Variational Autoencoder

Chunyan Li
Math Department
University of South Carolina

December 2020

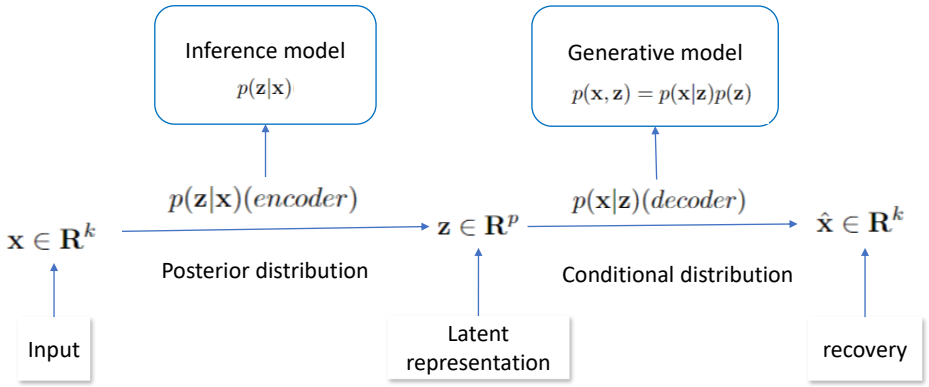
1 Introduction

In this chapter, we will give a tutorial for VAE that includes the following sections.

- Motivation of the construction of a VAE through the theory of probability and Bayes' Rule
- The derivation of loss function which is the lower bound of likelihood $p(\mathbf{x})$ of the observations.
- Maximize the lower bound via Stochastic gradient descent(SGD). (need reparameterization trick to pass the gradient w.r.t ϕ to the expectation.)
- MC(Monte Carlo) Method to approximate the expectation.
- Use neural networks for the probabilistic encoder and decoder.
- After training the VAE, how to find the latent representation/lower dimensional representation for any given new data which is assumed sampled from the same distribution with the training set.
- For the well trained VAE model, for a fixed given hidden variable, no many how many times you sampling, the decoder gives the same output. ????????
-

2 Motivation of the construction of a VAE

Consider the data set $X = [\mathbf{x}_1, ..., \mathbf{x}_n]^T$ with size n by k where n is the number of samples and k is the number of features.



Bayesian's rule: $$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{\int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}}$$

From the schematic diagram, we could see the following: we assume each $\mathbf{x}_i \in R^k$ is sampled from distribution $p(\mathbf{x})$, then for the given $\mathbf{x}_i$, we could determine the latent variable $\mathbf{z}_i \in R^p$ where $p << k$ by the posterior (density) distribution $p(\mathbf{z}|\mathbf{x} = \mathbf{x}_i)$, then for each given $\mathbf{z}_i$, we could recover $\mathbf{x}_i$ by conditional (density)distribution $p(\mathbf{x}|\mathbf{z} = \mathbf{z}_i)$. Hence, in this case, the posterior (density) distribution $p(\mathbf{z}|\mathbf{x})$ serves as an encoder and the conditional (density) distribution $p(\mathbf{x}|\mathbf{z})$ serves as a decoder. Then, using neural network to learn these two distributions gives us the variational autoencoder where we use another simple distribution $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ which is intractable in most of time. We call $q_\phi(\mathbf{z}|\mathbf{x})$ inference model or recognition model or an encoder or an approximated posterior. With parameter ϕ we indicate the parameters of this inference model, also called variational parameters. We optimize the variational parameters ϕ such that:

$$q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x}) \tag{1}$$

As we will explain, this approximation to the posterior will help us optimize the marginal likelihood of $\mathbf{x}$.

