A Brief Overview on Generative Adversarial Network and Variational Bayes

by

Ziyue Yang

Final Report

STA492, Winter 2022

University of Toronto

Contents

1 Introduction **1**

2 Generative Modeling **2**

2.1 Generative Adversarial Networks [4] 2

2.1.1 Global Optimality of $p_g = p_{\text{data}}$ 3

2.1.2 Optimization Algorithm 5

2.2 Why We Study Generative Modelling [3] 7

3 Variational Bayes [6] **10**

3.1 Setup 10

3.2 Mean Field Variational Bayes 12

4 Conclusion **20**

Chapter 1

Introduction

In the report, we will be discussing two main topics:

- Generative Adversarian Networks, or GANs, is a generative framework that captures the distribution of data through an adversarial process, by training a generative model and a discriminative model simultaneously.

- Variational Bayes: Variational Bayes is a family of methods for approximating intractable integrals with the use Bayesian inference. They are often used to efficiently compute estimations of parameters in complex and high-dimensional statistical models.

These two topics were my main research focus throughout seminar STA492 at the University of Toronto during the 2022 winter term. In the following sections, we will briefly go through the broad ideas of both topics, as well as providing some intuitive demonstrations using simple examples.

Chapter 2

Generative Modeling

2.1 Generative Adversarial Networks [4]

The **Generative Adversarial Network**, or **GAN**, is a framework for *estimating generative models* through an **adversarial** process. In the GAN framework, we train the following models simultaneously on input data $\mathbf{x}$:

- G: generative model that captures the distribution of data.
- D: discriminative model that estimates the probability that a sample is from the training data, rather than generated from model G.

Additionally, we will add independent random noises $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$, on the input, where μ and σ are the parameters.

We would want to learn p_g, the distribution generated by G, from data input $\mathbf{x}$. To do so, we define a prior, $p_{\mathbf{z}}(\mathbf{z})$, on the input noise variables $\mathbf{z}$, then we represent the generator using a differentiable function G as a mapping from the noises to the data space, as $G(\mathbf{z}; \theta_g)$.

For the output, we also need to define a mapping $D(\mathbf{x}; \theta_d)$ that outputs a scalar.

Now, our goal is to train D to maximize the probability of assigning the correct labelling to **both** samples from G and training samples. Therefore, we want to train the generative model G to minimize the log-likelihood $\log(1 - D(G(\mathbf{z})))$. This gives us a *two-player minimax game* with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]. \tag{2.1.1}$$

In the following section, we will carry out a theoretical analysis showing that 2.1.1 allows us to recover the data-generating distribution when G and D are in the non-parametric limit. In practice, the game will be implemented using an iterative approach, which will also be discussed in a later section.

2.1.1 Global Optimality of $p_g = p_{\text{data}}$

Definition 2.1.1. Suppose that $f : X \to \mathbb{R}$ is a real-valued function with an arbitrary set X as the domain. The set-theoretic support of f, written $Supp(f)$, is the set of points in X where f is non-zero:

$$Supp(f) = \{x \in X : f(x) \neq 0\}. \tag{2.1.2}$$

Consider the optimal discriminator D for any given generator G.

Proposition 2.1.1. For a fixed G, the optimal discriminator D is given by

$$D_G^*(\mathbf{x}) = \frac{p_{\text{data}(\mathbf{x})}}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}. \tag{2.1.3}$$

Proof. We want to maximize the quantity $V(G, D)$

$$V(G, D) = \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x}))dx + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z}) \log(1 - D(g(\mathbf{z})))dz \tag{2.1.4}$$

$$= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x}))dx. \tag{2.1.5}$$

For any $(a, b) \in \mathbb{R}^2 \backslash \{0, 0\}$, the function $y \to a \log(y) + b \log(1-y)$ achieves a maximum in $[0, 1]$ at point $a/(a + b)$. Since we do not need to define the discriminator outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, the proof is concluded. ■

