Variational Auto Encoders

Federico Pavesi

March 12, 2024

1 Variational Auto Encoders

Variational Auto Encoders [8] [10] are a specific class of generative models which take advantage of both neural networks and Variational Bayes. In particular, this approach offers a way to learn both a deep latent-variable model and its corresponding inference model using stochastic gradient descent[7].

We are in the setting in which we have a dataset X composed by observations $\mathbf{x}$ belonging to a space $\mathcal{X}$, and we would like to estimate the distribution of any $\mathbf{x}$, $p(\mathbf{x})$, using information contained in dataset X. We suppose there exists a hidden data generating process which is random and involves a latent variable $\mathbf{z} \in \mathcal{Z}$, distributed as $p(\mathbf{z})$. This way we now have two random variables which interaction is captured by their joint distribution $p(\mathbf{x}, \mathbf{z})$. In a slightly different perspective, we can image the process of generating an observation $\mathbf{x}$ starts from a sample from $p(\mathbf{z})$, $\mathbf{z}$, which is then used to generate $\mathbf{x}$ via the conditional $p(\mathbf{x}|\mathbf{z})$. It is now clear that, if we have accesso to the $p(\mathbf{x}|\mathbf{z})$ and $p(\mathbf{z})$, we can replicate the data generating process. Throughout the following discussion, we shall refer to $\mathbf{x}$ and it is understood it can interchangeably represent a single observation or a set of observations (including the complete dataset). This allows to reason in terms of batch of data without having to rewrite any equation.

If we employ a Bayesian perspective of this problem, we can say that $p(\mathbf{z})$ is our prior distribution which in some sense, help govern the data distribution [3]. The likelihood task is represented by the estimation of $p(\mathbf{x}|\mathbf{z})$, while the inference task is to estimate the posterior $p(\mathbf{z}|\mathbf{x})$.

Let's first put a focus on the inference task: unfortunately, we do not have access directly to $p(\mathbf{z}|\mathbf{x})$ but, following the Variational Bayes perspective, we can choose an approximate posterior $q(\mathbf{z}|\mathbf{x})$, belonging to class $\mathcal{Q}$,

which we hypothize it is close to the true posterior. In classic Variational Bayes, the solution is represented by choosing a class of known distributions (for example Gaussians), parametrized by ϕ, and estimate ϕ via Mean Field approximation.

Another fundamental quantity we need for this process again comes from Variational Inference: the Evidence Lower Bound (ELBO). We can derive it using Jensen Inequality. The log likelihood of any datapoint $\mathbf{x}$ is

$$\begin{aligned}\log \mathcal{L}(\mathbf{x}) = \log p(\mathbf{x}) &= \log \left(\int p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \right) \\ &= \log \left(\int \frac{q(\mathbf{z})}{q(\mathbf{z})} p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \right) \\ &= D_{KL}\left[q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}) \right] + \mathcal{F}(\mathbf{x}) \end{aligned} \tag{1}$$

where

$$\begin{aligned}\mathcal{F}(\mathbf{x}) &= \mathbb{E}_q \left[-\log q(\mathbf{z}) + \log p(\mathbf{x}, \mathbf{z}) \right] \\ &= -D_{KL}\left[q(\mathbf{z}) \parallel p(\mathbf{z}) \right] + \mathbb{E}_q \left[\log p(\mathbf{x}|\mathbf{z}) \right] \end{aligned} \tag{2}$$

is the ELBO, also called Free Energy [10]. Because $D_{KL}\left[q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}) \right]$ is always non-negative, we can write

$$\log \mathcal{L}(\mathbf{x}) \geq \mathcal{F}(\mathbf{x}) \tag{3}$$

thus the Free Energy lower bounds the log likelihood: whenever it is maximized, we are approximately maximizing the likelihood objective.

