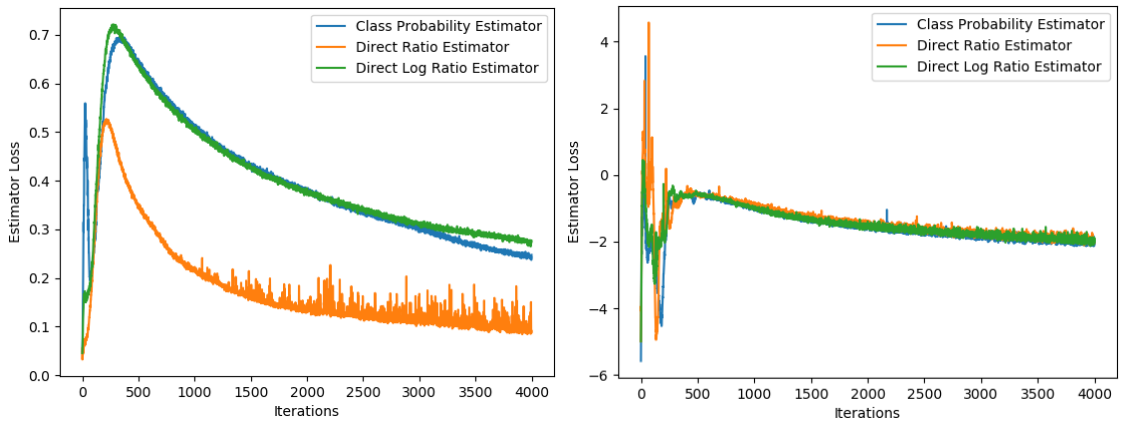
Table 7.2: In these joint-contrastive results, there is a considerable difference between the three estimators when the reverse KL bound is used. However, when the GAN divergence is used to formulate the estimator loss function, only the direct log ratio estimator demonstrates substantially worse posterior convergence than the other two estimators.



(a) GAN Divergence

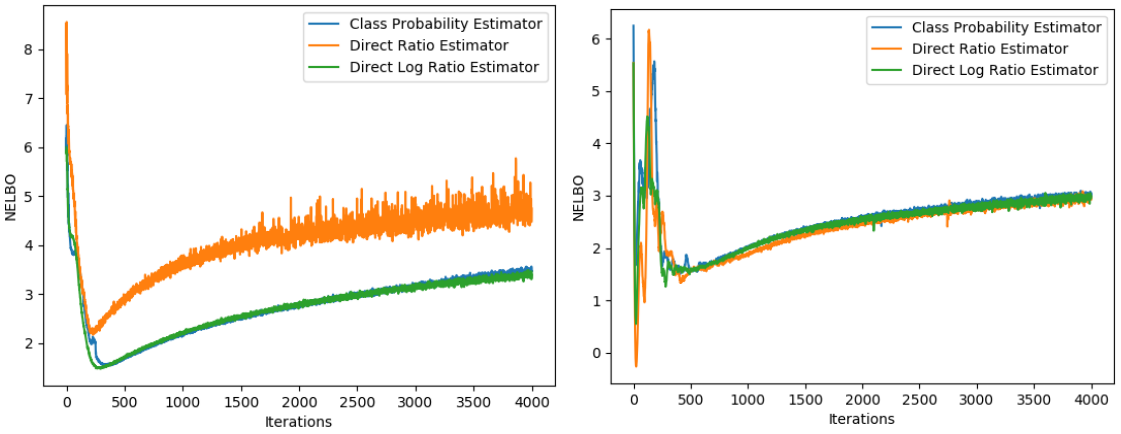
(b) Reverse KL Divergence

Figure 7.5: Similar to the true KL divergence plots in Figure 7.2, these joint-contrastive plots show that the estimators formulated with the GAN divergence experience faster initial convergence. However, this is unreliable as the reverse KL divergence experiments appear to initialise at a much higher true KL divergence.



(a) GAN Divergence (b) Reverse KL Divergence

Figure 7.6: In these estimator loss plots, we again see that the reverse KL divergence plot demonstrates initial estimator instability. This is in stark contrast to the relatively stable GAN divergence plot. From sub-figure (a), we note that the direct ratio estimator is more unstable and has a lower loss value than the other two estimators. The cause of this is unknown, as the direct ratio estimator leads to similar posterior convergence to the more consistent class probability estimator.



(a) GAN Divergence (b) Reverse KL Divergence

Figure 7.7: The joint-contrastive estimated $NELBO$ plots above generally have similar trends to the corresponding estimator plots. Here we note that the final $NELBO$ value of the reverse KL divergence experiments is lower than that of the GAN divergence. However, this may be due to its initial fluctuations delaying the start of the increasing $NELBO$ trend by approximately 250 iterations.

Chapter 8

Autoencoder Experiment - (MNIST Dataset)

In this chapter we compare the different f-divergence bounds and estimator parametrisations in the density ratio estimation involved with the autoencoder formulation. Specifically, we use the MNIST dataset: a popular dataset used in machine learning algorithms. Again, we undertrain the estimators to determine which parametrisation performs the best. The primary goals of this experiment are to confirm the conclusions presented in Chapter 7, and to gain further insights by applying the algorithms to a different experimental setting. Due to limited computational time, we only test the prior-contrastive algorithms. We repeat the experiment with two different latent spaces, one with 2 dimensions and one with 20 dimensions, to determine if the dimensionality of the densities has any effect on the results.

8.1 Experiment Outline

The MNIST (Modified National Institute of Standards and Technology) dataset is widely used for testing machine learning algorithms related to image analysis. It contains 60,000 labelled training images and 10,000 testing images of handwritten digits, each greyscale and of 28x28 pixel size. A sample of the images can be seen in Figure 8.1. Our algorithms do not require the images to be labelled so we combine the two image sets and ignore the labelling.

Figure 8.1: Samples from the MNIST Dataset

In this problem, we train autoencoders on the MNIST dataset using various density ratio estimation algorithms. Specifically, we use the "Adversarial Variational Bayes" variation of the autoencoder as described in Subsection 3.8.1. Recall that the associated optimisation problem is:

$$\min_{\phi,\theta} \left\{ \mathbb{E}_{q^*(x)} \left[-\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] + KL(q_\phi(z|x) \| p(z)) \right] \right\}.$$

We therefore have three neural networks, illustrated in Figures 8.2–8.4. The notation in these figures is the same as in Figures 5.2 and 5.3, and the variational posterior and estimator networks have a relatively similar structure to the neural networks used in Chapters 6 and 7.

- An encoder network $\mathscr{G}_\phi(\varepsilon; x)$ that outputs samples from the variational posterior $q_\phi(z|x)$, effectively creating a latent representation of a data sample x.

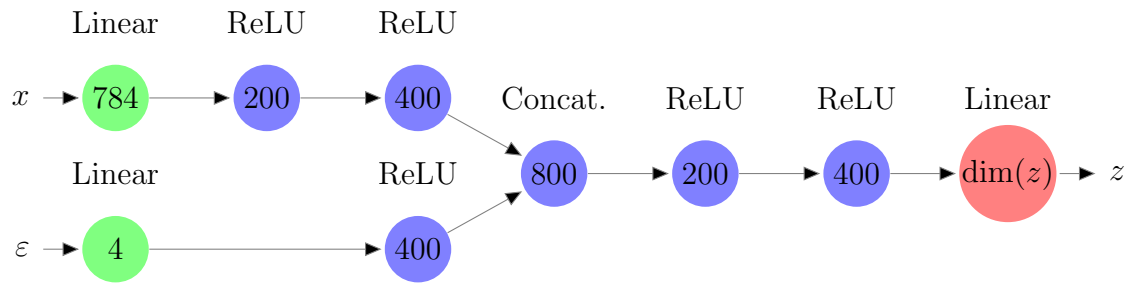


Figure 8.2: This figure illustrates the structure of the generator network $\mathscr{G}(x, \boldsymbol{\varepsilon})$ used in the MNIST data generation experiment. The dimensionality of latent variable z is either 2 or 20 depending on the experimental setting. We have arbitrarily chosen the number of random noise inputs ε to be 4.

- A decoder network representing the likelihood density $p_\theta(x|z)$ that reconstructs a data sample x from a latent variable input z.

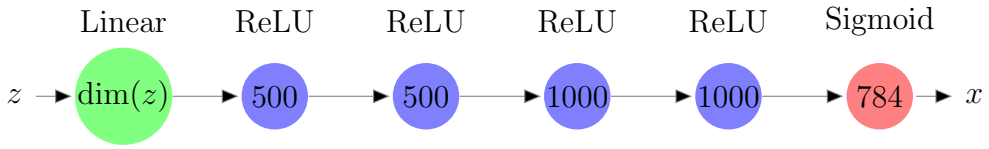


Figure 8.3: This is a diagram of the decoder network used in the MNIST data generation experiment. A sigmoid output layer is used to map the network output to $(0, 1)$, suiting the grayscale nature of the data. Although this problem involves image analysis, we refrain from using convolutional layers due to the relatively small image size. This is consistent with other similar MNIST experiments [Nowozin et al., 2016; Uehara et al., 2016].

- A density ratio estimator network ($D_\alpha(z, x)$, $r_\alpha(z, x)$ or $T_\alpha(z, x)$) used to evaluate the intractable $KL(q_\phi(z|x) \| p(z))$ term in the optimisation problem.

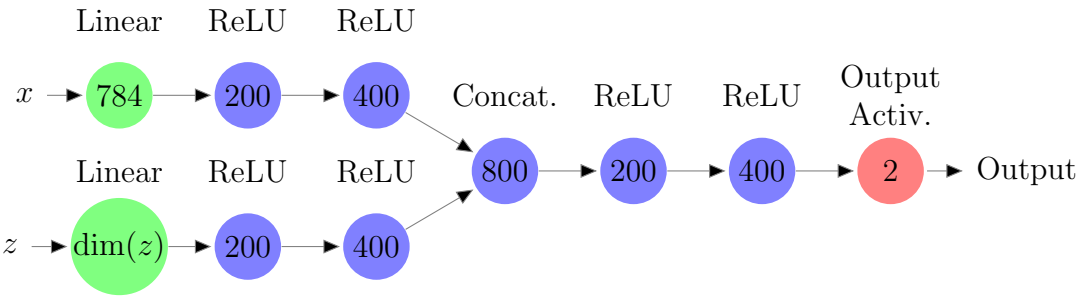


Figure 8.4: This figure depicts the estimator network for the MNIST data generation experiment. Depending on the estimator parametrisation, the output activation function is either sigmoid for the class probability estimator $D_\alpha(z, x) \simeq q_\phi(z|x)/(q_\phi(z|x) + p(z))$, exponential for the direct ratio estimator $r_\alpha(z, x) \simeq q_\phi(z|x)/p(z)$ or linear for the direct log ratio estimator $T_\alpha(z, x) \simeq \log(q_\phi(z|x)/p(z))$.

