Variational Autoencoders with variance loss

Blaž Stojanovič

Cavendish Laboratory, Department of Physics, J J Thomson Avenue, Cambridge. CB3 0HE

Abstract

Contents

1. Introduction

1.1. Generative modelling

Depending on what we learn in machine learning we differentiate between *discriminative* and *generative* modelling. While there is some inconsistency in the terminology, at their core discriminative models only have the ability to discriminate between targets Y and inputs X while generative models "simulate" the underlying data generation process and have the ability to generate new samples from the underlying distribution of data. Generative modelling is very much similar to the scientific process of providing competing models of how physical phenomena work and testing them through observations. The models are considered highly intuitive and interpretable as we can encode our prior knowledge into them while treating unknown details as noise [1]. While there has been a recent surge of popularity in generative modeling, with models like the *variational autoencoder* (VAE) [2, 3], *Generative Adversarial Network* (GAN) [4] and *Normalizing Flows* [5] generative modelling in ML has a rich history. Models such as Gaussian mixture models [6], Hidden Markov models [7], Autoregressive models, Latent Dirichlet allocation [8], Boltzmann and Helmholtz machines [9, 10] have long been known and used. With increase in computational power, modern generative models have achieved astounding results in speech synthesis [11], image super-resolution [12] and have been implemented to aid supervised learning [13].

Ian Goodfellow provided a helpful taxonomy of generative models in his tutorial on GANs [4]. He differentiates generative models that maximise likelihood between how they compute or approximate the likelihood and its gradients. The model can construct an explicit probability density $p_{\theta}(\mathbf{x})$ and consequently an explicit likelihood which is then optimised. This explicit density can either be tractable as is the case in *fully observed Bayesian networks*, or it could be intractable requiring either variational approaches as in the VAE, or MCMC for example in Boltzmann machines. On the other hand we have models that do not construct the probability distribution explicitly, but instead provide a less direct way to interact with it i.e. draw samples from it. GANs fall into this category.

1.2. Deep latent variable models

We assume that an observable value $\mathbf{x}$ is generated by an unknown underlying random process with unknown probability distribution $p^{*}(\mathbf{x})$. This is the process we try to approximate using a generative model

$p_{\boldsymbol{\theta}}(\mathbf{x})$, which is parameterised with parameters $\boldsymbol{\theta}$. The process of learning is finding parameters $\boldsymbol{\theta}$ such that our model distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$ is as close to the underlying distribution $p^*(\mathbf{x})$. Normally though, this is the case for regression and classification problems, we are modelling the conditional distribution $p^*(\mathbf{y}|\mathbf{x})$ with our model $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$, where $\mathbf{y}$ is conditioned on the input $\mathbf{x}$. The approaches described and implemented in this written assignment are equally applicable to both conditional and unconditional modelling. In either case we want our model to be versatile enough to represent the underlying structure of the data. The last two decades have seen a renaissance of neural networks as models of complex processes [14], they are *universal function approximators*, generally made of compositions of smaller differentiable sub-units with nonlinear activation functions and trained using gradient-based approaches via backpropagation. They can be used to parameterize conditional distributions and form basis of probabilistic models thus combining the best of both worlds, a probabilistic description and *hopefully* better scalability, making use of gradient-based learning. The categorical distribution $p_{\boldsymbol{\theta}}(y|\mathbf{x})$ in the case of classification, where y is the class, given the input $\mathbf{x}$, can be parameterised as

$$\begin{aligned} \mathbf{p} &= f(\mathbf{x}; \boldsymbol{\theta}), \\ p_{\boldsymbol{\theta}}(y|\mathbf{x}) &= \text{Categorical}(y; \mathbf{p}), \end{aligned} \tag{1}$$

where $f(\cdot, \boldsymbol{\theta}) : \mathbb{R}^{n_{\text{in}}} \longrightarrow \mathbb{R}^{n_{\text{out}}}$ is the neural network, with a final layer such that the probabilities $\sum_i^{n_{\text{out}}} p_i = 1$.