Usually, in the context of variational autoencoders, the interest is not on an uncondintional approximate posterior $q(\mathbf{z})$, but on a conditional one $q(\mathbf{z}|\mathbf{x})$, such that we can 'assign' to each observation a latent code. Plugging the conditional in (1) it is simple to obtain a version of (4) with the conditional posterior. To simplify learning process, we can parametrize both the posterior and the prior. If we denote parameters of the posterior as ϕ and the parameters of the prior as θ, free energy can be rewritten as

$$\mathcal{F}(\mathbf{x}) = -D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}) \right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \right] \tag{4}$$

Maximizing $\mathcal{F}(\mathbf{x})$, even with parametrization, it's a complex task and rarely there exist tractable closed form solutions. A promising approach is represented by optimization via Stochastic Gradient Descent (SGD).

In variational auto encoders, the idea is to parametrize $q_\phi(\mathbf{z}|\mathbf{x})$ via a neural network, called the Encoder and, similarly, to parametrize $p_\theta(\mathbf{x}|\mathbf{z})$ via another neural network called the Decoder. Notice with this formulation parameters ϕ, θ are shared among observations $\mathbf{x} \in X$ [7]. In the original formulation in [8], the prior $p(\mathbf{z})$ is chosen to be an isotropic standard gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Minimizing $\mathcal{F}(\mathbf{x})$ using SGD is not as straightforward as it might seem. Unfortunately, the individual datapoint gradient $\nabla_{\phi,\theta}\mathcal{F}(\mathbf{x})$ is, in general, intractable[7]. While the gradient w.r.t. Decoder parameters θ is relatively simple to estimate

$$\begin{aligned}
\nabla_\theta \mathcal{F}(\mathbf{x}) &= \nabla_\theta \left(D_{KL} \left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}) \right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \right] \right) \\
&= \nabla_\theta \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right]
\end{aligned} \tag{5}$$

We can switch integration and derivation because the expectation does not depend on θ, this quantity can be simply estimated via Monte Carlo approximation using a random sample from $q_\phi(\mathbf{z}|\mathbf{x})$. Instead, this passage in general cannot be done for the gradient w.r.t. ϕ

$$\begin{aligned}
\nabla_\phi \mathcal{F}(\mathbf{x}) &= \nabla_\phi \left(D_{KL} \left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}) \right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \right] \right) \\
&= \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] \\
&\neq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\nabla_\phi \left(\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right) \right]
\end{aligned} \tag{6}$$

This limitation can be solved in the case we have continuous latent variable via a change of variable called the *reparametrization trick*.

If latent random variable $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ can be expressed as some differentiable and invertible transformation of another random variable $\boldsymbol{\epsilon}$, independent from $\mathbf{x}$ or ϕ

$$\mathbf{z} = g(\boldsymbol{\epsilon}, \phi, \mathbf{x}) \tag{7}$$

It is possible to rewrite expectation in terms of $\boldsymbol{\epsilon}$ [8][7]

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[f(\mathbf{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(\mathbf{z})\right] \tag{8}$$

For any well defined function f where $\mathbf{z}$ is in its reparametrized form. We are now in the situation to estimate the gradient for (6) as

$$\begin{aligned}
\nabla_\phi \mathcal{F}(\mathbf{x}) &= \nabla_\phi \left(D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})\right]\right) \\
&= \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\
&= \nabla_\phi \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\
&= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\phi \left(\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right)\right]
\end{aligned} \tag{9}$$

which can be easily estimated via Monte Carlo approximation using a random sample from $q_\phi(\mathbf{z}|\mathbf{x})$ as for θ. Henceforth, for simplicity, we shall keep the notation $q_\phi(\mathbf{z}|\mathbf{x})$, keeping in mind it is subjected to reparametrization (which anyway, does not affect this quantity).

In some cases in which we know the datapoint posterior distribution (for example, in [8] it is a Gaussian) it is possible to evaluate $D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right]$ in closed form. In all other cases, we need to estimate density $\log q_\phi(\mathbf{z}|\mathbf{x})$ subject to reparametrization

$$\log q_\phi(\mathbf{z}|\mathbf{x}) = \log p(\boldsymbol{\epsilon}) - \log \left| \det \left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} \right) \right| \tag{10}$$

Where the right hand side of term is absolute value of logarithm of the determinant of the derivative of $\mathbf{z} = g(\boldsymbol{\epsilon})$ w.r.t $\boldsymbol{\epsilon}$. Equation (10) will be especially important when we will define flow transformations.

