A Short (Nice) Tutorial on Variational Auto Encoder

Ehsan Shareghi

University of Cambridge

May 2018

Introduction

1.1 Inference under Posterior

Imagine the posterior probability of a latent variable of interest z given an input data x:

$$P(z|x) = \frac{P(x,z)}{P(x)} = \frac{P(x|z)p(z)}{P(x)} \quad \text{where} \quad P(x) = \int P(x|z)P(z)dz. \tag{1.1}$$

This is a classic situation during inference time, when given an input a label is required to be predicted. Computing $P(z|x)$ happens to be hard, due to the denominator $P(x)$ which relies on the computation of an intractable integral. The intractability becomes more obvious if we assume z to be continuous, for instance a real-valued N-dimensional representation associated with an input sequence of words in a generative scenario, where knowing the latent variable z (i.e. a latent semantic space) will allow us to generate sentences.

An approach to tackle this problem is to construct a Markov Chain Monte Carlo (MCMC) sampling to simulate samples from the posterior distribution. MCMC tends to be slow (several samples are required for convergence), sampling procedure may never mix, and it is also hard to assess whether it converges or not (and in many cases the decision about a proposal distribution is also required). An orthogonal approach to MCMC is to aim for approximating $P(z|x)$ with another distribution $Q(z|x)$, by starting from a family of distributions (i.e., normal distributions considering all possible realization for its parameters) and searching for the optimized parameters (i.e., optimal mean and variance) such that the `KL-divergence`[1] between the optimal distribution $Q(z|x)$ and the target distribution $P(z|x)$ is minimum. The resulting $Q(z|x)$ through this optimization procedure is called the variational distribution.

1.1.1 `KL-divergence`

Noting that there are two ways of writing the `KL-divergence`:

$$\texttt{KL}(Q(z|x)||P(z|x)) = \int Q(z|x) \log \frac{Q(z|x)}{P(z|x)} dz = \left\langle \log \frac{Q(z|x)}{P(z|x)} \right\rangle_{Q(z|x)}$$

$$\texttt{KL}(P(z|x)||Q(z|x)) = \int P(z|x) \log \frac{P(z|x)}{Q(z|x)} dz = \left\langle \log \frac{P(z|x)}{Q(z|x)} \right\rangle_{P(z|x)}$$

Given that we are interested in having a good approximation of P in its high density area, we choose the first definition of `KL-divergence`, because:

- for both definitions: when both Q and P are high, or low the log of the ratio will be close to 0
- for the first definition:
  - when Q is low, regardless of P, the `KL-divergence` will be low.
  - when Q is high and P is low, the `KL-divergence` will be high.
- for the second definition: the opposite of the above properties is captured

[1] In addition to `KL-divergence` there are also other types of divergence metrics, but this is beyond the focus of this draft.

1.1.2 Reducing KL-divergence to ELBO

Using Eq. 1.1, the KL-divergence can be rewritten as,

$$\begin{aligned}
\texttt{KL}(Q(z|x)||P(z|x)) &= \left\langle \log \frac{Q(z|x)}{\frac{P(x,z)}{P(x)}} \right\rangle_{Q(z|x)} = \left\langle \log \frac{Q(z|x)}{P(x,z)} P(x) \right\rangle_{Q(z|x)} \\
&= \left\langle \log \frac{Q(z|x)}{P(x,z)} + \log P(x) \right\rangle_{Q(z|x)} \\
&= \left\langle \log \frac{Q(z|x)}{P(x,z)} \right\rangle_{Q(z|x)} + \left\langle \log P(x) \right\rangle_{Q(z|x)} \\
&= \left\langle \log \frac{Q(z|x)}{P(x,z)} \right\rangle_{Q(z|x)} + \log P(x)
\end{aligned}$$

The last equation was driven via expanding the following expectation in the second term,

$$\left\langle \log P(x) \right\rangle_{Q(z|x)} = \int Q(z|x) \log P(x) dz = \log P(x) \int Q(z|x) dz = \log P(x).$$

