Auto-Encoding Variational Bayes: a Review

Chris Mattioli
Tufts University

Abstract

Auto-Encoding Variational Bayes by Kingma and Welling (2014) is concerned with variational inference in complicated continuous latent variable models. Typically, the likelihood function involved depends on some complicated function of the latent variables $\boldsymbol{z}$. In such cases, finding the form of the posterior is not only intractable, but hard to estimate via mean field approaches. Kingma and Welling present a estimator for the variational lower bound as well as a simple algorithm for optimization. This work will explore the theoretical motivation of such an estimator as well as apply it to two the MNIST and Frey Face datasets.

1. Introduction

1.1. Continuous Latent Variable Models

One of the major aspects of this course was the discussion of generative models and latent variable models. The premise is that you have some observed data, $\boldsymbol{X}$, a column vector with N components, that you believe to be generated by or described by some latent information, $\boldsymbol{z}$ (sometimes called hidden information or hidden variables). Typically, our treatment of $\boldsymbol{z}$ was that of a discrete random variable; however, this review is focused solely on continuous $\boldsymbol{z}$. This is because Kingma and Welling are solely focused on continuous $\boldsymbol{z}$ in their paper. When it came to applying these latent variable models, we oftentimes were interested in finding or identifying the posterior distribution.

$$P(\boldsymbol{z}|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|\boldsymbol{z})P(\boldsymbol{z})}{\int_{\boldsymbol{z}} P(\boldsymbol{X}|\boldsymbol{z})P(\boldsymbol{z})d\boldsymbol{z}}$$

Bayesian inference and model selection hinge upon knowing the form of this posterior.

1.2. "Easy" Posteriors

Calculating this posterior distribution, $P(\boldsymbol{z}|\boldsymbol{X})$, can be relatively easy in many circumstances. Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$ and let $\boldsymbol{X}$ be normally distributed such that it depends on $\boldsymbol{z}$ through some function f. In other words,

$$\boldsymbol{X} \sim \mathcal{N}(f(\boldsymbol{z}), \beta^{-1}\boldsymbol{I})$$

Consider the following possibilities of f:

- $f(\boldsymbol{z}) = \boldsymbol{z}$
- $f(\boldsymbol{z}) = \boldsymbol{W}\boldsymbol{z}$

In the first case, the posterior can be written as:

$$P(\boldsymbol{z}|\boldsymbol{X}) \propto \mathcal{N}(\boldsymbol{z}, \beta^{-1}\boldsymbol{I})\mathcal{N}(\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$$

Via completing the square, this posterior is a normal distribution with mean $\frac{\beta}{\alpha+\beta}\boldsymbol{X}$ and variance $\frac{1}{\alpha+\beta}$. In the second case, the posterior can be written as:

$$P(\boldsymbol{z}|\boldsymbol{X}) \propto \mathcal{N}(\boldsymbol{W}\boldsymbol{z}, \beta^{-1}\boldsymbol{I})\mathcal{N}(\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$$

which is also a normal distribution but with mean $(\alpha + \beta\boldsymbol{W}^T\boldsymbol{W})^{-1}(\beta\boldsymbol{W}^T\boldsymbol{X})$ and variance $(\alpha + \beta\boldsymbol{W}^T\boldsymbol{W})^{-1}$. This is similar to our linear regression example from the beginning of the course.

1.3. "Hard" Posteriors

Calculating the form of the posterior distribution can also be problematic. Consider the following possibilities of f:

- $f(\boldsymbol{z}) = \sigma(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b})$
- $f(\boldsymbol{z}) = MLP(\boldsymbol{z})$

where $\sigma(\cdot)$ is the sigmoid function applied element-wise, and MLP is a multi-layer perceptron.

In the first case the posterior can be written as:

$$P(\boldsymbol{z}|\boldsymbol{X}) \propto \mathcal{N}(\sigma(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}), \beta^{-1}\boldsymbol{I})\mathcal{N}(\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$$

Thinking about completing the square for $\boldsymbol{z}$ is problematic since there are now non-linear, non-quadratic terms involving $\boldsymbol{z}$. Similarly, for the second case, the MLP represents a composition of non-polynomial functions.

Another example of a "hard" posterior taken directly from class is the example of Bayesian logistic regression. Assume that $f(\boldsymbol{z}) = \boldsymbol{z}$ and $\boldsymbol{X} \sim Ber(f(\boldsymbol{z}))$.

$$P(\boldsymbol{z}|\boldsymbol{X}) \propto \mathcal{N}(\boldsymbol{0}, \alpha^{-1}\boldsymbol{I}) \prod_i z_i^{x_i}(1 - z_i)^{1-x_i}$$

In this example we were able to find point estimates, but ultimately had to approximate the posterior because of the non-conjugate form.