At this point, it is simple to optimize (4) using SGD. From a practical perspective, it is to notice when we reach the optimal parameter configuration $\hat{\phi}, \hat{\theta}$ we actually obtained three important goals. First, we are maximizing the likelihood under a latent variable model, using an approximation of the posterior. Second, we can encode observations from their original space $\mathcal{X}$, to the latent space $\mathcal{Z}$ which, in most cases, it is less complex and has desirable properties. A good example is $\mathcal{X}$ as the space of images of handwritten digits and $\mathcal{Z} = \mathbb{R}^d$ [8]. Last, if $D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right]$ is sufficiently small, it means the posterior is very close to the prior $p(\mathbf{z})$, such that a we can directly sample from it and the decoder will be able to reconstruct the associated $\mathbf{x}$. This at the end allows to generate any possibly new point belonging to the data distribution.

2 Challenges of standard approach

This approach, despite all its advantages, does not come without issues. In particular, two main issues have been evidentiated in literature: optimization issues and blurriness of generative model.

Optimization issues have been evidentiated in [7][1][11] and are mainly related to the posterior. What happens is, at the beginning of neural networks training, $\log p(\mathbf{x}|\mathbf{z})$ is relatively weak, making the state $q(\mathbf{z}|\mathbf{x}) \approx p(\mathbf{z})$ attractive. This is an undesirable equilibrium because $\log p(\mathbf{x}|\mathbf{z})$, in general remains weak and it is difficult to escape from this area. The alternative proposed by [1][11] is to assign a weight β to $D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right]$ and slowly anneal it from 0 to 1 over training epochs. Free energy becomes

$$\mathcal{F}(\mathbf{x}) = \beta D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\right] \tag{11}$$

It is to notice [5] use a similar approach, but they find that a proper $\beta > 1$ allows to formulate the problem as constrained optimization, which balances latent channel capacity and independence constraints with reconstruction accuracy.

An alternative proposed in [6] is the *free bits* approach, which ensures, on average, a minimum number of bits of information is encoded per latent variable or per group of latent variables [7]. This is done by dividing latent dimensions in K disjoint groups $\{\mathbf{z}_1, \ldots, \mathbf{z}_K : \cup_{i=1}^{K} \mathbf{z}_i = \mathbf{z}\}$ and each group should encode a minimum of λ information for each $j = 1, 2, \ldots, K$

$$\mathcal{F}(\mathbf{x}) = \sum_{j=1}^{K} \max(\lambda, D_{KL}\left[q_\phi(\mathbf{z}_i|\mathbf{x}) \parallel p(\mathbf{z}_i)\right]) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\right] \tag{12}$$

This has still to be written.

3 Normalizing flows

A limitation of the classic approach of variational autoencoders has been evidentiated in [9]. In particular, it is easily understood that, because most of the time the posterior approximation $q_\phi(\mathbf{z}|\mathbf{x})$ has a simple form (for example Gaussians), the distance from the true posterior is not negligible. The solution proposed by [9] is to take advantage of normalizing flows.

Given a random variable $\mathbf{z}$ and an invertible smooth mapping $f : \mathbb{R}^d \mapsto \mathbb{R}^d$, clearly $f^{-1} \circ f(\mathbf{z}) = \mathbf{z}$. If $\mathbf{z}$ has distribution $q(\mathbf{z})$, we can write

$$q(f(\mathbf{z})) = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial f(\mathbf{z})} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1} \tag{13}$$

This fact allows to build arbitrary complex densities composing a sequence of maps $(f_k)_{k \in I}$ for $I = 1, 2, \ldots, K$ and K being the flow length.

$$\mathbf{z}_K = f_K \circ f_{K-1} \circ \ldots \circ f_1(\mathbf{z}_0) \tag{14}$$

and the density of $\mathbf{z}_K$ can be easily recovered as

$$q_K(\mathbf{z}_K) = q_0(\mathbf{z}_0) \prod_{k=1}^{K} \left| \det \frac{\partial f_K}{\partial \mathbf{z}_{k-1}} \right|^{-1} \tag{15}$$

so

$$\log q_K(\mathbf{z}_K) = \log q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_K}{\partial \mathbf{z}_{k-1}} \right| \tag{16}$$

