Variational Autoencoders

Dalin Guo
Department of Cognitive Science
dag082@ucsd.edu

Kuei-da Liao
Department of Electrical and Computer Engineering
lkueida@ucsd.edu

Corey Zhou
Department of Cognitive Science
yiz329@ucsd.edu

1 Introduction

Generative models allow an explicit expression of constraints and causal relations pertinent to the real world. Unlike discriminative models, generative models may not be as accurate in specific discriminative tasks, yet they are immensely helpful when it comes to discovering the "hidden" structure underlying observed data and thereby predicting new observations. This property in general leads to more interpretable models that "explain" data by capturing causal factors of variation in observations.

In light of providing semantically meaningful descriptions of data, the Variational Autoencoder (VAE) was proposed [Kingma and Welling, 2014]. VAE achieves such representation learning by explicitly regularizing the distribution of the learned latent representation, so that the latent space has desirable properties to exploit (e.g. to easily generate never-seen data instances). The specific regularization it achieves is made possible through variational inference. VAE also bridges the two worlds of graphical models and deep neural networks, taking advantage of the representatonal power of the former as well as the flexibility and scalability of the latter.

In this paper, we focus primarily on the vanilla Variational Autoencoder - that is, VAE that only considers a Gaussian inference model. We will first present previous work that are relevant to the ideas and methods used by VAE, and then introduce the general setup of VAE. We will also derive the key results from Kingma and Welling [2019], including the Evidence Lower Bound (ELBO) and a key algorithm to learn ELBO associated with a Bernoulli generative model from the monograph, and show how to extend it to a Gaussian generative model for data reconstruction. Finally, we conduct a few experiments to empirically demonstrate the capabilities and limitations of VAE.

2 Related Work

2.1 Unsupervised Learning & Dimensionality Reduction

Unsupervised representation learning is driven by the approach mentioned above: to discover the underlying process that generates the observed data, and potentially use such (causal) understanding to make predictions. In addition, it is often the case that the generative process involves fewer factors than the observation (e.g. think of human faces), which gives rise to the idea of dimensionality reduction.

Dimensionality reduction methods seek to learn a representation of high-dimensional data in a (much) lower space while preserving as much information as possible. Some of the most influential methods include Principle Component Analysis (PCA), the Helmholtz Machine, and Autoencoders (AE). Both the Helmholtz Machine and AE employ a *recognition/inference model*, which transforms input data into some representation in a latent space, defined over hidden variables that are believed to underlie

Preprint. Under review.

the data generating process. The recognition model thus learns to approximate the posterior. Both also consist a *generative model*, which (re)constructs data in the original high-dimensional space based on the hidden variables. i.e. it learns to approximate the likelihood of data given latent variables. Because the latent space is usually set to have a much lower dimensionality than the data space, while the objective is to understand the true data generative process, this setup poses a bottleneck that forces the model to learn an effective and compact representation.

2.2 Autoencoder (AE)

In AE, the inference model is what's called the *encoder*, and the generative model is the *decoder*, since they encode and decode a latent representation respectively. AE is often trained to reconstruct input data. Therefore, the encoder is responsible for dimensionality reduction. Both are instantiated as neural networks. The encoder approximates a function $\mathbf{f}(\cdot)$ parameterized by θ. The decoder approximates a function $\mathbf{g}(\cdot)$ parameterized by ϕ. Let $\mathbf{x}$ denote the input. The corresponding latent vector is then $\mathbf{z} = \mathbf{f}_\theta(\mathbf{x})$, and the Reconstruction is $\mathbf{x}' = \mathbf{g}_\phi(\mathbf{z}) = \mathbf{g}_\phi(\mathbf{f}_\theta(\mathbf{x}))$.

To ensure the latent representation faithfully captures the factors given rise to the observed data, one could maximize the overall data likelihood $p_\theta(\mathbf{x})$. With access to latent variables $\mathbf{z}$, using the Chain rule, the marginal distribution over observed variables $\mathbf{x}$ can be rewritten as

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z}. \tag{1}$$

3 Variational Autoencoder (VAE)

Variational Autoencoder (VAE) is structurally similar to AE: both has a decoder network and an encoder network. AE learns a *function* to approximate the latent representation based on input data, by mapping each input $\mathbf{x}$ to a latent encoding vector $\mathbf{z}$ of fixed size. Because the learning is restricted to a particular dataset, the learned function form may be discontinuous or meaningless in some places corresponding to a never-seen input. This may be sufficient when the goal is data reconstruction, but is clearly undesirable when the goal is to explain the underlying generative process, encapsulating all possible observations, or to generate novel observations. VAE is therefore proposed to further regularize the representation of the data for more meaningful generation. It tries to amend this problem by approximate the true posterior with a probability distribution, as opposed to a single function. i.e. $p_\theta(\mathbf{z}|\mathbf{x})$. θ denotes network parameters.