2. Variational Inference

2.1. Objective

The goal of variational inference is to find a distribution, $Q(\boldsymbol{z})$, that approximates the true posterior, $P(\boldsymbol{z}|\boldsymbol{X})$. Its study is useful because of the "hard" cases described above. Being able to find a approximate posterior is ideal since it can be used for model selection and prediction. One way of measuring the similarity between distributions is the KL-Divergence defined as:

$$KL(Q(\boldsymbol{z})||P(\boldsymbol{z}|\boldsymbol{X})) = \int_{\boldsymbol{z}} Q(\boldsymbol{z}) \log \frac{Q(\boldsymbol{z})}{P(\boldsymbol{z}|\boldsymbol{X})} d\boldsymbol{z}$$

One if its most important properties of this definition is that it is always non-negative:

$$KL(Q||P) \geq 0, \quad \forall Q, P$$

In practice, using the KL divergence to find an approximate posterior takes some manipulation of the definition:

$$\log P(\boldsymbol{X}) = KL(Q||P(\boldsymbol{z}|\boldsymbol{X})) - KL(Q||P(\boldsymbol{X}, \boldsymbol{z}))$$

The final term in the difference is so important to variational inference it is given its own name, the variational lower bound:

$$L(Q) = -KL(Q||P(\boldsymbol{X}, \boldsymbol{z}))$$

Kingma and Welling prefer to work with a different, but equivalent, expression of $L(Q)$ which I will adopt in the remainder of the review:

$$L(Q) = \mathbb{E}_{Q(\boldsymbol{z})}[\log P(\boldsymbol{X}|\boldsymbol{z})] - KL(Q(\boldsymbol{z})||P(\boldsymbol{z})) \quad (1)$$

Since the log marginal probability, $\log P(\boldsymbol{X})$, remains constant with respect to Q, maximizing $L(Q)$ will maximize $\log P(\boldsymbol{X})$ and minimize the divergence between Q and P. The objective of any variational technique then becomes focused on maximizing $L(Q)$.

2.2. Mean Field Approximation

One such technique covered in the lectures is the mean field approximation technique. In mean field approximation, the approximating posterior is assumed to factorize over the latent variables so that:

$$Q(\boldsymbol{z}) = \prod_i Q(\boldsymbol{z}_i)$$

Given this formulation, the optimal choice of $\boldsymbol{z}_i$ is given by the following:

$$\log Q(\boldsymbol{z}_i) = \mathbb{E}_{i \neq j}[\log P(\boldsymbol{X}, \boldsymbol{z})] + const$$

Now consider applying mean field to the first case in section 1.3:

$$\boldsymbol{X}|\boldsymbol{z} \sim \mathcal{N}(\sigma(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}), \beta^{-1}\boldsymbol{I}),$$
$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$$

where α and β are assumed to be constant, and for simplicity assume that $Q(\boldsymbol{z}) = Q(\boldsymbol{z}_1)Q(\boldsymbol{z}_2)$. The optimal solution for $\log Q(\boldsymbol{z}_1)$ is given by:

$$\log Q(\boldsymbol{z}_1) = \mathbb{E}_{Q(\boldsymbol{z}_2)}[\log P(\boldsymbol{X}, \boldsymbol{z})] + const$$

However, evaluating this expectation will require, among other calculations, the evaluation of:

$$\mathbb{E}_{Q(\boldsymbol{z}_2)}[\sigma(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b})^T \sigma(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b})]$$

which does not have a nice closed form. So the "hard" posterior is also difficult to approximate under this variational method.

3. Kingma and Welling's Approach

3.1. Preliminaries

Kingma and Welling are interested in applying a gradient update regime, such as gradient ascent, to $L(Q)$ directly. In order to do that, the gradient of $L(Q)$ has to be found. But the gradient with respect to what? To up this point, the parameters of Q and P have been left out for notational clarity, but both Q and P are assumed to be parameterized by a parameter set $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ respectfully such that $Q(\boldsymbol{z})$ is really $Q(\boldsymbol{z}; \boldsymbol{\phi})$ and $P(\boldsymbol{X}|\boldsymbol{z})$ is really $P(\boldsymbol{X}|\boldsymbol{z}; \boldsymbol{\theta})$. Therefore the gradient sought is with respect to both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$.

To make this concrete we will consider the example given in section 2.2:

$$\boldsymbol{X}|\boldsymbol{z} \sim \mathcal{N}(\sigma(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}), \beta^{-1}\boldsymbol{I}),$$
$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$$

where α and β are assumed to be constant.