Notice that the training objective for D is equivalent to maximizing the log-likelihood for estimating the conditional probability $P(Y = y \mid \mathbf{x})$, where Y is a binary variable indicating whether $\mathbf{x}$ comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). The minimax game in Eq. 2.1.1 can be reformulated as:

$$C(G) = \max_D V(G, D) \tag{2.1.6}$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log D_G^*(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}}[\log(1 - D_G^*(G(\mathbf{z})))] \tag{2.1.7}$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log D_G^*(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}}[\log(1 - D_G^*(\mathbf{x}))] \tag{2.1.8}$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right] + \mathbb{E}_{\mathbf{x} \sim p_g} \left[\log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right]. \tag{2.1.9}$$

Theorem 2.1.1. The global minimum of the criterion $C(G)$ is achieved if and only if $p_g = p_{\text{data}}$, where $C(G)$ achieves the value $-\log(4)$.

Proof. For $p_g = p_{\text{data}}$, $D^*_G(\mathbf{x}) = 1/2$ (consider Eq. 2.1.3. Therefore by Eq. 2.1.7 at $D^*_G(\mathbf{x}) = 1/2$, we find that $C(G) = \log(1/2) + \log(1/2) = -\log(4)$.

Note that this is the best possible value of $C(G)$, which is only reached when $p_g = p_{\text{data}}$. Observe that

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[-\log(2)] + \mathbb{E}_{\mathbf{x}\sim p_g}[-\log(2)] = -\log(4), \tag{2.1.10}$$

and that by subtracting this from $C(G) = V(D^*_G, G)$, we obtain that

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \| \frac{p_g + p_{\text{data}}}{2}\right) + KL\left(p_g \| \frac{p_g + p_{\text{data}}}{2}\right), \tag{2.1.11}$$

where KL denotes the Kullback-Leibler divergence. It is recognized that the Jensen-Shannon divergence between the model's distribution and the data generating process is

$$C(G) = -\log(4) + 2 \cdot JSD(p_{\text{data}} \| p_g), \tag{2.1.12}$$

since the Jensen-Shannon divergence between two distributions is always nonnegative, and is zero only when they are equal, we know that $C^* = -\log(4)$ is the global minimum of $C(G)$ and the only solution is $p_g = p_{\text{data}}$, at the point where the generative model perfectly replicates the data generating process. ■

2.1.2 Optimization Algorithm

Here we show an iterative algorithm to optimize Eq. 2.1.1.

Algorithm 1. Minibatch gradient descent training of GANs. The number of steps to apply to D is a hyperparameter k.

- **for** the number of training iterations **do**

– **for** k steps **do**

* Sample m noise samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$[1].

* Sample m examples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.

* Update the discriminator D by ascending the *stochastic gradient*:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)}))) \right]. \tag{2.1.13}$$

end for

- Again, sample m noise samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.

- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}^{(i)}))). \tag{2.1.14}$$

end for

Proposition 2.1.2. If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, such that p_g is updated to improve the criterion

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log D_G^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g}[\log(1 - D_G^*(\mathbf{x}))], \tag{2.1.15}$$

then p_g converges to p_{data}.

[1] The m samples here are called a *mini batch*.

Proof. This proof is adapted from the original GAN paper ([4]).

Consider $V(G, D) = U(p_g, D)$ as a function of p_g as in Eq. 2.1.15. Since $U(p_g, D)$ is convex in p_g, and the subderivates of a supremum of convex functions include the derivative of the function at the point where the optimum is obtained. Therefore equivalently, we compute the updates for p_g at optimal D given the corresponding G. Then $\sup_D U(p_g, D)$ is convex in p_g with a unique global optima, as proven in Theorem 2.1.1 ■

2.2 Why We Study Generative Modelling [3]

A question may arises when we mention generative modelling: why is generative modelling worth studying, particularly those that are only capable of generating data rather than estimating the density function. Well, there are a few reasons:

- Sampling from generative models allows us to test our capability to manipulate high-dimensional probability distributions, which are important in many domains such as engineering and math.

- Generative models can be integrated into reinforcement learning[2].

  - Reinforcement learning algorithms can be separated into two categories: *model-based* and *model-free*. Here model-based algorithms are the ones that contain a generative model.

  - For time-series data, we can use generative models to simulate possible futures by learning a conditional distribution over future states given the current state and hypothetical actions (see [2] for an example of using such model for *planning*).