The program alternates between multiple iterations of estimator training and a single, simultaneous training step of the encoder and decoder weights.

Unlike the "Adversarial Variational Bayes" specification in Subsection 3.8.1, we do not input random noise to the decoder, therefore resulting in a deterministic output. This is depicted in Figure 8.5. To express the likelihood term $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ in the optimisation problem, we use the likelihood function of a Bernoulli density, with probability equal to the output of the decoder. This likelihood representation is common in literature, and is used for several reasons [Nowozin et al., 2016; Tiao et al., 2018; Uehara et al., 2016]:

- Image blurriness is reduced as there is no random noise in the pixel values.
- We use the reconstruction error $\|x - \tilde{x}\|^2$ to evaluate convergence, so a noisy sample reconstruction can make it difficult to compare the algorithms.
- In this experiment, we are primarily interested in the accuracy of the density ratio estimation between the variational posterior and the prior: this is independent of the decoder parametrisation.

We avoid using kernel density estimation to evaluate convergence as it is inaccurate in high dimensions.

Our experiment parameters have been configured to be similar to the MNIST experiment performed in "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization" by Nowozin et al. [2016]: a large batch size of 2048 is used, and for each network, a training rate of 0.0004 is used in the low dimensional configuration, whilst the high dimensional setting uses a training rate of 0.0001. Again, the estimator is pre-trained for 5000 iterations, afterwards the program alternates between 20 iterations of estimator optimizaton and 1 iteration of posterior training, for 4000 total posterior iterations. Algorithm 9 in Appendix B.4 depicts

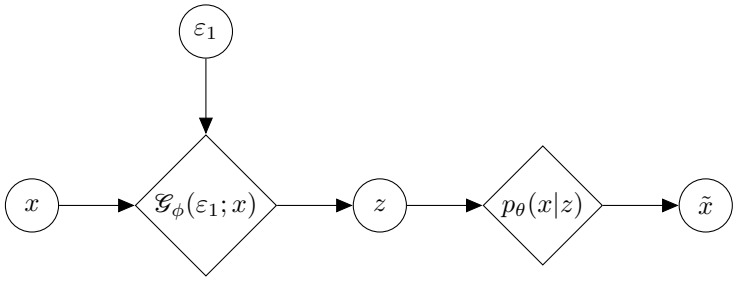


Figure 8.5: This diagram depicts the "Adversarial Variational Bayes" formulation of the variational autoencoder used in the MNIST data generation experiment. Note that there is no random noise added to the decoder.

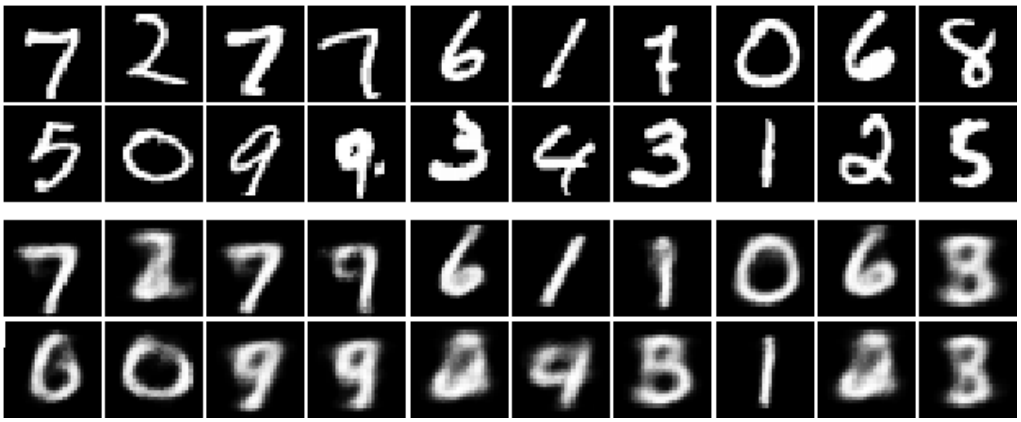
pseudocode for this experiment. Like in the "Sprinkler" experiments, the estimator loss and estimated $NELBO$ at each posterior iteration was saved, and every 10 posterior iterations, 500 MNIST samples were passed through the autoencoder and the average reconstruction error $\|x - \tilde{x}\|^2$ was saved.

8.2 Low Dimensional Experiment Results

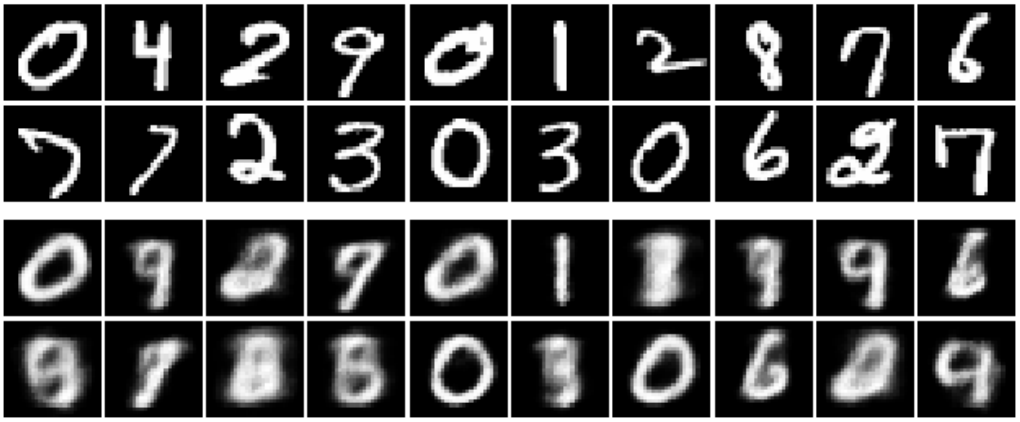
Table 8.1 below tabulates the mean and standard deviation of the autoencoder's final reconstruction error between the 30 experiment repetitions. Figures 8.6 depicts a visual representation of these values. As reflected in Figures 8.7–8.9, most of the distinctions are relatively minor, likely because the estimators experience a higher level of training in comparison to the experiment in Chapter 7. Additionally, the presence of the likelihood term in the $NELBO$ loss function reduces the importance of accurate density ratio estimation. In this experiment, the most notable observation is that the direct log ratio estimator trained with a GAN divergence leads to substantially higher reconstruction error.

Prior-Contrastive Results			
Algorithm		Mean Reconstruction Error	Standard Deviation
Reverse KL	$D_\alpha(z, x)$	**0.08662**	0.00154
	$r_\alpha(z, x)$	0.08710	0.00214
	$T_\alpha(z, x)$	0.08730	0.00157
GAN	$D_\alpha(z, x)$	0.08673	**0.00129**
	$r_\alpha(z, x)$	0.08716	0.00151
	$T_\alpha(z, x)$	0.10683	0.00199

Table 8.1: The low-dimensional MNIST autoencoder experiment results are tabulated above. We note that the relative performances of the different estimator loss functions is the same as in Chapter 7.

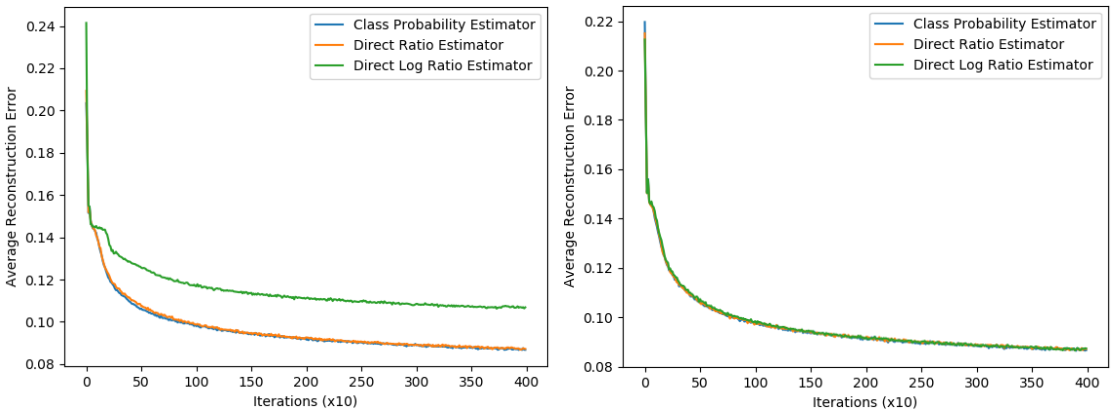


(a) Reconstruction Error of 0.08673



(b) Reconstruction Error of 0.10533

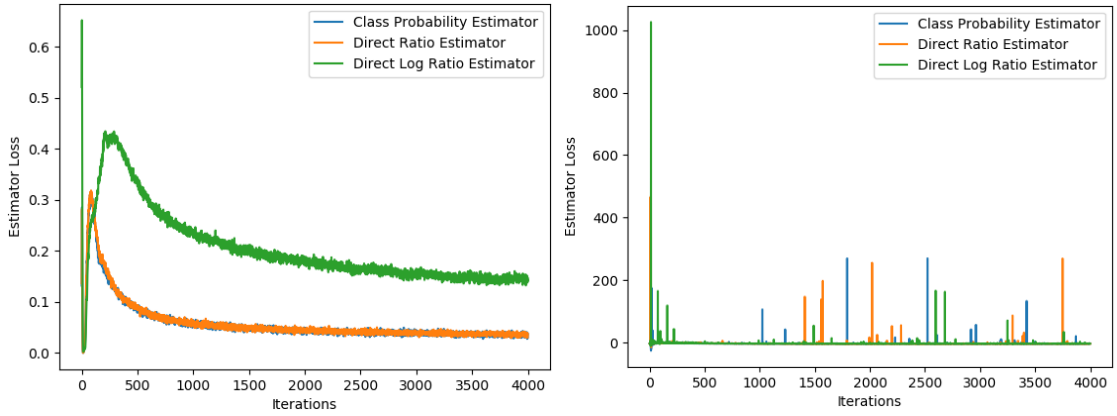
Figure 8.6: These figures exemplify data reconstruction corresponding to different reconstruction errors. The first two rows of each sub-figure depict example data input x, and the last two rows show the reconstructed output $\tilde{x}$.



