Auto-Encoding Variational Bayes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Can we efficiently learn the parameters of directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions? We introduce an unsupervised on-line learning algorithm that efficiently optimizes the variational lower bound on the marginal likelihood and that, under some mild conditions, even works in the intractable case. The algorithm optimizes a probabilistic encoder (also called a recognition network) to approximate the intractable posterior distribution of the latent variables. The crucial element is a reparameterization of the variational bound with an independent noise variable, yielding a stochastic objective function which can be jointly optimized w.r.t. variational and generative parameters using standard gradient-based stochastic optimization methods. Theoretical advantages are reflected in experimental results.

1 Introduction

How to efficiently learn the parameters of directed probabilistic models whose continuous latent variables have intractable posterior distributions? The variational approach to approximate Bayesian inference involves the introduction of an approximate posterior to the intractable posterior, used to maximize the variational lower bound on the marginal likelihood. Unfortunately, the common mean-field approach requires analytical solutions to expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how for continuous latent variables, a reparameterization of the expectation w.r.t. the approximate posterior yields a novel and practical estimator of the variational lower bound that can be differentiated and jointly optimized w.r.t. all parameters, i.e. both the variational parameters and regular parameters, using standard stochastic gradient ascent techniques.

The objective contains, in addition to regularization terms dictated by the variational bound, a noisy data reconstruction term, exposing a novel connection between auto-encoders and stochastic variational inference. In contrast to a typical objective for auto-encoders [BCV13], all parameters updates, including those of the noise distribution, correspond to optimization of the variational lower bound on the marginal likelihood. From the learned generative model it is straightforward to generate samples, without the typical requirement of running Markov chains. The probabilistic encoder can be used for fast approximate inference of latent variables, i.e. for recognition, representation or visualization purposes. Furthermore, the lower bound estimator can be used for unsupervised inference tasks such as denoising and inpainting.

2 Method

The strategy in the following section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform ML or MAP inference on the (global) parameters, and

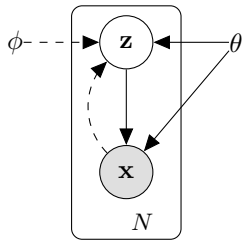


Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. The variational parameters $\boldsymbol{\phi}$ are learned jointly with the generative model parameters $\boldsymbol{\theta}$.

variational inference on the latent variables. It is, for example, straightforward to extend this scenario to the case where we also perform variational inference on the global parameters; that algorithm is put in the appendix, but experiments with that case are left to future work. Note that our method can be applied to online, non-stationary settings, e.g. streaming data, but here we assume a fixed dataset for simplicity.

2.1 Problem scenario

Let us consider some dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ consisting of N i.i.d. samples of some continuous or discrete variable $\mathbf{x}$. We assume that the data are generated by some random process, involving an unobserved continuous random variable $\mathbf{z}$. The process consists of two steps: (1) a value $\mathbf{z}^{(i)}$ is generated from some prior distribution $p_{\boldsymbol{\theta}^*}(\mathbf{z})$; (2) a value $\mathbf{x}^{(i)}$ is generated from some conditional distribution $p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$. We assume that the prior $p_{\boldsymbol{\theta}^*}(\mathbf{z})$ and likelihood $p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$ come from parametric families of distributions $p_{\boldsymbol{\theta}}(\mathbf{z})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, and that their PDFs are differentiable almost everywhere w.r.t. both $\boldsymbol{\theta}$ and $\mathbf{z}$. Unfortunately, a lot of this process is hidden from our view: the true parameters $\boldsymbol{\theta}^*$ as well as the values of the latent variables $\mathbf{z}^{(i)}$ are unknown to us.

Very importantly, we *do not* make the usual simplifying assumptions common in the literature. Conversely, we are here interested in a general algorithm that even works in the case of:

1. *Intractability*: the case where the integral of the marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, d\mathbf{z}$ is intractable (so we cannot evaluate or differentiate the marginal likelihood), where the true posterior density $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})/p_{\boldsymbol{\theta}}(\mathbf{x})$ is intractable (so the EM algorithm cannot be used), and where the required integrals for any reasonable mean-field Variational Bayes are also intractable. These intractabilities are quite common and already appear in case of moderately complicated likelihood functions $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, e.g. a neural network with a nonlinear hidden layer.
2. *A large dataset*: we have so much data that batch optimization is too costly; we would like to make parameter updates using small minibatches or even single datapoints. Sampling-based solutions, e.g. Monte Carlo EM, would in general be too slow, since it involves a typically expensive sampling loop per datapoint.

We are interested in, and propose a solution to, three related problems in the above scenario:

1. Efficient approximate maximum likelihood (ML) or maximum a posteriori (MAP) estimation for the parameters $\boldsymbol{\theta}$. The parameters can be of interest themselves, e.g. if we are analyzing some natural process. They also allow us to mimic the hidden random process and generate artificial data that resembles the real data.
2. Efficient approximate posterior inference of the latent variable $\mathbf{z}$ given an observed value $\mathbf{x}$ for a choice of parameters $\boldsymbol{\theta}$. This is useful for coding or data representation tasks.
3. Efficient approximate marginal inference of the variable $\mathbf{x}$. This allows us to perform all kinds of inference tasks where a prior over $\mathbf{x}$ is required. Common applications in computer vision include image denoising, inpainting and super-resolution.

For the purpose of solving the above problems, let us introduce the parametric variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$: an approximation to the intractable true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Note that in contrast with the approximate posterior in mean-field variational inference, it is not necesarilly factorial and its parameters are not computed from some closed-form expectation. Instead, its parameters $\boldsymbol{\phi}$ are learned jointly with the parameters of the generative model.

From a coding theory perspective, the unobserved variables $\mathbf{z}$ have an interpretation as a latent representation or *code*. In this paper we will therefore also refer to $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as a *(variational) encoder* or *recognition model*, since given a datapoint $\mathbf{x}$ it produces a distribution (e.g. a Gaussian) over the possible values of the code $\mathbf{z}$ from which the datapoint $\mathbf{x}$ could have been generated. In a similar vein we will refer to $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ as a *(generative) decoder*, since given a code $\mathbf{z}$ it produces a distribution over the possible corresponding values of $\mathbf{x}$.