[2]Tongfei has covered this topic very well, please refer to his report for more details. Here we will go over some big ideas.

- Generative models can be trained when there are missing data by predicting inputs that are missing data. In modern machine learning algorithms, particularly in deep learning, it typically requires an enormous amount of *labeled data*[3] for training a well-generalized model. With the use of generative modelling, we will be able to improve model generalization by studying unlabeled data, which are usually easier to obtain than labeled data.

- Generative models will enable machine learning algorithms to work with multi-modal outputs. For many tasks, one input may correspond to lots of correct answers that are acceptable. Traditional machine learning models, such as minimizing the mean square error between the model's predicted output and a desired output, are not able to train models that can produce multiple correct but different answers. For example, Lotter *et al.* [5] has an illustration of the importance of modelling multi-modal data. In the example shown in figure 2.1 (the three rows on the right), a model is trained to predict the next frame in a video sequence. The video captures a computer rendering of a moving 3D model of a person's head. The image on the left shows the actual frame of the video, which is treated as the true data; the center image shows the results when the model is trained using MSE between the actual frame and the model's predicted frame; the image on the right, where the model is forced to choose a single answer for what the frame will look like. Since there are lots of possible futures, the single answer chosen by the model corresponds to a mean over many slightly different results. The image on the right uses a GAN loss, where the model is able to understand that there are lots of possible outputs, each of which is sharp and recognizable as a realistic and detailed image.

[3]Labeled data are data that have been tagged with one or more labels. For instance, an image containing a horse labeled with "horse".

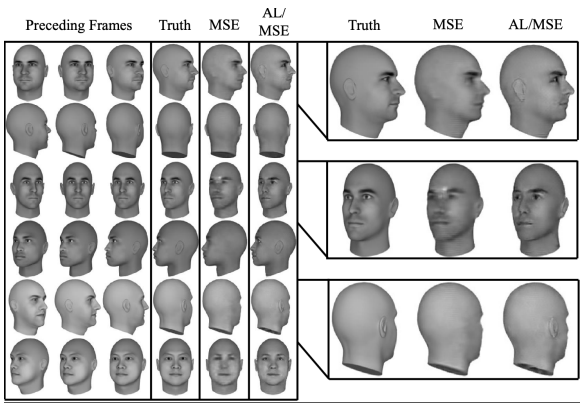

Figure 2.1: Lotter *et al.*

Chapter 3

Variational Bayes [6]

In this chapter, we will briefly discuss about variational Bayes (VB). Variational Bayes is an optimization-based method for approximating Bayesian inference, which provides a computationally efficient alternative to sampling methods: sampling methods such as Markov Chain Monte Carlo (MCMC) and Sequential Monte Carlo (SMC) are currently not capable of dealing with large data sets, and are not applicable in complex models with a huge number of unknown parameters.

In the following sections, we will be discussing the Mean Field Variational Bayes (MFVB) algorithm for estimating parameters.

3.1 Setup

We will begin by introducing the Bayesian Inference setup. Let y denote the data, and let $p(y \mid \theta)$ denote the likelihood function with $\theta \in \Theta$ being the vector of model parameters to be estimated. Let $p(\theta)$ be the prior.

Bayesian inference has all the information about the model parameter θ in the pos-

terior distribution with density

$$p(\theta \mid y) = \frac{p(y,\theta)}{p(y)} = \frac{p(\theta)p(y \mid \theta)}{(y)} \propto p(\theta)p(y \mid \theta), \tag{3.1.1}$$

where

$$p(y) = \int_{\Theta} p(\theta)p(y \mid \theta)d\theta. \tag{3.1.2}$$

Here $p(y)$ is known as the *marginal likelihood*.

In Bayesian Inference, it normally requires us to compute expectations with respect to the posterior distribution. For instance, the posterior mean is an expectation of θ with respect to the posterior distribution $p(\theta \mid y)$; however, it is usually impractical to compute such expectations analytically, since the density is intractable and the normalizing constant of $p(y)$ is often unknown. In many cases, the inference is performed using MCMC, where the expectations are estimated by sampling from $p(\theta \mid y)$; however, when the dimension gets higher, or when an efficient algorithm is required, VB is often a better alternative to MCMC.