(a) GAN Divergence

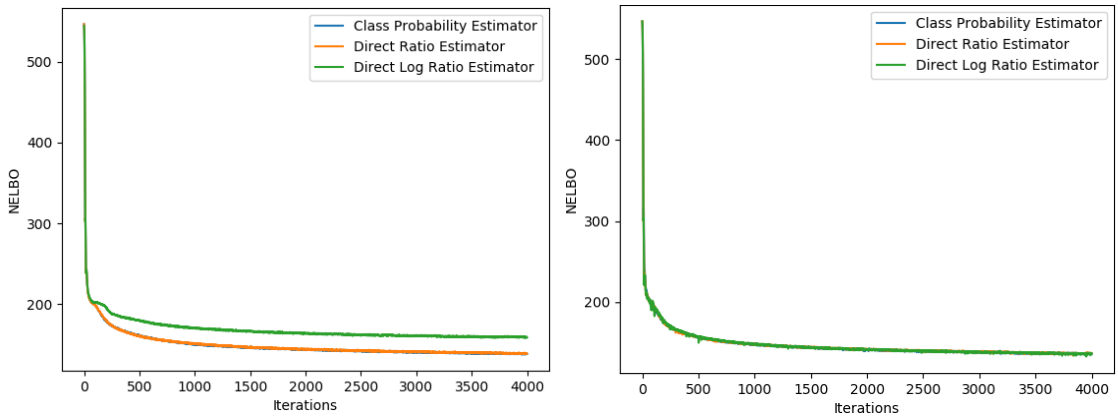
(b) Reverse KL Divergence

Figure 8.7: As shown in the above average reconstruction error plots, the direct log ratio estimator in sub-figure (a) consistently has a higher reconstruction error.



(a) GAN Divergence (b) Reverse KL Divergence

Figure 8.8: Noting the scale of the estimator loss plot in sub-figure (b), it is evident that the reverse KL divergence leads to unstable estimator training. Unlike the estimators plotted in Figure 7.3 (b), these estimators don't appear to stabilise after a certain period. Sub-figure (a) shows that the direct log ratio estimator loss is consistently higher than the other two estimators, correlating with its poorer reconstruction.



(a) GAN Divergence (b) Reverse KL Divergence

Figure 8.9: Despite the apparent instability of the estimators trained with the reverse KL divergence, the $NELBO$ plot in sub-figure (b) is relatively consistent, leading to a smooth convergence as plotted in Figure 8.7 (b). The high direct log ratio estimator loss associated with the GAN divergence corresponds to a relatively large $NELBO$, as depicted in sub-figure (a).

8.3 High Dimensional Experiment Results

When the dimensionality of the latent space was increased to 20, the direct ratio and log ratio estimator loss functions involved quantities greater than the largest representable 64-bit floating point number, overflowing to `Inf` and outputting `NaN` for the remainder of the program runtime. This is because the density ratio $q_\phi(z|x)/p(z)$ increases with the dimensionality of the latent space, eventually reaching a value too large to be represented by a 64-bit float. The direct ratio estimator $r_\alpha(z, x) \simeq$

$q_\phi(z|x)/p(z)$ therefore fails in this scenario. This is consistent with the results in "Generative Adversarial Nets from a Density Ratio Estimation Perspective" by Uehara et al. [2016]. Now recall that the direct log ratio estimator loss function involves taking the exponential of the estimator output. For example, the loss function formulated by the reverse KL divergence is

$$-\mathbb{E}_{q^*(x)\pi(\varepsilon)}[T_\alpha(\mathscr{G}(\varepsilon;x),x)] + \mathbb{E}_{p(z)q^*(x)}[\exp(T_\alpha(z,x))].$$

So despite $T_\alpha(z,x)$ outputting the log of the density ratio, the exact density ratio is expressed in the loss function when the exponential is taken, causing the program to fail. On the other hand, the class probability estimator does not experience this problem as its output is bound in $(0,1)$.

Lemma 8.3.1. *In the prior-contrastive setting, the output of the class probability estimator network before passing through the sigmoid activation function* $g(x) = (1+e^{-x})^{-1}$ *is the estimated log density ratio* $\log(q_\phi(z|x)/p(z))$. *A similar result holds in the joint-contrastive setting.*

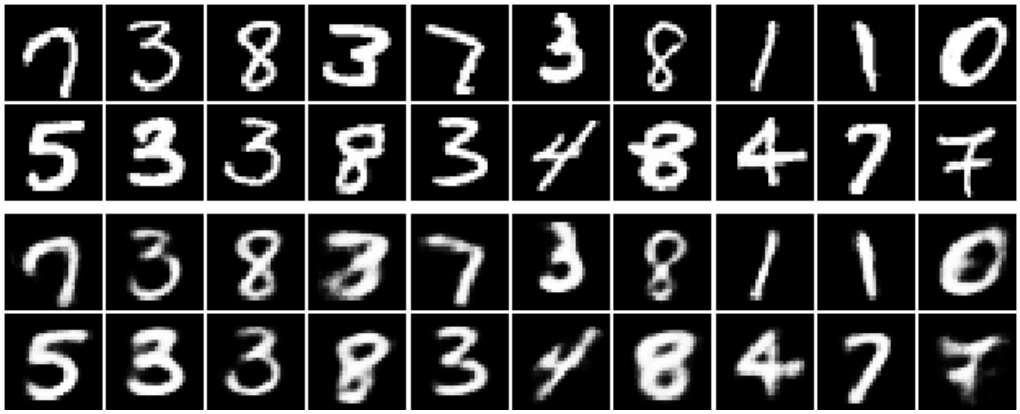
Proof. Proof of this lemma can be found in Appendix A.7. □

Remark 8.3.2. *The calculations with the class probability estimator* $D_\alpha(z,x)$ *involve values that lie within a larger bound of floating point representations than the values concerned with the direct ratio* $r_\alpha(z,x)$ *and log ratio* $T_\alpha(z,x)$ *estimators.*

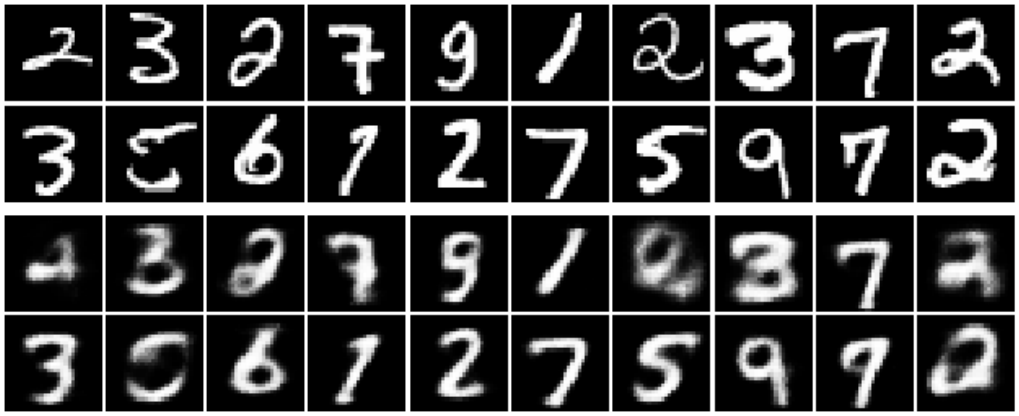
We can therefore conclude that the class probability estimator is superior in the sense that it is the only feasible parametrization. It remains to compare the f-divergence used to formulate the estimator loss function. The reconstruction errors are tabulated in Table 8.2, and a visual representation of the errors is found in Figure 8.10. We notice that unlike our previous experimental results, the reverse KL divergence correlates to inferior results; this is reflected in the instability of its estimator throughout the program runtime, as shown in Figures 8.11–8.13. This contrasts with the results in Chapter 7, where the estimators formulated by the reverse KL divergence experienced initial instability, but have more accurate density ratio estimation when stable. It is likely that the lack of estimator stabilisation in this experiment occurs from the increased complexity of the MNIST dataset.

Prior-Contrastive Results		
Algorithm	Mean Reconstruction Error	Standard Deviation
Reverse KL - $D_\alpha(z, x)$	0.06470	0.01949
GAN - $D_\alpha(z, x)$	**0.04440**	**0.00174**

Table 8.2: In the high dimensional MNIST autoencoder experiment results, it is evident that the GAN divergence correlates with substantially lower reconstruction error mean and standard deviation, contradicting the superiority of the reverse KL divergence shown in previous experiments.



(a) Reconstruction Error of 0.04494



(b) Reconstruction Error of 0.064834

Figure 8.10: Similar to Figure 8.6, these figures exemplify data reconstruction corresponding to different reconstruction errors.

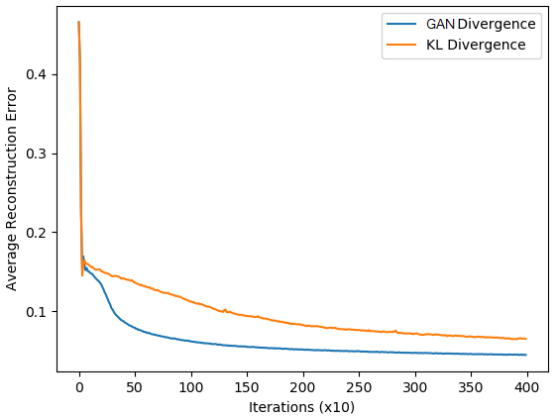


Figure 8.11: This plot shows that the reconstruction error associated with the reverse KL divergence is consistently higher for the majority of the program runtime.

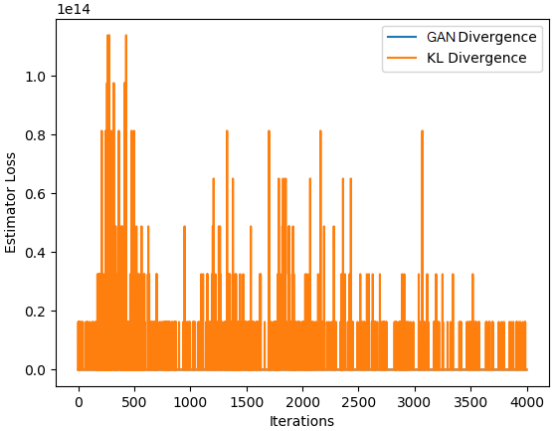


Figure 8.12: The estimator loss corresponding to the reverse KL divergence appears to be extremely unstable for the entire program runtime, spiking to values exceeding 10^{14}. On the other hand, the GAN divergence estimator loss appears to be relatively stable.