2.2 The variational bound

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$, which can each be rewritten as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{1}$$

The first RHS term is the KL divergence of the approximate from the true posterior, which is non-negative. The second RHS term $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ denotes the variational lower bound on the marginal likelihood of datapoint i:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right) d\mathbf{z} \tag{2}$$

Note that the bound equals the true marginal when the divergence of the approximate from true posterior distribution is zero.

The expectation on the RHS of eq. (2) can obviously be written as a sum of three separate expectations, of which the second and third component can sometimes be analytically solved, e.g. when both $p_{\boldsymbol{\theta}}(\mathbf{x})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ are Gaussian. For generality we will here assume that each of these expectations are intractable.

We would like to optimize the lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ (eq. (2)) using stochastic gradients. Note that following these gradients would either decrease the KL divergence between the approximate and true posterior distributions, or increase the marginal likelihood, or both. A naïve attempt to compute a stochastic gradient would be to draw samples $\{\mathbf{z}^{(l)}\}_{l=1}^{L}$ from $q_{\boldsymbol{\phi}}$ and then differentiate the following Monte Carlo estimate of the lower bound:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{L} \sum_{l=1}^{L} \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(l)}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}^{(l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)}|\mathbf{x}^{(i)}) \right) \quad \text{where} \quad \mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$$

While the above expression is an unbiased estimator of the marginal likelihood (i.e. it will equal the lower bound in the limit $L \to \infty$), differentiating it w.r.t. the parameters $\boldsymbol{\phi}$ will not result in an unbiased gradient: the variational parameters $\boldsymbol{\phi}$ indirectly influence the estimate through the samples $\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, and it is impossible to differentiate through this sampling process. Existing work on stochastic variational bayes provide workarounds [BJP12], but not a solution to this problem.

2.3 Our estimator of the lower bound

Under certain mild conditions outlined in section 2.4 for a chosen approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ we can reparameterize its conditional samples $\widetilde{\mathbf{z}} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as

$$\widetilde{\mathbf{z}} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{3}$$

where we choose a prior $p(\boldsymbol{\epsilon})$ and a function $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ such that the following holds:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) &= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right) d\mathbf{z} \\ &= \int p(\boldsymbol{\epsilon}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right) \Big|_{\mathbf{z}=g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}^{(i)})} d\boldsymbol{\epsilon} \end{aligned} \tag{4}$$

Algorithm 1 Pseudocode for computing a stochastic gradient using our estimator. See section 2.3 for meaning of the functions $f_{\boldsymbol{\theta},\boldsymbol{\phi}}$ and $g_{\boldsymbol{\phi}}$. The minibatch $\mathbf{X}^M = \{\mathbf{x}^{(i)}\}_{i=1}^M$ is a randomly drawn subset of the full dataset $\mathbf{X}$. We use settings $M = 100$ and $L = 1$ in experiments.

Require: $\boldsymbol{\theta}, \boldsymbol{\phi}$ (Current value of parameters)
  $\mathbf{g} \leftarrow 0$
  $\mathbf{X}^M \leftarrow$ Random subset (minibatch) of M datapoints from dataset
  for each $\mathbf{x} \in \mathbf{X}^M$ **do**
    for l is 1 to L **do**
      $\boldsymbol{\epsilon} \leftarrow$ Random sample from $p(\boldsymbol{\epsilon})$
      $\mathbf{g} \leftarrow \mathbf{g} + \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}))$
    end for
  end for
  return $(N/(M \cdot L)) \cdot \mathbf{g}$

For notational conciseness we introduce a shorthand notation $f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z})$ for the sum of three PDFs:

$$f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z}) = \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \tag{5}$$

Using eq. (4), the Monte Carlo estimate of the variational lower bound, given datapoint $\mathbf{x}^{(i)}$, is:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{L} \sum_{l=1}^{L} f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}^{(i)}, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})) \quad \text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{6}$$

The estimator only depends on samples from $p(\boldsymbol{\epsilon})$ which are obviously not influenced by $\boldsymbol{\phi}$, therefore we can use it as an objective function that can be differentiated and jointly optimized w.r.t. both $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Given multiple datapoints from the dataset $\mathbf{X}$, we can easily construct a minibatch-based version of the estimator: $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \simeq \frac{N}{M} \sum_{i=1}^{M} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ where the minibatch $\mathbf{X}^M = \{\mathbf{x}^{(i)}\}_{i=1}^M$ is a randomly drawn subset of the full dataset $\mathbf{X}$. In our experiments we found that the number of samples L per datapoint can be set to 1 as long as the minibatch size M was large enough, e.g. $M = 100$. Derivatives $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}(\boldsymbol{\theta}; \mathbf{X}^M)$ can be taken, and the resulting gradients can be used in conjunction with stochastic optimization methods such as SGD or Adagrad [DHS10]. See algorithm 1 for a basic approach to compute the stochastic gradients.

A connection with auto-encoders becomes clear when looking at the objective function given at eq. (6). The variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$ (the encoder) maps a datapoint $\mathbf{x}^{(i)}$ to a distribution over latent variables $\mathbf{z}$ from which the datapoint could have been generated. The function $g_{\boldsymbol{\phi}}(.)$ is chosen such that it maps a datapoint $\mathbf{x}^{(i)}$ and a random noise vector $\boldsymbol{\epsilon}^{(l)}$ to a sample from the approximate posterior for that datapoint: $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})$ where $\mathbf{z}^{(i,l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$. Subsequently, the sample $\mathbf{z}^{(i,l)}$ is then input to function $f_{\boldsymbol{\theta},\boldsymbol{\phi}}(.)$, which consists of three parts. The first part ($\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$) can be interpreted as the *negative reconstruction error* in neural network parlance. The second and third part can be interpreted as regularization terms that make sure the code activations have high entropy due to the term $\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, while not being too far from the prior due to the term $\log p_{\boldsymbol{\theta}}(\mathbf{z})$.