We can combine random variables $(\mathbf{x_1}, \ldots, \mathbf{x_M})$ and represent their conditional dependence structure as a network to construct a *probabilistic graphical model*. If the structure of the network is a directed acyclic graph, the joint distribution of the random variables is factorised as a product of conditionals and priors

$$p_{\boldsymbol{\theta}}(\mathbf{x}_1, \ldots, \mathbf{x}_M) = \prod_{j=1}^{M} p_{\boldsymbol{\theta}}\left(\mathbf{x}_j \mid \text{pa}(\mathbf{x}_j)\right), \tag{2}$$

where $\text{pa}(\mathbf{x}_j)$ are the parent nodes of node j in the directed graph. A graphical model with this structure is termed a *Bayesian network* and we can parameterise the conditional distributions in a Bayesian network using neural networks

$$\begin{aligned} \boldsymbol{\eta} &= f(\text{pa}(\mathbf{x}); \boldsymbol{\theta}) \\ p_{\boldsymbol{\theta}}(\mathbf{x} \mid \text{pa}(\mathbf{x})) &= p_{\boldsymbol{\theta}}(\mathbf{x} \mid \boldsymbol{\eta}). \end{aligned} \tag{3}$$

A model which only contains observable variables $\mathbf{x}_i$ is a *fully observed* model and is straightforward to learn. Given a dataset $\mathcal{D}$ of *independently and identically distributed* observations $\mathbf{x}_i$ we can perform *maximum likelihood estimation*, i.e. search for parameters $\boldsymbol{\theta}$ such that the observed data is most probable, by simply estimating the gradients of the *log-likelihood*

$$\log p_\theta(\mathcal{D}) = \sum_{\mathbf{x} \in \mathcal{D}} \log p_\theta(\mathbf{x}) \tag{4}$$

Moreover, *maximum a posteriori estimation* of which maximum likelihood is a special case, can be performed and full Bayesian treatment of the posterior is also possible [1]. A complication arises when we include *latent* variables into the model, these are variables which are not observable and do not form a part of the dataset $\mathcal{D}$, usually denoted with $\mathbf{z}$. For unconditional model learning the graphical model now represents a joint probability distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$, which when marginalized gives the single point *model evidence* $p_{\boldsymbol{\theta}}(\mathbf{x})$ as a function of the parameters $\boldsymbol{\theta}$. In the simplest deep latent variable models, the joint probability is factored into the prior distribution $p_\theta(\mathbf{z})$ and the conditional distribution, which is represented by a neural network $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, as

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}). \tag{5}$$

Alternatively, we can have a model with many layers of latent variables $\mathbf{z}_m$,

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p\left(x \mid \mathbf{z}_1\right) p\left(\mathbf{z}_1 \mid \mathbf{z}_2\right) \cdots p\left(\mathbf{z}_{m-1} \mid \mathbf{z}_m\right) p\left(\mathbf{z}_m\right), \tag{6}$$

which are then able to learn hierarchical latent representation. In either case learning the posterior $p_\theta(\mathbf{x}, \mathbf{z})$, i.e.

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}, \tag{7}$$

is intractable, meaning it is (without tricks) impossible to differentiate the marginal likelihood w.r.t parameters $\boldsymbol{\theta}$ and perform gradient-based optimisation. Advantage of DLVMs is their flexibility, even using very simple priors and conditionals they can express complicated distributions because of the nonlinearity of the neural network representations.

1.2.1. Learning in DLVMs

There are two main issues with learning in deep latent variable models, succinctly illustrated with two optimisation approaches. The first is the Expectation-Maximization (EM) [15] algorithm, it consists of two steps which are repeated until convergence.

- **E-step**: For each datapoint $\mathbf{x}_i$ compute the posterior $p_{\boldsymbol{\theta}_t}(\mathbf{z}_i|\mathbf{x_i})$.

- **M-step**: Estimate new parameters by maximsing the expected log-likelihood $\theta_{t+1} = \arg\max_\theta \sum_{x \in D} \mathbb{E}_{z \sim p(z|x;\theta_t)} \log p(x, z; \theta)$.