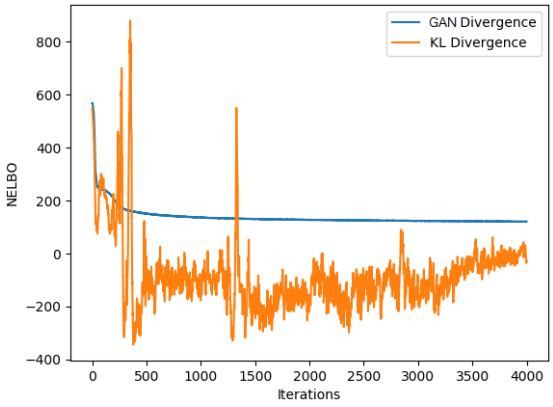


Figure 8.13: The instability of the reverse KL divergence estimator loss propagates to its *NELBO* estimation, which appears to fluctuate wildly. This correlates to the superiority of the GAN divergence in this particular experiment. It appears that the reverse KL estimator in this experiment has not stabilised, leading to poorer network convergence.

Chapter 9

Conclusion and Further Research

In this thesis, we have compared various density ratio estimator loss functions in both prior-contrastive and joint-contrastive contexts for a simple Bayesian posterior inference problem, and in the prior-contrastive context for an autoencoder trained on the MNIST dataset. In Chapter 5, we generalised the estimator loss function specification to the choice of f-divergence and estimator parametrisation. To the best of our knowledge, this generalisation has not yet been presented in the current literature and can be used to gain further insights into these loss functions. Applying this framework, we showed in Chapters 6 and 7 that the different estimator parametrisations have similar accuracies when optimally trained, but when under-trained, higher posterior convergence is correlated with faster estimator convergence, exhibited by the class probability estimator. These chapters also demonstrated that estimators trained with the reverse KL divergence may be initially unstable, particularly when undertrained, but demonstrate high density ratio estimation accuracy when stable. On the other hand, estimators trained with the GAN divergence are relatively stable but lead to lower posterior convergence. These results were verified in the MNIST autoencoder experiment discussed in Chapter 8. This cemented the class probability estimator as the superior estimator parametrisation, as the values associated with the direct ratio and direct log ratio estimators were too large to be efficiently stored in a digital representation. We believe these results are unique contributions to the field.

9.1 Further Research

In this thesis, we have only compared the use of the reverse KL divergence and the GAN divergence in formulating the estimator loss function. "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimisation" by Nowozin et al. [2016] compares several different f-divergences such as the Pearson divergence and the forward KL divergence, but only assesses the posterior convergence. An analysis on the estimator losses and estimated $NELBO$s over the

program runtime may reveal insights on the tradeoff between estimator stability and accuracy. Nowozin's paper also uses different estimator parametrisations for each f-divergence; comparing the f-divergences with the same class probability estimator parametrisation may lead to more reliable results.

There are alternative estimator parametrisations to test. For example, to alleviate the issues associated with an excessively high density ratio, "Generative Adversarial Nets from a Density Ratio Estimation Perspective" by Uehara et al. [2016] proposes that the direct density ratio estimator be parametrised as $r_\alpha(u) \simeq q(u)/(\beta q(u) + (1-\beta)p(u))$ where $\beta > 0$ is small.

In our experiments, we have excluded the joint-contrastive autoencoder formulation, as described in "Adversarially Learned Inference" by Dumoulin et al. [2016] due to lack of computational time, but comparing the estimators in this context may present new insights.

There are alternate methods of estimating the intractable KL divergence term in the $NELBO$ that do not involve Theorem 4.2.2. "Divergence Estimation for Multidimensional Densities via k-Nearest-Neighbor Distances" by Wang et al. [2009] describes a non-parametric method of estimating the KL divergence between two high dimensional densities. Rather than estimate the density ratio directly, a denoiser estimates the gradient of the log density, which can be used to minimise the KL divergence term in the $NELBO$ [Vincent et al., 2008]. These methods can be used to formulate new variational inference algorithms, and their performances can be compared with the algorithms discussed in this thesis.

Posterior convergence in our autoencoder experiment can be assessed more effectively by using Annealed Importance Sampling (AIS) to estimate the negative of the marginal density $-p(x)$, which can be compared with the estimated $NELBO$ which attains its minimum at that value. This process is detailed in "On the Quantitative Analysis of Decoder-Based Generative Models" by Wu et al. [2016].

References

Beck, A. (2014). *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Bengio, Y. (2012). *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Cheng, B. and Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. *Statist. Sci.*, 9(1):2–30.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Doersch, C. (2016). Tutorial on Variational Autoencoders. *ArXiv*.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. (2016). Adversarially Learned Inference. *ArXiv*.

E. Rumelhart, D., E. Hinton, G., and J. Williams, R. (1986). Learning representations by back propagating errors. 323:533–536.

Floudas, C. A. (2005). *Deterministic Global Optimization: Theory, Methods and (NONCONVEX OPTIMIZATION AND ITS APPLICATIONS Volume 37) (Nonconvex Optimization and Its Applications)*. Springer-Verlag, Berlin, Heidelberg.

Fuglede, B. and Topsoe, F. (2004). Jensen-shannon divergence and hilbert space embedding. In *International Symposium on Information Theory, 2004. ISIT 2004. Proceedings.*, page 31.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd ed. edition.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Goodman, L. A. (1960). On the exact variance of products. *Journal of the American Statistical Association*, 55(292):708–713.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Huszár, F. (2017). Variational Inference using Implicit Distributions. *ArXiv*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv*.

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *ArXiv*.

Kolen, J. F. and Kremer, S. C. (2001). *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*. IEEE.

Kullback, S. (1959). *Information Theory and Statistics*. Wiley, New York.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg.

Li, M., Zhang, T., Chen, Y., and Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 661–670, New York, NY, USA. ACM.

Mescheder, L. M., Nowozin, S., and Geiger, A. (2017). Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks. *ArXiv*.

Mohamed, S. and Lakshminarayanan, B. (2016). Learning in Implicit Generative Models. *ArXiv*.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2010). Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization. *IEEE Trans. Inf. Theor.*, 56(11):5847–5861.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems 29*, pages 271–279. Curran Associates, Inc.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv*.

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*.

Snyman, J. (2005). *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Applied Optimization. Springer.

Sugiyama, M., Suzuki, T., and Kanamori, T. (2012). *Density Ratio Estimation in Machine Learning*. Cambridge University Press.

Tiao, L. C., Bonilla, E. V., and Ramos, F. (2018). Cycle-Consistent Adversarial Learning as Approximate Bayesian Inference. *ArXiv*.

Tran, D., Ranganath, R., and Blei, D. (2017). Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems 30*, pages 5523–5533. Curran Associates, Inc.

Uehara, M., Sato, I., Suzuki, M., Nakayama, K., and Matsuo, Y. (2016). Generative Adversarial Nets from a Density Ratio Estimation Perspective. *ArXiv*.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA. ACM.

Wang, Q., Kulkarni, S. R., and Verdú, S. (2009). Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Trans. Inf. Theor.*, 55(5):2392–2405.

Wellman, M. P. and Henrion, M. (1993). Explaining 'explaining away'. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(3):287–292.

Wu, H. (2009). Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions. *Inf. Sci.*, 179(19):3432–3441.

Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. B. (2016). On the quantitative analysis of decoder-based generative models. *ArXiv*.

Zanin Zambom, A. and Dias, R. (2012). A Review of Kernel Density Estimation with Applications to Econometrics. *ArXiv*.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. (2017). Advances in variational inference. *ArXiv*.

Zilles, K. (1992). Neuronal plasticity as an adaptive property of the central nervous system. *Annals of Anatomy - Anatomischer Anzeiger*, 174(5):383–391.

Appendix A

Proofs

A.1 Proof of Proposition 2.6.2

First we show that

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} f\left(\boldsymbol{x}^{(n)}+\alpha \boldsymbol{s}^{(n)}\right)=\nabla f\left(\boldsymbol{x}^{(n)}+\alpha \boldsymbol{s}^{(n)}\right)^{\top} \boldsymbol{s}^{(n)},$$

where $\boldsymbol{x}^{(n)}=\left[x_1^{(n)}, \ldots, x_k^{(n)}\right]^{\top}$ and $\boldsymbol{s}^{(k)}=\left[s_1^{(n)}, \ldots, s_k^{(n)}\right]^{\top}$.
Let

$$x_i^{(n)}(\alpha)=x_i^{(n)}+\alpha s_i^{(n)}, \quad i=1, \ldots, n,$$

so that $\boldsymbol{x}^{(n)}+\alpha \boldsymbol{s}^{(n)}=\left[x_1^{(n)}(\alpha), \ldots, x_k^{(n)}(\alpha)\right]^{\top}$. We have

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\alpha} f\left(\boldsymbol{x}^{(n)}+\alpha \boldsymbol{s}^{(n)}\right) &= \frac{\mathrm{d}}{\mathrm{d}\alpha} f\left(x_1^{(n)}, \ldots, x_k^{(n)}(\alpha)\right) \\
&= \sum_{i=1}^{k} \frac{\partial f(\boldsymbol{x})}{\partial x_i}|_{\boldsymbol{x}=\boldsymbol{x}^{(n)}+\alpha \boldsymbol{s}^{(n)}} \frac{\mathrm{d}\left(x_i^{(n)}(\alpha)\right)}{\mathrm{d}\alpha} \\
&= \sum_{i=1}^{k} \frac{\partial f(\boldsymbol{x})}{\partial x_i}|_{\boldsymbol{x}=\boldsymbol{x}^{(n)}+\alpha \boldsymbol{s}^{(n)}} \boldsymbol{s}_i^{(n)} \\
&= \nabla f\left(\boldsymbol{x}^{(n)}+\alpha \boldsymbol{s}^{(n)}\right)^{\top} \boldsymbol{s}^{(n)}.
\end{aligned}$$