2.4 The alternative parameterization trick

In order to solve our problem we invoked an alternative method for generating samples from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. The essential parameterization trick is quite simple. Let $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ be some conditional distribution parameterized by $\boldsymbol{\phi}$. It is then often possible to express the random variable $\mathbf{z}$ given $\mathbf{x}$ as a determinstic variable $\mathbf{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$, where $\boldsymbol{\epsilon}$ is an auxiliary variable with independent marginal $p(\boldsymbol{\epsilon})$, and $g_{\boldsymbol{\phi}}(.)$ is some vector-valued function parameterized by $\boldsymbol{\phi}$.

This reparameterization is useful for our case since it can be used to rewrite an expectation w.r.t $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ such that the Monte Carlo estimate of the expectation is differentiable w.r.t. $\boldsymbol{\phi}$. A proof is as follows. Given the deterministic mapping $\mathbf{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ we know that $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \prod_i dz_i = p(\boldsymbol{\epsilon}) \prod_i d\epsilon_i$. Therefore[1], $\int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) f(\mathbf{z})\, d\mathbf{z} = \int p(\boldsymbol{\epsilon}) f(\mathbf{z})\, d\boldsymbol{\epsilon} = \int p(\boldsymbol{\epsilon}) f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}))\, d\boldsymbol{\epsilon}$. It follows

[1] Note that for infinitesimals we use the notational convention $d\mathbf{z} = \prod_i dz_i$

that a differentiable estimator can be constructed: $\int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) f(\mathbf{z})\, d\mathbf{z} \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon}^{(l)}))$ where $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$. In section 2.3 we applied this trick to obtain a differentiable estimator of the variational lower bound.

Take, for example, the univariate Gaussian case: let z be distributed as $p(z|x) = \mathcal{N}(x, \sigma)$. The random variable z is partially explained by x, but there is some uncertainty left indicated by σ. In this case, a deterministic parameterization is $z = x + \sigma\epsilon$, where ϵ is an independent auxiliary variable $\epsilon \sim \mathcal{N}(0, 1)$. In this univariate Gaussian case, $\boldsymbol{\phi} = \{\sigma\}$ and $g_{\boldsymbol{\phi}}(\epsilon, y) = y + \sigma\epsilon$.

When can we do this, i.e., for which $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can we choose such a $g_{\boldsymbol{\phi}}(.)$ and $p(\boldsymbol{\epsilon})$? There are three basic approaches:

1. Tractable inverse CDF. In this case, let $\boldsymbol{\epsilon} \sim \mathcal{U}(\mathbf{0}, \mathbf{I})$, and let $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ be the inverse CDF of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. Examples: Exponential, Cauchy, Logistic, Rayleigh, Pareto, Weibull, Reciprocal, Gompertz, Gumbel and Erlang distributions.
2. Analogous to the Gaussian example, for any "location-scale" family of distributions (with differentiable log-PDF) we can choose the standard distribution (with location $= 0$, scale $= 1$) as the auxiliary variable E, and let $g(.) =$ location $+$ scale $\cdot\, \boldsymbol{\epsilon}$. Examples: Laplace, Elliptical, Student's t, Logistic, Uniform, Triangular and Gaussian distributions.
3. Composition: It is often possible to express variables as functions of component variables with distributions that are reparameterizable using either of the above two approaches. Examples: Log-Normal (exponentiation of normally distributed variable), Gamma (a sum over exponentially distributed variables), Dirichlet (weighted sum of Gamma variates), Beta, Chi-Squared, and F distributions.

When all three approaches fail, good approximations to the inverse CDF exist requiring computations with time complexity comparable to the PDF (see e.g. [Dev86] for some methods).

3 Example

Here we'll give an example generative model and posterior approximation used in experiments.

Let the prior over the latent variables be the centered isotropic Gaussian $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Note that in this case, the prior lacks parameters. Let $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ (the decoder) be a multivariate Bernoulli whose probabilities are computed from $\mathbf{z}$ with a fully-connected neural network with a single hidden layer:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) &= \sum_{i=1}^{D} x_i \log y_i - (1 - x_i) \cdot \log(1 - y_i) \\ \text{where } \mathbf{y} &= f_\sigma(\mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2) \end{aligned} \tag{7}$$

where $f_\sigma(.)$ is the elementwise sigmoid activation function. While there is much freedom in the choice of the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ (encoder / recognition model), we'll for a moment assume a relatively simple case: let's assume that the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ takes on a approximate Gaussian form with an approximately diagonal covariance. In this case, we can let the variational approximate posterior be a multivariate Gaussian with a diagonal covariance structure[2]:

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I}) \tag{8}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are yet unspecified functions of $\mathbf{x}$. We can sample from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ using $\widetilde{\mathbf{z}} = h_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. With $\odot$ we signify an element-wise product. Therefore, given a minibatch $\mathbf{X}^M$ of data, and using the $f_{\boldsymbol{\theta},\boldsymbol{\phi}}(.)$ abbrevation of eq. (5), our estimator of the lower bound is:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{L}\sum_{l=1}^{L} f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)})\big|_{\mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)}} \quad \text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{9}$$

[2]Note that this is just a (simplifying) choice, and not a limitation of our method.

where $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\sigma}^{(i)}$ denote the mean and s.d. of the approximation of the posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$, which we didn't yet specify. Let the mean $\boldsymbol{\mu}^{(i)}$ and variance $\boldsymbol{\sigma}^{(i)}$ of the Gaussian encoding distribution be the following nonlinear function of $\mathbf{x}$, (a neural network):

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$$
$$\text{where } \boldsymbol{\mu} = \mathbf{W}_4\mathbf{h} + \mathbf{b}_4, \quad \text{and} \quad \log \boldsymbol{\sigma}^2 = \mathbf{W}_5\mathbf{h} + \mathbf{b}_5, \quad \text{and} \quad \mathbf{h} = \tanh(\mathbf{W}_3\mathbf{x} + \mathbf{b}_3) \quad (10)$$

Note that the generative (decoding) parameters are $\boldsymbol{\theta} = \{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^2$ and the variational (encoding) parameters are $\boldsymbol{\phi} = \{\mathbf{W}_j, \mathbf{b}_j\}_{j=3}^5$. These definitions for the encoder and decoder can be plugged in eq. 6, and the lower bound can subsequently be differentiated and optimized w.r.t. the parameters.