2.1 intractabilities[1]

The data likelihood

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z} \tag{2}$$

It is intractable to compute $p_\theta(\mathbf{x}|\mathbf{z})$ for every $\mathbf{z}$! This integral is a high dimensional integration which has no analytic form or an efficient estimator.
Posterior density

$$p_\theta(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})/p_\theta(\mathbf{x}) \tag{3}$$

It is intractable due to the intractability of likelihood and vice versa. The marginal probability of data under the model is typically intractable. This is due to the integral $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})d\mathbf{z}$ for computing the marginal likelihood not having an analytic solution or efficient estimator. Due to this intractability, we can not differentiate it w.r.t its parameters and optimize it.
The intractablity of $p_\theta(\mathbf{x})$, is related to the intractability of the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. Note that the joint distribution $p_\theta(\mathbf{x}, \mathbf{z})$is efficient to compute, and that the densities are related through the basic identity (Bayesian's rule):

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})} \tag{4}$$

Since $p_\theta(\mathbf{x}, \mathbf{z})$ is tractable to compute, a tractable marginal likelihood $p_\theta(\mathbf{x})$ leads to a tractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$ and vice versa. Both are intractable in VAE.

3 The derivation of loss function/Evidence lower bound objective (ELBO)

3.1 The derivation of loss function

The marginal likelihood is composed of a sum over the marginal likelihoods of individual data points. That is,

$$logp_\theta(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n) = \sum_{i=1}^{n} logp_\theta(\mathbf{x}_i) \tag{5}$$

The likelihood of one sample point $\mathbf{x}_i$ is $logp_\theta(\mathbf{x}_i)$. We would like to maximize the log-likelihood of the data. But the data likelihood is intractable which make the posterior is also intractable, so one can play a trick to address this issue. That is to approximate the posterior by a tractable distribution. Then one can use KL divergence to measure how close these two distributions, then, we have

$$\begin{aligned}
D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) &= E_{\mathbf{z}\sim q}[logq_\phi(\mathbf{z}|\mathbf{x}) - logp_\theta(\mathbf{z}|\mathbf{x})] \\
&= E_{\mathbf{z}\sim q}[logq_\phi(\mathbf{z}|\mathbf{x})] - E_{\mathbf{z}\sim q}[log\frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})}] \\
&= E_{\mathbf{z}\sim q}[logq_\phi(\mathbf{z}|\mathbf{x})] - E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})] + E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x})] \\
&= E_{\mathbf{z}\sim q}[logq_\phi(\mathbf{z}|\mathbf{x})] - E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})] + logp_\theta(\mathbf{x})
\end{aligned}$$

Hence, one can write the likelihood in the following expression:

$$logp_\theta(\mathbf{x}_i) = E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x}_i|\mathbf{z})p_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q}[logq_\phi(\mathbf{z}|\mathbf{x}_i)] + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z}|\mathbf{x}_i)) \tag{6}$$

Now, let's look at the equation (6), since $D_{KL} \geq 0$, we get the lower bound of the likelihood, also called the variational lower bound or the evidence lower bound, which is denoted by $L(\theta, \phi, \mathbf{x}_i)$.

$$\mathcal{L}(\theta, \phi, \mathbf{x}_i) = logp_\theta(\mathbf{x}_i) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z}|\mathbf{x}_i)) \tag{7}$$

$$= E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x}_i|\mathbf{z})p_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q}[logq_\phi(\mathbf{z}|\mathbf{x}_i)] \tag{8}$$

In order to maximize the likelihood and minimize the LK divergence $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z}|\mathbf{x}_i))$, we could maximize the lower bound $L(\theta, \phi, \mathbf{x}_i)$. We could rewrite the lower bound expression in the following way:

$$\begin{aligned}
\mathcal{L}(\theta, \phi, \mathbf{x}_i) &= E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x}_i|\mathbf{z})p_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q}[logq_\phi(\mathbf{z}|\mathbf{x}_i)] \\
&= E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x}_i|\mathbf{z}) + logp_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q}[logq_\phi(\mathbf{z}|\mathbf{x}_i)] \\
&= E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x}_i|\mathbf{z})] + E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{z}) - logq_\phi(\mathbf{z}|\mathbf{x}_i)] \\
&= E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x}_i|\mathbf{z})] - E_{\mathbf{z}\sim q}log[\frac{q_\phi(\mathbf{z}|\mathbf{x}_i)}{p_\theta(\mathbf{z})}] \\
&= E_{\mathbf{z}\sim q}[logp_\theta(\mathbf{x}_i|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z}))
\end{aligned}$$