Now we need to make an assumption of the form of the approximate posterior, Q, which is not something required in mean field approximation. In this example, we'll assume:

$$Q(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$$

where each z_i is independent and has variance σ_i^2. Here $\boldsymbol{\sigma}^2$ is a diagonal $D \times D$ matrix. This implies that the variational parameters are $\boldsymbol{\phi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$ and the model parameters are $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$.

3.2. Reformulation of $L(Q)$

The objective given by (1) has two parts which will be considered one at a time. First consider the final term:

$$-KL(Q(\boldsymbol{z})||P(\boldsymbol{z}))$$

Kingma and Welling give an analytic solution to this (though their's was without the α):

$$-KL(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)||\mathcal{N}(\mathbf{0}, \alpha^{-1}\boldsymbol{I}))$$
$$= \frac{1}{2}\sum_i^D [1 + \log \sigma_i^2] - \frac{\alpha}{2}\sum_i^D [\mu_i^2 + \sigma_i^2] + \frac{D}{2}\log\alpha$$
$$= \frac{D}{2} + \frac{1}{2}\log\det\boldsymbol{\sigma}^2 - \frac{\alpha}{2}\boldsymbol{\mu}^T\boldsymbol{\mu} - \frac{\alpha}{2}Tr(\boldsymbol{\sigma}^2) + \frac{D}{2}\log\alpha$$

Now consider the first term in (1):

$$\mathbb{E}_{Q(\boldsymbol{z})}[\log P(\boldsymbol{X}|\boldsymbol{z})]$$

Just like in the mean field approximation, calculating this involves calculating: $\mathbb{E}_{Q(\boldsymbol{z})}[\sigma(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b})^T\sigma(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b})]$, which is problematic. Kingma and Welling's solution to this is to use a Monte Carlo approximation:

$$\mathbb{E}_{Q(\boldsymbol{z})}[\log P(\boldsymbol{X}|\boldsymbol{z})] \approx \frac{1}{M}\sum_m \log P(\boldsymbol{X}|\boldsymbol{z}_m)$$

where each $\boldsymbol{z}_\ell$ is sampled from $Q(\boldsymbol{z})$. This estimation changes the original objective slightly. The new objective to maximize is given by the following:

$$\tilde{L}(Q) = \frac{1}{M}\sum_m \log P(\boldsymbol{X}|\boldsymbol{z}_m) - KL(Q(\boldsymbol{z})||P(\boldsymbol{z})) \quad (2)$$

3.3. Stochastic Gradient Variational Estimator

Since we are ultimately interested in taking gradients with respect to $\boldsymbol{\mu}$, $\boldsymbol{\sigma}^2$, $\boldsymbol{W}$, and $\boldsymbol{b}$, all of the terms including these variables have to be exposed in the objective. The second term, the KL term, does that exactly for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$. However, the first term, the expectation approximation, requires a sampling of $\boldsymbol{z}$ from $Q(\boldsymbol{z})$ which depends on $\boldsymbol{\phi}$. This sampling causes the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ to become implicit in the objective, which is not conducive for gradient ascent optimization.

The solution to this problem is termed by Kingma and Welling as the "reparameterization trick". The trick enables the random variable $\boldsymbol{z}$ to be written as a deterministic function of an uncorrelated random variable. This allows the variational parameters to be present in the objective despite using the Monte Carlo estimate. In the example set up here, the reparameterization for $\boldsymbol{z}$ looks like:

$$\boldsymbol{z}_m = g(\boldsymbol{\epsilon}_m) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_m$$

where $\boldsymbol{\epsilon}_m \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\odot$ refers to element-wise multiplication. Using this deterministic function, g, the objective given by (2) can be rewritten:

$$\tilde{L}(Q, \boldsymbol{\epsilon}) = \frac{1}{M}\sum_m \log P(\boldsymbol{X}|g(\boldsymbol{\epsilon}_m)) - KL(Q(\boldsymbol{z})||P(\boldsymbol{z})) \quad (3)$$

This objective is termed by Kingma and Welling as the Stochastic Gradient Variational Bayes Estimator (SGVB Estimator).

3.4. AEVB Algorithm

Using (3), the variational lower bound is maximized using gradient ascent or another stochastic gradient optimization method (e.g., adagrad). The algorithm, termed the Auto-Encoding Variational Bayes (AEVB) Algorithm, is given by the following:

1. Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
2. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta\nabla_{\boldsymbol{\theta}}\tilde{L}(Q, \boldsymbol{\epsilon})$
3. $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \eta\nabla_{\boldsymbol{\phi}}\tilde{L}(Q, \boldsymbol{\epsilon})$
4. Repeat 1-3 until convergence of $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$

where η is a learning rate parameter.