In this model both $p_{\boldsymbol{\theta}}(\mathbf{z})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ are Gaussian; in this special case, the second and third term of $f_{\boldsymbol{\theta},\boldsymbol{\phi}}$ (eq. (5)) can be solved analytically. This results in an estimator with a lower variance than the generic estimator given in eq. (9). The resulting estimator is:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}) \quad (11)$$

See the appendix for the derivation.

4 Related work

Perhaps the most relevant related method is the Wake-Sleep algorithm [HDFN95]. Like AEVB, the wake-sleep algorithm employs an encoder (called a recognition network) that approximates the true posterior. A well-known drawback of the wake-sleep algorithm is that it lacks a theoretically justified method for learning the parameters of the recognition network: its updates correspond to optimization of the divergence $KL(p||q)$ instead of the divergence $KL(q||p)$ dictated by the lower bound. A theoretical advantage of the wake-sleep algorithm is that it also applies to models with discrete latent variables. Wake-Sleep has the same computational complexity as AEVB per datapoint.

AEVB corresponds to the optimization of a type of auto-encoder, exposing a connection between directed probabilistic models and auto-encoders. A connection between *linear* auto-encoders and a certain class of generative linear-Gaussian models has long been known. In [Row98] it was shown that PCA corresponds to the maximum-likelihood (ML) solution of a special case of the linear-Gaussian model with a prior $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ and a conditional distribution $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{W}\mathbf{z}, \epsilon\mathbf{I})$, specifically the case with infinitesimally small ϵ.

In relevant recent work on autoencoders [VLL+10] it was shown that the training criterion of unregularized autoencoders corresponds to maximization of a lower bound (see the infomax principle [Lin89]) of the mutual information between input X and latent representation Z. Maximizing (w.r.t. parameters) of the mutual information is equivalent to maximizing the conditional entropy, which is lower bounded by the expected loglikelihood of the data under the autoencoding model [VLL+10], i.e. the negative reconstrution error. However, it is well known that this reconstruction criterion is in itself not sufficient for learning useful representations [BCV13]. Regularization techniques have been proposed to make autoencoders learn useful representations, such as denoising, contractive and sparse autoencoder variants [BCV13]. Our objective function contains (hyper-parameter free) regularization terms dictated by the variational bound (see eq. (5)). Related are also encoder-decoder architectures such as the predictive sparse decomposition (PSD) [KRL08], from which we drew some inspiration. Also relevant are the recently introduced Generative Stochastic Networks [BTL13] where noisy auto-encoders learn the transition operator of a Markov chain that samples from the data distribution. In [SL10] a recognition network was employed for efficient learning with Deep Boltzmann Machines. These methods are targeted at either unnormalized models (i.e. undirected models like Boltzmann machines) or limited to sparse coding models, in contrast to our proposed algorithm for learning a general class of directed probabilistic models.

The recently proposed NADE method [GMW13], also learns a directed probabilistic model using an auto-encoding structure, however their method applies to binary latent variables, their objective does not correspond to the variational bound and their encoder doesn't correspond to a variational approximation of the posterior distribution.

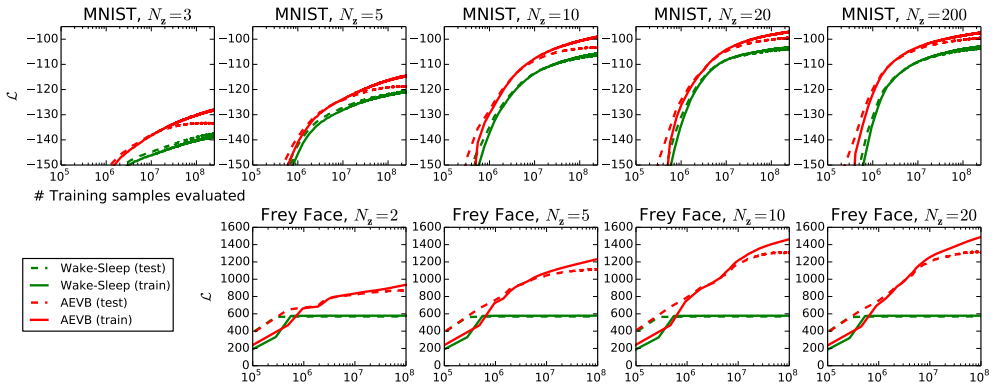


Figure 2: Comparison of our AEVB method to the wake-sleep algorithm, in terms of optimizing the lower bound, for different dimensionality of latent space ($N_{\mathbf{z}}$). Our method converged considerably faster and reached a better solution in all experiments. Interestingly enough, more latent variables does not result in more overfitting, which is explained by the regularizing effect of the lower bound. Vertical axis: the estimated average variational lower bound per datapoint. The estimator variance was small (< 1) and omitted. Horizontal axis: amount of training points evaluated. Computation took around 20-40 minutes per million training samples with a Intel Xeon CPU running at an effective 40 GFLOPS.

5 Experiments

We trained generative models of images from the MNIST and Frey Face datasets[3] and compared learning algorithms in terms of the variational lower bound, and the estimated marginal likelihood.

The generative model (encoder) and variational approximation (decoder) from section 3 were used, where the described encoder and decoder have an equal number of hidden units. Since the Frey Face data are continuous, we used a decoder with Gaussian outputs, identical to the encoder, except that the means were constrained to the interval $(0, 1)$ using a sigmoidal activation function at the decoder output. Note that with *hidden units* we refer to the hidden layer of the neural networks of the encoder and decoder.

Parameters are updated using stochastic gradient ascent where gradients are computed by differentiating the lower bound estimator $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X})$ (see algorithm 1), plus a small weight decay term corresponding to a prior $p(\boldsymbol{\theta}) = \mathcal{N}(0, \mathbf{I})$. Optimization of this objective is equivalent to approximate MAP estimation, where the likelihood gradient is approximated by the gradient of the lower bound.

