Variational Autoencoder Mathematics

Andreas Spanopoulos
andrewspanopoulos@gmail.com

December 4, 2020

Introduction

The **Variational Autoencoder** (aka **VAE**) is a generative model. This means that it is a model which produces new unseen data. Unlike the normal Autoencoder, VAE focuses on understanding the distribution of a smaller representation of the data. This lower-dimensional representation of the data is known as “latent vector **z**”.

The dimension of the latent vector z is a hyperparameter which we choose along with the architecture of the Network. Keep in mind that we don’t want **z** to be too large. It should be a relatively small vector, so that an information bottleneck is created. One other reason for **z** being small, is that we want to be able to sample easily new vectors, without having to take into consideration many features.

With that said, the question arises: How can we pick the values of **z** which will make sense, that is, which will generate a new data point from the distribution of our original data?

Here is the beauty of the **Variational Autoencoder**: We will learn the distribution of **z**. That is, for every component of **z**, we will learn a mean and a standard deviation.

Suppose **z** has k components:

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix}$$

Then, the mean and standard deviation vectors are defined as:

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_k \end{bmatrix}, \quad \sigma = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_k \end{bmatrix}$$

Our goal is to learn the μ and σ vectors in order to be able to sample **z** as follows

$$z = \mu + \epsilon \odot \sigma$$

where $\epsilon \sim N(0,1)$ is a gaussian with mean 0 and standard deviation 1.

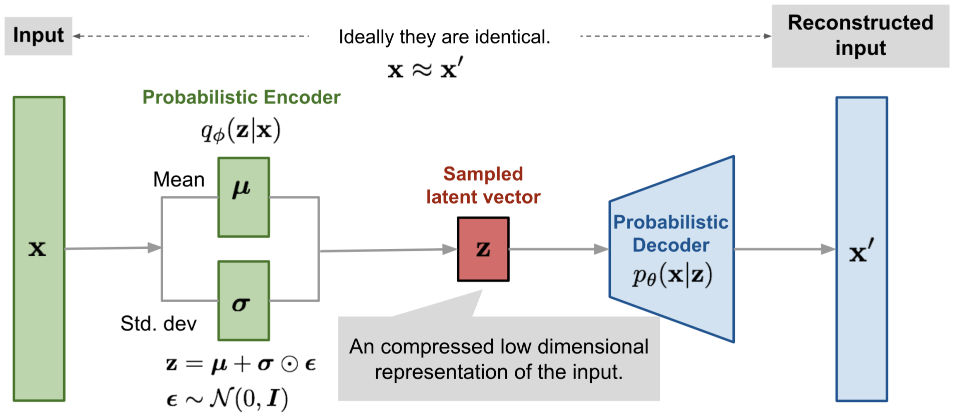

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
$\mathbf{x}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

Figure 1: This picture demonstrates the architecture of a Variational Autoencoder. The input $\mathbf{x}$ gets fed in a Probabilistic Encoder $q_\phi(z|x)$, which in turns connects with the μ and σ layers. Note that usually there is a encoder Network before the mean and std layers, but here in the figure it is ommitted. Then, they sample $\mathbf{z}$ which in turn is fed to the Probabilistic Decoder $p_\theta(x|z)$. The result is then fed to an output layer which represents the reconstructed input data. The original picture can be found here.

Brief Explanation of architecture

The architecture of a **VAE** is briefly portrayed in Figure 1. Let's take a closer look in each part:

1. The encoder part consists of a Probabilistic Encoder $q_\phi(z|x)$. Given some parameters ϕ (which are parameters of the model), $q_\phi(z|x)$ models the probability of obtaining the latent vector $\mathbf{z}$ given input data $\mathbf{x}$. Afterwards, it connects to the μ and σ layers, as there might a whole encoder network before those.

2. The latent vector $\mathbf{z}$.

3. The decoder part which consists of a Probabilistic Decoder $p_\theta(x|z)$. As with the probabilistic encoder, given some parameters θ which are parameters of the model, we want to learn the probability of obtaining a data point $\mathbf{x}$ given a latent vector $\mathbf{z}$.

4. The reconstructed input $\hat{x}$.

Loss function

The loss function of the VAE is:

$$L(\theta, \phi, x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + D_{KL}\left[q_\phi(z|x) \,\|\, p_\theta(z)\right]$$

It may seem daunting at first, but if we break it down into pieces then it gets much simpler.