3.5. Sample Gradient

For the example laid out in section 3.1, let $\boldsymbol{W}_{i,*}$ represent a row vector corresponding to the ith row of $\boldsymbol{W}$. The expression $\boldsymbol{W}_{i,*}\boldsymbol{z}$ is an inner product since $\boldsymbol{z}$ is a column vector of the same dimensionality. For simplicity, assume $M = 1$. The gradients of $\tilde{L}(Q)$ are as follows:

$$\frac{\partial\tilde{L}(Q)}{\partial W_{ij}} = \beta x_i\sigma(\boldsymbol{W}_{i,*}\boldsymbol{z}+b_i)(1-\sigma(\boldsymbol{W}_{i,*}\boldsymbol{z}+b_i))(\mu_j+\sigma_j\epsilon_j)$$
$$-\beta\sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i)^2(1 - \sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i))(\mu_j + \sigma_j\epsilon_j),$$

$$\frac{\partial\tilde{L}(Q)}{\partial b_i} = \beta x_i\sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i)(1 - \sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i))$$
$$-\beta\sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i)^2(1 - \sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i)),$$

$$\frac{\partial\tilde{L}(Q)}{\partial\mu_j} = \beta\sum_i x_i\sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i)(1 - \sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i))W_{ij}$$
$$-\beta\sum_i \sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i)^2(1 - \sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i))W_{ij}$$
$$-\alpha\mu_j,$$

$$\frac{\partial\tilde{L}(Q)}{\partial\sigma_j^2} = \frac{\beta}{2}\sum_i x_i\sigma(\boldsymbol{W}_{i,*}\boldsymbol{z}+b_i)(1-\sigma(\boldsymbol{W}_{i,*}\boldsymbol{z}+b_i))\frac{W_{ij}\epsilon_j}{\sigma_j^2}$$
$$-\frac{\beta}{2}\sum_i \sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i)^2(1 - \sigma(\boldsymbol{W}_{i,*}\boldsymbol{z} + b_i))\frac{W_{ij}\epsilon_j}{\sigma_j^2}$$
$$+\frac{1}{2}\left[\prod_i \sigma_i^2\right]^{-1}\prod_{i\neq j}\sigma_i^2 - \frac{\alpha}{2}$$

3.6. Reparameterization Examples

Assume the latent variable $\boldsymbol{z}$ is not normally distributed, but is distributed according to a different continuous probability distribution. The reparameterization trick will have to be adjusted to this new distribution, i.e., a deterministic function g of the parameters of the distribution and a random variable uncorrelated to $\boldsymbol{z}$ will need to be constructed.

Take, for example, $z \sim Exp(\lambda)$. The expression $g(\epsilon) = \mu + \sigma\epsilon$ that was used above no longer makes any sense, so a new g must be constructed to include λ in the expression of $\tilde{L}(Q)$. To do that, note that the exponential distribution has an inverse CDF that can be written in closed form:

$$F_z(\epsilon)^{-1} = -\frac{1}{\lambda}\log(-\epsilon + 1), \quad \epsilon \in [0, 1)$$

Inverse transform sampling can now be applied to create the function g. Let $\epsilon \sim Uniform(0, 1)$,

$$z = g(\epsilon) = -\frac{1}{\lambda}\log(-\epsilon + 1)$$

Now take the example where $z \sim Gamma(r, \theta)$. Unfortunately, the gamma distribution does not have an inverse tractable CDF, so the above method cannot be applied. However, the exponential distribution is a special case of the gamma distribution when $r = 1$ and $\theta = 1$. Therefore $z = g(\epsilon) = -\frac{1}{\lambda}\log(-\epsilon + 1)$ is distributed as gamma with $r = 1$ and $\theta = 1$. The gamma distribution has an important property whereby a sum of independent gamma random variables with the same value of θ is also gamma distributed. In other words, let $z_i \sim Gamma(r_i, \theta)$ then

$$\sum_i z_i \sim Gamma\left(\sum_i r_i, \theta\right)$$

This implies that for any $R \geq 1$ and $R \in \mathbb{Z}$,

$$z = \sum_i^R -\frac{1}{\lambda}\log(-\epsilon_i + 1) \sim Gamma\left(R, 1\right)$$

where $\epsilon_i \sim Uniform(0, 1)$. In order to change θ in this case, the gamma distribution also has the following property:

$$cz \sim Gamma(r, c\theta), \quad c > 0$$

so if $z \sim Gamma(r, \theta)$ then

$$z = g(\boldsymbol{\epsilon}) = -\frac{\theta}{\lambda}\sum_{i=1}^{r}\log(-\epsilon_i + 1)$$

Kingma and Welling offer several suggestions on how to construct g, but not all distributions of $\boldsymbol{z}$ will offer a closed form. Take for example the gamma distribution when $0 < r < 1$. This requires numerical techniques to be able to sample from.