We compared performance of AEVB to the wake-sleep algorithm [HDFN95]. We employed the same encoder (also called recognition network) for the wake-sleep algorithm and the variational auto-encoder. All parameters, both variational and generative, were initialized by random sampling from $\mathcal{N}(0, 0.01)$, and were jointly stochastically optimized using the MAP criterion. Stepsizes were adapted with Adagrad [DHS10]; the Adagrad global stepsize parameters were chosen from $\{0.01, 0.02, 0.1\}$ based on performance on the training set in the first few iterations. Minibatches of size $M = 100$ were used, with $L = 1$ samples per datapoint.

Likelihood lower bound We trained generative models (decoders) and corresponding encoders (a.k.a. recognition networks) having 500 hidden units in case of MNIST, and 200 hidden units in case of the Frey Face dataset (to prevent overfitting, since it is a considerably smaller dataset). Figure 2 shows the results when comparing the lower bounds. Interestingly, superfluous latent variables did not result in overfitting, which is explained by the regularizing nature of the variational bound.

[3] Available at `http://www.cs.nyu.edu/~roweis/data.html`

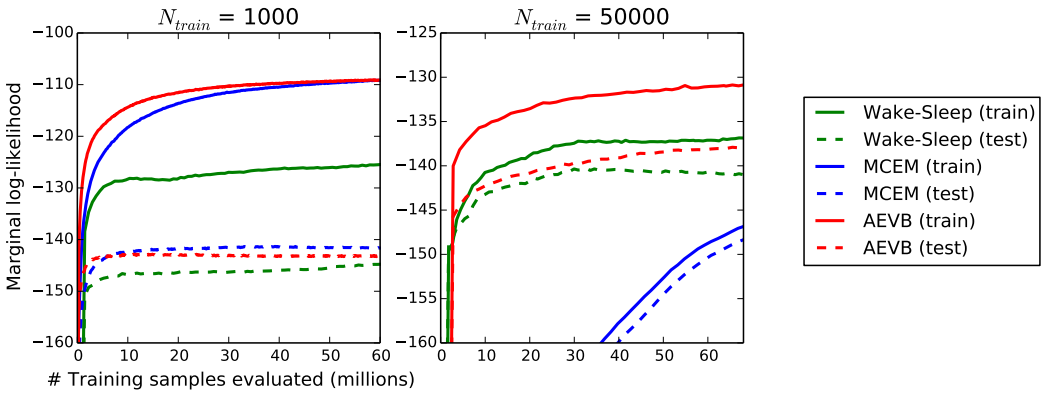
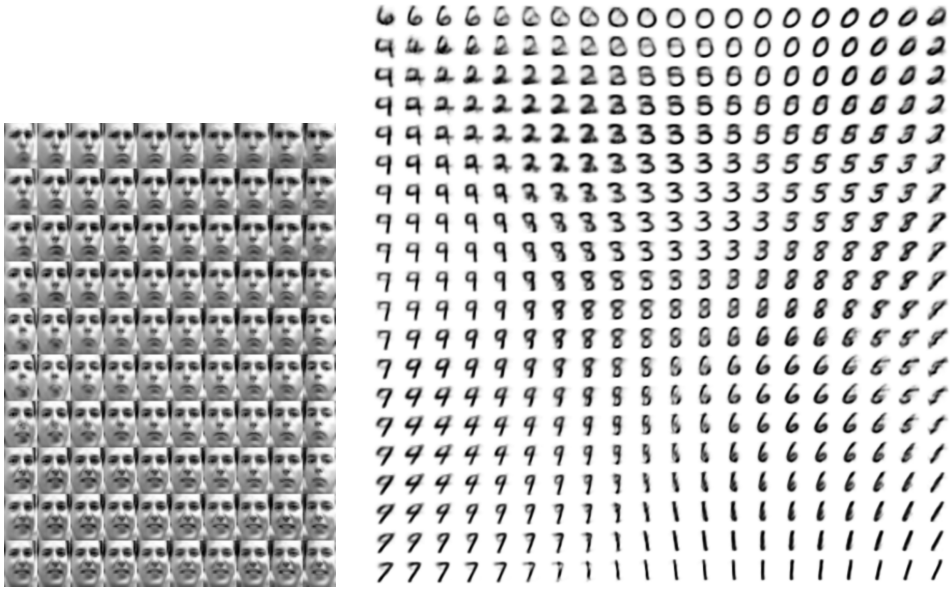


Figure 3: Comparison of AEVB to the wake-sleep algorithm and Monte Carlo EM, in terms of the estimated marginal likelihood, for a different number of training points. The Monte Carlo EM algorithm is (unlike AEVB and the wake-sleep method) asymptotically unbiased but cannot be applied online such that it becomes inefficient for large datasets (right figure).

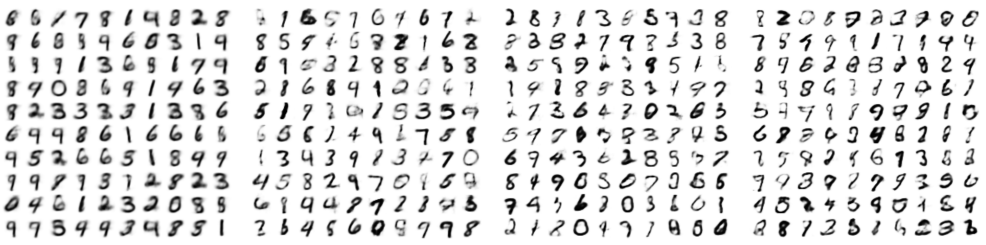


(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Marginal likelihood For very low-dimensional latent space it is possible to estimate the marginal likelihood of the learned generative models using an MCMC estimator. More information about the marginal likelihood estimator is available in the appendix. For the encoder and decoder we again used neural networks, this time with 100 hidden units, and 3 latent variables; for higher dimensional latent space the estimates became unreliable. Again, the MNIST dataset was used. The AEVB and Wake-Sleep methods were compared to Monte Carlo EM (MCEM) with a Hybrid Monte Carlo (HMC) [DKPR87] sampler; details are in the appendix. We compared the convergence speed for the three algorithms, for a small and large training set size. Results are in figure 3.