Setting $\alpha=0$ and using Definition 2.7.1,

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} f\left(\boldsymbol{x}^{(n)}+\alpha \boldsymbol{s}^{(n)}\right)|_{\alpha=0}=\nabla f\left(\boldsymbol{x}^{(n)}\right)^{\top} \boldsymbol{s}^{(n)}<0.$$

Therefore for sufficiently small $\alpha>0$,

$$f\left(x^{(n)}+\alpha s^{(n)}\right)<f\left(x^{(n)}\right).$$

A.2 Proof of Lemma 3.2.4

$$\begin{aligned} D_{RKL}(p(x)\|q(x)) &= \mathbb{E}_{p(x)}\left[\frac{q(x)}{p(x)}\log\left(\frac{q(x)}{p(x)}\right)\right] \\ &= \int_{\mathbb{R}} p(x)\frac{q(x)}{p(x)}\log\left(\frac{q(x)}{p(x)}\right)\,\mathrm{d}x \\ &= \int_{\mathbb{R}} q(x)\log\left(\frac{q(x)}{p(x)}\right)\,\mathrm{d}x \\ &= \mathbb{E}_{q(x)}\left[\log\left(\frac{q(x)}{p(x)}\right)\right] \\ &= KL(q(x)\|p(x)). \end{aligned}$$

A.3 Proof of Lemma 3.2.5

We prove this in the case where $p(x)$ and $q(x)$ are continuous densities: a similar proof holds when they are discrete.

$$\begin{aligned} KL(p(x)\|q(x)) &= \int_{\mathbb{R}} p(x)\log\left(\frac{p(x)}{q(x)}\right)\,\mathrm{d}x \\ &= -\int_{\mathbb{R}} p(x)\log\left(\frac{q(x)}{p(x)}\right)\,\mathrm{d}x \\ &\geq -\int_{\mathbb{R}} p(x)\left(\frac{q(x)}{p(x)}-1\right)\,\mathrm{d}x \\ &= -\int_{\mathbb{R}} q(x)dx + \int_{\mathbb{R}} p(x)\,\mathrm{d}x \\ &= 0. \end{aligned}$$

In the third line we use $-\log x \geq -(x-1)$ for all $x > 0$ with equality if and only if $x = 1$, therefore $KL(q(x)\|p(x)) = 0$ if and only if $q(x) = p(x)$. The last line is due to $p(x)$ and $q(x)$ being probability densities.

A.4 Proof of Lemma 3.7.1

Denoting the identity matrix as I and the covariance matrix of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ as Σ, we state the probability density functions of the two densities:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \frac{1}{(2\pi)^{M/2}\det(\Sigma)^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^\top\Sigma^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right),$$

$$p(\boldsymbol{z}) = \frac{1}{(2\pi)^{M/2}}\exp\left(-\frac{1}{2}\boldsymbol{z}^\top I\boldsymbol{z}\right).$$

From Definition 3.2.2, we have:

$$\begin{aligned}
KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z})] \\
&= -\frac{1}{2}\log(\det\Sigma) + \frac{1}{2}\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[-(\boldsymbol{z}-\boldsymbol{\mu})^\top\Sigma^{-1}(\boldsymbol{z}-\boldsymbol{\mu}) + \boldsymbol{z}^\top I\boldsymbol{z}\right] \\
&= -\frac{1}{2}\log(\det\Sigma) + \frac{1}{2}\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[-\operatorname{tr}\left(\Sigma^{-1}\Sigma\right) + \operatorname{tr}\left(I\boldsymbol{z}\boldsymbol{z}^\top\right)\right] \\
&= -\frac{1}{2}\log(\det\Sigma) - \frac{M}{2} + \frac{1}{2}\operatorname{tr}\left(I + \left(\Sigma + \mu\mu^\top\right)\right) \\
&= \frac{1}{2}\sum_{i=1}^{M}\left(\sigma_i^2 + \mu_i^2 - \log(\sigma_i^2) - 1\right).
\end{aligned}$$

A.5 Proof of Lemma 4.1.1

First we write the discriminator loss function in integral form:

$$L_D = -\int_U q(u)\log D_\alpha(u)\,\mathrm{d}u - \int_U p(u)\log(1 - D_\alpha(u))\,\mathrm{d}u.$$

Now we take the functional derivative of the expression and equate it to 0:

$$\frac{\partial L_D}{\partial D_\alpha(u)} = 0$$

$$\begin{aligned}
-\frac{q(u)}{D_\alpha(u)} + \frac{p(u)}{1 - D_\alpha(u)} &= 0 \\
D_\alpha(u)(q(u) + p(u)) &= q(u) \\
D_\alpha(u) &= \frac{q(u)}{q(u) + p(u)}.
\end{aligned}$$

Observing that $q(u)$ and $p(u)$ are densities, this expression is a minimum as

$$\begin{aligned}
\frac{\partial^2 L_D}{\partial D_\alpha^2(u)} &= \frac{q(u)}{D_\alpha^2(u)} + \frac{p(u)}{(1 - D_\alpha(u))^2} \\
&> 0.
\end{aligned}$$

It can be shown that the conditions to take the first and second functional derivative using the Euler-Lagrange equation are met.

A.6 Proof of Lemma 4.2.2

First we write the ratio loss function in integral form:

$$L_r = -\int_U q(u)\log r_\alpha(u)\,\mathrm{d}u + \int_U p(u) r_\alpha(u)\,\mathrm{d}u.$$

Now we take the functional derivative of the expression and equate it to 0:

$$\frac{\partial L_r}{\partial r_\alpha(u)} = 0$$

$$-\frac{q(u)}{r_\alpha(u)} + p(u) = 0$$

$$r_\alpha(u) = \frac{q(u)}{p(u)}.$$

Observing that $q(u)$ is a density, this expression is a minimum as:

$$\begin{aligned}\frac{\partial^2 L_r}{\partial r_\alpha^2(u)} &= \frac{q(u)}{r_\alpha^2(u)} \\ &> 0.\end{aligned}$$

A.7 Proof of Lemma 8.3.1

Letting x be the neural network output before being mapped to $(0, 1)$, we have:

$$\begin{aligned}\frac{1}{1+e^{-x}} &\simeq \frac{q_\phi(z|x)}{p(z)+q_\phi(z|x)} \\ e^{-x}+1 &\simeq \frac{p(z)+q_\phi(z|x)}{q_\phi(z|x)} \\ e^{-x} &\simeq \frac{p(z)}{q_\phi(z|x)} \\ x &\simeq \log\frac{q_\phi(z|x)}{p(z)}.\end{aligned}$$

Appendix B

Algorithms

B.1 Back-Propagation Algorithm

Data: Training Data $\{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, \boldsymbol{y}^{(N)})\}$
Result: Cost Function Partial Derivatives $\frac{\partial}{\partial \Theta^{(j)}_{m,n}} L(\Theta)$

begin
  Initialize weights Θ using Xavier Initialisation;
  Set $\Delta^{(j)}_{m,n} = 0 \quad \forall j, m, n$;
  for $I = 1$ **to** N **do**
    Set $\boldsymbol{a}^{(1)} = \boldsymbol{x}^{(I)}$;
    for $j = 2$ **to** J **do**
      Set $\boldsymbol{a}^{(j)} = \Theta^{(j-1)^\top} \boldsymbol{a}^{(j-1)}$;
    end
    Set $\boldsymbol{\delta}^{(J)} = \boldsymbol{a}^{(J)} - \boldsymbol{y}^{(I)}$;
    for $j = J - 1$ **to** 2 **do**
      Set $\boldsymbol{\delta}^{(j)} = ((\Theta^{(j)})^\top \boldsymbol{\delta}^{(j+1)}) .* g'(\Theta^{(j)^\top} \boldsymbol{a}^{(j)})$;
    end
    for $j = 1$ **to** $J - 1$ **do**
      Set $\Delta^{(j)} = \Delta^{(j)} + \boldsymbol{\delta}^{(j+1)} (\boldsymbol{a}^{(j)})^\top$;
    end
  end
  for *all* j, m, n **do**
    Set $\frac{\partial}{\partial \Theta^{(j)}_{m,n}} L(\Theta) = \frac{1}{N} \Delta^{(j)}_{m,n}$;
  end
end

Algorithm 5: Back-Propagation Algorithm

B.2 Coordinate Ascent Variational Inference Algorithm

Data: Dataset $\boldsymbol{x}$ and Bayesian Model $\mathrm{p}(\boldsymbol{x}, \boldsymbol{z})$

Result: Variational density $q(\boldsymbol{z}) = \prod_{i=1}^{M} q_i(z_i)$

begin

  Initialize random variational factors $q_j(z_j)$;

  while *ELBO(q) has not converged* **do**

    for $j = 1$ **to** m **do**

      Set $q_j(z_j) \propto \exp(\mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(z_j | \boldsymbol{z}_{-j}, \boldsymbol{x})])$;

    end

    Calculate $ELBO(q) = \mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{z}, \boldsymbol{x})] - \mathbb{E}_{q(\boldsymbol{z})}[\log q(\boldsymbol{z})]$;

  end

  Return $q(\boldsymbol{z})$;

end

Algorithm 6: Coordinate Ascent Variational Inference (CAVI)

B.3 Algorithms for "Sprinkler" Experiment

Input: Dataset density $q^*(x) = \{0, 5, 8, 12, 20\}$,
(Implicit) Prior density $z \sim \mathcal{N}(0, 2I_{2\times 2})$,
Likelihood density $x|z \sim Exp(3 + \max(0, z_1)^3 + \max(0, z_2)^3)$,
Noise density $\varepsilon \sim \mathcal{N}(0, I_{3\times 3})$
Result: Optimized posterior generator $\mathscr{G}_\phi(\varepsilon; x)$

for $j = 1$ **to** 5000 **do**
  Sample $\{\varepsilon^{(i,j)}\}_{i=1}^{1000} \sim \pi(\varepsilon)$;
  Sample $\{0, 5, 8, 12, 50\}_{i=1}^{200} = \{x^{(i,j)}\}_{i=1}^{1000} \sim q^*(x)$;
  Sample $\{z_p^{(i,j)}\}_{i=1}^{1000} \sim p(z)$;
  foreach $\varepsilon^{(i,j)}, x^{(i,j)}$ **do**
    Sample $z_q^{(i,j)} = \mathscr{G}(\varepsilon^{(i,j)}; x_q^{(i,j)})$;
  end
  update *Estimator weights* α
end
for $j = 1$ **to** 10000 **do**
  for $k = 1$ **to** 100 **do**
    Sample $\{\varepsilon^{(i,k)}\}_{i=1}^{1000} \sim \pi(\varepsilon)$;
    Sample $\{0, 5, 8, 12, 50\}_{i=1}^{200} = \{x^{(i,k)}\}_{i=1}^{1000} \sim q^*(x)$;
    Sample $\{z_p^{(i,k)}\}_{i=1}^{1000} \sim p(z)$;
    foreach $\varepsilon^{(i,k)}, x^{(i,k)}$ **do**
      Sample $z_q^{(i,k)} = \mathscr{G}(\varepsilon^{(i,k)}; x_q^{(i,k)})$;
    end
    update *Estimator weights* α
  end
  Sample $\{\varepsilon^{(i)}\}_{i=1}^{1000} \sim \pi(\varepsilon)$;
  Sample $\{0, 5, 8, 12, 50\}_{i=1}^{200} = \{x^{(i)}\}_{i=1}^{1000} \sim q^*(x)$;
  update *Variational posterior weights* ϕ
end