In the encoder, VAE swaps the single-layer bottleneck in AE with multiple layers parameterizing the probability distribution. Consequently, in the decoder, a latent representation is sampled from the learned distribution, from which the input is reconstructed based on the likelihood distribution $p_\theta(\mathbf{x}|\mathbf{z})$ using the decoder network. Ideally, the reconstructed data will be identical to the input. With access to the prior distribution $p_\theta(\mathbf{z})$, the network could then be used to compute the joint distribution as well as marginal data likelihood according to eq. (1). A latent variable model $p_\theta(\mathbf{x}, \mathbf{z})$ with distributions parameterized by neural networks is termed a *deep latent variable model* (DLVM), and VAE is one of such models. The usage of neural networks permits approximation of potentially very complex marginal data likelihoods.

3.1 Approximate & Variational Inference

One issue immediately arises as a result of DLVM is the intractability of marginal likelihood computation. Eq. (1) lacks a general analytic solution or efficient estimators due to the integral. Importantly, this means the approximated posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is also intractable, as it is related to the marginal likelihood through Bayes' rule:

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})}.$$

Approximate inference is adopted to address this problem. Specifically, as the exact posterior (and marginal likelihood) is intractable to compute, approximate inference allows approximation of these distributions. In the case of VAE, the *amortized variational inference* technique is used in the encoder so it considers a more restricted set of known distributions with good computational properties and

learns one that is closest to the true distribution in terms of KL divergence (see section 3.2 below for details). The family of variational distributions is usually (multivariate) Gaussian. i.e.

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

Furthermore, Variational parameters ϕ are shared across all datapoints (hence "amortized"). Crucially, variation inference acts like a regularization force: it encourages the posterior distribution to be as closed to a standard Gaussian as possible, so the latent space is continuous and meaningful for data generation.

The resultant setup of VAE is illustrated in Figure 1: the inference model (encoder) with parameters ϕ (including both weights and biases of the network) approximates $p_\theta(\mathbf{z}|\mathbf{x})$ using a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$; the generative model (decoder) parameterized by θ approximates $p_\theta(\mathbf{x}|\mathbf{z})$.

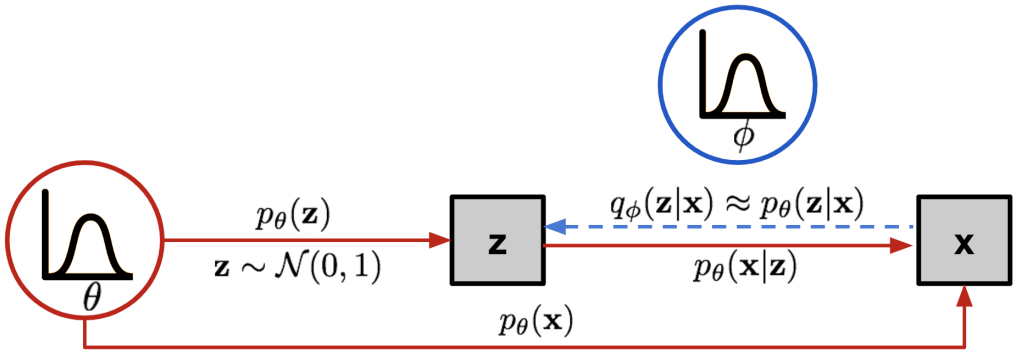


Figure 1: The graphical model of a vanilla Variational Autoencoder. The solid lines correspond to the generative distribution $p_\theta(\cdot)$. The dashed line indicates the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$, which approximates the (intractable) posterior $p_\theta(\mathbf{z}|\mathbf{x})$. Aopted from Weng [2018].

3.2 Forward versus Backward KL Divergence

To quantify the distance from the current approximate $q_\phi(\mathbf{z}|\mathbf{x})$ to the target distribution $p_\theta(\mathbf{z}|\mathbf{x})$, the Kullback-Leibler (KL) divergence appears to be the natural choice. Since KL divergence is not symmetric, there are two options for measuring the "closeness" between the true distribution P and the variational distribution Q: forward KL $D_{KL}(P||Q)$, and reversed KL $D_{KL}(Q||P)$. When *forward* KL is small, Q will attempt to provide a large support for the P. When it is not of the right distribution class, some areas might be incorrectly covered by Q. Thus minimizing forward KL may result in false positives in the approximation. On the other hand, when *reversed* KL is small, Q has a small support. It may approximate some areas of the true distribution very well by sacrificing all effort in other areas (i.e. $Q(Z) = 0$ for some Z). Thus minimizing reverse KL may lead to false negatives in the approximation. As the goal of variational inference in VAE is to fit at least some portion of the true posterior accurately, usually reversed KL is used. i.e. using a network parameterized by ϕ to approximate the posterior of a (true) latent model parameterized by θ, the encoder part of VAE attempts to minimize $D_{KL}(q_\phi(z|x)||p_\theta(z|x))$.

Unsurprisingly, reversed KL could lead to failure to comprehensively approximate certain distributions. e.g. when P is a mixture of Gaussians, a vanilla VAE approximating it with a multivariate Gaussian Q may be only able to capture one of the components. We will show that this is indeed the case in the next section on experiments.