Generally, VB approximate the posterior distribution by a probability distribution with density $q(\theta)$, which belongs to some tractable family of distributions $\mathcal{Q}$ (e.g. Gaussians, Exponentials). The optimal VB approximation, $q^* \in \mathcal{Q}$, is given by minimizing the Kullback-Leibler divergence from $q(\theta)$ to $p(\theta \mid y)$:

$$q^* = \arg\min_{q \in \mathcal{Q}} \left\{ KL(q \| p(\cdot|y)) = \int q(\theta) \log \frac{q(\theta)}{p(\theta|y)} d\theta \right\}. \tag{3.1.3}$$

Then the Bayesian inference is performed with the (intractable) posterior $p(\theta \mid y)$

replaced by the tractable VB approximation $q^*(\theta)$. Notice that

$$KL(q\|p(\cdot|y)) = -\int q(\theta)\log\frac{p(\theta)p(y|\theta)}{q(\theta)}d\theta + \log(p(y)), \tag{3.1.4}$$

therefore the objective of minimizing the above KL divergence is equivalent to maximizing the lower bound (denoted LB) on $\log(p(y))$:

$$\text{LB}(q) = \int q(\theta)\log\frac{p(\theta)p(y|\theta)}{q(\theta)}d\theta = \mathbb{E}_q\left(\log\frac{p(\theta)p(y|\theta)}{q(\theta)}\right). \tag{3.1.5}$$

Some notes on notation: For random variable X and any function $g(x)$, $\mathbb{E}_f(g(X))$ denotes the expectation of $g(X)$ with X following a probability distribution with density function $f(x)$.

Hence our objective now is to maximize the lower bound derived in Eqn. 3.1.5.

3.2 Mean Field Variational Bayes

We begin with a simple case with dimension of 2. Write $\theta = (\theta_1^T, \theta_2^T)^T$, and we assume that q can be written in the following factorization form:

$$q(\theta) = q_1(\theta_1)q_2(\theta_2). \tag{3.2.1}$$

In other words, we ignore the posterior dependence between θ_1 and θ_2. Our goal is to approximate $p(\theta_1, \theta_2|y)$ given $q(\theta)$[1].

Then we can derive the lower bound in Eqn. 3.1.5:

[1]Note that this is the only assumption on class $\mathcal{Q}$

$$LB(q_1, q_2) = \int q_1(\theta_1)q_2(\theta_2) \log\left(\frac{p(\theta, y)}{q_1(\theta_1)q_2(\theta_2)}\right) d\theta_1\theta_2 \tag{3.2.2}$$

$$= \int q_1(\theta_1)q_2(\theta_2) \log[p(\theta, y)]d\theta_1 d\theta_2 \tag{3.2.3}$$

$$- \int q_1(\theta_1) \log[q_1(\theta_1)]d\theta_1 - \int q_2(\theta_2)log[q_2(\theta_2)]d\theta_2 \tag{3.2.4}$$

$$= \int q_1(\theta_1)\mathbb{E}_{-\theta_1}[\log(p(y, \theta))]d\theta_1 - \int q_1(\theta_1) \log[q_1(\theta_1)]d\theta_1 + C(q_2), \tag{3.2.5}$$

where

$$\mathbb{E}_{-\theta_1}[\log(p(y, \theta))] = \mathbb{E}_{q_2(\theta_2)}[\log(p(y, \theta))] = \int q_2(\theta_2) \log[p(y, \theta)]d\theta_2, \tag{3.2.6}$$

and $C(q_2)$ is the term that is independent to q_1. Note that the weird notation $\mathbb{E}_{-\theta_1}(\cdot)$ here denotes the expectation with respect to everything except θ_1[2].