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

6 Conclusion

We have introduced a novel online learning and approximate inference method for models with of continuous latent variables, that works for the case where mean-field VB and EM methods are intractable. The proposed estimator can be straightforwardly differentiated and optimized w.r.t. all parameters, resulting in stochastic gradients that are easily plugged into existing stochastic gradient optimization methods. The method learns an encoder, or variational approximation to the posterior, that can be used for fast approximate inference of the distribution of the latent variables. The theoretical advantages are reflected in experimental results.

References

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. 2013.

[BJP12] David M Blei, Michael I Jordan, and John W Paisley. Variational bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1367–1374, 2012.

[BTL13] Yoshua Bengio and Éric Thibodeau-Laufer. Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*, 2013.

[Dev86] Luc Devroye. Sample-based non-uniform random variate generation. In *Proceedings of the 18th conference on Winter simulation*, pages 260–265. ACM, 1986.

[DHS10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2010.

[DKPR87] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[GMW13] Karol Gregor, Andriy Mnih, and Daan Wierstra. Deep autoregressive networks. *arXiv preprint arXiv:1310.8499*, 2013.

[HDFN95] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *SCIENCE*, pages 1158–1158, 1995.

[KRL08] Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical Report CBLL-TR-2008-12-01, Computational and Biological Learning Lab, Courant Institute, NYU, 2008.

[Lin89] Ralph Linsker. *An application of the principle of maximum information preservation to linear systems*. Morgan Kaufmann Publishers Inc., 1989.

[Row98] Sam Roweis. EM algorithms for PCA and SPCA. *Advances in neural information processing systems*, pages 626–632, 1998.

[SL10] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 693–700, 2010.

[VLL+10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 9999:3371–3408, 2010.

A Derivation of the alternative estimator

Here we give the derivation of the alternative estimator for the example model introduced in the paper.

The estimator in our example can be improved upon, in terms of variance, by realizing that the expectation of the second and third term of $f_{\boldsymbol{\theta}}(.)$ are tractable and can be solved analytically, since both the variational approximation $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ and the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$ are Gaussian. Let J be the dimensionality of $\mathbf{z}$. Then in this case:

$$\begin{aligned}\int q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \log p(\mathbf{z})\, d\mathbf{z} &= \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})\, d\mathbf{z} \\ &= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2)\end{aligned}$$

And:

$$\begin{aligned}-\int q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\, d\mathbf{z} &= -\int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, d\mathbf{z} \\ &= \frac{J}{2}\log(2\pi) + \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma_j^2)\end{aligned}$$

The both the variational encoder $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ and the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$ are Gaussian, so the $\int q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{z})\, d\mathbf{z}$ and $\int q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\, d\mathbf{z}$ are tractable and can be solved analytically. This results in the following estimator:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}; \mathbf{x}^{(i)}) &= \int q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})\right) d\mathbf{z} \\ &= \int q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})\right) d\mathbf{z} + \int p(\boldsymbol{\epsilon}) \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\big|_{\mathbf{z}=g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathbf{x}^{(i)})} d\boldsymbol{\epsilon} \\ &= \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \int p(\boldsymbol{\epsilon}) \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\big|_{\mathbf{z}=g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathbf{x}^{(i)})} d\boldsymbol{\epsilon} \\ &\simeq \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L}\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})\big|_{\mathbf{z}^{(i,l)}=\boldsymbol{\mu}^{(i)}+\boldsymbol{\sigma}^{(i)}\odot\boldsymbol{\epsilon}^{(i,l)}} \\ &\quad \text{where} \quad \boldsymbol{\epsilon}^{(i,l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})\end{aligned}$$

Note that $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ were defined earlier. $\boldsymbol{\mu}^{(i)}$ simply denotes the variational mean evaluated at datapoint i, and $\mu_j^{(i)}$ is simply the j-th element of that vector. The same notation is used for $\boldsymbol{\sigma}$.

B Marginal likelihood estimator

We derived the following marginal likelihood estimator that produces good estimates of the marginal likelihood as long as the dimensionality of the sampled space is low (less then 5 dimensions), and sufficient samples are taken. Let $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ be the generative model we are sampling from, and for a given datapoint $\mathbf{x}^{(i)}$ we would like to estimate the marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$.

The estimation process consists of three stages:

1. Sample L values $\{\mathbf{z}^{(l)}\}$ from the posterior using gradient-based MCMC, e.g. Hybrid Monte Carlo, using $\nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{z}) + \nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$.
2. Fit a density estimator $q(\mathbf{z})$ to these samples $\{\mathbf{z}^{(l)}\}$.
3. Again, sample L new values from the posterior. Plug these samples, as well as the fitted $q(\mathbf{z})$, into the following estimator:

$$p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \simeq \left(\frac{1}{L} \sum_{l=1}^{L} \frac{q(\mathbf{z}^{(l)})}{p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(l)})} \right)^{-1} \quad \text{where} \quad \mathbf{z}^{(l)} \sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})$$

Derivation of the estimator:

$$\begin{aligned}
\frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})} &= \frac{\int q(\mathbf{z})\, d\mathbf{z}}{p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})} = \frac{\int q(\mathbf{z}) \frac{p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z})}\, d\mathbf{z}}{p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})} \\
&= \int \frac{p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})} \frac{q(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z})}\, d\mathbf{z} \\
&= \int p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)}) \frac{q(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z})}\, d\mathbf{z} \\
&\simeq \frac{1}{L} \sum_{l=1}^{L} \frac{q(\mathbf{z}^{(l)})}{p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(l)})} \quad \text{where} \quad \mathbf{z}^{(l)} \sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})
\end{aligned}$$

C Monte Carlo EM

The Monte Carlo EM algorithm does not employ an encoder, instead it samples from the posterior of the latent variables using gradients of the posterior computed with $\nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{z}) + \nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. The Monte Carlo EM procedure consists of 10 HMC leapfrog steps with an automatically tuned stepsize such that the acceptance rate was 90%, followed by 5 weight updates steps using the acquired sample. For all algorithms the parameters were updated using the Adagrad stepsizes (with accompanying annealing schedule).