3.3 Evidence Lower Bound (ELBO)

Adopted from other variational methods, the optimization objective of VAE is the evidence lower bound (ELBO). For an arbitrary parameterization of the inference model $q_\phi(\mathbf{z}|\mathbf{x})$, we have:

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x})] = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right] = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right]$$

Splitting the product term inside expectation into two:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right] \\ &= \mathcal{L}_{\theta,\phi}(x) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))\end{aligned}$$

The first term $\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right]$ on the right hand side is the variational lower bound ELBO. Note that because KL divergence is non-negative,

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(x) &= \log p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) &(2)\\ &\leq \log p_\theta(\mathbf{x}). &(3)\end{aligned}$$

i.e. the ELBO is a lower bound on the data log-likelihood. Thus it justifies the reason for VAE to optimize with respect to ELBO using SGD methods, as the marginal data likelihood is guaranteed to be at least as good as ELBO.

Rearranging the terms in eq. (2),

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \log p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) \\ &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}) d\mathbf{z} + \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{z}|\mathbf{x}) d\mathbf{z} - \int q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] &(4)\end{aligned}$$

The right hand side of eq. (4) can be approximated using Monte Carlo methods. Specifically, the Monte Carlo estimator $\tilde{\mathcal{L}}_{\theta,\phi}(\mathbf{x})$ is

$$\tilde{\mathcal{L}}_{\theta,\phi}(\mathbf{x}) = \log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \quad (5)$$

The three terms could be seen as three forces that drives the variational inference: $\log p_\theta(\mathbf{x}|\mathbf{z})$ encourages better likelihood fit but could potentially lead to overfit of data; $\log p_\theta(\mathbf{z})$ encourages fit to the prior but may cause posterior collapse; $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log q_\phi(\mathbf{z}|\mathbf{x})]$, the entropy of samples drawn from the posterior, tries to push variational samples apart while also avoiding posterior collapse (see Figure 2).

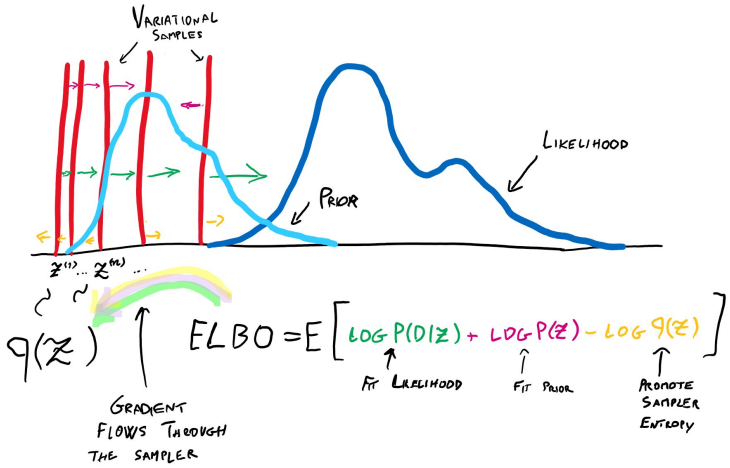


Figure 2: Illustration of the components of ELBO, adopted from @LucaAmb [Ambrogioni, 2020].

3.4 Reparameterization Trick

If we were to proceed with a naive implementation of VAE - sampling a $\mathbf{z}$ directly from $q_\phi(\mathbf{z}|\mathbf{x})$ (e.g. the multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ in vanilla VAE), in general the gradient $\nabla_\phi q_\phi(\mathbf{z}|\mathbf{x})$

is not attainable. Thus there is no way to perform backpropagation through the entire network, as this particular sampling step is not guaranteed to be differentiable w.r.t. ϕ.

To address this problem, a reparameterization trick could be introduced to allow ELBO to be differentiated w.r.t. $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. In particular, we can write $\mathbf{z} = \mathbf{g}(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})$, where $\mathbf{g}(\cdot)$ is a differentiable function, and $\boldsymbol{\epsilon}$ is a random variable with some predetermined prior distribution $p(\boldsymbol{\epsilon})$ independent of both $\mathbf{x}$ and $\boldsymbol{\phi}$. For instance, in the univariate Gaussian case, while previously $\mathbf{z}$ is drawn from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, by introducing a noise parameter $\boldsymbol{\epsilon}$, $\mathbf{z}$ can be rewritten as $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$. This reparameterization effectively loads all stochastivity on $\boldsymbol{\epsilon}$, which follows a fixed distribution that doesn't require any learning. As a result, we could propagate errors through the network parameterized by ϕ by taking $\nabla_{\phi}\mathbf{g}(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})$.

As a result of applying the reparameterization trick to eq. (4), we have

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{p(\epsilon)}[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})], \text{ where } \mathbf{z} = \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}$$

3.5 Derivation of Algorithm from Kingma and Welling [2019]

In this section, we will derive the key algorithm of VAE, Algorithm 1 from the monograph Kingma and Welling [2019]. Note that eq. (5) is exactly line 9. For clarity, we first look at the case when the inference (and generative) model assumes a isotropic Gaussian prior.

Algorithm 1: Computation of unbiased estimate of single-datapoint ELBO for example VAE with a full-covariance Gaussian inference model and a factorized Bernoulli generative model. $\mathbf{L}_{mask}$ is a masking matrix with zeros on and above the diagonal, and ones below the diagonal.