By further combining two integrals, we obtain

$$LB(q_1, q_2) = \int q_1(\theta_1) \log \frac{\exp(\mathbb{E}_{-\theta_1}[\log(p(y, \theta)])}{q_1(\theta_1)} d\theta_1 + C(q_2) \tag{3.2.7}$$

$$= \int q_1(\theta_1) \log \frac{\tilde{q_1}(\theta_1)}{q_1(\theta_1)} d\theta_1 + C(q_2) + \log(\tilde{C}(q_2)) \tag{3.2.8}$$

$$= -KL(q_1 \| \tilde{q_1}) + C(q_2) + \log \tilde{C}(q_2), \tag{3.2.9}$$

where $\tilde{q_1}(\theta_1)$ is the probability density function given by

[2]This seems unnecessary in our 2D case, but when the dimension gets higher, this notation can be very convenient, since we do not really want to write all the parameters in one vector under the $\mathbb{E}$ notation

$$\tilde{q_1}(\theta_1) = \frac{\exp(\mathbb{E}_{-\theta_1}[\log(p(y,\theta))])}{\tilde{C}(q_2)} \propto \exp(\mathbb{E}_{-\theta_1}[\log(p(y,\theta))]), \tag{3.2.10}$$

where

$$\tilde{C}(q_2) = \int \exp(\mathbb{E}_{-\theta_1}[\log(p(y,\theta))] \tag{3.2.11}$$

is independent of q_1. Now we have

$$LB(q_1, q_2) = -KL(q_1 \| \tilde{q_2}) + \text{constant independent of } q_1. \tag{3.2.12}$$

And similarly, we can also obtain that

$$LB(q_1, q_2) = -KL(q_2 \| \tilde{q_2}) + \text{constant independent of } q_2, \tag{3.2.13}$$

where $\tilde{q_2} \propto \exp(Exp_{-\theta_2}[\log(p(y,\theta))])$ with

$$\mathbb{E}_{-\theta_2} = \int q_1(\theta_1) \log[p(y,\theta)] d\theta_1. \tag{3.2.14}$$

With Eqn. 3.2.12 and Eqn. 3.2.13, it suggests that we can use a coordinate ascent optimization method to maximize the lower bound. That is, given q_2, we minimize $KL(q_1 \| \tilde{q_1})$ to find q_1; and similarly given q_1, we minimize $KL(q_2 \| \tilde{q_2})$ to find q_2. Our goal is to solve the optimization problems

$$\min_{q_1}\{KL(q_1 \| \tilde{q_1})\} \qquad \text{and} \qquad \min_{q_2}\{KL(q_2 \| \tilde{q_2}\}. \tag{3.2.15}$$

Hopefully, solving the minimizing problems in Eqn. 3.2.15 is easier than minimizing the KL divergence in Eqn. 3.1.4.

Example 3.2.1 (Conjugate Prior)**.** Suppose that prior $p(\theta_1)$ belongs to a parametric density family $\mathcal{F}_1$, and prior $p(\theta_2)$ belongs to a parametric density family $\mathcal{F}_2$, then $\tilde{q_1}$ belongs to $\mathcal{F}_1$ and $\tilde{q_2}$ belongs to $\mathcal{F}_2$.

In the scenario, the solutions to 3.2.15 will be

$$q_1(\theta_1) = \tilde{q_1}(\theta_1) \in \mathcal{F}_1 \tag{3.2.16}$$

$$q_2(\theta_2) = \tilde{q_2}(\theta_2) \in \mathcal{F}_2 \tag{3.2.17}$$

To identify q_1 and q_2, we need to compute their parameters. To compute the parameters for q_1, we need to know q_2, and vice versa. Therefore, consider the following coordinate ascent-type algorithm, called **Mean Field Variational Bayes (MFVB)**, for maximizing the lower bound:

Algorithm (Mean Field Variational Bayes).