The marginal likelihood was estimated with the first 1000 datapoints from the train and test sets, for each datapoint sampling 50 values from the posterior of the latent variables using Hybrid Monte Carlo with 4 leapfrog steps.

D Full VB

As written in the paper, it is possible to perform variational inference on both the parameters $\boldsymbol{\theta}$ and the latent variables $\mathbf{z}$, as opposed to just the latent variables as we did in the paper. Here, we'll derive our estimator for that case.

Let $p_{\boldsymbol{\alpha}}(\boldsymbol{\theta})$ be some hyperprior for the parameters introduced above, parameterized by $\boldsymbol{\alpha}$. The marginal likelihood can be written as:

$$\log p_{\boldsymbol{\alpha}}(\mathbf{X}) = D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||p_{\boldsymbol{\alpha}}(\boldsymbol{\theta}|\mathbf{X})) + \mathcal{L}(\boldsymbol{\phi}; \mathbf{X}) \tag{12}$$

where the first RHS term denotes a KL divergence of the approximate from the true posterior, and where $\mathcal{L}(\boldsymbol{\phi}; \mathbf{X})$ denotes the variational lower bound to the marginal likelihood:

$$\mathcal{L}(\boldsymbol{\phi}; \mathbf{X}) = \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{X}) + \log p_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \right) d\boldsymbol{\theta} \tag{13}$$

Note that this is a lower bound since the KL divergence is non-negative; the bound equals the true marginal when the approximate and true posteriors match exactly. The term $\log p_{\boldsymbol{\theta}}(\mathbf{X})$ is composed of a sum over the marginal likelihoods of individual datapoints $\log p_{\boldsymbol{\theta}}(\mathbf{X}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$, which can each be rewritten as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{14}$$

where again the first RHS term is the KL divergence of the approximate from the true posterior, and $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x})$ is the variational lower bound of the marginal likelihood of datapoint i:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right) d\mathbf{z} \tag{15}$$

The expectations on the RHS of eqs (13) and (15) can obviously be written as a sum of three separate expectations, of which the second and third component can sometimes be analytically solved, e.g. when both $p_{\boldsymbol{\theta}}(\mathbf{x})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ are Gaussian. For generality we will here assume that each of these expectations is intractable.

Given the equations above it is easy to construct a Monte Carlo estimator of the lower bound, for example the following:

$$\mathcal{L}(\boldsymbol{\phi}; \mathbf{X}) \simeq \frac{1}{L} \sum_{l=1}^{L} N \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(l)}|\mathbf{z}^{(l)}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}^{(l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)}) \right) + \log p_{\boldsymbol{\alpha}}(\boldsymbol{\theta}^{(l)}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}^{(l)}) \tag{16}$$

where $\mathbf{x}^{(l)} \sim \mathbf{X}$ (random datapoints from the dataset) and $\boldsymbol{\theta}^{(l)} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ and $\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(l)})$.

We would like to optimize the lower bound $\mathcal{L}(\boldsymbol{\phi}; \mathbf{X})$ (eq. (13)) w.r.t. $\boldsymbol{\phi}$ using stochastic gradient-based optimization. Note that following these gradients would either decrease the KL divergence between the approximate and true posteriors, or increase the marginal likelihood. While the estimator above is unbiased, differentiating it w.r.t. the parameters $\boldsymbol{\phi}$ will not result in an unbiased gradient: the variational parameters $\boldsymbol{\phi}$ indirectly influence the estimate through the samples $\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, and it is impossible to differentiate through this sampling process.

D.1 Our estimator of the lower bound

Under certain mild conditions outlined in section (see paper) for chosen approximate posteriors $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ we can reparameterize its conditional samples $\widetilde{\mathbf{z}} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as

$$\widetilde{\mathbf{z}} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{17}$$

where we choose a prior $p(\boldsymbol{\epsilon})$ and a function $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ such that the following holds:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) &= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right) d\mathbf{z} \\ &= \int p(\boldsymbol{\epsilon}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right) \Big|_{\mathbf{z}=g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}^{(i)})} d\boldsymbol{\epsilon} \end{aligned} \tag{18}$$

The same can be done for the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$:

$$\widetilde{\boldsymbol{\theta}} = h_{\boldsymbol{\phi}}(\boldsymbol{\zeta}) \quad \text{with} \quad \boldsymbol{\zeta} \sim p(\boldsymbol{\zeta}) \tag{19}$$

where we, similarly as above, choose a prior $p(\boldsymbol{\zeta})$ and a function $h_{\boldsymbol{\phi}}(\boldsymbol{\zeta})$ such that the following holds:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\phi}; \mathbf{X}) &= \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{X}) + \log p_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \right) d\boldsymbol{\theta} \\ &= \int p(\boldsymbol{\zeta}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{X}) + \log p_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \right) \Big|_{\boldsymbol{\theta}=h_{\boldsymbol{\phi}}(\boldsymbol{\zeta})} d\boldsymbol{\zeta} \end{aligned} \tag{20}$$

For notational conciseness we introduce a shorthand notation $f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z}, \boldsymbol{\theta})$:

$$f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z}, \boldsymbol{\theta}) = N \cdot (\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) + \log p_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \tag{21}$$

Using equations (20) and (18), the Monte Carlo estimate of the variational lower bound, given datapoint $\mathbf{x}^{(i)}$, is:

$$\mathcal{L}(\boldsymbol{\phi}; \mathbf{X}) \simeq \frac{1}{L} \sum_{l=1}^{L} f_{\boldsymbol{\phi}}(\mathbf{x}^{(l)}, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(l)}), h_{\boldsymbol{\phi}}(\zeta^{(l)})) \tag{22}$$

Algorithm 2 Pseudocode for computing a stochastic gradient using our estimator. See section D.1 for meaning of the functions $f_{\boldsymbol{\phi}}$, $g_{\boldsymbol{\phi}}$ and $h_{\boldsymbol{\phi}}$.