KL-Divergence and multivariate Normal Distribution

Let's start by explaining what the second term of the loss function is. The **K**ullback **L**eibler Divergence, also known as Relative Entropy, is a measure of similarity between two probability distributions. It is denoted by $D_{KL}(\cdot \,\|\, \cdot)$, its unit of measure it called **nat** and it can computed by the formula (for continuous probability distributions P, Q):

$$D_{KL}\left[P \,\|\, Q\right] = \int P(x)\log\left(\frac{P(x)}{Q(x)}\right)dx \tag{1}$$

Of course, this implies that $D_{KL}(P \,\|\, Q) \neq D_{KL}(Q \,\|\, P)$.

Now, let's suppose that both P, Q are multivariate normal distributions with means μ_1, μ_2 and covariance matrices Σ_1, Σ_2:

$$\begin{aligned} P(x) &= N(x;\, \mu_1,\, \Sigma_1) = \frac{1}{\sqrt{(2\pi)^k|\Sigma_1|}}e^{-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)} \\ Q(x) &= N(x;\, \mu_2,\, \Sigma_2) = \frac{1}{\sqrt{(2\pi)^k|\Sigma_2|}}e^{-\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)} \end{aligned}$$

where k is the magnitude (length) of vector x.
Hence

$$\begin{aligned} \log(P(x)) &= \log\left(\frac{1}{\sqrt{(2\pi)^k|\Sigma_1|}}e^{-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)}\right) \\ &= \log\left(\frac{1}{\sqrt{(2\pi)^k|\Sigma_1|}}\right) + \log\left(e^{-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)}\right) \\ &= -\frac{k}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma_1|) - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) \end{aligned}$$

Following the exact same steps, we also get that

$$\log(Q(x)) = -\frac{k}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma_2|) - \frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)$$

With the help of the above equalities, expanding (1) yields:

$$\begin{aligned} D_{KL}[P\|Q] &= \int P(x)\left[\log(P(x)) - \log(Q(x))\right]dx \\ &= \int P(x)\left[\frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right. \\ &\qquad\qquad \left. + \frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right]dx \end{aligned}$$

We can rewrite the above term an an Expectation over P:

$$\begin{aligned} D_{KL}[P\|Q] &= \mathbb{E}_P\left[\frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right. \\ &\qquad\qquad \left. + \frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right] \end{aligned}$$

Since the logarithmic term is independent of x, we can move it outside the expectation. This leaves us with

$$\begin{aligned} D_{KL}[P\|Q] = &\frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) \\ &-\frac{1}{2}\mathbb{E}_P\left[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right] \\ &+\frac{1}{2}\mathbb{E}_P\left[(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right] \end{aligned} \tag{2}$$

Let's now try to simplify the 2nd and 3rd terms of the above expression.

First, we have to recall the trace function and some of its properties. The trace of a square matrix A, denoted as $tr(A)$, is the sum of the elements along the main diagonal of A. The properties of the trace function which we will need are:

1. Trace of scalar: Considering the scalar as a 1×1 matrix, gives: $x = tr(x)$
2. Trace of Expectation: From 1: $\mathbb{E}[x] = \mathbb{E}[tr(x)] \Rightarrow tr(\mathbb{E}[x]) = \mathbb{E}[tr(x)]$
3. Cyclic Property: $tr(ABC) = tr(CAB)$

Having these properties in mind, we are now ready to simplify the expectation terms computed before during the simplification of the KL Divergence.

- Term 2. Note that the matrix multiplications inside the expectation reduce to a scalar value.

$$\begin{aligned}\mathbb{E}_P\left[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right] &\overset{(1)}{=} \mathbb{E}_P\left[tr\left((x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right)\right]\\ &\overset{(3)}{=} \mathbb{E}_P\left[tr\left((x-\mu_1)(x-\mu_1)^T\Sigma_1^{-1}\right)\right]\\ &\overset{(2)}{=} tr\left(\mathbb{E}_P\left[(x-\mu_1)(x-\mu_1)^T\Sigma_1^{-1}\right]\right)\end{aligned}$$

  Σ_1^{-1} is independent from the expectation over P, so it can be moved outside, thus giving:

$$\mathbb{E}_P\left[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right] = tr\left(\mathbb{E}_P\left[(x-\mu_1)(x-\mu_1)^T\right]\Sigma_1^{-1}\right)$$