Data:

$\mathbf{x}$: a datapoint, and optionally other conditioning information
$\boldsymbol{\epsilon}$: a random sample from $p(\boldsymbol{\epsilon}) = \mathcal{N}(0, \mathbf{I})$
$\boldsymbol{\theta}$: Generative model parameters
$\boldsymbol{\phi}$: Inference model parameters
$q_\phi(\mathbf{z}|\mathbf{x})$: Inference model
$p_\theta(\mathbf{z}|\mathbf{x})$): Generative model

Result:

$\tilde{\mathcal{L}}$:unbiased estimate of the single-datapoint ELBO $\mathcal{L}_{\theta,\phi}(\mathbf{x})$

1 $(\boldsymbol{\mu}, \log \boldsymbol{\sigma}, \mathbf{L}') \leftarrow \text{EncoderNeuralNet}_\phi(\mathbf{x})$
2 $\mathbf{L} \leftarrow \mathbf{L}_{mask} \odot \mathbf{L}' + \text{diag}(\boldsymbol{\sigma})$
3 $\boldsymbol{\epsilon} \sim \mathcal{N}(o, \mathbf{I})$
4 $\mathbf{z} \leftarrow \mathbf{L}\boldsymbol{\epsilon} + \boldsymbol{\mu}$
5 $\tilde{\mathcal{L}}_{\text{logqz}} \leftarrow -\sum_i(\frac{1}{2}(\epsilon_i^2 + \log(2\pi) + \log \sigma_i^2))_i$ // $q_\phi(\mathbf{z}|\mathbf{x})$
6 $\tilde{\mathcal{L}}_{\text{logpz}} \leftarrow -\sum_i(\frac{1}{2}(z_i^2 + \log(2\pi)))$ // $p_\theta(\mathbf{z})$
7 $\mathbf{p} \leftarrow \text{DecoderNeuralNet}_\theta(\mathbf{z})$
8 $\tilde{\mathcal{L}}_{\text{logpx}} \leftarrow \sum_i(x_i \log p_i + (1 - x_i)\log(1 - p_i))$ // $p_\theta(\mathbf{z}|\mathbf{x}))$
9 $\tilde{L} = \tilde{\mathcal{L}}_{\text{logpx}} + \tilde{\mathcal{L}}_{\text{logpz}} - \tilde{\mathcal{L}}_{\text{logqz}}$

3.5.1 Posterior (line 5)

In the case of isotropic Guassian posteriors, after reparameterization, we have

$$\begin{aligned}
\boldsymbol{\epsilon} &\sim \mathcal{N}(0, \mathbf{I}) \\
(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) &\leftarrow \text{EncoderNeuralNet}_\phi(\mathbf{x}) \\
\mathbf{z} &= \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \\
q_\phi(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^{2(i)}\mathbf{I}) = \Pi_i \mathcal{N}(z_i; \mu_i, \sigma_i^2)
\end{aligned}$$

As illustrated above, the random variable $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ can be reexpressed using a differentiable function $\mathbf{g}(\cdot)$ and a random variable $\boldsymbol{\epsilon}$. Under this change of variable in the probability density

function, we have

$$p(\boldsymbol{\epsilon}) = q_{\boldsymbol{\phi}}(\mathbf{g}(\boldsymbol{\epsilon}))\left|\det\left[\frac{\partial \mathbf{g}(\boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}}\right]\right|$$

Since $\frac{\partial \mathbf{g}(\boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}} = \frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}}$ and $q_{\boldsymbol{\phi}}(\mathbf{g}(\epsilon)) = q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, taking logarithm of both sides:

$$\log p(\boldsymbol{\epsilon}) = \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log\left|\det\left[\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}}\right]\right|$$

Rearranging terms, we have

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \log p(\boldsymbol{\epsilon}) - \log\left|\det\left[\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}}\right]\right|$$

Let J be dimensionality of $\mathbf{z}$. When the posterior is isotropic Gaussian, the Jacobian matrix ($\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}}$) is diagonal. Thus the log of the determinant is simply the sum of the log of all diagonal terms, i.e.

$$\log\left|\det\left[\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}}\right]\right| = \sum_{i=1}^{J}\log\sigma_i = \frac{1}{2}\sum_{i=1}^{J}\log\sigma_i^2. \tag{6}$$

Additionally,

$$\log p(\boldsymbol{\epsilon}) = \sum_{i=1}^{J}\log\mathcal{N}(\epsilon_i; 0, 1) = \sum_{i=1}^{J}\log\left[(2\pi)^{-\frac{1}{2}}e^{-\frac{1}{2}\epsilon_i^2}\right] = -\sum_{i=1}^{J}\frac{1}{2}(\epsilon_i^2 + \log(2\pi)) \tag{7}$$

Putting eq. (6) and (7) all together, the Monte Carlo estimate of $\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = -\sum_{i=1}^{J}\frac{1}{2}(\epsilon_i^2 + \log(2\pi) + \log\sigma_i^2). \tag{8}$$