4. Software Package

Software was developed in Python 2.7 for training these variational models. It makes extensive use of the popular deep learning package, Keras, which is publicly available under the MIT license. It is important to note that the optimization schemes in Keras are only capable of minimization, so the $\tilde{L}(Q)$ will appear negated in the code.

The software that was developed is robust to any sort of MLP architecture. In the example given above, the mean parameters of the data were given by a sigmoid of a linear transformation of $\boldsymbol{z}$. However, more complicated MLP's can be encoded such as multi-sigmoid transformations, hyperbolic tangent transformations, etc. The interface is fully customizable through a .json parameter file, an example of which is attached.

The package can handle input data that is normally distributed, Bernoulli distributed, or Poisson distributed, each of which is customized through this parameter file. This package can also handle normal or exponential $\boldsymbol{z}$. Where the prior, $P(\boldsymbol{z})$, is standard normal for the former and exponential ($\lambda = 1$) for the latter. The normal $\boldsymbol{z}$ can be paired with the normal, Bernoulli, and Poisson input. The exponential $\boldsymbol{z}$ can only be paired with the Bernoulli input.

5. Experiments

5.1. MNIST

Three different variants of the generative model were trained on the MNIST dataset. The MNIST dataset is comprised of 70,000 images of hand written digits along with labels (though the labels are not used during training). Of the 70,000, there were 60,000 examples that were randomly split into a training set (without replacement). The remaining 10,000 comprised the validation set.

Model 1 was a Bernoulli/Gaussian model where the input data were binarized to the values 0 and 1 and assumed independent. In Model 2, the data were assumed to be homoscedastic Gaussian (constant variance) and independent. Model 3 also assumed the input data as independent Gaussian, but set the variance to be a function of the hidden variables as well. See the table below for a summary.

	$P(\boldsymbol{X}\|\boldsymbol{z})$	$P(\boldsymbol{z})$	$Q(\boldsymbol{z})$
Model 1	$\boldsymbol{X} \sim Ber(f(\boldsymbol{z}))$	$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$	$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
Model 2	$\boldsymbol{X} \sim \mathcal{N}(f(\boldsymbol{z}), \beta^{-1}\boldsymbol{I})$	$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$	$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
Model 3	$\boldsymbol{X} \sim \mathcal{N}(f(\boldsymbol{z}), g(\boldsymbol{z}))$	$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$	$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$

Each function of $\boldsymbol{z}$ can be described by the following:

	$f(\boldsymbol{z}), g(\boldsymbol{z})$
Model 1	$f(\boldsymbol{z}) = \sigma(\boldsymbol{W}_2 Relu(\boldsymbol{W}_1\boldsymbol{z} + \boldsymbol{b}_1) + \boldsymbol{b}_2)$
Model 2	$f(\boldsymbol{z}) = Relu(\boldsymbol{W}_2 Relu(\boldsymbol{W}_1\boldsymbol{z} + \boldsymbol{b}_1) + \boldsymbol{b}_2)$
Model 3	$f(\boldsymbol{z}) = Relu(\boldsymbol{W}_2 Relu(\boldsymbol{W}_1\boldsymbol{z} + \boldsymbol{b}_1) + \boldsymbol{b}_2)$
	$g(\boldsymbol{z}) = Relu(\boldsymbol{W}_3 Relu(\boldsymbol{W}_1\boldsymbol{z} + \boldsymbol{b}_1) + \boldsymbol{b}_3)$

where $RELU$ is the rectified linear unit function.

In all cases $\boldsymbol{z}$ was assumed to be only two-dimensional where $\boldsymbol{X}$ had 784 dimensions. $\boldsymbol{W}_1$ was set to a 256×2 matrix and $\boldsymbol{b}_1$ was set to a 256 dimensional vector. $\boldsymbol{W}_2$ and $\boldsymbol{W}_3$ were both 784×256 matrices and $\boldsymbol{b}_2$ and $\boldsymbol{b}_3$ were both 784 dimensional vectors.

5.2. Frey Faces

Model 2 described above was also trained on the Frey Face dataset. The Frey Face dataset is a set of 1900 images of a face taken from different frames in a continuous shot. 1800 of these images were taken as training data and remaining 100 were used as validation data.

The same dimensions apply to $\boldsymbol{z}$ as they did in the MNIST experiments. However, the Frey Face data is 560 dimensional and so the size of the weight matrices $\boldsymbol{W}_2$ and $\boldsymbol{W}_3$ were adjusted accordingly along with the bias vectors $\boldsymbol{b}_2$ and $\boldsymbol{b}_3$. $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$ remained the same size as the previous experiment.