The algorithm falls apart in two ways, firstly the E-step requires computation of the posterior distribution which is intractable in DLVMs. This can be solved by employing a sampling technique (MCMC EM algorithm) but at a high computational cost, requiring a per datapoint MCMC step. Secondly, the M-step optimises over all datapoints requiring vast amounts of memory, although it can be done in mini-batches (online EM [16]. Hence, the first issue with learning in DLVMs is computational efficiency. The second algorithm is the *wake-sleep* algorithm [17], it employs a recognition model that approximates the posterior distribution and is applicable to both discrete and continuous latent variable models. While it is not computationally expensive relative to EM, its optimisation objectives do not coincide with optimising the marginal likelihood, thus giving a bad estimate of the posterior. Therefore the second issue is inaccurate estimates of the posterior. Both these issues were addressed in two seminal papers that constitute the framework of the variational autoencoder [2, 3].

1.3. Variational Autoencoders

The strength of variational methods lies in recasting inference as an optimisation problem, which scales better to large datasets than sampling methods. This is done by introducing an approximate posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, with parameters $\boldsymbol{\phi}$ and optimising to maximise the similarity between $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. In the case of the VAE $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is also parameterised using a neural network. The similarity between the two distributions is measured by the Kullback-Leibler divergence

$$D_{KL}(q\|p) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})}, \tag{8}$$

which has two important properties $D_{KL}(q\|p) \geq 0$ for all q and p and it is equal to zero only when $q = p$. It can be rewritten as

$$\begin{aligned} &D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})) = \\ &= \sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{x})} + \log p_{\boldsymbol{\theta}}(\mathbf{x}) \right] \\ &= \sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{x}) \right] + q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x}), \end{aligned} \tag{9}$$

from which we can express the log likelihood as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})) + \mathcal{L}_{\theta,\phi}(\mathbf{x}), \tag{10}$$

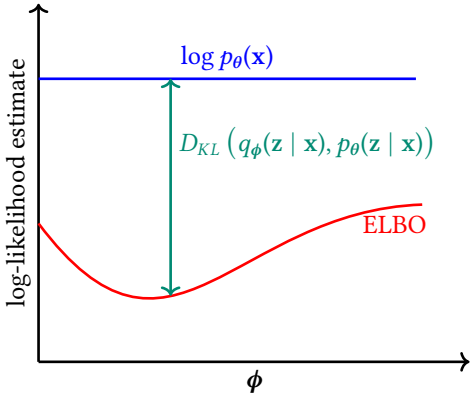


Figure 1: ELBO is a lower bound of the log likelihood, the difference between the two is the KL divergence. The better the approximation of the posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, the smaller is KL and consequently the closer is ELBO to $\log p_{\boldsymbol{\theta}}(\mathbf{x})$.

where $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$ is the *evidence lower bound* ELBO, defined as

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \right]. \tag{11}$$

ELBO gives a lower bound on the log likelihood, because Kullback-Leiber divergence is non-negative. From equation (10) we see, that maximizing the ELBO with regards to both $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ does two things, it minimizes the KL divergence between the true and approximate posteriors, and consequently better approximates the log evidence, see Figure 1. This makes ELBO a good optimisation objective for learning in VAEs.

1.3.1. Gradient estimation

In order to perform gradient descent on the ELBO objective, we need to be able to evaluate its gradients with respect to parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Taking the gradient w.r.t generative parameters $\boldsymbol{\theta}$ is straightforward, because we can change the order of the expectation operator and the gradient, leaving us with

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right] \\ &\simeq \nabla_{\boldsymbol{\theta}} \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right) \\ &= \nabla_{\boldsymbol{\theta}} \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) \right), \end{aligned} \tag{12}$$

where $\simeq$ denotes an unbiased estimator. This reversing of the order of operations is not possible when taking gradients w.r.t variational parameters $\boldsymbol{\phi}$, because the expectation $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}$ is performed w.r.t the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. The gradient could be estimated with a vanilla Monte Carlo estimator, but it has very high variance and is not practical [2].

