Stochastic Gradient VB and the Variational Auto-Encoder

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

Can we efficiently learn the parameters of directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? 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, Stochastic Gradient Variational Bayes (SGVB), optimizes a probabilistic encoder (also called a recognition model) to approximate the intractable posterior distribution of the latent variables. Crucial 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 approximation 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 a reparameterization of the variational lower bound yields a practical differentiable estimator of the lower bound. This SGVB (Stochastic Gradient Variational Bayes) estimator can be straightforwardly used as a stochastic objective function, and jointly optimized w.r.t. all parameters, i.e. both the variational parameters and regular parameters, using standard stochastic gradient ascent techniques.

The SGVB algorithm can be applied to learning almost any generative model with continuous latent variables. When we use a neural networks for the posterior approximation, we arrive at a *variational auto-encoder*. The SGVB objective for this case contains a regularization terms dictated by the variational bound, and a stochastic data reconstruction term. From the learned generative model it is straightforward to generate samples simply by ancestral sampling. 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 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

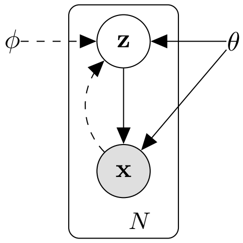


Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to the intractable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

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 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 common simplifying assumptions about the marginal or posterior probabilities. Conversely, we are here interested in a general algorithm that is even works efficiently 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 a recognition model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$: 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 necessarily factorial and its parameters $\boldsymbol{\phi}$ are not computed from some closed-form expectation. Instead, we'll introduce a method for learning the recognition model parameters $\boldsymbol{\phi}$ jointly with the generative model parameters $\boldsymbol{\theta}$.

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 the recognition model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as a probabilistic *encoder*, 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 probabilistic *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. Since this KL-divergence is non-negative, the second RHS term $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ is called the (variational) *lower bound* on the marginal likelihood of datapoint i, and can be written as:

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

which can also be written as:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) \right] \tag{3}$$

We want to differentiate and optimize the lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ w.r.t. both the variational parameters $\boldsymbol{\phi}$ and generative parameters $\boldsymbol{\theta}$. However, the gradient of the lower bound w.r.t. $\boldsymbol{\phi}$ is a bit problematic. The usual (naïve) Monte Carlo gradient estimator for this type of problem is: $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})} \left[f(\mathbf{z}) \right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})} \left[f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z})} \log q_{\boldsymbol{\phi}}(\mathbf{z}) \right] \simeq \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})$ where $\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$. This gradient estimator exhibits exhibits very high variance (see e.g. [BJP12]) and is impractical for our purposes.

2.3 Stochastic Gradient Variational Bayes (SGVB)

In this section we introduce a practical estimator of the lower bound and its derivatives w.r.t. the parameters. We assume an approximate posterior in the form $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, but please note that the technique can be applied to the case $q_{\boldsymbol{\phi}}(\mathbf{z})$, i.e. where we do not condition on $\mathbf{x}$, as well.

Under certain mild conditions outlined in section 2.4 for a chosen approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ we can reparameterize the random variable $\widetilde{\mathbf{z}} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ using a differentiable transformation $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ of an (auxiliary) noise variable $\boldsymbol{\epsilon}$:

$$\widetilde{\mathbf{z}} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{4}$$

See section 2.4 for general strategies for chosing such an approriate distribution $p(\boldsymbol{\epsilon})$ and function $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$. We can now form Monte Carlo estimates of expectations of some function $f(\mathbf{z})$ w.r.t. $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as follows:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})} \left[f(\mathbf{z}) \right] = \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}^{(i)})) \right] \simeq \frac{1}{L} \sum_{l=1}^{L} f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})) \quad \text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{5}$$

Algorithm 1 Pseudocode of SGVB algorithm based on minibatches. Either of the two estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

```
$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
    $\mathbf{g} \leftarrow 0$
    $\mathbf{X}^M \leftarrow$ Random minibatch of $M$ datapoints (drawn from full dataset)
    $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
    $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
    $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$
```

We apply this technique to the variational lower bound (eq. (2)), yielding our generic SGVB estimator $\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$:

$$\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)} | \mathbf{x}^{(i)})$$
$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{6}$$