  But the term $E_P\left[(x-\mu_1)(x-\mu_1)^T\right]$ is equal to the Covariance Matrix Σ_1, thus yielding

$$\mathbb{E}_P\left[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right] = tr\left(\Sigma_1\Sigma_1^{-1}\right) = tr(I_k) = k \qquad (3)$$

- Term 3. Again, the term inside the expectation reduces to a scalar.

$$\mathbb{E}_P\left[(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right]$$

  Add and subtract μ_1 :

$$= \mathbb{E}_P\left[[(x-\mu_1)+(\mu_1-\mu_2)]^T\Sigma_2^{-1}[(x-\mu_1)+(\mu_1-\mu_2)]\right]$$

$$= \mathbb{E}_P\left[\left[(x-\mu_1)^T+(\mu_1-\mu_2)^T\right]\Sigma_2^{-1}[(x-\mu_1)+(\mu_1-\mu_2)]\right]$$

$$(A^T+B^T)C(A+B) = A^TCA + A^TCB + B^TCA + B^TB :$$

$$= \mathbb{E}_P\Big[(x-\mu_1)^T\Sigma_2^{-1}(x-\mu_1) + (x-\mu_1)^T\Sigma_2^{-1}(\mu_1-\mu_2) +$$

$$(\mu_1-\mu_2)^T\Sigma_2^{-1}(x-\mu_1) + (\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2)\Big]$$

Taking the expectation over each term individually:

$$\begin{aligned}\mathbb{E}_P\left[(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right] = \; & \mathbb{E}_P\left[(x-\mu_1)^T\Sigma_2^{-1}(x-\mu_1)\right] + \\ & \mathbb{E}_P\left[(x-\mu_1)^T\Sigma_2^{-1}(\mu_1-\mu_2)\right] + \\ & \mathbb{E}_P\left[(\mu_1-\mu_2)^T\Sigma_2^{-1}(x-\mu_1)\right] + \\ & \mathbb{E}_P\left[(\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2)\right]\end{aligned}$$

- The first sub-term has the same derivation as the first term from before:

$$\mathbb{E}_P\left[(x-\mu_1)^T\Sigma_2^{-1}(x-\mu_1)\right] = tr(\Sigma_1\Sigma_2^{-1})$$

- The second and third sub-terms are equal to 0 due to the expectation over $(x-\mu_1)^T$. Specifically, the factor $\Sigma_2^{-1}(\mu_1-\mu_2)$ can be moved out of the expectation as it is a constant:

$$\begin{aligned}\mathbb{E}_P\left[(x-\mu_1)^T\Sigma_2^{-1}(\mu_1-\mu_2)\right] &= \mathbb{E}_P\left[(x-\mu_1)^T\right]\Sigma_2^{-1}(\mu_1-\mu_2) \\ &= 0_k\,\Sigma_2^{-1}(\mu_1-\mu_2) = 0\end{aligned}$$

$$\begin{aligned}\mathbb{E}_P\left[(\mu_1-\mu_2)^T\Sigma_2^{-1}(x-\mu_1)\right] &= (\mu_1-\mu_2)^T\Sigma_2^{-1}\,\mathbb{E}_P\left[(x-\mu_1)^T\right] \\ &= \Sigma_2^{-1}(\mu_1-\mu_2)\,0_k = 0\end{aligned}$$

- The fourth sub-term is the expectation of a constant, so it is equal to the constant itself:

$$\mathbb{E}_P\left[(\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2)\right] = (\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2)$$

These simplifications leave us with:

$$\begin{aligned}\mathbb{E}_P\left[(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right] = \; & tr(\Sigma_1\Sigma_2^{-1}) + \\ & (\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2)\end{aligned} \tag{4}$$

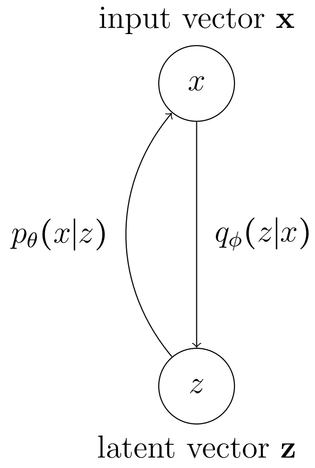
Finally, formula (2) for the KL-Divergence can be ultimately simplified using equations (3), (4) to:

$$D_{KL}\left[P\,\|\,Q\right] = \frac{1}{2}\left[\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - k + tr(\Sigma_1\Sigma_2^{-1}) + (\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2)\right] \tag{5}$$