Only minor modifications are needed to extend this to the full-covariance case. In addition to standard deviation terms, we also need covariances of elements in $\mathbf{z}$. Since Σ is symmetric, learning a lower (or upper) triangle $\mathbf{L}$ would be sufficient given Cholesky decomposition $\boldsymbol{\Sigma} = L\mathbf{L}^T$. One way to obtain $\mathbf{L}$ for the inference model would be to further learn a matrix $\mathbf{L}'$ of the appropriate size, then to combine it with the learned standard deviations - for instance, with a mask $\mathbf{L}_{mask}$ that extracts covariances above the diagonal, and replace diagonal terms with $\boldsymbol{\sigma}$. With reparameterization, this gives the following alternative/additional relations:

$$\begin{aligned}(\boldsymbol{\mu}, \log\boldsymbol{\sigma}, \mathbf{L}') &\leftarrow \text{EncoderNeuralNet}_{\phi}(\mathbf{x})\\ \mathbf{L} &\leftarrow \mathbf{L}_{mask} \odot \mathbf{L}' + \text{diag}(\boldsymbol{\sigma})\\ \mathbf{z} &= \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}\\ q_{\phi}(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\end{aligned}$$

The Jacobian of the transformation from $\boldsymbol{\epsilon}$ to $\mathbf{z}$ is simply $\mathbf{L}$, meaning the log determinant is again the sum of the log of all diagonal terms [1]. The rest of the computation of $\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ follows exactly as above. We obtain eq. (8) in both isotropic and full-covariance cases, justifying line 5.

3.5.2 Prior (line 6)

The estimates of $\log p_{\theta}(\mathbf{z})$ is straightforward in both cases (isotropic and full-covariance Gaussian):

$$\log p_{\theta}(\mathbf{z}) = \sum_{i=1}^{J}\log\mathcal{N}(z_i; 0, 1) = \sum_{i=1}^{J}\log\left[(2\pi)^{-\frac{1}{2}}e^{-\frac{1}{2}z_i^2}\right] = -\sum_{i=1}^{J}\frac{1}{2}(z_i^2 + \log(2\pi)) \tag{9}$$

[1] The diagonal terms σ_i's here do not necessarily correspond to standard deviation.

3.5.3 Likelihood (line 8)

Finally, $\log p_\theta(\mathbf{x}|\mathbf{z})$ depends on the specific generative model used by the VAE. Algorithm 2 is based on a Bernoulli generative model with parameter p_i. For a Gaussian generative model. i.e. $\mathbf{x}|\mathbf{z} = z \sim \mathcal{N}(\boldsymbol{\mu}_{\theta,z}, \boldsymbol{\Sigma}_{\theta,z})$, line 7 is replaced by $\boldsymbol{\mu}_{\theta,z}, \boldsymbol{\Sigma}_{\theta,z} \leftarrow \text{DecoderNeuralNet}_\theta(\mathbf{z})$, and line 8 is modified so that

$$\begin{aligned}\log p_\theta(\mathbf{x}|\mathbf{z}) &= \log\left[(2\pi)^{-\frac{J}{2}}|\boldsymbol{\Sigma}_{\theta,z}|^{-\frac{1}{2}}\exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{\theta,z})^T\boldsymbol{\Sigma}_{\theta,z}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\theta,z}))\right] \\ &= -\frac{1}{2}\left[(\mathbf{x}-\boldsymbol{\mu}_{\theta,z})^T\boldsymbol{\Sigma}_{\theta,z}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\theta,z}) + J\log(2\pi) + \log|\boldsymbol{\Sigma}_{\theta,z}|\right]\end{aligned}$$

In the isotropic Gaussian case, this reduces to

$$\log p_\theta(\mathbf{x}|\mathbf{z}) = \sum_{i=1}^{J}\log\left[(2\pi)^{-\frac{1}{2}}\sigma_i^{-1}e^{-\frac{1}{2\sigma_i^2}(x_i-\mu_{\theta,z,i})^2}\right] = -\frac{1}{2}\sum_{i=1}^{J}\left[\frac{(x_i-\mu_{\theta,z,i})^2}{\sigma_i^2} + \log(2\pi) + \log\sigma_i^2\right],$$

where $\mu_{\theta,z,i}$ is the i^{th} element of $\boldsymbol{\mu}_{\theta,z}$, and σ_i^2 is the i^{th} diagonal entry of $\boldsymbol{\Sigma}_{\theta,z}$.

3.6 Alternative Monte Carlo Estimate of ELBO w.r.t single datapoint

Recall we derived

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{p(\epsilon)}\left[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right], \text{where } \mathbf{z} = \boldsymbol{\sigma}\odot\boldsymbol{\epsilon} + \boldsymbol{\mu}.$$

Furthermore,

$$\mathbb{E}_{p(\epsilon)}[\log q_\phi(\mathbf{z}|\mathbf{x})] = \mathbb{E}_{p(\epsilon)}\left[-\frac{1}{2}\sum_{i=1}^{J}(\epsilon_i^2 + \log(2\pi) + \log\sigma_i^2)\right] = -\frac{1}{2}\sum_{i=1}^{J}\mathbb{E}_{p(\epsilon)}[\epsilon_i^2] - \frac{1}{2}\sum_{i=1}^{J}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{J}\log\sigma_i^2.$$