The problem of stochastic gradient estimation of an expectation of a function is a well studied problem that transcends machine learning and has a variety of applications [18, 19]. Different estimators differ in from and their properties, variance being one of the most important. In their review [20] Mohamed et al. categorise MC gradient estimators into three categories

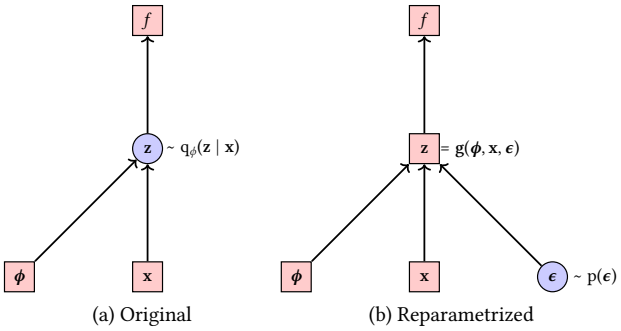
1. *Score-function estimator*: The score function is a logarithm of a probability distribution w.r.t to distributional parameters. It can be used as a gradient estimator

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}[f(\mathbf{x})] &= \nabla_{\boldsymbol{\theta}} \int p_{\boldsymbol{\theta}}(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \\ &= \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}\left[f(\mathbf{x}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})\right].\end{aligned} \tag{13}$$

   The score-function estimator is compatible with any cost function, it requires that the measure $p_{\boldsymbol{\theta}}(\mathbf{x})$ is differentiable and easy to sample. Very importantly it is applicable to both discrete and continuous distribution, but has a drawback of having high variance.
2. *Pathwise estimator*: Continuous distributions can be sampled either directly by generating samples from the distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$ or indirectly, by sampling from a simpler base distribution $p(\boldsymbol{\epsilon})$ and transforming the variate through a deterministic path $g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$. Using this, it is possible to move the source of randomness in such a way that the objective is differentiable. In essence this approach pushes the parameters of the measure into the cost function which is then differentiated. The estimator is

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}[f(\mathbf{x})] &= \nabla_{\boldsymbol{\theta}} \int p_{\boldsymbol{\theta}}(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \\ &= \nabla_{\boldsymbol{\theta}} \int p(\boldsymbol{\epsilon}) f(g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})) d\boldsymbol{\epsilon} \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\theta}} f(g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}))\right].\end{aligned} \tag{14}$$

   This was the gradient estimator originally used in the VAE implementation [2] there named as the *reparametrization trick*, see also Figure 2. In many cases the transformation paths are so simple they can be implemented in one line of code, referred to as *one-liners*. The pathwise-estimator can only be used on differentiable cost functions, but is easy to implement and crucially has lower variance than the score-function estimator.
3. *Measure-valued gradient estimator*, which exploits the properties of signed-measures is beyond the scope of this report.



(a) Original (b) Reparametrized

Figure 2: The reparametrization trick, adapted from [1]. The stochasticity of the $\mathbf{z}$ node is pushed out into a separate input to the same node, resulting in deterministic gradients w.r.t $\boldsymbol{\phi}$ through the node.

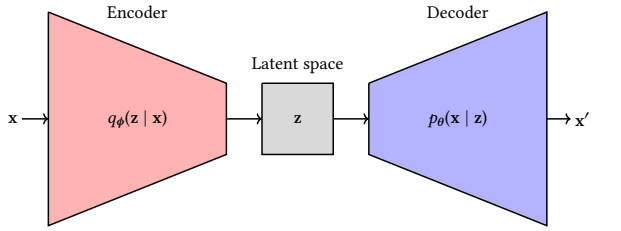


Figure 3: The variational autoencoder.

Unbiased gradient estimation makes training on the ELBO objective possible. During training the VAE learns a mapping between a complicated observed space of $\mathbf{x}$ and the latent space of $\mathbf{z}$, which is usually defined to be relatively simple. The *decoder* network or the *generative* part of the VAE $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ learns to map from the latent space to the data space, while the *inference* or *encoder* network $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ approximates the true posterior $p_{\theta}(\mathbf{z}|\mathbf{x})$ of this process, see Figure 3.