■

Goal of VAE

As mentioned in the introduction, the goal of the **VAE** is to learn the distribution $q_\phi(\mathbf{z}|\mathbf{x})$ of the latent vector $\mathbf{z}$. After the distribution of the latent vector has been learnt, we can sample $\mathbf{z}$ and feed it to the Generator Network defined by the distribution $p_\theta(\mathbf{x}|\mathbf{z})$ (aka decoder) in order to obtain a new data point $\tilde{x}$. Of course during training, we want $\tilde{x} \approx x$. The training process can be thought of graphically as follows



Derivation of Loss Function

We would like the two distributions to be approximately same, that is

$$q_\phi(z|x) \approx p_\theta(z|x)$$

How can we force the two distributions to come close? We could view this as a minimization problem of the KL-Divergence between Q and P.

$$D_{KL}\left[q_\phi(z|x)\,||\,p_\theta(z|x)\right] = \int q_\phi(z|x)\log\left(\frac{q_\phi(z|x)}{p_\theta(z|x)}\right)dz$$

Rewriting this term as an Expectation yields:

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

Since $\log p_\theta(x)$ is independent from the Expectation of z, it can be moved outside, thus giving

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

which is the loss function L, defined in terms of ϕ, θ, x:

$$L(\theta,\,\phi,\,x) \;=\; -\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] \;+\; D_{KL}\left[q_\phi(z|x)\,\|\,p_\theta(z)\right]$$

which is also known as Evidence Lower BOund (ELBO). Of course, our target is to find the θ, ϕ that minimize the loss $L(\theta, \phi, x)$ for all data points x in the training set, X that is:

$$\theta^*,\,\phi^* \;=\; \operatorname*{argmin}_{\phi,\theta} L(\theta,\,\phi,\,x) \text{ over all } x \in X$$

Where in our case θ^* are the weighs of the Recognition Network (aka encoder) and ϕ^* are the weights of the Generator Network (aka decoder).

Simplifying the Loss Function

How could we simplify the Loss function $L(\theta, \phi, x)$ such that it can be computed using known information? The problem is the KL-Divergence term, as we don't have a closed form for it. The first solution that comes to mind is to model both distributions of the KL-Divergence term as Gaussians, with $p_\theta(z)$ having mean 0 and standard deviation 1, thus pushing $q_\phi(z|x)$ to come close to it. Specifically:

1. $q_\phi(z|x) \sim N(\mu_\phi(x), \Sigma_\phi(x))$
2. $p_\theta(z) \sim N(0_k, I_k)$

This in turn allows us to use eq. (5), substituting the values

1. $\mu_1 = \mu_\phi(x),\ \Sigma_1 = \Sigma_\phi(x)$
2. $\mu_2 = 0_k,\ \Sigma_2 = I_k$

Thus yielding:

$$D_{KL}\left[q_\phi(z|x) \,\|\, p_\theta(z)\right]$$

$$= \frac{1}{2}\left[\log\left(\frac{|I_k|}{|\Sigma_\phi(x)|}\right) - k + tr(\Sigma_\phi(x)\, I_k) + (\mu_\phi(x) - 0_k)^T I_k(\mu_\phi(x) - 0_k)\right]$$

$$= \frac{1}{2}\left[-\log|\Sigma_\phi(x)| - k + tr(\Sigma_\phi(x)) + \mu_\phi(x)^T \mu_\phi(x)\right]$$

Note that $\Sigma_\phi(x)$ is a diagonal matrix with k elements. Hence we can write

$$D_{KL}\left[q_\phi(z|x) \,\|\, p_\theta(z)\right]$$

$$= \frac{1}{2}\left[-\log\left(\prod_k \Sigma_{\phi_k}(x)\right) - k + \sum_{i=1}^{k} \Sigma_{\phi_i}(x) + \sum_{i=1}^{k} \mu_{\phi_i}^2\right]$$

$$= \frac{1}{2}\left[-\sum_{i=1}^{k} \log \Sigma_{\phi_i}(x) - \sum_{i=1}^{k} 1 + \sum_{i=1}^{k} \Sigma_{\phi_i}(x) + \sum_{i=1}^{k} \mu_{\phi_i}^2\right]$$

$$= \frac{1}{2}\sum_{i=1}^{k}\left[-\log \Sigma_{\phi_i}(x) - 1 + \Sigma_{\phi_i}(x) + \mu_{\phi_i}^2\right]$$

which is the fully simplified version of the KL-Divergence term. The final loss function will is

$$L(\theta, \phi, x) = -\mathbb{E}_{z \sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + \frac{1}{2}\sum_{i=1}^{k}\left[-\log \Sigma_{\phi_i}(x) - 1 + \Sigma_{\phi_i}(x) + \mu_{\phi_i}^2\right] \tag{6}$$