Assuming (i.i.d.) standard Gaussian noise, in the isotropic Gaussian case,

$$\mathbb{E}_{p(\epsilon)}[\epsilon_i^2] = Var(\epsilon_i) + \left(\mathbb{E}_{p(\epsilon)}[\epsilon_i]\right)^2 = 1 + 0 = 1.$$

Thus,

$$\mathbb{E}_{p(\epsilon)}[\log q_\phi(\mathbf{z}|\mathbf{x})] = -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{J}(1+\log\sigma_i^2).$$

Additionally,

$$\begin{aligned}\mathbb{E}_{p(\epsilon)}[\log p_\theta(\mathbf{z})] &= \mathbb{E}_{p(\epsilon)}\left[-\frac{1}{2}\sum_{i=1}^{J}(z_i^2 + \log(2\pi))\right] \\ &= -\frac{1}{2}\sum_{i=1}^{J}\mathbb{E}_{p(\epsilon)}[(\mu_i + \sigma_i\cdot\epsilon_i)^2] - \frac{1}{2}\sum_{i=1}^{J}\log(2\pi) \\ &= -\frac{1}{2}\sum_{i=1}^{J}(\mu_i^2 + \sigma_i^2\mathbb{E}_{p(\epsilon)}[\epsilon_i^2] + 2\mu_i\sigma_i\mathbb{E}_{p(\epsilon)}[\epsilon_i]) - \frac{J}{2}\log(2\pi) \\ &= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{J}(\mu_i^2 + \sigma_i^2).\end{aligned}$$

It then follows that

$$\mathcal{L}_{\theta,\phi}(x_j) \simeq \frac{1}{2}\sum_{i=1}^{J}(1 + \log((\sigma_i^{(j)})^2) - (\mu_i^{(j)})^2 - (\sigma_i^{(j)})^2) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(x_j|z_j)\right],$$

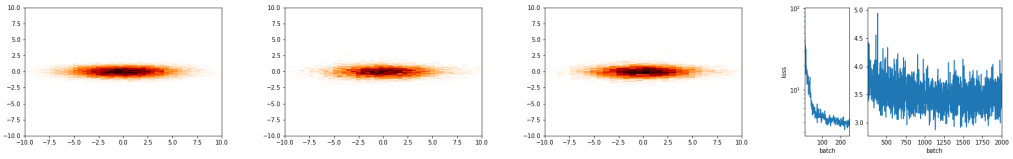
where the expected reconstruction error $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(x_j|z_j)\right]$ w.r.t each data point x_j can be estimated by sampling multiple z_j's from the posterior. This estimator was first introduced in Kingma and Welling [2014]. Here we have demonstrated the full derivation.

4 Experiments

In this section, we present a few experiments to illustrate the performance of a vanilla VAE. For all experiments, the same architecture is used: both the encoder and the decoder consist of 2 layers of [128, 128] densely connected neurons with ReLU activation. The first layer is shared. The second layer(s) approximates distribution parameters. In each experiment, synthetic data was generated to train the VAE, with an isotropic and/or full-covariance inference model (the generative model is chosen correspondingly). The first two experiments used 20000 training samples and VAE was trained for 50 epochs. The rest used 40000 training examples and 200 training epochs. All training was done end-to-end. Unless otherwise stated, the latent dimension is 2 (i.e. $\mathbf{z} = [z_1 z_2]^T$) and the training batch size was set to 32, with learning rate equal to $1e^{-3}$. Testing sets are of half of the size of the respective training set.

4.1 Isotropic Gaussian

In the first experiment, we trained our vanilla VAE on a dataset generated by an isotropic Gaussian process (see Fig. 3a). The inference and generative models both assumed a diagonal covariance matrix. After training, we tested (1) performance of the decoder (generative model) by sampling from the latent prior; (2) ability for the entire model to reconstruct a test dataset. As expected, VAE performed well, in terms of both data generation (Fig. 3b) and reconstruction (Fig. 3c). Training quickly converged after around 250 batches (Fig. 3d).



(a) Sample input. (b) Generative samples. (c) Reconstruction. (d) Batch loss.

Figure 3: Results from VAE when the true generative process is isotropic Gaussian: $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_2), \boldsymbol{\epsilon} \sim \mathcal{N}(0, 0.2 \cdot \mathbf{I}_2), \mathbf{x}|\mathbf{z} \sim \begin{bmatrix} 3 & 0 \\ 0 & 1/3 \end{bmatrix} \mathbf{z} + \boldsymbol{\epsilon}$.

4.2 Non-isotropic Gaussian

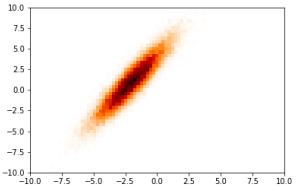
In the second experiment, we trained our vanilla VAE on a dataset generated by a non-isotropic Gaussian process (see Fig. 4a). The inference and generative models first assumed a diagonal covariance matrix. Again, VAE seems to have learned a fairly accurate picture of the ground truth latent process, in terms of both data generation (Fig. 4b) and reconstruction (Fig. 4c). With a full covariance matrix, VAE performed slightly better in both tests (Fig. 4d, 4e).