Algorithm 7: Sprinkler Prior-Contrastive Algorithm

Input: Dataset density $q^*(x) = \{0, 5, 8, 12, 20\}$,
(Implicit) Prior density $z \sim \mathcal{N}(0, 2I_{2\times 2})$,
(Implicit) Likelihood density $x|z \sim Exp(3 + \max(0, z_1)^3 + \max(0, z_2)^3)$,
Noise density $\varepsilon \sim \mathcal{N}(0, I_{3\times 3})$
Result: Optimized posterior generator $\mathscr{G}_\phi(\varepsilon; x)$

for $j = 1$ **to** 5000 **do**
  Sample $\{\varepsilon^{(i,j)}\}_{i=1}^{1000} \sim \pi(\varepsilon)$;
  Sample $\{0, 5, 8, 12, 50\}_{i=1}^{200} = \{x_q^{(i,j)}\}_{i=1}^{1000} \sim q^*(x)$;
  Sample $\{z_p^{(i,j)}\}_{i=1}^{1000} \sim p(z)$;
  foreach $\varepsilon^{(i,j)}, x_q^{(i,j)}$ **do**
    Sample $z_q^{(i,j)} = \mathscr{G}(\varepsilon^{(i,j)}; x_q^{(i,j)})$;
  end
  foreach $z_p^{(i,j)}$ **do**
    Sample $x_p^{(i,j)} \sim p(x|z)$;
  end
  update *Estimator weights* α
end
for $j = 1$ **to** 40000 **do**
  for $k = 1$ **to** 100 **do**
    Sample $\{\varepsilon^{(i,k)}\}_{i=1}^{1000} \sim \pi(\varepsilon)$;
    Sample $\{0, 5, 8, 12, 50\}_{i=1}^{200} = \{x_q^{(i,k)}\}_{i=1}^{1000} \sim q^*(x)$;
    Sample $\{z_p^{(i,k)}\}_{i=1}^{1000} \sim p(z)$;
    foreach $\varepsilon^{(i,k)}, x^{(i,k)}$ **do**
      Sample $z_q^{(i,k)} = \mathscr{G}(\varepsilon^{(i,k)}; x_q^{(i,k)})$;
    end
    foreach $z_p^{(i,k)}$ **do**
      Sample $x_p^{(i,k)} \sim p(x|z)$;
    end
    update *Estimator weights* α
  end
  Sample $\{\varepsilon^{(i)}\}_{i=1}^{1000} \sim \pi(\varepsilon)$;
  Sample $\{0, 5, 8, 12, 50\}_{i=1}^{200} = \{x_q^{(i)}\}_{i=1}^{1000} \sim q^*(x)$;
  update *Variational posterior weights* ϕ
end

Algorithm 8: Sprinkler Joint-Contrastive Algorithm

B.4 Algorithm for MNIST Experiment

Input: MNIST Dataset density $q^*(x)$,
(Implicit) Prior density $z \sim \mathcal{N}(0, I_{M\times M})$,
Noise density $\varepsilon \sim \mathcal{N}(0, I_{4\times 4})$
Result: Optimized posterior generator $\mathscr{G}_\phi(\varepsilon; x)$,
Optimized likelihood network $p_\theta(x|z)$

for $j = 1$ **to** 5000 **do**
  Sample $\{\varepsilon^{(i,j)}\}_{i=1}^{2048} \sim \pi(\varepsilon)$;
  Sample $\{x^{(i,j)}\}_{i=1}^{2048} \sim q^*(x)$;
  Sample $\{z_p^{(i,j)}\}_{i=1}^{2048} \sim p(z)$;
  foreach $\varepsilon^{(i,j)}, x^{(i,j)}$ **do**
    Sample $z_q^{(i,j)} = \mathscr{G}(\varepsilon^{(i,j)}; x^{(i,j)})$;
  end
  update *Estimator weights* α
end
for $j = 1$ **to** 4000 **do**
  for $k = 1$ **to** 20 **do**
    Sample $\{\varepsilon^{(i,k)}\}_{i=1}^{2048} \sim \pi(\varepsilon)$;
    Sample $\{x^{(i,k)}\}_{i=1}^{2048} \sim q^*(x)$;
    Sample $\{z_p^{(i,k)}\}_{i=1}^{2048} \sim p(z)$;
    foreach $\varepsilon^{(i,k)}, x^{(i,k)}$ **do**
      Sample $z_q^{(i,k)} = \mathscr{G}(\varepsilon^{(i,k)}; x^{(i,k)})$;
    end
    update *Estimator weights* α
  end
  Sample $\{\varepsilon^{(i)}\}_{i=1}^{2048} \sim \pi(\varepsilon)$;
  Sample $\{x^{(i)}\}_{i=1}^{2048} \sim q^*(x)$;
  update *Variational posterior weights* ϕ *and likelihood weights* θ
end

Algorithm 9: MNIST Prior-Contrastive Algorithm

Appendix C

Coordinate Ascent Variational Inference Derivation

Firstly, we express $ELBO(q)$ as an integral:

$$\begin{aligned} ELBO(q) &= \mathbb{E}_{q(z)}[\log p(\boldsymbol{x}|\boldsymbol{z})] - KL(q(\boldsymbol{z})\|p(\boldsymbol{z})) \\ &= \mathbb{E}_{q(z)}[\log p(\boldsymbol{x}|\boldsymbol{z}) + \log p(\boldsymbol{z}) - \log q(\boldsymbol{z})] \\ &= \mathbb{E}_{q(z)}[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z})] \\ &= \int_{\mathbb{R}^M} q(\boldsymbol{z})(\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}))\, \mathrm{d}\boldsymbol{z}. \end{aligned}$$

Substituting $q(\boldsymbol{z}) = \prod_{i=1}^{M} q_i(z_i)$ and factoring out $q_j(z_j)$ yields:

$$\begin{aligned} ELBO(q) &= \int_{\mathbb{R}^M} \left[\prod_{i=1}^{M} q_i(z_i)\right] \left(\log p(\boldsymbol{x}, \boldsymbol{z}) - \sum_{i=1}^{M} \log q_i(z_i)\right) \mathrm{d}\boldsymbol{z} \\ &= \int_{\mathbb{R}} q_j(z_j) \left(\int_{\mathbb{R}^{M-1}} \log p(\boldsymbol{x}, \boldsymbol{z}) \prod_{i\neq j} q_i(z_i)\, \mathrm{d}\boldsymbol{z}_{-j}\right) \mathrm{d}z_j \\ &\quad - \int_{\mathbb{R}} q_j(z_j) \left(\int_{\mathbb{R}^{M-1}} \left[\prod_{i\neq j} q_i(z_i)\right] \sum_{i=1}^{M} \log q_i(z_i)\, \mathrm{d}\boldsymbol{z}_{-j}\right) dz_j \\ &= \int_{\mathbb{R}} q_j(z_j) \mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})]\, \mathrm{d}z_j \\ &\quad - \int_{\mathbb{R}} q_j(z_j) \log q_j(z_j) \left(\int_{\mathbb{R}^{M-1}} \prod_{i\neq j} q_i(z_i)\, \mathrm{d}z_{-j}\right) \mathrm{d}z_j \\ &\quad - \int_{\mathbb{R}} q_j(z_j) \left(\int_{\mathbb{R}^{M-1}} \left[\prod_{i\neq j} q_i(z_i)\right] \sum_{i\neq j} \log q_i(z_i)\, \mathrm{d}\boldsymbol{z}_{-j}\right) \mathrm{d}z_j \\ &= \int_{\mathbb{R}} q_j(z_j) \mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})]\, \mathrm{d}z_j - \int_{\mathbb{R}} q_j(z_j) \log q_j(z_j)\, \mathrm{d}z_j \\ &\quad - \int_{\mathbb{R}^{M-1}} \left[\prod_{i\neq j} \log q_i(z_i)\right] \sum_{i\neq j} \log q_i(z_i)\, \mathrm{d}\boldsymbol{z}_{-j} \qquad \text{(C.0.1)} \\ &= \int_{\mathbb{R}} q_j(z_j) \left(\mathbb{E}_{z_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \log q_j(z_j)\right) \mathrm{d}z_j + C, \qquad \text{(C.0.2)} \end{aligned}$$

where C is a constant.