Moreover, because of the law of the unconsciuos statistician, any expectation under $q_K(\mathbf{z}_K)$ of an arbitrary function of $\mathbf{z}_K$, $h(\mathbf{z}_K)$, can be recovered as an expectation over the original density

$$\mathbb{E}_{q_K} \left[h(\mathbf{z}_K) \right] = \mathbb{E}_{q_0} \left[h(f_K \circ f_{K-1} \circ \ldots \circ f_1(\mathbf{z}_0)) \right] \tag{17}$$

We can make an interesting connection between equation (16) and the reparametrized density (10) if we notice that what in the latter equation was defined as g can be viewed as a map belonging to a flow (for simplicity, the first map f_0), so under this fashion we can write the flow assigning $\boldsymbol{\epsilon} = \mathbf{z}_0$ and $q_{\boldsymbol{\epsilon}} = q_0$

$$\log q_K(\mathbf{z}_K) = \log q_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_K}{\partial \mathbf{z}_{k-1}} \right| \tag{18}$$

This intuition is at the base of the implementation of normalizing flows in variational autoencoders.

An intuition by [9], which was explored in depth by [2], is that if we suppose $K \to \infty$, we obtain an infinitesimal flow. In other words, we see the

case with K finite as an Euler discretization of a partial differential equation flow [2] with K steps. The partial differential equation can be written as

$$\frac{\partial}{\partial t} q_t(\mathbf{z}) = \mathcal{T}_t \left[q_t(\mathbf{z}) \right] \tag{19}$$

where $\mathcal{T}$ describes continuous-time dynamics [9] and t has been used instead of k for notation consistency with differential equations framework.

EXPAND?

4 Application

In this section we shall explore in depth variatioanl autoencoders we used in applications. Specifically, we adopted four approaches: a classic VAE with diagonal covariance of the posterior, a VAE with full posterior covariance, a VAE with planar flows and a VAE with Inverse Autoregressive Flows. Each model has a isotropic standard Gaussian prior and the decoder architecture is the same for each model.

4.1 Decoder

Because we represented the Neural Network architecture $boldx \in \mathcal{X}$ as a time series of one hot encodings of layer types and feature values, the decoder is structured as follows. Sample from latent dimension $\mathbf{z}$ is first passed through a stack of linear modules and nonlinear activations (Rectified Linear Units specifically) which produce three outputs: a vector $\mathbf{x}_0$, the beginning of the sequence, and two vectors $\mathbf{h}$, $\mathbf{c}$ which are used to initialize the hidden and cell states of a Long Short Term Memory (LSTM) module. This LSTM has the task to reconstruct the sequence of one hot encodings of layer types and feature values. For computational efficiency reasons and stability, we decided to use teacher forcing during training. Below, we propose a pseudo code for the autoregressive LSTM. Notice we set $\mathbf{h}$, $\mathbf{c}$ to be vectors of arbitrary dimension, possibly different from input dimension (this allows to overcome complexity constraints). To go back to input dimension, an element-wise linear layer is applyed with output dimension equal to $\mathbf{x}$ dimension.

Pseudo code for the autoregressive step is reported in 4.1. Notice that, in the case of inference, we cannot use teacher forcing, so we have to recursively generate the current input using previous inputs. This is reported in 4.1.