Hence, we get the final expression of the loss function which is the negative lower bound of the likelihood for one sample point:

$$-\mathcal{L}(\theta, \phi, \mathbf{x}_i) = -E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[logp_\theta(\mathbf{x}_i|\mathbf{z})] + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z})) \tag{9}$$

The red part is the reconstruction error and the blue part is the regularizer.
We need to maximize the variational lower bound by optimizing the parameters ϕ and θ of the neural network. In simple words, on the RHS:

- We need to minimize the divergence between the estimated latent vector and the true latent vector.
- We need to maximize the expectation of the reconstruction of data points from the latent vector.

Let's explain why these two terms are named so in the following subsection.

3.2 Example of Loss function

The usual choice of encoder and decoder are multivariate Gaussian distribution, and assume the prior distribution $p_\theta(\mathbf{z})$ be normal distribution. Then, we use this case as an example to derive the explicit expression of the loss function.
The encoder is

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x})) \tag{10}$$

The decoder is

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})) \tag{11}$$

Assume the prior distribution is

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mu_3(\theta), \Sigma_3(\theta)) \tag{12}$$

Where μ_θ, μ_ϕ and $\Sigma_\theta, \Sigma_\phi$ are arbitrary deterministic functions with hyperparameter θ, ϕ that can be learned from data using neural network. Σ is usually constrained to be a diagonal matrix which is easy to calculate. The pdf of normal distribution of a multivariant r.v. $\mathbf{x} = [x_1, ..., x_k]$ is

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right), \tag{13}$$

where k is the dimension of $\mathbf{x}$.

Then, the first term of loss function is

$$E_{\mathbf{z}\sim\mathcal{N}(\mathbf{z}|\mu_{\phi i},\Sigma_{\phi i})}[log\mathcal{N}(\mathbf{x}_i; \mu_{\theta i}, \Sigma_{\theta i})] = -\frac{k}{2}log(2\pi) - \frac{1}{2}E_{\mathbf{z}\sim\mathcal{N}}[log|\Sigma_{\theta i}| + (\mathbf{x}_i - \mu_{\theta i})^T\Sigma_{\theta i}^{-1}(\mathbf{x}_i - \mu_{\theta i}))] \tag{14}$$

The expectation could be approximated by MC method which will be introduced later.

$$E_{\mathbf{z}\sim\mathcal{N}(\mathbf{z}|\mu_{\phi i},\Sigma_{\phi i})}[log\mathcal{N}(\mathbf{x}_i; \mu_{\theta i}, \Sigma_{\theta i})] \approx -\frac{k}{2}log(2\pi) - \frac{1}{2}\sum_{l=1}^{L}[log|\Sigma_{\theta i}(\mathbf{z}^{(l)})| + (\mathbf{x}_i - \mu_{\theta i}(\mathbf{z}^{(l)})^T\Sigma_{\theta i}^{-1}(\mathbf{z}^{(l)})(\mathbf{x}_i - \mu_{\theta i}(\mathbf{z}^{(l)})))] \tag{15}$$

The last term of the above equation (14) is square norm which could be treated as the reconstruction error. The second term of loss function is

$$D_{KL}(\mathcal{N}(\mu_{\phi i}, \Sigma_{\phi i})||\mathcal{N}(0, I)) = \frac{1}{2}\left[-log|\Sigma_{\phi i}| - p + tr(\Sigma_{\phi i}) + \mu_{\phi i}^T \mu_{\phi i}\right] \tag{16}$$