The term in Expression (C.0.1) is constant with respect to $q_j(z_j)$. We want to maximize $ELBO(q)$, so we formulate the Lagrangian equation with the constraint that $q_i(z_i)$ are probability density functions:

$$ELBO(q) - \sum_{i=1}^{M} \lambda_i \int_{\mathbb{R}} q_i(z_i)\,\mathrm{d}z_i = 0,$$

or using Expression (C.0.2),

$$\int_{\mathbb{R}} q_j(z_j) \left(\mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \log q_j(z_j)\right) \mathrm{d}z_j - \sum_{i=1}^{M} \lambda_i \int_{\mathbb{R}} q_i(z_i)\,\mathrm{d}z_i + C = 0. \quad \text{(C.0.3)}$$

Using the Euler-Lagrange equation, we then take the functional derivative of Equation (C.0.3) with respect to $q_j(z_j)$ [Bishop, 2006]:

$$\begin{aligned}\frac{\partial ELBO(q)}{\partial q_j(z_j)} &= \frac{\partial}{\partial q_j(z_j)} \left[q_j(z_j) \left(\mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \log q_j(z_j)\right) - \lambda_j q_j(z_j)\right] \\ &= \mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \log q_j(z_j) - 1 - \lambda_j. \end{aligned} \quad \text{(C.0.4)}$$

Equating Expression (C.0.4) to 0 and observing that $1 + \lambda_j$ is constant with respect to z, we have:

$$\begin{aligned}\log q_j^*(z_j) &= \mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathrm{C} \\ q_j^*(z_j) &= \frac{\exp(\mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})])}{\exp(\mathrm{C})} \\ &= \frac{\exp(\mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})])}{\int_{\mathbb{R}} \exp(\mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})])\,\mathrm{d}z_j}. \end{aligned} \quad \text{(C.0.5)}$$

The normalization constant on the denominator of Expression (C.0.5) is derived by observing $q_j^*(z_j)$ as a density. Finally, we derive a simpler expression of $q_j^*(z_j)$ by observing that terms independent of z_j can be treated as a constant:

$$\begin{aligned}q_j^*(z_j) &\propto \exp\left(\mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})]\right) \\ &\propto \exp\left(\mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(z_j | \boldsymbol{z}_{-j}, \boldsymbol{x})]\right). \end{aligned} \quad \text{(C.0.6)}$$

Appendix D

Mean Field Variational Inference Example

To illustrate the mean-field variational inference approach, we closely follow the "Bayesian mixture of Gaussians" example from "Variational Inference: A Review for Statisticians" by Blei et al. [2017].
Consider the hierarchical model

$$\begin{aligned} \mu_k &\sim \mathcal{N}(0, \sigma^2), && k = 1, \dots, K, \\ c_i &\sim \text{Categorical}\left(1; \frac{1}{K}, \dots, \frac{1}{K}\right), && i = 1, \dots, n, \\ x_i | c_i, \boldsymbol{\mu} &\sim \mathcal{N}(c_i^\top \boldsymbol{\mu}, 1), && i = 1, \dots, n, \end{aligned}$$

where $\boldsymbol{\mu} = (\mu_1, \dots, \mu_K)^\top$.
This is a Bayesian mixture of univariate Gaussian random variables with unit variance. In this model, we draw K identical copies of the variable μ_k from a prior Gaussian density $\mathcal{N}(0, \sigma^2)$ (σ^2 is a fixed hyperparameter), forming the vector $\boldsymbol{\mu}$. We then generate an indicator vector c_i of length K from a prior categorical density. This vector has zeros for every element except for one element, where it is 1. Each element has equal probability $1/K$ of being the non-zero element. The transpose of this c_i is then multiplied by $\boldsymbol{\mu}$. This is a practical implementation to choose one of the $\boldsymbol{\mu}_k$ at random. We then draw x_i from the resulting $\mathcal{N}(c_i^\top \boldsymbol{\mu}, 1)$.
Defining $\boldsymbol{c} = (c_1, \dots, c_n)^\top$, our latent variables are $\boldsymbol{z} = \{\boldsymbol{c}, \boldsymbol{\mu}\}$. Assuming n samples, our joint density is

$$p(\boldsymbol{\mu}, \boldsymbol{c}, \boldsymbol{x}) = p(\boldsymbol{\mu}) \prod_{i=1}^{n} p(c_i) p(x_i | c_i, \boldsymbol{\mu}). \tag{D.0.1}$$

To evaluate the posterior density over the latent variables $p(\boldsymbol{\mu}, \boldsymbol{c} | \boldsymbol{x})$, we apply variational inference, approximating it with a variational density $q(\boldsymbol{\mu}, \boldsymbol{c})$. We will assume

this density follows the mean-field variational family:

$$q(\boldsymbol{\mu}, \boldsymbol{c}) = \prod_{k=1}^{K} q(\mu_k; m_k, s_k^2) \prod_{i=1}^{n} q(c_i; \boldsymbol{\phi_i}).$$

In this density, we have K Gaussian factors with mean m_k and variance s_k^2, and n categorical factors with index probabilities defined by the vector $\boldsymbol{\phi_i}$, such that

$$\begin{aligned} \mu_k &\sim \mathcal{N}(m_k, s_k^2), && k = 1, \ldots, K, \\ c_i &\sim \text{Categorical}(\boldsymbol{\phi_i}), && i = 1, \ldots, n. \end{aligned}$$

Using this and Equation (C.0.1), we can derive the evidence lower bound as a function of the variational parameters $\boldsymbol{m} = (m_1, \ldots, m_k)^\top$, $\boldsymbol{s}^2 = (s_1^2, \ldots, s_k^2)^\top$ and $\boldsymbol{\phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n]^\top$:

$$\begin{aligned} ELBO(\boldsymbol{m}, \boldsymbol{s}^2, \boldsymbol{\phi}) &= \mathbb{E}_{p(\mathbf{z}, \boldsymbol{x})}[\log p(\mathbf{z}, \boldsymbol{x})] - \mathbb{E}_{q(\boldsymbol{z})}[\log q(\boldsymbol{z})] \\ &= \mathbb{E}_{p(\boldsymbol{\mu}, \boldsymbol{c}, \mathbf{x})}[\log p(\boldsymbol{\mu}, \boldsymbol{c}, \mathbf{x})] - \mathbb{E}_{q(\boldsymbol{\mu}, \boldsymbol{c})}[\log q(\boldsymbol{\mu}, \boldsymbol{c})] \\ &= \sum_{i=1}^{K} \mathbb{E}_{p(\mu_k)}[\log p(\mu_k); m_k, s_k^2] \\ &\quad + \sum_{i=1}^{n} \left(\mathbb{E}_{p(c_i)}[\log p(c_i); \boldsymbol{\phi}_i] + \mathbb{E}_{p(x_i | c_i, \boldsymbol{\mu})}[\log p(x_i | c_i, \boldsymbol{\mu}); \boldsymbol{\phi}_i, \boldsymbol{m}, \boldsymbol{s}^2] \right) \\ &\quad - \sum_{k=1}^{K} \mathbb{E}_{q(\mu_k; m_k, s_k^2)}[\log q(\mu_k; m_k, s_k^2)] - \sum_{i=1}^{n} \mathbb{E}_{q(c_i; \boldsymbol{\phi}_i)}[\log q(c_i; \boldsymbol{\phi}_i)]. \end{aligned}$$

From the CAVI algorithm in Equation (3.5.7), which we restate to be

$$q_j^*(z_j) \propto \exp\left(\mathbb{E}_{\boldsymbol{z}_{-j}}[\log p(\boldsymbol{x}, \boldsymbol{z})] \right),$$

we derive the optimal categorical factor by only considering terms from the true density $p(\cdot)$ dependent on c_i:

$$q^*(c_i; \boldsymbol{\phi}_i) \propto \exp\left(\log p(c_i) + \mathbb{E}_{p(x_i | c_i, \boldsymbol{\mu})}[\log p(x_i | c_i, \boldsymbol{\mu}); \boldsymbol{m}, \boldsymbol{s}^2] \right). \tag{D.0.2}$$

Now since $\boldsymbol{c}_i = (c_{i1}, \ldots, c_{iK})^\top$ is an indicator vector, we have:

$$p(x_i | \boldsymbol{c}_i, \boldsymbol{\mu}) = \prod_{k=1}^{K} p(x_i | \mu_k)^{c_{ik}}.$$

We can now evaluate the second term of Equation (C.0.2):

$$\begin{aligned}
\mathbb{E}_{p(x_i|\boldsymbol{c}_i,\boldsymbol{\mu})}\left[\log p(x_i|c_i,\boldsymbol{\mu});\boldsymbol{m},\boldsymbol{s}^2\right] &= \sum_{k=1}^{K} c_{ik}\mathbb{E}_{p(x_i|\mu_k)}[\log p(x_i|\mu_k);m_k,s_k^2] \\
&= \sum_{k=1}^{K} c_{ik}\mathbb{E}_{x_i}[-(x_i-\mu_k)^2/2;m_k,s_k^2] + C \\
&= \sum_{k=1}^{K} c_{ik}\left(\mathbb{E}_{\mu_k}[\mu_k;m_k,s_k^2]x_i - \mathbb{E}_{\mu_k^2}[\mu_k^2;m_k,s_k^2]/2\right) \\
&\quad + C.
\end{aligned}$$

In each line, terms constant with respect to c_{ik} have been absorbed into the constant C. Our optimal categorical factor becomes

$$q^*(c_i;\boldsymbol{\phi}_i) \propto \exp\left(\log p(c_i) + \sum_{k=1}^{K} c_{ik}\left(\mathbb{E}_{\mu_k}[\mu_k;m_k,s_k^2]x_i - \mathbb{E}_{\mu_k^2}[\mu_k^2;m_k,s_k^2]/2\right)\right).$$

By proportionality, we then have the variational update

$$\phi_{ik} \propto \exp\left(\mathbb{E}_{\mu_k}[\mu_k;m_k,s_k^2]x_i - \mathbb{E}_{\mu_k^2}[\mu_k^2;m_k,s_k^2]/2\right).$$

Now we find the variational density of the kth mixture component, again using Equation (3.5.7) with the ELBO and ignoring terms independent of $p(\cdot)$ and μ_k:

$$q(\mu_k;m_k,s_k^2) \propto \exp\left(\log p(\mu_k) + \sum_{i=1}^{n} \mathbb{E}_{p(x_i|c_i,\boldsymbol{\mu})}[\log p(x_i|c_i,\boldsymbol{\mu});\boldsymbol{\phi}_i,\boldsymbol{m}_{-k},\boldsymbol{s}_{-k}^2]\right).$$

The log of this density is

$$\begin{aligned}
\log q(\mu_k) &= \log p(\mu_k) + \sum_{i=1}^{n} \mathbb{E}_{p(x_i|c_i,\boldsymbol{\mu})}[\log p(x_i|c_i,\boldsymbol{\mu});\boldsymbol{\phi}_i,\boldsymbol{m}_{-k},\boldsymbol{s}_{-k}^2] + C \\
&= \log p(\mu_k) + \sum_{i=1}^{n} \mathbb{E}_{c_{ik},p(x_i|\mu_k)}[c_{ik}\log p(x_i|\mu_k);\boldsymbol{\phi}_i] + C \\
&= -\frac{\mu_k^2}{2\sigma^2} + \sum_{i=1}^{n} \mathbb{E}_{c_{ik}}[c_{ik};\boldsymbol{\phi}_i]\log p(x_i|\mu_k) + C \\
&= -\frac{\mu_k^2}{2\sigma^2} + \sum_{i=1}^{n} \phi_{ik}\frac{-(x_i-\mu_k)^2}{2} + C \\
&= -\frac{\mu_k^2}{2\sigma^2} + \sum_{i=1}^{n} \left(\phi_{ik}x_i\mu_k - \frac{\phi_{ik}\mu_k^2}{2}\right) + C
\end{aligned}$$

$$= \mu_k \left(\sum_{i=1}^{n} \phi_{ik} x_i \right) - \mu_k^2 \left(\frac{1}{2\sigma^2} + \frac{\sum_{i=1}^{n} \phi_{ik}}{2} \right) + C$$
$$= -\frac{1}{2} \left(\frac{1}{\sigma^2} + \sum_{i=1}^{n} \phi_{ik} \right) \left(\mu_k^2 - \frac{2 \sum_{i=1}^{n} \phi_{ik} x_i}{1/\sigma^2 + \sum_{i=1}^{n} \phi_{ik}} \mu_k \right) + C.$$