Often, the KL-divergence $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}))$ of eq. (3) can be integrated analytically (see appendix B), such that only the expected reconstruction error $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$ requires estimation by sampling. The KL-divergence term can then be interpreted as a regularizing $\boldsymbol{\phi}$, encouraging the approximate posterior to be close to the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$. This yields a second version of the SGVB estimator $\widetilde{\mathcal{L}}^B(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$, corresponding to eq. (3), which typically has less variance than the generic estimator:

$$\widetilde{\mathcal{L}}^B(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \frac{1}{L} \sum_{l=1}^{L} (\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}))$$
$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{7}$$

Given multiple datapoints from a dataset $\mathbf{X}$ with N datapoints, we can construct an estimator of the marginal likelihood lower bound of the full dataset, based on minibatches:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \simeq \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M) = \frac{N}{M} \sum_{i=1}^{M} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{8}$$

where the minibatch $\mathbf{X}^M = \{\mathbf{x}^{(i)}\}_{i=1}^M$ is a randomly drawn sample of M datapoints from the full dataset $\mathbf{X}$ with N datapoints. 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. (7). The first term is (the KL divergence of the approximate posterior from the prior) acts as a regularizer, while the second term is a an expected negative reconstruction error. 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 $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$, which equals the probability density (or mass) of datapoint $\mathbf{x}^{(i)}$ under the generative model, given $\mathbf{z}^{(i,l)}$. This term is a negative *reconstruction error* in auto-encoder parlance.

2.4 The reparameterization 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 $\mathbf{z}$ be a continuous random variable, and $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ be some conditional distribution. It is then often possible to express the random variable $\mathbf{z}$ as a deterministic 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 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 $z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$. In this case, a valid reparameterization is $z = \mu + \sigma\epsilon$, where ϵ is an auxiliary noise variable $\epsilon \sim \mathcal{N}(0, 1)$. Therefore, $\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}\left[f(z)\right] = \mathbb{E}_{\mathcal{N}(\epsilon;0,1)}\left[f(\mu + \sigma\epsilon)\right] \simeq \frac{1}{L} \sum_{l=1}^{L} f(\mu + \sigma\epsilon^{(l)})$ where $\epsilon^{(l)} \sim \mathcal{N}(0, 1)$.

For which $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can we choose such a differentiable transformation $g_{\boldsymbol{\phi}}(.)$ and auxiliary variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$? Three basic approaches are:

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 we can choose the standard distribution (with location $= 0$, scale $= 1$) as the auxiliary variable $\boldsymbol{\epsilon}$, and let $g(.) = \text{location} + \text{scale} \cdot \boldsymbol{\epsilon}$. Examples: Laplace, Elliptical, Student's t, Logistic, Uniform, Triangular and Gaussian distributions.
3. Composition: It is often possible to express random variables as different transformations of auxiliary variables. 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: Variational Auto-Encoder

In this section we'll give an example where we use a neural network for the probabilistic encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ (the approximation to the posterior of the generative model $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$) and where the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are optimized jointly with the SGVB algorithm.

Let the prior over the latent variables be the centered isotropic multivariate Gaussian $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Note that in this case, the prior lacks parameters. We let $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ be a multivariate Gaussian (in case of real-valued data) or Bernoulli (in case of binary data) whose distribution parameters are computed from $\mathbf{z}$ with a MLP (a fully-connected neural network with a single hidden layer, see appendix C). Note the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is in this case intractable. While there is much freedom in the form $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, we'll assume the true (but intractable) posterior 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]:

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\mathbf{I}) \tag{9}$$

where the mean and s.d. of the approximate posterior, $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\sigma}^{(i)}$, are outputs of the encoding MLP, i.e. nonlinear functions of datapoint $\mathbf{x}^{(i)}$ and the variational parameters $\boldsymbol{\phi}$ (see appendix C).

As explained in section 2.4, we sample from the posterior $\mathbf{z}^{(i,l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$ using $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}, \boldsymbol{\epsilon}^{(l)}) = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)}$ where $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. With $\odot$ we signify an element-wise product. In this model both $p_{\boldsymbol{\theta}}(\mathbf{z})$ (the prior) and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ are Gaussian; in this case, we can use the

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

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

estimator of eq. (7) where the KL divergence can be computed and differentiated without estimation (see appendix B). The resulting estimator for this model and datapoint $\mathbf{x}^{(i)}$ is:

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

$$\text{where} \quad \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \tag{10}$$

As explained above and in appendix C, the decoding term $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$ is a Bernoulli or Gaussian MLP, depending on the type of data we are modelling.