The MFVB algorithm updates the parameters of q_1, q_2 in the following way:

1. Initialize the parameter of $q_1(\theta_1)$.

2. Given $q_1(\theta_1)$, update the parameter of $q_2(\theta_2)$ using

$$q_2(\theta_2) \propto \exp(\mathbb{E}_{-\theta_2}[\log(p(y,\theta)]) = \mathbb{E}\left(\int q_1(\theta_1)\log[p(y,\theta_1,\theta_2)]d\theta_1\right) \tag{3.2.18}$$

3. Given $q_2(\theta_2)$, update the parameter of $q_1(\theta_1)$ using

$$q_1(\theta_1) \propto \exp(\mathbb{E}_{-\theta_1}[\log(p(y,\theta)]) = \mathbb{E}\left(\int q_2(\theta_2)\log[p(y,\theta_1,\theta_2)]d\theta_2\right) \tag{3.2.19}$$

4. Repeat the update steps 2 and 3 until a stopping condition is met.

A stopping condition here is to terminate the update once the change in the parameters in the posterior $q(\theta) = q_1(\theta_1)q_2(\theta_2)$ between two consecutive iterations is less than some threshold ϵ. In other words, denote the value of parameter of $q(\theta)$ at the i^{th} iteration as $\hat{q}^{(i)}(\theta)$, then the algorithm stops at the $i+1$ iteration if $|\hat{q}^{(i)}(\theta) - \hat{q}^{(i+1)}(\theta)| < \epsilon$[3].

Example 3.2.2. Let $y = (11; 12; 8; 10; 9; 8; 9; 10; 13; 7)$ be some obervations from distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2.

Suppose that we use a prior $\mathcal{N}(\mu_0, \sigma_0^2)$ for μ, and Inverse-Gamma(α_0, β_0) for σ^2, with hyperparameters

$$\mu_0 = 0 \quad \sigma_0 = 10 \quad \alpha_0 = 1 \quad \beta_0 = 1. \tag{3.2.20}$$

Suppose further that $q(\mu, \sigma^2)$ follows the VB factorization $q(\mu_\sigma^2) = q(\mu)q(\sigma^2)$. Now we derive the procedure for MFVB to approximate the posterior[4]

$$p(\mu, \sigma^2|y) \propto p(\mu)p(\sigma^2)p(y|\mu, \sigma^2). \tag{3.2.21}$$

[3]Note that for every iteration, $LB(q)$ increases.

[4]Notice in the MFVB algorithm above, we were using θ_1, θ_2 as our parameters. Here we can view μ and σ^2 as θ_1, θ_2 respectively.

From Eqn. 3.2.18, the optimal VB posterior for σ^2 is

$$q(\sigma^2) \propto \exp(\mathbb{E}_{-\sigma^2}[\log p(y, \mu, \sigma^2])) \tag{3.2.22}$$

$$= \exp(\mathbb{E}_{(\mu}[\log p(y, \mu, \sigma^2])) \tag{3.2.23}$$

$$\propto \exp(\mathbb{E}_{q(\mu)})(\log p(\sigma^2) + \log p(y|\mu, \sigma^2)) \tag{3.2.24}$$

$$\propto \exp(-(\alpha_0 + n/2 + 1)\log\sigma^2 - (\beta_0 + \mathbb{E}_{q(\mu)}\left[\sum(y_i - \mu)^2\right])/(2\sigma^2)) \tag{3.2.25}$$

Then we can see that $q(\sigma^2)$ is inverse-Gamma with parameters

$$\alpha_q = \alpha_0 + \frac{n}{2} \quad \text{and } \beta_q = \beta_0 + \frac{1}{2}\mathbb{E}_{q(\mu)}\left[\sum(y_i - \mu)^2\right]. \tag{3.2.26}$$

After obtaining $q(\mu)$, we can compute the expectation $\mathbb{E}_{q(\mu)}$:

$$q(\mu) \propto \exp(\mathbb{E}_{q(\sigma^2)}[\log p(y, \mu, \sigma^2)]) \tag{3.2.27}$$

$$\propto \exp(\mathbb{E}_{q(\sigma^2)}[\log p(\mu) + \log p(y|\mu, \sigma^2)]) \tag{3.2.28}$$

$$\propto \exp(-\frac{1}{2\sigma_0^2}(\mu^2 - 2\mu_0\mu) - \frac{n}{2}\mathbb{E}_{q(\sigma^2)}\frac{1}{\sigma^2}(-2\bar{y}\mu + \mu^2)) \tag{3.2.29}$$