To swap the left and right handsides we get,

$$\texttt{KL}(Q(z|x)||P(z|x)) + \left\langle \log \frac{P(x,z)}{Q(z|x)} \right\rangle_{Q(z|x)} = \log P(x)$$

where $\log P(x)$ is a **constant** term known as evidence. Therefore, in order to minimize $\texttt{KL}(Q(z|x)||P(z|x))$, we can instead maximize,

$$\left\langle \log \frac{P(x,z)}{Q(z|x)} \right\rangle_{Q(z|x)}$$

which is called the evidence lower bound (ELBO). The reason for this being the lower bound is due to non-negativity of KL-divergence as we proved in Appendix A. Therefore,

$$\underbrace{\texttt{KL}(Q(z|x)||P(z|x))}_{\text{always non-negative}} + \left\langle \log \frac{P(x,z)}{Q(z|x)} \right\rangle_{Q(z|x)} = \log P(x) \Rightarrow \left\langle \log \frac{P(x,z)}{Q(z|x)} \right\rangle_{Q(z|x)} \leq \log P(x)$$

We have reduced minimizing the KL-divergence to the problem of maximizing ELBO.

1.2 Maximizing ELBO via Monte Carlo Gradient

This relies on a very strong assumption,

$$\nabla_\theta \langle f_\theta(z) \rangle_{Q_\theta(z)} = \langle \nabla_\theta f_\theta(z) \rangle_{Q_\theta(z)} \tag{1.2}$$

which results (see Appendix B) in the following approximation of gradient with respect to the variational distribution parameter,

$$\left\langle \nabla \log Q(z|x) \left(\log \frac{P(x,z)}{Q(z|x)} \right) \right\rangle_{Q(z|x)} \approx \frac{1}{K} \sum_{z_k \in \{z_k\}_{k=1}^K} \nabla \log Q(z_k|x) \left(\log \frac{P(x,z_k)}{Q(z_k|x)} \right) \tag{1.3}$$

where $\{z_k\}_{k=1}^K$ are generated samples from $Q(z|x)$. And the gradient with respect to model (distribution P) parameter corresponds to,

$$\nabla \left(\left\langle \log \frac{P(x,z)}{Q(z|x)} \right\rangle_{Q(z|x)} \right) = \left\langle \nabla \log P(x,z) \right\rangle_{Q(z|x)} \approx \frac{1}{K} \sum_{z_k \in \{z_k\}_{k=1}^K} \nabla \log P(x,z_k) \tag{1.4}$$

It turns out that optimizing the parameters using the samples (i.e., MCMC-generated from $Q(z|x)$) exhibits high variance. And there are efforts in this direction to control the variance, see [Paisley et al., 2012, Mnih and Gregor, 2014, Mnih and Rezende, 2016, Maddison et al., 2017].

Note I Extremely important observation here is to note that $Q(z|x)$, the distribution over the latent variable z, is dependent on a given x. Therefore, in the optimization, we are essentially learning a separate distribution over latent variables for each data point. This is more expressive than just learning a distribution $Q(z)$ which is not sensitive to the input.

Variational Autoencoder

2.1 Analyzing ELBO

Let us take another path to expand ELBO as follows,

$$\begin{aligned}
\left\langle \log \frac{P(x,z)}{Q(z|x)} \right\rangle_{Q(z|x)} &= \left\langle \log \frac{P(x|z)P(z)}{Q(z|x)} \right\rangle_{Q(z|x)} \\
&= \left\langle \log P(x|z) + \log \frac{P(z)}{Q(z|x)} \right\rangle_{Q(z|x)} \\
&= \left\langle \log P(x|z) \right\rangle_{Q(z|x)} + \left\langle \log \frac{P(z)}{Q(z|x)} \right\rangle_{Q(z|x)} \qquad (2.5) \\
&= \left\langle \log P(x|z) \right\rangle_{Q(z|x)} - \underbrace{\texttt{KL-divergence}(Q(z|x)||P(z))}_{\text{Fitting the approximate posterior } Q(z|x) \text{ to the prior } P(z)} \qquad (2.6)
\end{aligned}$$

We haven't made any assumption about the mathematical form of Q and P. For instance, if we assume $P(z)$ and $Q(z|x)$ are both Gaussian distributions, the $\texttt{KL-divergence}(Q(z|x)||P(z))$ could be solved analytically, see Appendix C for the case with diagonal covariance. Hence, approximation is required only to be applied to first term in Eq. 2.6, which consequently reduces the variance during the optimization.