1.3.2. Variants and advances

In the previous section we have described the basic variational autoencoder as proposed in the original paper [2]. But as with any new paradigm/approach in machine learning, the variational autoencoder was quickly extended in a variety of ways. A large line of work tries to address the issues that VAEs have with learning, this was prompted by the discovery that ELBO optimisation can get stuck in suboptimal equilibria [21]. Suggestions include annealing the latent cost $D_{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ from 0 to 1 [21], or modifying the ELBO objective to ensure that a certain minimum number of bits is encoded in each latent variable [22]. The β-VAE [23] also modifies the ELBO objective by introducing the β hyperparameter to balance the regularization and reconstruction terms. One thing that the

β-VAE does particularly well is learn disentangled latent representations, which are latent representations where each component is sensitive to changes in one generative factor and is not much affected by others. Further work has been done in understanding how introducing the β parameter leads to these representations [24].

Alternative objectives to the evidence lower bound have also been proposed, instead of using the KL divergence one can use the chi-square divergence to arrive at an upper bound of the log likelihood (CUBO) and optimise that [25]. In addition to the objective of the VAE, any other component can be, and has been modified and studied. There has been work done on improving the variational posteriors by using conditional flow-based models [22, 5], implementing all sorts of architectures of the decoder, including convolutional neural networks, residual networks and recurrent neural networks [26]. Much in the same vein, variational autoencoders have been adapted for use with other data types than the original application to images. Among other uses, they have been used for text generation [27], molecular graph generation [28] and video compression [29]. Variational autoencoders can also be used in tandem with a classifier in a procedure called semi-supervised learning [13], allowing to learn from large datasets which have only partially been labeled.

We have alluded in section 1.2 that VAEs can have hierarchical latent space structure, such models have very recently been implemented and show promising results. The Bidirectional-Inference Variational Autoencoder (BIVA) [30] uses a skip-connected generative model and an inference network formed by a bidirectional stochastic inference path to learn a hierarchy of latent variables. Another hierarchical VAE is the Noveau-VAE (NVAE) [31] which uses depth-wise separable convolutions and batch normalization. Moreover, very deep VAE's [32] can represent autoregressive models and beat them on likelihood benchmarks.

Given the reliance of most variational autoencoders on the reparametrization trick, one of their main flaws is their inability to treat discrete latent variables. Current workarounds to this issue include using a continuous relaxation of the discrete latent variables [33], or pairing each discrete latent variable with an auxiliary continuous one and performing the reparametrization trick on the marginal distributions [34]. Current state of the art is the vector quantised VAE (VQ-VAE) which uses vector quantisation for the latent space and learns the prior [35].

Nevertheless, there are some problems with VAEs that have not yet been fully addressed. Perhaps the most striking issue, one that pertains to all deep generative models not just VAEs, is the *out-of-distribution problem*. Deep generative models have trouble distinguishing between a sample from the distribution they were trained on and another sample from a different distribution. This was discovered by Nalisnick et al. [36], when they noticed that the models could not distinguish common objects (CIFAR-10) from house numbers (SVHN). The second problem is the *posterior collapse*. Because of the variance of the stochastic gradient approximation the posterior approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ may provide too weak of a signal

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx q_{\boldsymbol{\phi}}(\mathbf{z}), \tag{15}$$

resulting in the decoder ignoring $\mathbf{z}$ samples from the posterior. Proposed solutions include skip connections [25] and aggressively optimising the inference model before performing each model update. [37]. There is also the *hole problem* [38], which occurs when there are discrepancies between the prior $p(\mathbf{z})$ and the aggregate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}) = \frac{1}{N}\sum_n q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_n)$, where $\mathbf{x}_n$ is a training example. Sampling from these discrepancies or *holes* provides poor latent variable samples which are then badly reconstructed by the generative network. Finally there is the *balancing problem*, the issue of finding a good balance between the sampling quality and the reconstruction quality.