This term is called regularizer which will make sure that the encoder will not too far away from the normal distribution. In other words, the variance of encoder can't be too small.
If Σ_ϕ is diagonal, then, $\mu_\phi = [\mu_1, ..., \mu_p]$, $\Sigma_\phi = \mathbf{diag}\{\sigma_1, ..., \sigma_p\}$,

$$D_{KL}(\mathcal{N}(\mu_\phi, \Sigma_\phi)||\mathcal{N}(0, I)) = \frac{1}{2}\left[-log|\Sigma_\phi| - p + tr(\Sigma_\phi) + \mu_\phi^T \mu_\phi\right] \tag{17}$$

$$= \frac{1}{2}\sum_{i=1}^{p}[-1 - log\sigma_i + \sigma_i + \mu_i^2] \tag{18}$$

The KL divergence of any two multivariate Gaussian is given as follows:

$$D_{KL}(\mathcal{N}(\mu_1, \Sigma_1)||\mathcal{N}(\mu_2, \Sigma_2)) = \frac{1}{2}\left[log\frac{|\Sigma_2|}{|\Sigma_1|} - p + tr(\Sigma_2^{-1}\Sigma_1) + (\mu_1 - \mu_2)^T\Sigma_2^{-1}(\mu_1 - \mu_2)\right] \tag{19}$$

3.3 encoder $\mathcal{N}(\mathbf{z}; \mu_\phi, diag\{\sigma_j\}I_{p\times p})$ and decoder $\mathcal{N}(\mathbf{x}; \mu_\theta, I_{k\times k})$

Note: In my notation in this subsection and next subsection, σ, ξ stand for variance instead of standard deviation as usual. This is aimed for simplicity of the expression of loss function. Since we will see later, the covariance matrix is computed by using a neural network approximate $log\sigma$ as a whole.

If for each data point $\mathbf{x}$, we only sample one $\mathbf{z}$, then, the loss function is defined as follows:

$$-\mathcal{L}(\theta, \phi, X) = \frac{k}{2}log(2\pi) + \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i - \mu_{\theta i})^T(\mathbf{x}_i - \mu_{\theta i}) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}(log(\sigma_{ij}) - \sigma_{ij}) + \frac{1}{2}\sum_{i=1}^{n}\mu_{\phi i}^T\mu_{\phi i} - \frac{p}{2} \tag{20}$$

3.4 encoder $\mathcal{N}(\mathbf{z}; \mu_\phi, diag\{\sigma_j\}I_{p\times p})$ and decoder $\mathcal{N}(\mathbf{x}; \mu_\theta, diag\{\xi_m\}I_{k\times k})$

If for each data point $\mathbf{x}$, we only sample one $\mathbf{z}$, then the loss function is defined by

$$-\mathcal{L}(\theta, \phi, X) = \frac{k}{2}log(2\pi) + \frac{1}{2}\sum_{i=1}^{n}\sum_{m=1}^{k}(log\xi_{im}) + \frac{1}{2}\sum_{i=1}^{n}\frac{(\mathbf{x}_i - \mu_{\theta i})^T}{\xi_{im}}(\mathbf{x}_i - \mu_{\theta i}) \tag{21}$$

$$-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}(log(\sigma_{ij}) - \sigma_{ij}) + \frac{1}{2}\sum_{i=1}^{n}\mu_{\phi i}^T\mu_{\phi i} - \frac{p}{2} \tag{22}$$

4 Maximize the loss function via reparameterization trick

Now we get the lower bound of the likelihood, then one could maximize the lower bound to maximize the likelihood. So we get the following optimization problem:

$$max_{\theta,\phi} \quad \sum_{i=1}^{n}\mathcal{L}(\theta, \phi, \mathbf{x}_i)$$

In order to maximize the lower bound, one need the derivatives $\nabla_\theta\mathcal{L}$ and $\nabla_\phi\mathcal{L}$. Now, let's compute the derivatives.