$$\propto \exp\Big(-\frac{1}{2}\underbrace{(\underset{0}{\frac{1}{2}} + n\mathbb{E}_{q(^2)}[\frac{1}{2}])}_{A}\mu^2 + \mu\underbrace{(\underset{0}{\frac{\mu_0}{2}} + n\bar{y}\mathbb{E}_{q(^2)}[\frac{1}{2}])}_{B}\Big) \tag{3.2.30}$$

$$= \exp\Big(-\frac{1}{2}A\mu^2 + B\mu\Big) \tag{3.2.31}$$

$$\propto \exp\Big(-\frac{1}{2}\frac{(\mu - B/A)^2}{1/A}\Big). \tag{3.2.32}$$

Therefore if $q(\mu)$ is Gaussian with mean μ_q and variance σ_q^2, then

$$\mu_q = \frac{\frac{\mu_0}{\sigma_0^2} + n\bar{y}\mathbb{E}_{q(\sigma^2)}[\frac{1}{\sigma^2}]}{\frac{1}{\sigma_0^2} + n\mathbb{E}_{q(\sigma^2)}[\frac{1}{\sigma^2}]} \qquad \text{and} \qquad \sigma_q^2 = \Big(\frac{1}{\sigma_0^2} + n\mathbb{E}_{q(\sigma^2)}[\frac{1}{\sigma^2}]\Big)^{-1}. \tag{3.2.33}$$

Now we have obtained the distributions $q(\mu)$ and $q(\sigma^2)$, so we are able to compute the expectations in 3.2.26:

$$\begin{aligned}
\beta_q &= \beta_0 + \frac{1}{2}\mathbb{E}_{q(\mu)}\big[\sum (y_i - \mu)^2\big] && (3.2.34)\\
&= \beta_0 + \frac{1}{2}\Big(\sum y_i^2 - 2n\bar{y}\mathbb{E}_{q(\mu)}[\mu] + n\mathbb{E}_{q(\mu)}[\mu^2]\Big) && (3.2.35)\\
&= \beta_0 + \frac{1}{2}\sum y_i^2 - n\bar{y}\mu_q + \frac{n}{2}(\mu_q^2 + {}^2_q). && (3.2.36)
\end{aligned}$$

And since $q(\sigma^2) \sim$ Inverse-Gamma(α_q, β_q) and $\mathbb{E}(\frac{1}{\sigma^2}) = \alpha_q/\beta_q$,

$$\mu_q = \Big(\frac{\mu_0}{\sigma_0^2} + n\bar{y}\frac{\alpha_q}{\beta_q}\Big)/\Big(\frac{1}{\sigma_0^2} + n\frac{\alpha_q}{\beta_q}\Big) \qquad \text{and} \qquad \sigma_q^2 = \Big(\frac{1}{\sigma_0^2} + n\frac{\alpha_q}{\beta_q}\Big)^{-1}. \tag{3.2.37}$$

Now we can move on to our iterative MFVB algorithm.