2. Implementation of Variance loss

The evidence lower bound (ELBO) pathwise gradient estimator (SGVB), described in section 1.3.1, has found great success but is limited to models with continuous latent variables. It would be more convenient to have a loss function that would deal with both discrete and continuous latent variables within the same framework. Recent work by Richter et al. [39] analysed a score-function gradient estimator for ELBO and showed that it can be obtained from a different divergence measure, which they termed the *log-variance loss*

$$\mathcal{L}_{\text{LV}}\left(q_{\boldsymbol{\phi}}(\mathbf{z})\|p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})\right) = \frac{1}{2}\mathop{\mathbb{V}\text{ar}}_{\mathbf{z}\sim r(\mathbf{z})}\left[\log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})}\right)\right], \tag{16}$$

where $r(\mathbf{z})$ is some arbitrary distribution. Moreover, they showed that the gradient of $\mathcal{L}_{\text{LV}}$ w.r.t $\boldsymbol{\phi}$ corresponds to that of the KL divergence if $r(\mathbf{z}) = q_{\boldsymbol{\phi}}(\mathbf{z})$ and provided a gradient estimator of the log-variance, thus suggested an optimisation procedure that works for both discrete and continuous latent distributions. In this work we will proceed in a similar fashion but

with a different loss function. Motivated by the fact that when the posterior $q(z|x) = p(z|x)$ is exact we have

$$\frac{p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})} = p(\mathbf{x}). \tag{17}$$

The variance of $p(\mathbf{x})$ with regards to samples drawn from an arbitrary probability distribution $r(\mathbf{z})$ which suggests the variance of the left hand side

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \underset{\mathbf{z}\sim r(\mathbf{z})}{\mathbb{V}\text{ar}}\left[\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})}\right)\right] \tag{18}$$

as an objective function. We can estimate the loss with sample variance using S Monte Carlo samples

$$\begin{gathered}\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) \approx \frac{1}{(S-1)}\sum_{s=1}^{S}\left(f_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{z}^{(s)},\mathbf{x}) - \bar{f}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x})\right)^2, \\ \mathbf{z}^{(s)} \overset{\text{i.i.d.}}{\sim} r(\mathbf{z}),\end{gathered} \tag{19}$$

where

$$f_{\boldsymbol{\phi},\boldsymbol{\theta}} = \log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})}\right) \tag{20}$$

and $f_{\boldsymbol{\phi},\boldsymbol{\theta}}$ is the average

$$\bar{f}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{S}\sum_{s=1}^{S} f_{\boldsymbol{\phi},\boldsymbol{\theta}}\left(\mathbf{z}^{(s)},\mathbf{x}\right). \tag{21}$$

The gradient of expression (19) can be computed using automatic differentiation without any issue. The form of the gradient estimator of $\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x})$ outlined above, is the same as the leave-one-out estimator described in [39], which means that it is an unbiased estimator of the gradient. The procedure of calculating the gradients for a given example $\mathbf{x}$ and parameters $\boldsymbol{\phi}, \boldsymbol{\theta}$ is given in Algorithm 1. Importantly the samples $\mathbf{z}^{(s)}$, which will be for simplicity sampled from the model probability $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$, must be detached from the computational graph so that backpropagation is possible.

To test the applicability of the loss function we will test two variational autoencoders, one with continuous latent space and one with discrete latent space. The architecture of the neural networks that were used to parameterise the encoder and decoder will be discussed in section 3.