4.1 estimate $\nabla_\theta\mathcal{L}$ of individual point

For the first one $\nabla_\theta\mathcal{L}$, one can directly differentiate $\mathcal{L}$ to get:

$$\begin{aligned}\nabla_\theta\mathcal{L}(\theta, \phi, \mathbf{x}_i) &= \nabla_\theta E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[logp_\theta(\mathbf{x}_i|\mathbf{z})] + \nabla_\theta E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[logp_\theta(\mathbf{z})] \\ &= E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}\nabla_\theta[logp_\theta(\mathbf{x}_i|\mathbf{z})] + E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}\nabla_\theta[logp_\theta(\mathbf{z})] \\ &= E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[\frac{p'_\theta(\mathbf{z}|\mathbf{x}_i)}{logp_\theta(\mathbf{x}_i|\mathbf{z})}] + E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[\frac{p'_\theta(\mathbf{z})}{logp_\theta(\mathbf{z})}]\end{aligned}$$

One can use Monte Carlo Method to get the approximation of $\nabla_\theta \mathcal{L}$. The MC estimates of the expectations of some function $f(\mathbf{z})$ w.r.t $q_\phi(\mathbf{z}|\mathbf{x})$ could be formulated as follows:

$$E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[f(\mathbf{z})] \simeq \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)}) \quad \text{where } \mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x}_i) \tag{23}$$

Hence, we apply this MC method (23) to the above expression of $\nabla_\theta \mathcal{L}$, obtaining our estimation of $\nabla_\theta \mathcal{L}$:

$$\nabla_\theta \mathcal{L}(\theta, \phi, \mathbf{x}_i) \simeq \frac{1}{L}\sum_{l=1}^{L} \frac{p'_\theta(\mathbf{z}^{(l)}|\mathbf{x}_i)}{logp_\theta(\mathbf{x}_i|\mathbf{z}^{(l)})} + \frac{p'_\theta(\mathbf{z}^{(l)})}{logp_\theta(\mathbf{z}^{(l)})} \quad where \quad \mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x}_i) \tag{24}$$

For the example in last section, the loss function is given by (14-17), then the corresponding estimation of $\nabla_\theta \mathcal{L}$ of one point $\mathbf{x}_i$is given by

$$\begin{aligned}
\nabla_\theta \mathcal{L} &= \nabla_\theta E_{\mathbf{z}\sim\mathcal{N}(\mathbf{z}|\mu_{\phi i},\Sigma_{\phi i})}[log\mathcal{N}(\mathbf{x}_i;\mu_{\theta i},\Sigma_{\theta i})] \\
&= \nabla_\theta \left[-\frac{1}{2}E_{\mathbf{z}\sim\mathcal{N}} log|\Sigma_{\theta i}| - \frac{1}{2}tr(\Sigma_{\theta i}^{-1}\Sigma_{\phi i}) - \frac{1}{2}(\mu_{\phi i}-\mu_{\theta i})^T\Sigma_{\theta i}^{-1}(\mu_{\phi i}-\mu_{\theta i})\right] \\
&= -\frac{1}{2}E_{\mathbf{z}\sim\mathcal{N}(\mathbf{z};\mu_{\phi i},\Sigma_{\phi i})}\nabla_\theta log|\Sigma_{\theta i}| - \frac{1}{2}\nabla_\theta[tr(\Sigma_{\theta i}^{-1}\Sigma_{\phi i}) + (\mu_{\phi i}-\mu_{\theta i})^T\Sigma_{\theta i}^{-1}(\mu_{\phi i}-\mu_{\theta i})] \\
&\simeq -\frac{1}{2L}\sum_{l=1}^{L}\nabla_\theta log|\Sigma_{\theta i}(\mathbf{z}^{(l)})| - \frac{1}{2}\nabla_\theta[tr(\Sigma_{\theta i}^{-1}\Sigma_{\phi i}) + (\mu_{\phi i}-\mu_{\theta i})^T\Sigma_{\theta i}^{-1}(\mu_{\phi i}-\mu_{\theta i})]
\end{aligned}$$

$where \quad \mathbf{z}^{(l)} \sim \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}_i), \Sigma_\phi(\mathbf{x}_i))$.

4.2 estimate $\nabla_\phi \mathcal{L}$ of individual point

The gradients w.r.t the variational parameters ϕ are more difficult to obtain, since the ELBO's expectation is taken w.r.t the distribution $q_\phi(\mathbf{z}|\mathbf{x})$, which is a function of ϕ. In general, the gradient operator and expectation operator is not commutative.