2.2 Maximizing ELBO via Autoencoder

The path to VAE starts by considering the alternative way of expanding ELBO, as shown in Eq. 2.6. In this expansion, we can observe an interesting interpretation. Let us focus on the first term:

$$\left\langle \log P(x|z) \right\rangle_{Q(z|x)}$$

To evaluate this term, we require to take all possible $z \sim Q(z|x)$ draws which are conditioned on the input x, and then for each of the latent variables compute $P(x|z)$, which measures how likely it is to predict the given data point x from z. Now assume we want to maximize this (which we do when maximizing the whole ELBO). We are essentially finding the optimal parameter of Q and P, such that z could be successfully predicted from x, and then x could be successfully predicted from z. This is reminiscent of autoencoding framework, where we in the first part encode x into z via $Q(z|x)$, and in the second part we decode z into x via $P(x|z)$. A trivial solution to maximize this, is to just learn Q such that it takes the input x and produces x (a copy of x as z), and then learn P such that it takes the output from Q, which is now x, and do the same thing.

Note II An important point in this analogy was ignored and requires clarification. A random variable is a function defined over a particular space. This space is fixed, meaning that it only allows certain random outcomes (i.e., {head, tail} in coin tossing experiments as inputs to this function will be mapped via random variable to {1,0}). Therefore, a probability distribution defined over a particular random variable z will be undefined if we just pass to it an element which doesn't belong to the domain of the random variable z. So, it basically makes no sense to say either Q or P can copy their inputs because x is a data point (i.e., whose domain is a sequence of words) and z is a latent variable we want to predict (i.e., whose domain is a sentiment label {positive,negative}). This is just to clarify, we are making this analogy because we know in the next step we are going to replace P and Q with neural networks. Cheers!

Therefore, to avoid simulating copy-paste which prevents us from learning informative representations, we need to (similar to autoencoding literature) add more constraints in the maximization process. One reasonable assumption would be to make sure the approximate posterior distribution $Q(z|x)$ is not going to be too far away from our belief about parameter z. to formulate this, we can consider a prior distribution over the latent variables, $P(z)$, and make sure $Q(z|x)$, remains close to the prior. This, right away, prevents the copying procedure because the prior probability is independent from x, and contains our prior belief about how the distribution over z should look like. Mathematically speaking, a way to formulate this is to minimize the distance between the two distributions. This is where the second term enters:

$$\texttt{KL-divergence}(Q(z|x)||P(z))$$

As discussed in Subsection 2.1 and proved in Appendix C, this term can be computed analytically assuming $Q(z|x)$ and $P(z)$ are N-dimensional Multivariate Gaussian distributions $Q(z|x) = \mathcal{N}(\mu_1, \Sigma_1)$ and $P(z) = \mathcal{N}(\mu_2, \Sigma_2)$, (both with *diagonal covariances*[2]). In fact, for computational reasons, [Kingma and Welling, 2013] goes one step further and assumes instead $P(z) = \mathcal{N}(0, \mathcal{I})$. This simplifies this computation even further,

$$\begin{aligned}
\texttt{KL-divergence}(Q(z|x)||P(z)) &= \int Q(z|x) \log \frac{Q(z|x)}{P(z)} dz \\
&= \frac{1}{2} \sum_{i=1}^{N} \log \sigma_{i,2}^2 - \log \sigma_{i,1}^2 - 1 + \frac{\sigma_{i,1}^2 + (\mu_{i,1} - \mu_{i,2})^2}{\sigma_{i,2}^2} \\
&= \frac{1}{2} \sum_{i=1}^{N} \log 1 - \log \sigma_{i,1}^2 - 1 + \frac{\sigma_{i,1}^2 + (\mu_{i,1} - 0)^2}{1} \\
&= \frac{1}{2} \sum_{i=1}^{N} - \log \sigma_{i,1}^2 - 1 + \sigma_{i,1}^2 + \mu_{i,1}^2
\end{aligned} \tag{2.7}$$

where $\sigma_{i,1}$ and $\mu_{i,1}$ denote the ith elements on the diagonal of the covariance matrix Σ_1, and mean vector μ_1 of $Q(z|x)$, respectively. Similarly, $\sigma_{i,2}$ and $\mu_{i,2}$ are defined for $P(z)$.