Algorithm 1 Autoregressive LSTM: training time (teacher forcing)

Input: : $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}_0 \in \mathbb{R}^d$, $\mathbf{h} \in \mathbb{R}^h$, $\mathbf{c} \in \mathbb{R}^h$
Output: : $\hat{\mathbf{x}}$
  $\mathbf{y} \leftarrow \mathbf{x}_0 \parallel \mathbf{x}[1 : n-1]$ ▷ Concatenate $\mathbf{x}_0$ and $\mathbf{x}$ without last element
  $\hat{\mathbf{x}} \leftarrow \text{LSTM}(\mathbf{y}, \mathbf{h}, \mathbf{c})$
  for $i = 1$ to N **do**
    $\hat{\mathbf{x}}[i] \leftarrow$ Element-wise Linear($\hat{\mathbf{x}}[i]$)
  end for
  return $\hat{\mathbf{x}}$

Notice in both algorithms, after the passage through the element-wise linear, each output $\hat{\mathbf{x}} \in \mathbb{R}^d$ is split into layer type prediction $\mathbf{x}_{[1:d-1]}$ and feature value prediction $\mathbf{x}_{[d]}$ and the former is passed through a Softmax activation, while the latter through a Sigmoid if we normalized features to be in $[0, 1]$, otherwise a ReLU to ensure they are positive. Additionally, during inference, the argmax approximation is applied at each iteration such that inputs resembles the original one hot encoding of layer types.

Algorithm 2 Autoregressive LSTM: inference time

Input: : $\mathbf{x}_0 \in \mathbb{R}^d$, $\mathbf{h} \in \mathbb{R}^h$, $\mathbf{c} \in \mathbb{R}^h$
Output: : $\hat{\mathbf{x}}$
  $\hat{\mathbf{x}}[0] \leftarrow \mathbf{x}_0$ ▷ An additional position '0' is added to prediction, but it will be removed at the end
  for $i = 1$ to N **do**
    $\hat{\mathbf{x}}[i] \leftarrow \text{LSTM}(\hat{\mathbf{x}}, \mathbf{h}, \mathbf{c})$ ▷ Only the last element of the output is kept for each iteration
    $\hat{\mathbf{x}}[i] \leftarrow$ Element-wise Linear($\hat{\mathbf{x}}[i]$)
  end for
  return $\hat{\mathbf{x}}[1 : n]$ ▷ remove the '0' positin element

4.2 Classic VAE

The first model we wish to implement is a standard VAE with diagonal covariance as in [8]. The encoder outputs for each datapoint mean and variance of a k-variate gaussian. In this case the reparametrization trick

has a very simple form as we can take advantage of properties of gaussian distributions.

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \tag{20}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation of the posterior, $\odot$ is the element-wise product and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

In this setting, it is possible to evaluate the Kullback-Leibler divergence from a standard isotropic Gaussian in closed form

$$\begin{aligned} D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right] &= \frac{1}{2}\left[\boldsymbol{\mu}^T\boldsymbol{\mu} + tr\Sigma - h - \log|\Sigma|\right] \\ &= \frac{1}{2}\left[\boldsymbol{\mu}^T\boldsymbol{\mu} + \sum_{i=1}^{h}\sigma_i^2 - h - \sum_{i=1}^{h}\log\sigma_i^2\right] \end{aligned} \tag{21}$$

Where $\boldsymbol{\mu}$ is the mean of the posterior, Σ is the covariance matrix and h is the dimension of the latent space. The passage from the first equation to the second is possible because Σ is a diagonal matrix.

4.3 VAE with full posterior covariance

The case in which we want to allow for correlation between elements of the posterior needs the estimation of a full covariance matrix from the encoder. This task is, in principle, quite complex as Σ must be symmetric, positive semi-definite. A common approach [7] is to use Cholesky decomposition of Σ as a lower triangular matrix $\mathbf{L}$ such that $\Sigma = \mathbf{L}\mathbf{L}^T$. This way, we can write the reparametrization trick as

$$\mathbf{z} = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon} \tag{22}$$

In practice, we let the encoder estimate three quantities: $\boldsymbol{\mu}$, $\mathbf{L}^*$ and $\boldsymbol{\sigma}$. $\mathbf{L}^*$ is a lower-diagonal matrix with all 0's on the diagonal. $\mathbf{L}$ is obtained by $\mathbf{L} = \mathbf{L}^* + \text{diag}(\boldsymbol{\sigma})$. This way, we can ensure $\Sigma = \mathbf{L}\mathbf{L}^T$ is well definite. Again, it is possible to estimate the Kullback-Leibler divergence from a standard isotropic Gaussian in closed form

$$D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right] = \frac{1}{2}\left[\boldsymbol{\mu}^T\boldsymbol{\mu} + tr\Sigma - k - \log|\Sigma|\right] \tag{23}$$