2.1. Continuous latent space

For the continuous case we will recreate the model from the original VAE paper [2]. The inference and generative models will both be parameterised by fully connected multilayer perceptrons. The approximate

Algorithm 1: Gradients of $\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x})$

Input: Encoder parameters $\boldsymbol{\phi}$, decoder parameters $\boldsymbol{\theta}$, example $\mathbf{x}$
Result: Gradients w.r.t $\boldsymbol{\phi}, \boldsymbol{\theta}$ of the objective function 1
for $s = 1, \ldots, S$ **do**
  $z^{(s)} \leftarrow \text{sample}\left(q_{\boldsymbol{\phi}}(\cdot \mid \mathbf{x})\right)$
  $z^{(s)} \leftarrow \text{stop\_gradient}\left(z^{(s)}\right)$
  $f^{(s)}_{\boldsymbol{\phi},\boldsymbol{\theta}} \leftarrow \log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}|\boldsymbol{z}^{(s)}\right) p\left(\boldsymbol{z}^{(s)}\right) - \log q_{\phi}\left(z^{(s)}\right)$
end
$\widehat{\mathcal{L}} \leftarrow \text{Variance}\left(\left\{f^{(s)}_{\boldsymbol{\phi},\boldsymbol{\theta}}\right\}_{s=1}^{S}\right)$
return $\text{grad}\left(\widehat{\mathcal{L}}\right)$

posterior $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ will be a multivariate Gaussian with a diagonal covariance matrix

$$\begin{aligned}q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) &= \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}, \text{diag}\left(\sigma^2\right)\right) \\ q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) &= \prod_i q_{\boldsymbol{\phi}}\left(z_i \mid \mathbf{x}\right) = \prod_i \mathcal{N}\left(z_i; \mu_i, \sigma_i^2\right)\end{aligned} \tag{22}$$

where

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\mathbf{x}). \tag{23}$$

The choice of decoder depends on the type of data. For real-valued outputs the generative $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ model is again a Gaussian, with mean and variance parameterised by a neural network

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{DecoderNeuralNet}_{\boldsymbol{\theta}}(\mathbf{z}). \tag{24}$$

Alternatively if the outputs are binary we use the Bernoulli distribution for the logarithm of $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) &= \sum_{j=1}^{D} \log p\left(x_j \mid \mathbf{z}\right) \\ &= \sum_{j=1}^{D} \log \text{Bernoulli}\left(x_j; y_j\right) \\ &= \sum_{j=1}^{D} x_j \log y_j + \left(1 - x_j\right) \log\left(1 - y_j\right).\end{aligned} \tag{25}$$

Where D is the dimension of $\mathbf{x}$ and $\mathbf{y}$ is the output of a sigmoid activation in the last layer of the NN

$$\mathbf{y} = f_{\sigma}\left(\text{DecoderNeuralNet}_{\boldsymbol{\theta}}(\mathbf{z})\right). \tag{26}$$

The prior $p(\mathbf{z})$ will be a unit isotropic Gaussian in either case $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$.

Bibliography

[1] D. P. Kingma, Variational inference & deep learning: A new synthesis (2017).

[2] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).

[3] D. J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: International conference on machine learning, PMLR, 2014, pp. 1278–1286.

[4] I. Goodfellow, Nips 2016 tutorial: Generative adversarial networks, arXiv preprint arXiv:1701.00160 (2016).

[5] D. Rezende, S. Mohamed, Variational inference with normalizing flows, in: International Conference on Machine Learning, PMLR, 2015, pp. 1530–1538.

[6] B. S. Everitt, Finite mixture distributions, Wiley StatsRef: Statistics Reference Online (2014).

[7] L. Rabiner, B. Juang, An introduction to hidden markov models, ieee assp magazine 3 (1) (1986) 4–16.

[8] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, the Journal of machine Learning research 3 (2003) 993–1022.

[9] D. H. Ackley, G. E. Hinton, T. J. Sejnowski, A learning algorithm for boltzmann machines, Cognitive science 9 (1) (1985) 147–169.

[10] P. Dayan, G. E. Hinton, R. M. Neal, R. S. Zemel, The helmholtz machine, Neural computation 7 (5) (1995) 889–904.

[11] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, arXiv preprint arXiv:1609.03499 (2016).

[12] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4681–4690.

[13] D. P. Kingma, D. J. Rezende, S. Mohamed, M. Welling, Semi-supervised learning with deep generative models, arXiv preprint arXiv:1406.5298 (2014).

[14] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep learning, Vol. 1, MIT press Cambridge, 2016.

[15] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, Journal of the Royal Statistical Society: Series B (Methodological) 39 (1) (1977) 1–22.