The density is therefore

$$q(\mu_k) \propto \sqrt{\frac{1/\sigma^2 + \sum_{i=1}^{n} \phi_{ik}}{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{1}{\sigma^2} + \sum_{i=1}^{n} \phi_{ik} \right) \left(\mu_k - \frac{\sum_{i=1}^{n} \phi_{ik} x_i}{1/\sigma^2 + \sum_{i=1}^{n} \phi_{ik}} \right)^2 \right).$$

It can be seen that $q(\mu_k)$ is a Gaussian density, so our variational updates for m_k and s_k^2 are its mean and variance:

$$m_k = \frac{\sum_{i=1}^{n} \phi_{ik} x_i}{1/\sigma^2 + \sum_{i=1}^{n} \phi_{ik}}, \qquad s_k^2 = \frac{1}{1/\sigma^2 + \sum_{i=1}^{n} \phi_{ik}}.$$

We can now formulate the CAVI algorithm (Algorithm 10), which simply iterates the cluster assignment probabilities ϕ_{ik} and the variational density parameters m_k and s_k^2 until the ELBO converges.

Data: Data $\boldsymbol{x}$, Number of Gaussian components K, Hyperparameter value σ^2

Result: Optimal variational factors $q(\mu_k; m_k, s_k^2)$ and $q(c_i; \boldsymbol{\phi_i})$

begin

  Randomly initialize parameters $\boldsymbol{m}, \boldsymbol{s}^2$ and $\boldsymbol{\phi}$;

  while *ELBO has not converged* **do**

    for $i = 1$ **to** n **do**

      Set $\phi_{ik} \propto \exp\left(\mathbb{E}_{\mu_k}[\mu_k; m_k, s_k^2] x_i - \mathbb{E}_{\mu_k^2}[\mu_k^2; m_k, s_k^2]/2 \right)$;

    end

    for $k = 1$ **to** K **do**

      Set $m_k = \frac{\sum_i \phi_{ik} x_i}{1/\sigma^2 + \sum_i \phi_{ik}}$;

      Set $s_k^2 = \frac{1}{1/\sigma^2 + \sum_i \phi_{ik}}$;

    end

    Compute $ELBO(\boldsymbol{m}, \boldsymbol{s}^2, \boldsymbol{\phi})$;

  end

  Return $q(\boldsymbol{m}, \boldsymbol{s}^2, \boldsymbol{\phi})$;

end

Algorithm 10: CAVI Algorithm for Bayesian mixture of Gaussians

Appendix E

Kernel Density Estimation

Kernel density estimation is a non-parametric method used to estimate the probability density function of a distribution, using only samples [Zanin Zambom and Dias, 2012]. It can therefore be used to estimate implicit densities. For simplicity we only explain the univariate form of the kernel density estimator.

Let $\{x^{(i)}\}_{i=1}^{n}$ be an i.i.d. sample from a distribution with unknown probability density function f. Its kernel density estimator is defined as

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x^{(i)}}{h}\right).$$

K is the kernel, a symmetric non-negative weighting function that integrates to 1. Examples of kernel functions are:

- Epanechnikov: $K(u) = \frac{3}{4}(1 - u^2), |u| \leq 1$,
- Uniform: $K(u) = \frac{1}{2}, |u| \leq 1$,
- Gaussian: $K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}u^2\right)$.

Typically, the Gaussian kernel is used due to its statistical properties.

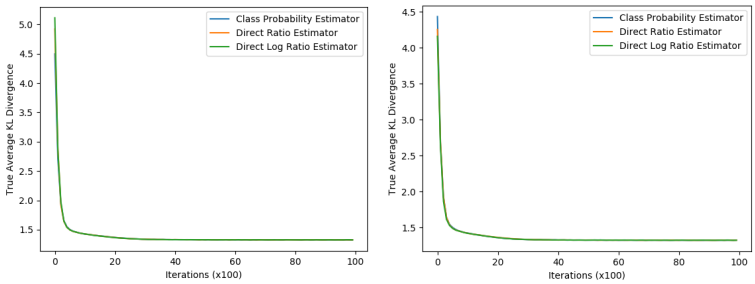
The bandwidth $h > 0$ acts as a smoothing parameter, determining the width of the kernel. If h is too small, $\hat{f}$ will be 'undersmoothed' as too much weight is placed on the areas nearest the data-points, leading to a spiky estimate with high variance. On the other hand, if h is too large, $\hat{f}$ will be 'oversmoothed' with too little weight on areas nearest to the data-points, resulting in a relatively flat estimate with high bias. It is therefore ideal to choose h such that the mean integrated square error $MISE(h) = \mathbb{E}\left[\int (\hat{f}_h(x) - f(x))^2 dx\right]$ is minimized. For a Gaussian kernel, this occurs at approximately $h = 1.06\hat{\sigma}n^{-1/5}$ where $\hat{\sigma}$ is the sample standard deviation.

The kernel density estimator works by placing a kernel on each data point and summing up the kernels to produce a smooth curve. Each point on the curve is essentially a weighted average of nearby data points. Regions of the curve with many data points will therefore have a high estimated probability density.

Appendix F

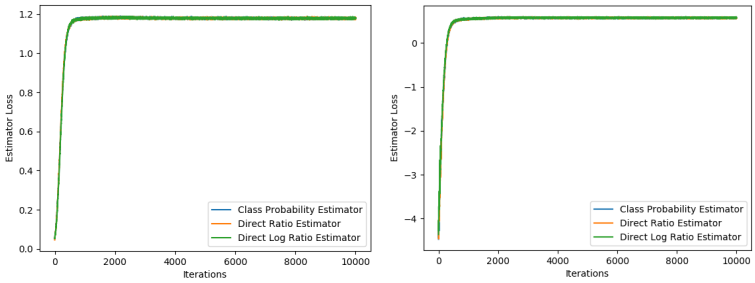
Prior-Contrastive Optimal Estimator Experiment Plots

Note that in the following plots, there is no noticeable difference between the estimator parametrisations or the f-divergences.



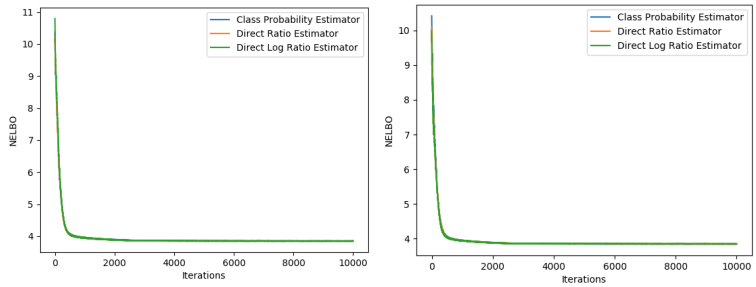
(a) GAN Divergence (b) Reverse KL Divergence

Figure F.1: Prior-Contrastive Average KL Divergence



(a) GAN Divergence (b) Reverse KL Divergence

Figure F.2: Prior-Contrastive Estimator Loss



(a) GAN Divergence (b) Reverse KL Divergence

Figure F.3: Prior-Contrastive $NELBO$

Appendix G

Second Functional Derivatives of Direct Log Ratio Estimator Losses

Reverse KL Divergence Bound

Direct Log Ratio Estimator:

$$\begin{aligned}
L_{RKL}(u) &\coloneqq -\mathbb{E}_{q(u)}[T_\alpha(u)] + \mathbb{E}_{p(u)}\left[e^{T_\alpha(u)}\right] \\
&= -\int_U q(u)T_\alpha(u)\,\mathrm{d}u + \int_U p(u)e^{T_\alpha(u)}\,\mathrm{d}u. \\
\frac{\partial L_{RKL}(u)}{\partial T_\alpha(u)} &= -q(u) + p(u)e^{T_\alpha(u)}. \\
\frac{\partial^2 L_{RKL}(u)}{\partial T_\alpha^2(u)} &= p(u)e^{T_\alpha(u)}.
\end{aligned}$$

GAN Divergence Bound

Direct Log Ratio Estimator:

$$\begin{aligned}
L_{GAN}(u) &\coloneqq -\mathbb{E}_{q(u)}\left[T_\alpha(u) - \log\left(e^{T_\alpha(u)}+1\right)\right] + \mathbb{E}_{p(u)}\left[\log\left(e^{T_\alpha(u)}+1\right)\right] \\
&= \int_U q(u)\left[\log\left(e^{T_\alpha(u)}+1\right) - T_\alpha(u)\right]\,\mathrm{d}u + \int_U p(u)\left[\log\left(e^{T_\alpha(u)}+1\right)\right]\,\mathrm{d}u. \\
\frac{\partial L_{GAN}(u)}{\partial T_\alpha(u)} &= -q(u) + \frac{(q(u)+p(u))e^{T_\alpha(u)}}{e^{T_\alpha(u)}+1}. \\
\frac{\partial^2 L_{GAN}(u)}{\partial T_\alpha^2(u)} &= \frac{(q(u)+p(u))e^{T_\alpha(u)}}{e^{T_\alpha(u)}+1} - \frac{(q(u)+p(u))e^{2T_\alpha(u)}}{(e^{T_\alpha(u)}+1)^2} \\
&= \frac{(q(u)+p(u))(e^{2T_\alpha(u)}+e^{T_\alpha(u)}) - (q(u)+p(u))e^{2T_\alpha(u)}}{(e^{T_\alpha(u)}+1)^2} \\
&= \frac{(q(u)+p(u))e^{T_\alpha(u)}}{(e^{T_\alpha(u)}+1)^2}.
\end{aligned}$$