1. Initialize μ_q and σ_q^2

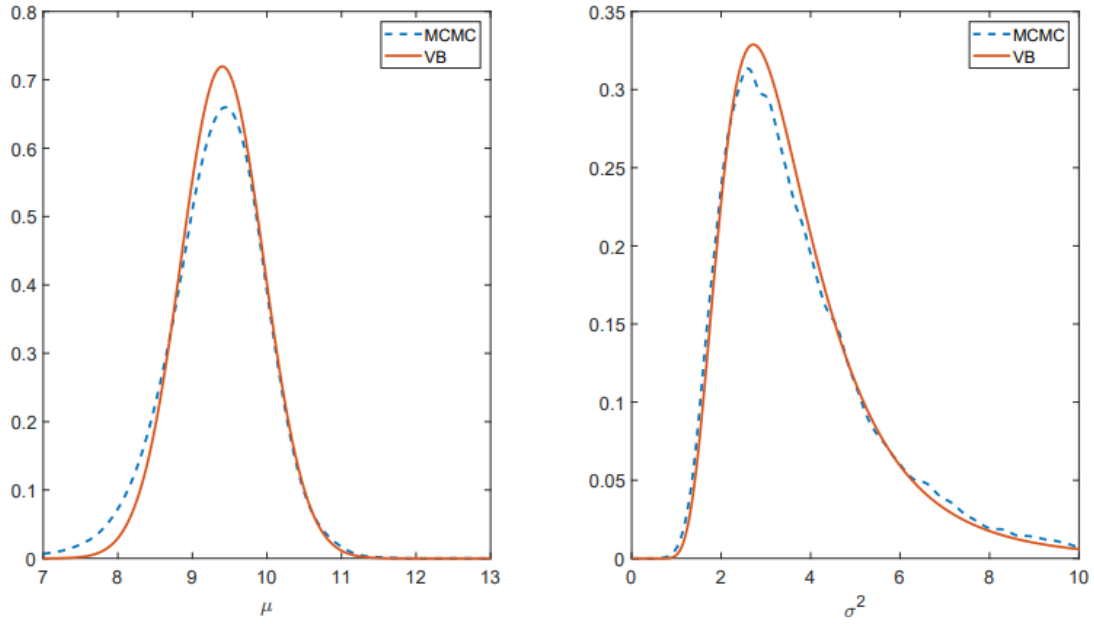

Figure 3.1: Comparison of Posterior Density for μ and σ^2 Estimated by MFVB and MCMC using Gibbs Sampling [6].

2. Recursively update the parameters until convergence:

$$\alpha_q \leftarrow \alpha_0 + \frac{n}{2}, \tag{3.2.38}$$

$$\beta_q \leftarrow \beta_0 + \frac{1}{2}\sum y_i^2 - n\bar{y}\mu_q + \frac{n}{2}(\mu_q^2 + \sigma_q^2), \tag{3.2.39}$$

$$\mu_q \leftarrow \Big(\frac{\mu_0}{\sigma_0^2} + n\bar{y}\frac{\alpha_q}{\beta_q}\Big)/\Big(\frac{1}{\sigma_0^2} + n\frac{\alpha_q}{\beta_q}\Big), \tag{3.2.40}$$

$$\sigma_q^2 \leftarrow \Big(\frac{1}{\sigma_0^2} + n\frac{\alpha_q}{\beta_q}\Big)^{-1}, \tag{3.2.41}$$

Notice we have mentioned about setting a threshold. Here we terminate the algorithm when the L2 norm of vector $(\alpha_q, \beta_q, \mu_q, \sigma_q^2)^T$ is smaller than, say, $\epsilon = 10^{-6}$.

Figure **??** shows the comparison of posterior densities for μ and σ^2 estimated by MFVB and MCMC using Gibbs sampling.

Chapter 4

Conclusion

Throughout the term, I mainly investigated GANs and Variational Bayes. The main reason is that combining GAN and Variational Autoencoders (VAE) can lead to very powerful models with better generalization than vanilla GAN model (i.e. the GAN model mentioned in chapter 2, without any modifications). Up to now, Generative modelling is one of the most trending research topic in the area of deep learning applications and researches. Currently, we are building a GAN model that augments more features to the data through adding perturbation drawn from a Gaussian distribution (inspired by [1]).

The course STA492 really provided me an opportunity to do in-depth research on topics that I am interested in. It was such a wonderful time, and I hope to have opportunities to work with everyone from the seminar in the future.

Bibliography

[1] Clément Chadebec and Stéphanie Allassonnière. "Data Augmentation with Variational Autoencoders and Manifold Sampling". In: *Deep Generative Models, and Data Augmentation, Labelling, and Imperfections*. Springer, 2021, pp. 184–192.

[2] Chelsea Finn, Ian Goodfellow, and Sergey Levine. "Unsupervised learning for physical interaction through video prediction". In: *Advances in neural information processing systems* 29 (2016).

[3] Ian Goodfellow. "Nips 2016 tutorial: Generative adversarial networks". In: *arXiv preprint arXiv:1701.00160* (2016).

[4] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[5] William Lotter, Gabriel Kreiman, and David Cox. "Unsupervised learning of visual structure using predictive generative networks". In: *arXiv preprint arXiv:1511.06380* (2015).

[6] Václav Šmıdl and Anthony Quinn. *The variational Bayes method in signal processing*. Springer Science & Business Media, 2006.