4 Related work

The wake-sleep algorithm [HDFN95] is, to the best of our knowledge, the only other on-line learning method in the literature that is applicable to the same general class of continuous latent variable models. Like our method, the wake-sleep algorithm employs a recognition model that approximates the true posterior. A drawback of the wake-sleep algorithm is that it requires a concurrent optimization of two objective functions, which together do not correspond to optimization of (a bound of) the marginal likelihood. An advantage of wake-sleep is that it also applies to models with discrete latent variables. Wake-Sleep has the same computational complexity as SGVB per datapoint.

Stochastic variational inference has recently received increasing interest. Recently, [BJP12] introduced a control variate schemes to reduce the high variance of the naïve gradient estimator discussed in section 2.1, and applied to exponential family approximations of the posterior. In [SK13], a similar reparameterization as in this paper was used in an efficient version of a fixed-point variational inference algorithm for learning the natural parameters of exponential-family approximating distributions.

The SGVB algorithm exposes a connection between directed probabilistic models (trained with a variational objective) 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]. From an auto-encoder perspective, our method is attractive since it offers a regularization of activations of Z that is dictated by the variational bound (see second and third RHS terms of), lacking the usual nuisance hyperparameter for regularization that is typically required to learn useful representations. Regularization techniques have been proposed to make autoencoders learn useful representations, such as denoising, contractive and sparse autoencoder variants [BCV13]. The SGVB objective contains a regularization term dictated by the variational bound (e.g. eq. (10)). 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 model 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. Even more recently, [RMW14] also make the connection between auto-encoders, directed proabilistic models and stochastic variational inference using the

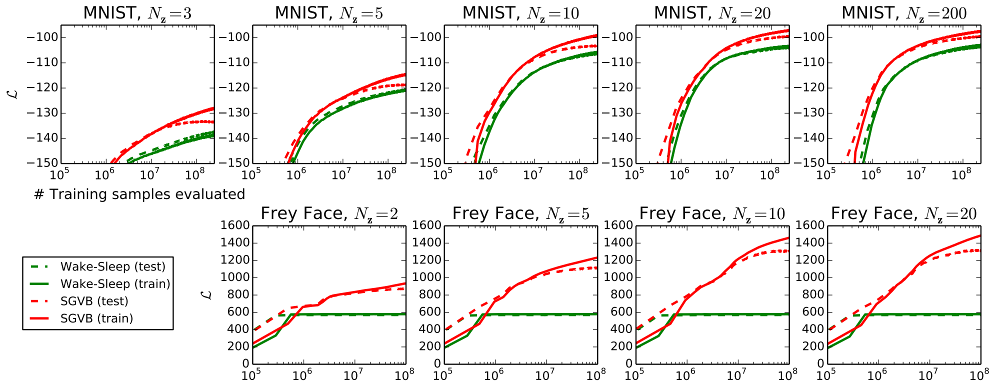


Figure 2: Comparison of our SGVB method to the wake-sleep algorithm, in terms of optimizing the lower bound, for different dimensionality of latent space ($N_{\mathbf{z}}$). Our method converged considerably faster and reached a better solution in all experiments. Interestingly enough, more latent variables 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.

reparameterization trick we describe in this paper. Their work was developed independently of ours and provides an additional perspective on SGVB.

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 SGVB to the wake-sleep algorithm [HDFN95]. We employed the same encoder (also called recognition model) 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 models) 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`