4.3 Figure 8

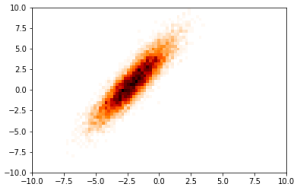
Next, we tried a dataset that cannot be precisely described by a Gaussian process (see Fig. 5a). The inference and generative models first assumed a diagonal covariance matrix, and then a full covariance. VAE completely failed to capture the latent dynamics, and was only able to learn a isotropic Gaussian distribution with either a diagonal (Fig. 5b, 5c), or a full covariance model (Fig. 5d, 5e).

4.4 Three Blobs

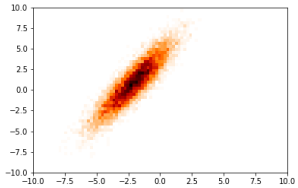
In another experiment, we used synthetic data that has a "three blob" structure (Fig. 6a), which is again hard to approximate with any Gaussian prior. As expected, VAE wasn't able to learn the generative process underlying this structure with either a diagonal (Fig. 6b, 6c) or a full-covariance model (Fig. 6d, 6e), as the results were reminiscent of an isotropic Gaussian.



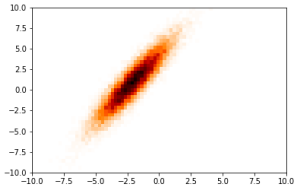
(a) Sample input.



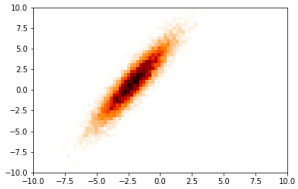
(b) Generative samples (diagonal).



(c) Reconstruction (diagonal).

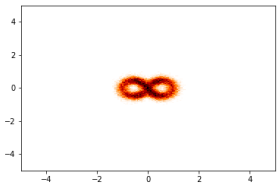


(d) Generative samples (full).

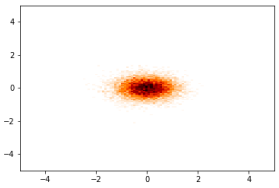


(e) Reconstruction (full).

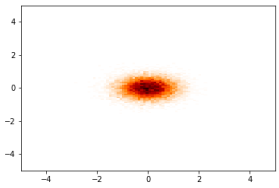
Figure 4: Results from VAE when the true generative process is non-isotropic Gaussian: $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_2), \boldsymbol{\epsilon} \sim \mathcal{N}(0, 0.2 \cdot \mathbf{I}_2), \mathbf{x}|\mathbf{z} \sim \begin{bmatrix} \cos(\pi/3) & -\sin(\pi/3) \\ \sin(\pi/3) & \cos(\pi/3) \end{bmatrix} \cdot \begin{bmatrix} 3 & 0 \\ 0 & 1/3 \end{bmatrix} \cdot \mathbf{z} + \boldsymbol{\epsilon}$.



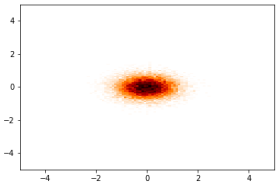
(a) Sample input.



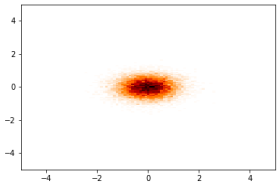
(b) Generative samples (diagonal).



(c) Reconstruction (diagonal).



(d) Generative samples (full).

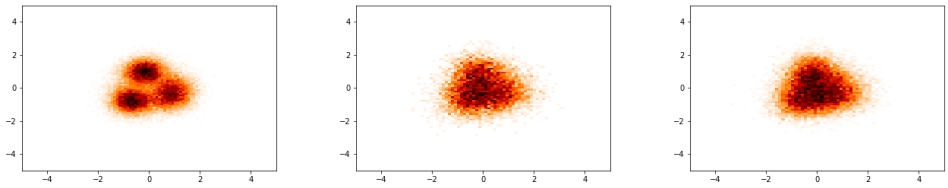


(e) Reconstruction (full).

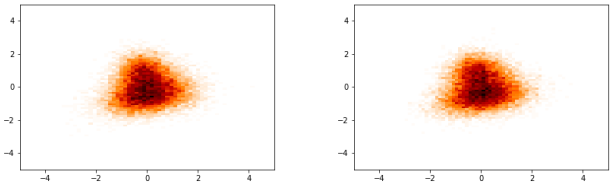
Figure 5: Results from VAE w.r.t. a figure "8" dataset: $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_2), \boldsymbol{\epsilon} \sim \mathcal{N}(0, 0.2 \cdot \mathbf{I}_2), u(\mathbf{z}) = (0.6 + 1.8 \cdot \Phi(\mathbf{z})), \mathbf{x}|\mathbf{z} \sim \begin{bmatrix} \frac{\sqrt{2}}{2} \cdot \frac{\cos(u(\mathbf{z}))}{\sin(u(\mathbf{z}))^2+1} \\ \sqrt{2} \cdot \frac{\cos(u(\mathbf{z}))\sin(u(\mathbf{z}))}{\sin(u(\mathbf{z}))^2+1} \end{bmatrix} \mathbf{z} + \boldsymbol{\epsilon}$. $\Phi(\mathbf{z})$ denotes the Gaussian CDF.