6. Discussion

6.1. MNIST

One the of fun features of these models is that they can be used to encode, decode and reconstruct data. Since this variational technique was utilized in an unsupervised manner, most results will appear qualitative.

Three different models were trained on the MNIST dataset described in section 5.1. Model 1, which assumed a Bernoulli input and a Gaussian latent variable, was trained and then used to encode data, decode data and reconstruct data. Encoding seeks to transform high dimensional data into a low dimensional space in a manner that retains the most information about the original representation. In the case of MNIST, the encoding step transformed 784 pixel images into two dimensions. Figure 1 shows the encoded MNIST validation data in the latent two-dimensional space. The coloration originates from the labels provided with the dataset rather than originating from the model. Note the separation of colors. The color matching the 0 digit is clustered toward the right and the color matching the 1 digit is clustered toward left. This separation makes intuitive sense because the 0 and 1 digit look very different from each other.

Note that there is also overlap between colors. The colors representing 8, 3, and 5 are all overlapping which says that the difference between them is not as clear as the difference between 0 and 1.

The decoding of data with this model transforms a two dimensional Gaussian random variable into a 784 dimensional Bernoulli variable. Figure 2 shows the decoded result for four two dimensional Gaussians selected from different spots in figure 1. As you can see, the decoded data match their respective labels closely. The 7 is the only one that may look different (looks almost like a 9); however, judging from figure 1, 7 and 9 are very close to each other in latent space.

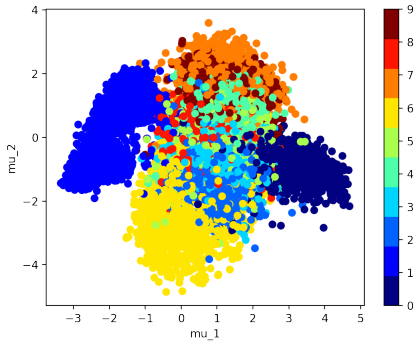

Figure 1: Encoded representation of the MNIST validation set into the latent space $\boldsymbol{z}$ for model 1 (Bernoulli-to-Gaussian). The mean of z_1 is plotted on the x-axis and the mean of z_2 is plotted on the y-axis for each validation example.

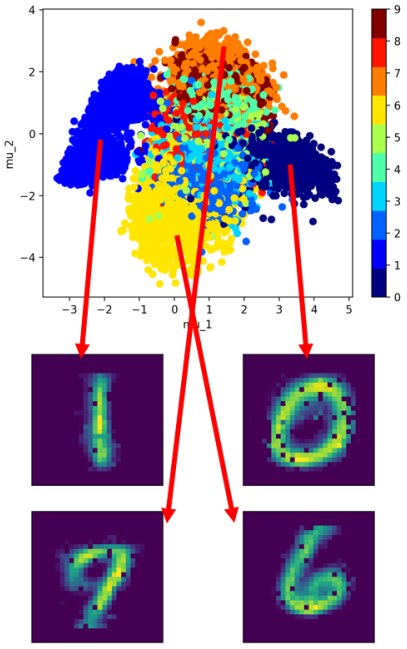

Figure 2: Decoding $\boldsymbol{z}$ into the original 784 dimensional space. Four instances of $\boldsymbol{z}$ are drawn to see if their original location matches with the true label.

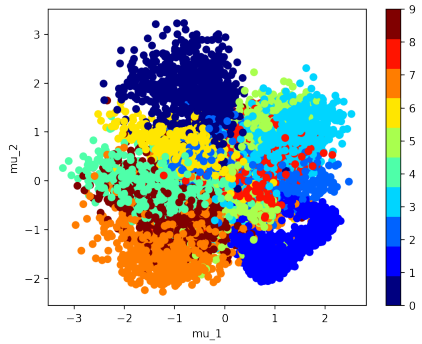

Figure 3: Encoded representation of the MNIST validation set into the latent space $\boldsymbol{z}$ for model 2 (Homoscedastic-Gaussian-to-Gaussian). The mean of z_1 is plotted on the x-axis and the mean of z_2 is plotted on the y-axis for each validation example.

Model 2, which assumed a constant variance Gaussian input and a Gaussian latent variable, was also tested in its ability to encode the 784 dimensional information into two dimensions. Figure 3 shows a plot of the two mean values for $\boldsymbol{z}$, the latent variable. This result is very similar to the Bernoulli-Gaussian model, model 1. There is a clear separation between 0 and 1; however, there is still overlap between some digits, e.g., 8's, 5's, and 3's.