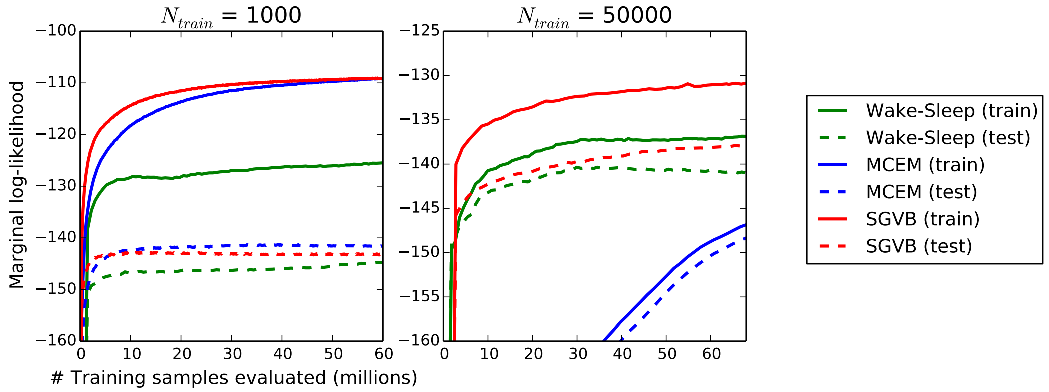


Figure 3: Comparison of SGVB to the wake-sleep algorithm and Monte Carlo EM, in terms of the estimated marginal likelihood, for a different number of training points. Monte Carlo EM is not an on-line algorithm, and (unlike SGVB and the wake-sleep method) can't be applied efficiently for the full MNIST dataset.

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 SGVB 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.

Visualisation of high-dimensional data If we choose a low-dimensional latent space (e.g. 2D), we can use the learned encoders (recognition model) to project high-dimensional data to a low-dimensional manifold. See appendix A for visualisations of the 2D latent manifolds for the MNIST and Frey Face datasets.

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.

7 Future work

Since the SGVB algorithm can be applied to almost any directed model with continuous latent variables, there are plenty of future directions: (i) learning hierarchical generative architectures with deep neural networks (e.g. convolutional networks) used for the encoders and decoders, trained jointly with SGVB; (ii) time-series models (i.e. dynamic Bayesian networks); (iii) application of SGVB to the global parameters; (iv) supervised models with latent variables, useful for learning complicated noise distributions.

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.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and variational inference in deep latent gaussian models. *arXiv preprint arXiv:1401.4082*, 2014.

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

[SK13] Tim Salimans and David A Knowles. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4), 2013.

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

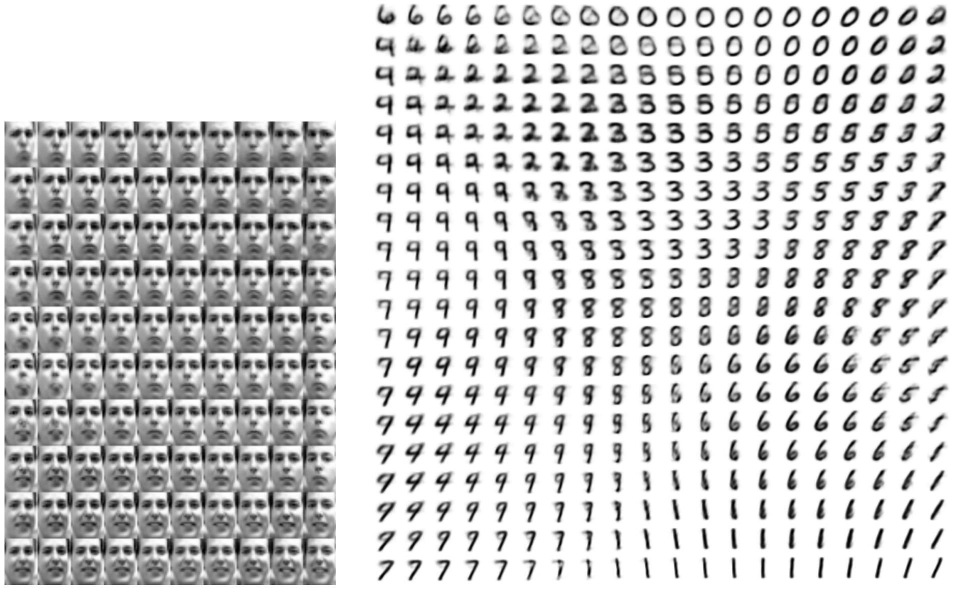
[VLL$^+$10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 9999:3371–3408, 2010.

A Visualisations

See figures 4 and 5 for visualisations of latent space and corresponding observed space of models learned with SGVB.