Backpropagation

In order to perform backpropagation in the VAE model, we need to compute the partial derivatives

$$\frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial \phi}$$

1. Partial derivative w.r.t. θ:

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

   Using a Monte Carlo Estimator for the Expectation, we get

   $$\frac{\partial L}{\partial \theta} = -\frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial \theta}\log p_\theta(x|z^{(i)})$$

   where $z^{(i)} \sim q_\theta(z|x)$

2. Partial derivative w.r.t. ϕ:

   Here, a problem arises. If we try to take the partial derivative of the first term w.r.t. ϕ, then the gradient is being blocked by the distribution q_ϕ for which the expectation is taken. This problem can be solved by using the Reparameterization Trick, that is, performing a linear substituion $z = g_\phi(\epsilon, x)$ with $\epsilon \sim N(0, 1)$ in order to "push" the stochasticity out of the latent vector z, into the newly introduced ϵ node.

   The linear transformation g can be as simple as

   $$g_\phi(\epsilon, x) = \mu_\phi(x) + \epsilon \odot \Sigma_\phi^{\frac{1}{2}}(x) = z \sim N(\mu_\phi(x), \Sigma_\phi(x))$$

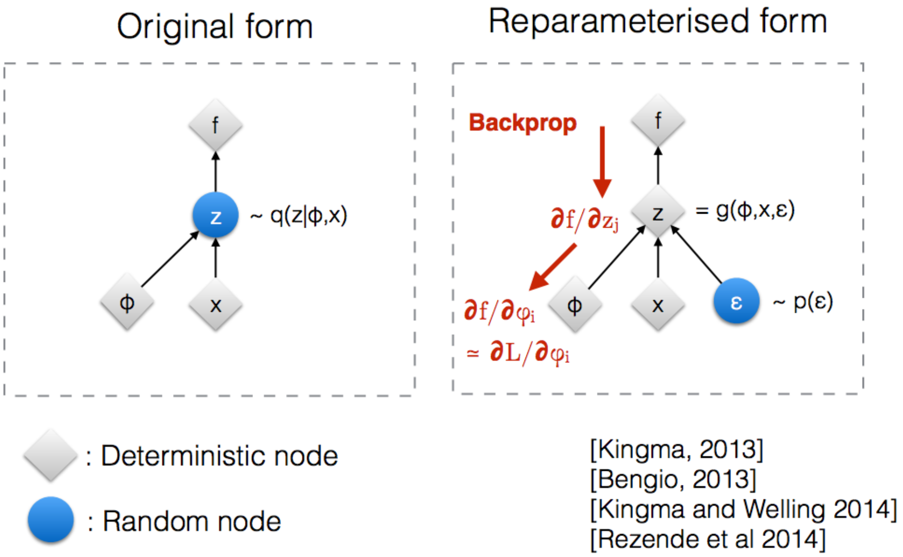


Figure 2: Visual Representation of the effect of the Reparameterization Trick. The linear substituion introduces the stochastic node ϵ, thus removing the stochasticity from z and allowing the gradients to flow backwards. This fig. can be found in the Introduction to Variational Autoencoders paper, here.

Figure 2 is a visual representation of the reparameterization trick. Now, again using a Monte Carlo Estimator for the Expectation of the first term, we can compute the gradient of the loss function L w.r.t. ϕ as follows

$$\frac{\partial L}{\partial \theta} = -\frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial \theta}\log p_\theta(x|z^{(i)}) + \frac{\partial}{\partial \theta}\left[\frac{1}{2}\sum_{i=1}^{k}\left[-\log \Sigma_{\phi_i}(x) - 1 + \Sigma_{\phi_i}(x) + \mu_{\phi_i}^{2}\right]\right]$$

Resources

- Papers:
  - Auto-Encoding Variational Bayes
  - An Introduction to Variational Autoencoders
  - Early Visual Concept Learning with Unsupervised Deep Learning
- Online Lectures:
  - Ahlad Kumar
  - Ali Ghodsi