Figure 4 is the result of decoding data using model 2. As you can see, selecting digits close to their centers yields an almost perfect decoding of the digit into 784 dimensional space. This is a very similar result to the decoded output of model 1. The 7 digit is close to 9 in this latent space as well, and that can be seen in its decoded image.

Another test of this was in regards to reconstruction, that is, given an input $\boldsymbol{X}_i$, encode it and then decode it. How close is the output, $\tilde{\boldsymbol{X}}_{\boldsymbol{i}}$, that results in the decoding? Figure 5 is a qualitative example where the left-hand image is the input example $\boldsymbol{X}_i$ and the right-hand is the reconstruction, $\tilde{\boldsymbol{X}}_{\boldsymbol{i}}$. The result still appears to be a 3, however it looks smoothed over all.

Taking the absolute difference between $\boldsymbol{X}_i$ and $\tilde{\boldsymbol{X}}_{\boldsymbol{i}}$ yields a reconstruction error. Figure 6 is a comparison between the reconstruction error of Model 1 and Model 2 over the validation set. Interestingly, the results are very close to each other. Model 1 has more of a pronounced change compared to model 2's smoother transition.

The final model experimented with was model 3. This model assumed the input was Gaussian only instead of having constant variance, the variance was a function of the hidden variables. Figure 7 shows the encoding into the latent space. Notice that there is no clear separation between the different colors (digits).

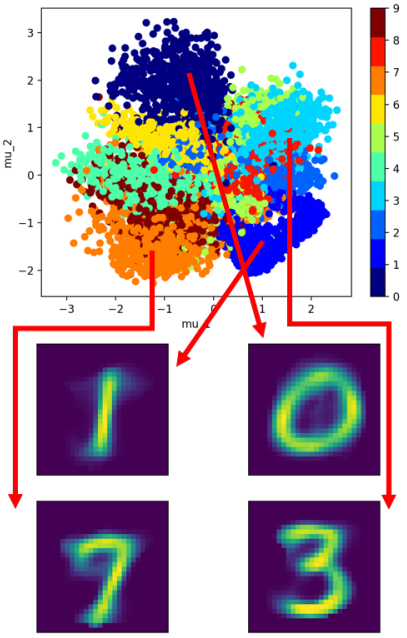

Figure 4: Decoding $\boldsymbol{z}$ into its original representation.

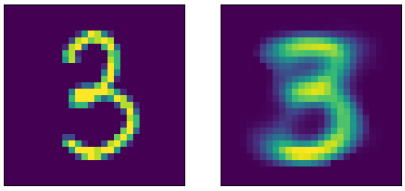

Figure 5: Reconstruction of the digit 3. The original (left) is encoded into the two dimensional latent space $\boldsymbol{z}$ and then decoded back into the original space (right). Encoded/decoded using Model 2.

This model was much more difficult to train than the homoscedastic version, and the results do not show any sort of pattern. It is possible that only two dimensions is not enough to describe this data, but its also possible that trying to train model parameters for both the mean and variance causes issues.

As a point of comparison, a multinomial mixture model was trained via the EM-Algorithm in an attempt to model

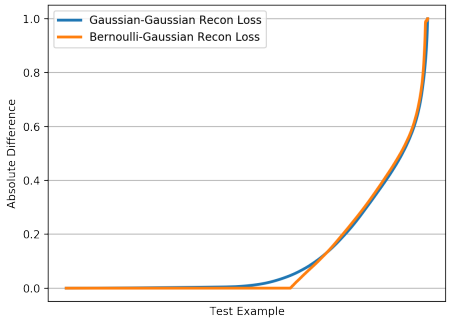

Figure 6: Comparison of the absolute reconstruction error from model 1 (Bernoulli-to-Gaussian, orange) and from model 2 (Homoscedastic-Gaussian-to-Gaussian, blue).

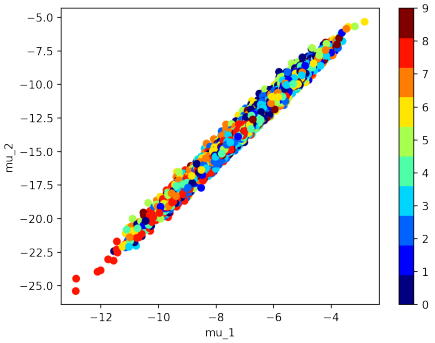

Figure 7: Encoded representation of the MNIST validation set into the latent space $\boldsymbol{z}$ for model 2 (Heteroscedastic-Gaussian-to-Gaussian) The mean of z_1 is plotted on the x-axis and the mean of z_2 is plotted on the y-axis for each validation example.