B Solution of $-D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z}))$, Gaussian case

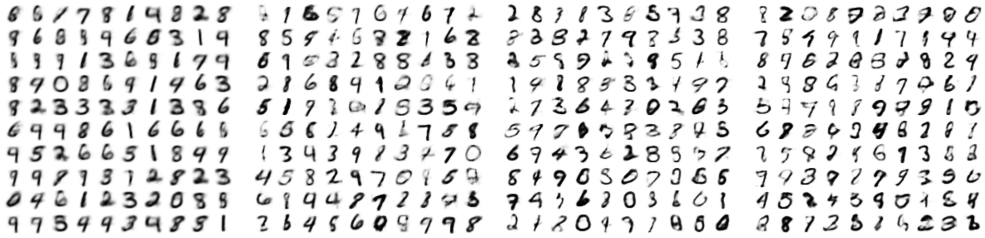
The variational lower bound (the objective to be maximized) contains a KL term that can often be integrated analytically. Here we give the solution when both the prior $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ and the posterior approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$ are Gaussian. Let J be the dimensionality of $\mathbf{z}$. Let $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$



(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 SGVB. 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 $\boldsymbol{\theta}$.



(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.

denote the variational mean and s.d. evaluated at datapoint i, and let μ_j and σ_j simply denote the j-th element of these vectors. Then:

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

And:

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

Therefore:

$$
\begin{aligned}
-D_{KL}((q_{\boldsymbol{\phi}}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z})) &= \int q_{\boldsymbol{\theta}}(\mathbf{z}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\theta}}(\mathbf{z})\right) \, d\mathbf{z} \\
&= \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2\right)
\end{aligned}
$$

When using a recognition model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ then $\boldsymbol{\mu}$ and s.d. $\boldsymbol{\sigma}$ are simply functions of $\mathbf{x}$ and the variational parameters $\boldsymbol{\phi}$, as exemplified in the text.

C MLP's as probabilistic encoders and decoders

In variational auto-encoders, neural networks are used as probabilistic encoders and decoders. There are many possible choices of encoders and decoders, depending on the type of data and model. In our example we used relatively simple neural networks, namely multi-layered perceptrons (MLPs). For the encoder we used a MLP with Gaussian output, while for the decoder we used MLPs with either Gaussian or Bernoulli outputs, depending on the type of data.

C.1 Bernoulli MLP as decoder

In this case let $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ be a multivariate Bernoulli whose probabilities are computed from $\mathbf{z}$ with a fully-connected neural network with a single hidden layer:

$$
\begin{aligned}
\log p(\mathbf{x}|\mathbf{z}) &= \sum_{i=1}^{D} x_i \log y_i + (1 - x_i) \cdot \log(1 - y_i) \\
\text{where } \mathbf{y} &= f_\sigma(\mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2) \qquad (11)
\end{aligned}
$$

where $f_\sigma(.)$ is the elementwise sigmoid activation function, and where $\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ are the weights and biases of the MLP.

C.2 Gaussian MLP as encoder or decoder

In this case let encoder or decoder be a multivariate Gaussian with a diagonal covariance structure:

$$
\begin{aligned}
\log p(\mathbf{x}|\mathbf{z}) &= \log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I}) \\
\text{where } \boldsymbol{\mu} &= \mathbf{W}_4 \mathbf{h} + \mathbf{b}_4 \\
\log \boldsymbol{\sigma}^2 &= \mathbf{W}_5 \mathbf{h} + \mathbf{b}_5 \\
\mathbf{h} &= \tanh(\mathbf{W}_3 \mathbf{z} + \mathbf{b}_3) \qquad (12)
\end{aligned}
$$

where $\{\mathbf{W}_3, \mathbf{W}_4, \mathbf{W}_5, \mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5\}$ are the weights and biases of the MLP and part of $\boldsymbol{\theta}$ when used as decoder. Note that when this network is used as an encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, then $\mathbf{z}$ and $\mathbf{x}$ are swapped, and the weights and biases are variational parameters $\boldsymbol{\phi}$.

D 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:

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

E 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.

F 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:

$$\log p_{\boldsymbol{\alpha}}(\mathbf{X}) = D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||p_{\boldsymbol{\alpha}}(\boldsymbol{\theta}|\mathbf{X})) + \mathcal{L}(\boldsymbol{\phi}; \mathbf{X}) \tag{13}$$

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:

$$\mathcal{L}(\boldsymbol{\phi}; \mathbf{X}) = \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{X}) + \log p_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \right) \, d\boldsymbol{\theta} \tag{14}$$

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:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{15}$$

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:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right) \, d\mathbf{z} \tag{16}$$

Algorithm 2 Pseudocode for computing a stochastic gradient using our estimator. See text 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 $\mathbf{g}$
```

The expectations on the RHS of eqs (14) and (16) 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.

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 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:

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

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:

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

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:

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

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.

F.1 Example

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:

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

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:

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

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:

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

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