Require: $\boldsymbol{\phi}$ (Current value of variational parameters)
  $\mathbf{g} \leftarrow 0$
  for l is 1 to L **do**
    $\mathbf{x} \leftarrow$ Random draw from dataset $\mathbf{X}$
    $\boldsymbol{\epsilon} \leftarrow$ Random draw from prior $p(\boldsymbol{\epsilon})$
    $\boldsymbol{\zeta} \leftarrow$ Random draw from prior $p(\boldsymbol{\zeta})$
    $\mathbf{g} \leftarrow \mathbf{g} + \frac{1}{L} \nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}(\mathbf{x}, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}), h_{\boldsymbol{\phi}}(\boldsymbol{\zeta}))$
  end for
  return $\mathbf{g}$

where $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$ and $\zeta^{(l)} \sim p(\boldsymbol{\zeta})$. The estimator only depends on samples from $p(\boldsymbol{\epsilon})$ and $p(\boldsymbol{\zeta})$ which are obviously not influenced by $\boldsymbol{\phi}$, therefore the estimator can be differentiated w.r.t. $\boldsymbol{\phi}$. The resulting stochastic gradients can be used in conjunction with stochastic optimization methods such as SGD or Adagrad [DHS10]. See algorithm 1 for a basic approach to computing stochastic gradients.

D.2 Example

D.2.1 Gaussian priors and variational approximations

While there is much freedom in the choice of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$, we'll for a moment assume a relatively simple case. Let the prior over the parameters and latent variables be the centered isotropic Gaussian $p_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$ and $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Note that in this case, the prior lacks parameters. Let's also assume that the true posteriors are approximatily Gaussian with an approximately diagonal covariance. In this case, we can let the variational approximate posteriors be multivariate Gaussians with a diagonal covariance structure:

$$\begin{aligned} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) &= \log \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}^2_{\boldsymbol{\theta}} \mathbf{I}) \\ \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) &= \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}^2_{\mathbf{z}} \mathbf{I}) \end{aligned} \tag{23}$$

where $\boldsymbol{\mu}_{\mathbf{z}}$ and $\boldsymbol{\sigma}_{\mathbf{z}}$ are yet unspecified functions of $\mathbf{x}$. Since they are Gaussian, we can parameterize the variational approximate posteriors:

$$\begin{aligned} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \quad &\text{as} \quad \widetilde{\boldsymbol{\theta}} = \boldsymbol{\mu}_{\boldsymbol{\theta}} + \boldsymbol{\sigma}_{\boldsymbol{\theta}} \odot \boldsymbol{\zeta} \qquad &&\text{where} \quad \boldsymbol{\zeta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \quad &\text{as} \quad \widetilde{\mathbf{z}} = \boldsymbol{\mu}_{\mathbf{z}} + \boldsymbol{\sigma}_{\mathbf{z}} \odot \boldsymbol{\epsilon} \qquad &&\text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \end{aligned}$$

With $\odot$ we signify an element-wise product. These can be plugged into the lower bound defined above (eqs (21) and (22)).

In this case it is possible to construct an alternative estimator with a lower variance, since in this model $p_{\boldsymbol{\alpha}}(\boldsymbol{\theta})$, $p_{\boldsymbol{\theta}}(\mathbf{z})$, $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ are Gaussian, and therefore four terms of $f_{\boldsymbol{\phi}}$ can be solved analytically. The resulting estimator is:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\phi}; \mathbf{X}) \simeq \frac{1}{L} \sum_{l=1}^{L} N \cdot \left(\frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma^{(l)}_{\mathbf{z},j})^2) - (\mu^{(l)}_{\mathbf{z},j})^2 - (\sigma^{(l)}_{\mathbf{z},j})^2 \right) + \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)} \mathbf{z}^{(i)}) \right) \\ + \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma^{(l)}_{\boldsymbol{\theta},j})^2) - (\mu^{(l)}_{\boldsymbol{\theta},j})^2 - (\sigma^{(l)}_{\boldsymbol{\theta},j})^2 \right) \end{aligned} \tag{24}$$

$\mu^{(i)}_j$ and $\sigma^{(i)}_j$ simply denote the j-th element of vectors $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\sigma}^{(i)}$.

D.2.2 Example with auto-encoder

In the example above, we still left unspecified the generative PDF $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ (the decoder), as well as the mean $\boldsymbol{\mu}_{\mathbf{z}}$ and s.d. $\boldsymbol{\sigma}_{\mathbf{z}}$ of the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ (the encoder). Although we can choose any differentiable PDF for both, let, for example, the probabilistic decoder be a multivariate

Bernoulli whose probabilities are computed from $\mathbf{z}$ with a fully-connected neural network with a single hidden layer:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) &= \sum_{i=1}^{D} x_i \log y_i - (1 - x_i) \cdot \log(1 - y_i) \\ \text{where } \mathbf{y} &= \exp(\mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2) \end{aligned} \tag{25}$$

Let the Gaussian mean and covariance of the encoder be a function of $\mathbf{x}$, using a similar neural network:

$$\begin{aligned} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) &= \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I}) \\ \text{where } \boldsymbol{\mu} &= \mathbf{W}_4 \mathbf{h} + \mathbf{b}_4 \\ \log \boldsymbol{\sigma}^2 &= \mathbf{W}_5 \mathbf{h} + \mathbf{b}_5 \\ \mathbf{h} &= \tanh(\mathbf{W}_3 \mathbf{x} + \mathbf{b}_3) \end{aligned} \tag{26}$$

Note that the generative parameters are $\boldsymbol{\theta} = \{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^{2}$. The variational parameters are $\boldsymbol{\phi} = \boldsymbol{\phi}_{\boldsymbol{\theta}} \cup \boldsymbol{\phi}_{\mathbf{z}}$ where $\boldsymbol{\phi}_{\boldsymbol{\theta}} = \{\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}_{\boldsymbol{\theta}}\}$ and $\boldsymbol{\phi}_{\mathbf{z}} = \{\mathbf{W}_j, \mathbf{b}_j\}_{j=3}^{5}$. These definitions above can be plugged in eq. 22 or eq. 24 (for a lower variance), and the lower bound can subsequently can be differentiated and optimized w.r.t. the parameters.