MNIST data. Ten separate models were selected corresponding to the ten digits in the data set. Figure 8 plots the $P(\boldsymbol{X}|\boldsymbol{z}_k)$ for each model. The probability of being in any one k is about even which makes sense because the digits in the training set are pretty evenly distributed. Notice that this model repeats some representations such as 9. This is most likely because the 9 digit looks very similar to the 4 digit. The 7, 6, 0, and 1 digit are clearly represented, but the 5, 3, and 8 appear to overlap. Both the continuous latent variable model and this discrete latent variable model have the same flaws: images that are similar appear closer together in the latent representation.

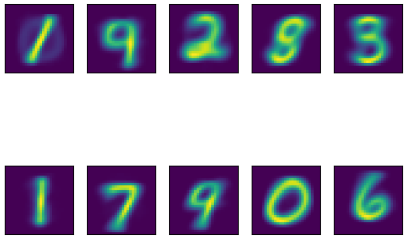

Figure 8: Multinomial mixture model applied to the MNIST dataset. Each image, k, corresponds to $P(\boldsymbol{X}|\boldsymbol{z}_k)$.

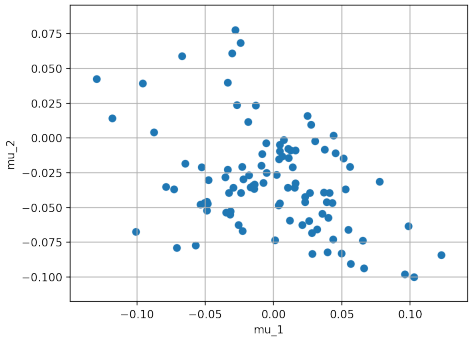

Figure 9: Encoded representation of the Frey Face validation set into the latent space $\boldsymbol{z}$ for a Homoscedastic-Gaussian-to-Gaussian model. The mean of z_1 is plotted on the x-axis and the mean of z_2 is plotted on the y-axis for each validation example.

6.2. Frey Faces

There was only one model trained on the Frey Face dataset, a homoscedastic Gaussian to Gaussian model. This model was without any sort of labels, so an analysis akin to the MNIST data is not possible; however, figure 9 shows the representation of the validation images in a two dimensional latent space.

A decoding technique was applied similar to the decoding of the MNIST data. However, in order to highlight differences, more extreme values of $\boldsymbol{z}$ were selected. Figure 10 shows the result of four decoded images differenced with the decoded image resulting from $\boldsymbol{z} = (0, 0)$. In other words, the image the represented by $(0, 0)$ in latent space was differenced with four other decoded images: an image corresponding to $(-10, -10)$, an image corresponding to $(10, 10)$, an image corresponding to $(-10, 10)$, and an im-

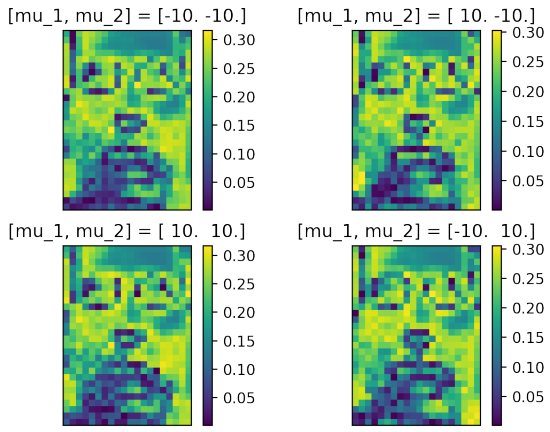

Figure 10: Four decoded images of the Frey Face data, differenced with the image generated by $(0, 0)$.

age corresponding to $(10, -10)$. What is interesting to note here is that for all the images, the highlights of nose, eyes, and mouth had the lowest difference. This makes intuitive sense since they are pretty constant across the training data. The latent space did not choose those features to represent because they are so common. The major differences appear in the cheeks. That is because Frey is moving throughout the scene and the training includes many different angles and light reflections off those facial features.

7. Conclusion

The SGVB estimator presented by Kingma and Welling is a powerful variational tool. It allows for the optimization of the variational lower bound with respect to many different continuous distributions. These distributions can be parameterized via complicated functions of the latent variables. The estimator and technique have proven useful when applied to both the MNIST and Frey Face dataset.

Future work will consistent of adding different distribution options for the models. This involves formulating the reparameterization trick for those distributions as well as the finding a closed form representation of the KL divergence between it and the prior distribution. Though exponential $\boldsymbol{z}$ was implemented in the software package, it was not able to be experimented with. Same goes for the Poisson-Gaussian model. Future work will consider examples involving these distributions.

8. References

1. Kingma, D.,P., Welling, M. *Auto-Encoding Variational Bayes*. The International Conference on Learning Representations (ICLR), Banff, 2014.