In order to change the gradient operator and expectation operator, we will play the following trick on the loss function.

$$\begin{aligned}
\nabla_\phi E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[f(\mathbf{z})] &= \nabla_\phi \int q_\phi(\mathbf{z}|\mathbf{x}_i) f(\mathbf{z}) d\mathbf{z} \\
&= \int \nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}_i) f(\mathbf{z}) d\mathbf{z} \\
&= \int \frac{\nabla_\phi q_\phi(\mathbf{z}|\mathbf{x}_i)}{q_\phi(\mathbf{z}|\mathbf{x}_i)} f(\mathbf{z}) q_\phi(\mathbf{z}|\mathbf{x}_i) d\mathbf{z} \\
&= \int [\nabla_\phi log q_\phi(\mathbf{z}|\mathbf{x}_i) f(\mathbf{z})] q_\phi(\mathbf{z}|\mathbf{x}_i) d\mathbf{z} \\
&= E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[\nabla_\phi log q_\phi(\mathbf{z}|\mathbf{x}_i) f(\mathbf{z})] \\
&\approx \frac{1}{L}\sum_{l=1}^{L} \nabla_\phi log q_\phi(\mathbf{z}^{(l)}|\mathbf{x}_i) f(\mathbf{z}^{(l)}), \quad z^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x}_i)
\end{aligned}$$

This gradient estimator exhibits very high variance (see e.g. [BJP12]) and is impractical for our purposes.

For the above gradient, in order to get a practical gradient estimator, we play a reparameterization trick here so that we could sample from a fixed distribution instead of the keep changing distribution $q_\phi(\mathbf{z}|\mathbf{x}_i)$ when ϕ changed.

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

$$\mathbf{z} = \mathbf{g}_\phi(\epsilon, \mathbf{x}) \quad with \quad \epsilon \sim p(\epsilon) \tag{25}$$

The strategies for choosing such an appropriate distribution $p(\epsilon)$ and function $\mathbf{g}_\phi(\epsilon, \mathbf{x})$ will be added later.

$$E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[f(\mathbf{z})] = E_{\epsilon\sim p(\epsilon)}[f(\mathbf{g}_\phi(\epsilon, \mathbf{x}_i))] \simeq \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{g}_\phi(\epsilon^{(l)}, \mathbf{x}_i)) \quad \text{where } \epsilon^{(l)} \sim p(\epsilon) \tag{26}$$

With this reparameterization, we can formulate the MC estimates of $\nabla_\phi \mathcal{L}$ with the loss function given by (14-17) as follows: since encoder is Gaussian, then the function $\mathbf{g}_\phi(\epsilon, \mathbf{x}) = \mu_\phi(\mathbf{x}) + \Sigma_\phi(\mathbf{x}) \cdot \epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$.

$$\begin{aligned}
\nabla_\phi \mathcal{L} &= \nabla_\phi E_{\mathbf{z}\sim\mathcal{N}(\mathbf{z};\mu_\phi,\Sigma_\phi)}[log\mathcal{N}(\mathbf{x};\mu_\theta(\mathbf{z}),\Sigma_\theta(\mathbf{z}))] - \nabla_\phi[\frac{1}{2}\left[-log|\Sigma_\phi| - k + tr(\Sigma_\phi) + \mu_\phi^T\mu_\phi\right]] \\
&= E_{\epsilon\sim\mathcal{N}(\epsilon;0,I)}\nabla_\phi[log\mathcal{N}(\mathbf{x};\mu_\theta(\mathbf{g}_\phi),\Sigma_\theta(\mathbf{g}_\phi))] - \nabla_\phi[\frac{1}{2}\left[-log|\Sigma_\phi| - k + tr(\Sigma_\phi) + \mu_\phi^T\mu_\phi\right]] \\
&\simeq \sum_{l=1}^{L} \nabla_\phi[log\mathcal{N}(\mathbf{x};\mu_\theta(\mathbf{g}_\phi(\epsilon^{(l)}),\Sigma_\theta(\mathbf{g}_\phi(\epsilon^{(l)})] - \nabla_\phi[\frac{1}{2}\left[-log|\Sigma_\phi| - k + tr(\Sigma_\phi) + \mu_\phi^T\mu_\phi\right]]
\end{aligned}$$

where $\mathbf{z} = \mu_\phi(\mathbf{x}) + \Sigma_\phi(\mathbf{x}) \cdot \epsilon, \epsilon \sim \mathcal{N}(0, I)$. Note that μ_θ, μ_ϕ and $\Sigma_\phi, \Sigma_\theta$ are approximated / implemented by a neural network which will be discussed in the next section.