4.5 Mixture of Gaussians

Finally, we trained VAE using a mixture of three Gaussians (Fig. 7a). i.e. $i \in 1, 2, 3$. VAE actually achieved decent data generation (Fig. 7b, 7e) and reconstruction (Fig. 7c, 7f). However, this is not guaranteed: Fig. 7d shows a case during training where VAE failed miserably, likely because the variational distribution only managed to fit one of the peaks. This is consistent with our prediction from section 3.2.



(a) Sample input. (b) Generative samples (diagonal). (c) Reconstruction (diagonal).



(d) Generative samples (full). (e) Reconstruction (full).

Figure 6: Results from VAE w.r.t. a "3 blobs" dataset: $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_2), \boldsymbol{\epsilon} \sim \mathcal{N}(0, 0.2 \cdot \mathbf{I}_2), u(\mathbf{z}) = \frac{2\pi}{1+\exp(-\frac{1}{2}\pi\mathbf{z})}, t(u) = 2 \cdot \tanh(10 \cdot u - 20 \cdot \lfloor u/2 \rfloor - 10) + 4 \cdot \lfloor u/2 \rfloor + 2, \mathbf{x}|\mathbf{z} \sim \begin{bmatrix} \cos(t(u(\mathbf{z})) \\ \sin(t(u(\mathbf{z})) \end{bmatrix} \mathbf{z} + \boldsymbol{\epsilon}$.

5 Conclusion

We introduced the basic Variational Autoencoder (VAE) in this paper by outlining its general structure and proving some key theoretical results related to VAE. We derived a algorithm to compute the optimization objective of VAE, namely the variational lower bound. Lastly, we conducted a series of experiments to shed light on the scenarios where VAE succeeds and fails to reconstruct or generate synthetic data.

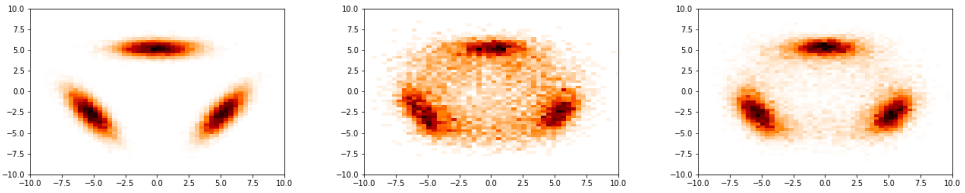
References

Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

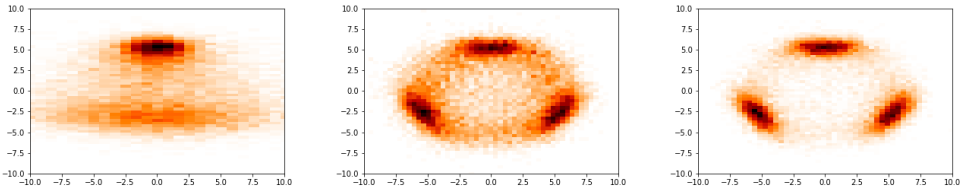
Diederik P. Kingma and M. Welling. An introduction to variational autoencoders. *Found. Trends Mach. Learn.*, 12:307–392, 2019.

Lilian Weng. From autoencoder to beta-vae, Aug 2018. URL `https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html#vae-variational-autoencoder`.

Luca Ambrogioni, Dec 2020. Tweet: "The dynamics of stochastic variational inference is a balance of three forces. One pushes the samples towards the likelihood, the other towards the prior and the third pushes the samples apart to avoid the collapse into the MAP estimator.".



(a) Sample input. (b) Generative samples (diagonal). (c) Reconstruction (diagonal).



(d) Generative samples (diagonal, failed). (e) Generative samples (full). (f) Reconstruction (full).

Figure 7: Results from VAE when the true generative process is a mixture of Gaussians: $\mathbf{y}_i \sim \mathcal{N}\left(0, \begin{bmatrix} 3 & 0 \\ 0 & 1/3 \end{bmatrix}\right), z \sim U[0,1], \mathbf{x}_1 = \mathbf{y}_1 + \begin{bmatrix} 0 \\ 3\sqrt{3} \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} \cos(2\pi/3) & -\sin(2\pi/3) \\ \sin(2\pi/3) & \cos(2\pi/3) \end{bmatrix} \mathbf{y}_2 + \begin{bmatrix} -3\sqrt{3} \\ -3\sqrt{3}/2 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} \cos(\pi/3) & -\sin(2/3) \\ \sin(\pi/3) & \cos(\pi/3) \end{bmatrix} \mathbf{y}_3 + \begin{bmatrix} -3\sqrt{3} \\ -3\sqrt{3}/2 \end{bmatrix}, \mathbf{x}|\mathbf{z} = \mathbf{x}_{\lceil 3z \rceil}$.