[16] M. Hoffman, F. R. Bach, D. M. Blei, Online learning for latent dirichlet allocation, in: advances in neural information processing systems, Citeseer, 2010, pp. 856–864.

[17] G. E. Hinton, P. Dayan, B. J. Frey, R. M. Neal, The" wake-sleep" algorithm for unsupervised neural networks, Science 268 (5214) (1995) 1158–1161.

[18] N. A. Chriss, N. Chriss, Black Scholes and beyond: option pricing models, McGraw-Hill, 1997.

[19] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al., Mastering atari, go, chess and shogi by planning with a learned model, Nature 588 (7839) (2020) 604–609.

[20] S. Mohamed, M. Rosca, M. Figurnov, A. Mnih, Monte carlo gradient estimation in machine learning, Journal of Machine Learning Research 21 (132) (2020) 1–62.

[21] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, S. Bengio, Generating sentences from a continuous space, arXiv preprint arXiv:1511.06349 (2015).

[22] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, M. Welling, Improving variational inference with inverse autoregressive flow, arXiv preprint arXiv:1606.04934 (2016).

[23] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, beta-vae: Learning basic visual concepts with a constrained variational framework (2016).

[24] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, A. Lerchner, Understanding disentangling in β-vae, arXiv preprint arXiv:1804.03599 (2018).

[25] A. B. Dieng, D. Tran, R. Ranganath, J. Paisley, D. M. Blei, Variational inference via χ-upper bound minimization, arXiv preprint arXiv:1611.00328 (2016).

[26] K. Gregor, I. Danihelka, A. Graves, D. Rezende, D. Wierstra, Draw: A recurrent neural network for image generation, in: International Conference on Machine Learning, PMLR, 2015, pp. 1462–1471.

[27] S. Semeniuta, A. Severyn, E. Barth, A hybrid convolutional variational autoencoder for text generation, arXiv preprint arXiv:1702.02390 (2017).

[28] W. Jin, R. Barzilay, T. Jaakkola, Junction tree variational autoencoder for molecular graph generation, in: International Conference on Machine Learning, PMLR, 2018, pp. 2323–2332.

[29] A. Habibian, T. v. Rozendaal, J. M. Tomczak, T. S. Cohen, Video compression with rate-distortion autoencoders, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7033–7042.

[30] L. Maaløe, M. Fraccaro, V. Liévin, O. Winther, Biva: A very deep hierarchy of latent variables for generative modeling, arXiv preprint arXiv:1902.02102 (2019).

[31] A. Vahdat, J. Kautz, Nvae: A deep hierarchical variational autoencoder, arXiv preprint arXiv:2007.03898 (2020).

[32] R. Child, Very deep vaes generalize autoregressive models and can outperform them on images, arXiv preprint arXiv:2011.10650 (2020).

[33] C. J. Maddison, A. Mnih, Y. W. Teh, The concrete distribution: A continuous relaxation of discrete random variables, CoRR abs/1611.00712 (2016). arXiv:1611.00712.

[34] J. T. Rolfe, Discrete variational autoencoders, arXiv preprint arXiv:1609.02200 (2016).

[35] A. van den Oord, O. Vinyals, K. Kavukcuoglu, Neural discrete representation learning, CoRR abs/1711.00937 (2017). arXiv:1711.00937.

[36] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, B. Lakshminarayanan, Do deep generative models know what they don't know?, arXiv preprint arXiv:1810.09136 (2018).

[37] J. He, D. Spokoyny, G. Neubig, T. Berg-Kirkpatrick, Lagging inference networks and posterior collapse in variational autoencoders, arXiv preprint arXiv:1901.05534 (2019).

[38] D. J. Rezende, F. Viola, Taming vaes, arXiv preprint

arXiv:1810.00597 (2018).
[39] L. Richter, A. Boustati, N. Nüsken, F. J. Ruiz, Ö. D. Akyildiz, Vargrad: A low-variance gradient estimator for variational inference, arXiv preprint arXiv:2010.10436 (2020).