5 Neural Network serves as Encoder and Decoder

Now we use the neural network to approximate the distributions. Here we assume the posterior distribution is a Multi-Gaussian distribution, then we use a neural network with one hidden layer to approximate the encoder:

5.1 Bernoulli MLP as decoder

Let $p_\theta(\mathbf{x}|\mathbf{z})$ be a multivariate Bernoulli:

$$\begin{aligned}
\mathbf{y} &= EncoderNeuralNet_\phi(\mathbf{z}) \\
q_\phi(\mathbf{z}|\mathbf{x}) &= Bernoulli(\mathbf{z};\mathbf{y})
\end{aligned}$$

More specific, probabilities $\mathbf{y}$ of multivariate Bernoulli are computed from $\mathbf{z}$ with a fully-connected neural network with a single hidden layer

$$logp(\mathbf{x}|\mathbf{z}) = \sum_{i=1}^{k} x_i logy_i + (1 - x_i) \cdot log(1 - y_i) \tag{27}$$

$$where\ \mathbf{y} = f_\sigma(W_2 tanh(W_1\mathbf{z} + b_1) + b_2) \tag{28}$$

where $f_\sigma(\cdot)$ is the elementwise sigmoid activation function, and $\theta = \{W_1, W_2, b_1, b_2\}$ are the weight and biases of the MLP.

5.2 Gaussian as encoder and decoder

The encoder or approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ can be parameterized using deep neural networks. Then, the variational parameters ϕ include the weights and biases of the neural network. For example,

$$\begin{aligned}
(\mu, diag\{log\sigma_i^2\}) &= EncoderNeuralNet_\phi(\mathbf{x}) \\
q_\phi(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z};\mu, diag(\sigma_i^2))
\end{aligned}$$

where $i = 1, ..., k$ with k be the dimensional of $\mathbf{x}$. σ_i^2 is the variance of $\mathbf{x}_i$ Typically, we use a single encoder neural network to perform posterior inference over all data points in our dataset. Hence, VAEs employ a strategy with global variational parameters which is difference to traditional variational inference methods where the variational parameters are not shared, but instead separately and iteratively optimized per data point.

5.2.1 Gaussian with diagonal covariance matrix

Let me give a more specific example to indicate how to implement encoder neural network. let encoder and decoder be a multivariate Gaussian with a diagonal covariance structure:

$$logp(\mathbf{x}|\mathbf{z}) = log\mathcal{N}(\mathbf{x};\mu,\sigma^2 I) \tag{29}$$

$$where\ \mu = W_4 h + b_4 \tag{30}$$

$$diag\{log\sigma_i^2\}_{i=1}^k = W_5 h + b_5 \tag{31}$$

$$h = tanh(W_3\mathbf{z} + b_3) \tag{32}$$

where $W_3, W_4, W_5, b_3, b_4, b_5$ are the weights and biases of MLP and part of θ when used as decoder. When this network is used as an encoder $q_\phi(\mathbf{z}|\mathbf{x})$, then $\mathbf{z}$ and $\mathbf{x}$ are swapped, and the weights and biases are variational parameters ϕ.

5.2.2 Gaussian with full covariance matrix

6 Implementation

In this section, we focus on the implementation of VAE in PyTorch. We create dataset by $sin(x + y)$, so (x, y) is the dataset, $sin(x + y)$ is the corresponding target.

References

[1] Diederik P Kingma and Max Welling. "An introduction to variational autoencoders". In: *arXiv preprint arXiv:1906.02691* (2019).