4.4 VAE with planar flows

Planar flows were first introduced in [9] and are a simple way to introduce a sequence of invertible transformations to the latent space. The flow is defined as

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b) \tag{24}$$

where $\mathbf{u}$, $\mathbf{w}$ and b are learnable parameters and h is a smooth non-linearity. The determinant of the Jacobian of this transformation is

$$\left|\det \frac{\partial f}{\partial \mathbf{z}}\right| = \left|1 + \mathbf{u}^T h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}\right| \tag{25}$$

Plugging this into a VAE means that the encoder must estimate $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ as in standard VAE. Then a sequence of planar stransformations is applyed to the latent variable, such that the final latent variable is

$$\mathbf{z}_K = f_K \circ f_{K-1} \circ \ldots \circ f_1 \circ g(\boldsymbol{\epsilon}) \tag{26}$$

where $g(\boldsymbol{\epsilon})$ is $\mathbf{z}_0$ under reparametrization trick. Here, the Kullback-Leibler divergence is not in closed form and must be estimated via Monte Carlo approximation. In particular, the density of the posterior under planar flow is

$$\log q_K(\mathbf{z}_K) = \log q_{\mathbf{z}_0}(\mathbf{z}_0) - \sum_{k=1}^{K} \log \left|1 + \mathbf{u}_k^T h'(\mathbf{w}_k^T \mathbf{z}_{k-1} + b_k)\mathbf{w}_k\right| \tag{27}$$

And $\log q_{\mathbf{z}_0}(\mathbf{z}_0)$ is easily estimated as it is an isotropic gaussian distribution with known parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$.

4.5 VAE with Inverse Autoregressive Flows (IAF)

Inverse Autoregressive Flows were first introduced in [6] and are a more complex way to introduce a sequence of invertible transformations to the latent space. The idea of inverse autoregressive flows is to use an autoregressive

transformation on the latent variable to define a complex, but with simple determinant transformation. Suppose $\mathbf{z} \in \mathbb{R}^h$ and we can give an order on dimensions h, such that we can see $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_t, \ldots, \mathbf{z}_h)$. The transformation is defined as

$$\mathbf{z}_{t+1} = \text{AR-transform}(\mathbf{z}_{1:t}) \tag{28}$$

In particular, if we apply a Gaussian autoregressive transformation, we can write

$$\mathbf{z}_{t+1} = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \mathbf{z}_t \tag{29}$$

Here, $\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t$ are the outputs of an autoregressive neural network with input $\mathbf{z}_{1:t}$. The determinant of the Jacobian of this transformation is

$$\left| \det \frac{\partial f}{\partial \mathbf{z}} \right| = \prod_{t=1}^{h} \sigma_t \tag{30}$$

For which the density of the posterior under IAF is

$$\log q_K(\mathbf{z}_K) = \log q_{\mathbf{z}_0}(\mathbf{z}_0) - \sum_{k=1}^{K} \sum_{t=1}^{h} \log \sigma_{k,t} \tag{31}$$

In our implementation, we follow the suggestion of [6] to use an autoregressive neural network similar to MADE [4] but with some slight modifications for numerical stability. Moreover, it is to notice we let the encoder estimate $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ as in standard VAE, but an additional hidden state $\mathbf{h}$ is also estimated and plugged into each autoregressive transform.

References

[1] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[2] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[3] Alp Kucukelbir David M. Blei and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[4] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 881–889, Lille, France, 07–09 Jul 2015. PMLR.

[5] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

[6] Diederik P. Kingma, Tim Salimans, and Max Welling. Improved variational inference with inverse autoregressive flow. *ArXiv*, abs/1606.04934, 2016.

[7] Diederik P. Kingma and Max Welling. 2019.

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[9] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. *ArXiv*, abs/1509.08731, 2015.

[10] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR.

[11] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Neural Information Processing Systems*, 2016.