As we maximize the `ELBO` according to Eq. 2.6, we need to maximize the quality of the encoding-decoding component in the first term, while minimizing the distance between our prior belief and the learned posterior distribution in the second term, acting as a regularizer.

Note III We did not make any assumption about the mathematical form of $P(z)$, $Q(z|x)$. It is worth pointing out that in Bayesian literature conjugacy between the prior and posterior distribution is an important property. It simply means if we choose the prior to be a certain distribution (i.e., a Gaussian), then multiplying it by the likelihood term will generate a posterior distribution that belongs to the same family (i.e., also a Gaussian).

2.3 Variational Autoencoder Architecture

Given the discussion above, it becomes clear that a corresponding VAE can be considered, where on the encoder side x is encoded as z, and on the decoder side z is decoded back to x. The function by which encoding and

[2]The choice of diagonal covariance is only to simplify the computations. Without imposing any assumption, the `KL-divergence` of two Multivariate Gaussians is always tractable.

decoding happens can have any architecture (MLP, RNN, LSTM, etc), we just need to make sure that the encoder simulates $Q(z|x)$, while the decoder simulates $P(x|z)$.

For the encoder component, due to computational reasons we assumed $Q(z|x)$ to be a Multivariate Gaussian with diagonal covariance. This means the function simulating Q is producing the μ and Σ of this Gaussian distribution. As for the decoder part, we haven't made any assumption about $P(x|z)$ (i.e., it can be Gaussian, Bernoulli, etc).

In order to evaluate the objective function in Eq. 2.6 for learning the autoencoder, we need to make an approximation,

$$\Big\langle \log P(x|z) \Big\rangle_{Q(z|x)} \approx \frac{1}{K} \sum_{z_i \in \{z_k\}_{k=1}^K} \log P(x|z_i)$$

where $\{z_k\}_{k=1}^K$ are sampled from $Q(z|x)$. Given that the function simulating $Q(z|x)$ generates its mean and covariance, there are two possibilities to generate samples from $Q(z|x)$: i) directly sample from $Q(z|x) = \mathcal{N}(\mu_1, \Sigma_1)$ using Gibbs sampling, or ii) to sample via reparametrization, as shown in Appendix D.[3]

But, is there any reason to choose one over the other? Consider the case where we need to generate K samples. In the case of Gibbs sampling, we need to wait for the simulating function to produce the (μ_1, Σ_1), then the sampling would begin. In the case of the reparametrization, we can sample $\{y_k\}_{k=1}^K$ random draws from $\mathcal{N}(0, \mathcal{I})$ even before we pass x to the simulating function, and once the function produced (μ_1, Σ_1), we would just need to evaluate the following,

$$_k = \mu_1 + \Sigma_1 \times y_k \quad \text{for each} \quad y_k \in \{y_k\}_{k=1}^K \tag{2.8}$$

in order to get $\{z_k\}_{k=1}^K$. This is much more efficient than Gibbs sampling. The inefficiency of Gibbs sampling becomes much more obvious if we consider the training phase, where we need to do this for each training instance. [4] But this is not the only reason!

Another reason reveals itself if we think about the training phase, where the gradient of the whole objective function in Eq. 2.6 is taken. As mentioned in Section 1.2, the Monte Carlo gradient estimation relies on a strong assumption, see Eq. 1.2. It turns out that the reparametrization, see Appendix D, allows us to simply avoid this assumption and at the same time get a well-behaved gradient estimator. This is due to the following equivalency,

$$\nabla_\theta \langle f_\theta(z) \rangle_{Q_\theta(z)} = \langle \nabla_\theta f_\theta(z) \rangle_{P_\gamma(y)} \quad \text{where} \quad z = g(\theta, y) \quad \text{and} \quad z \sim P_\gamma(y) \tag{2.9}$$

which replaces the expectation under the distribution $Q(z|x)$, which we are optimizing, with the expectation under a fixed independent distribution $P_\gamma(y)$. This improvement is more apparent in our case, Eq. 2.5, where $f_\theta(z)$ itself can be a function of $Q(z|x)$, for instance $\log \frac{P(z)}{Q(z|x)}$.

Rewriting our objective function, Eq. 2.6, with the reparametrization of Eq. 2.8 and the assumption for the `KL-divergence`$(Q(z|x)||P(z))$, Eq. 2.7, results in,

$$\Big\langle \log P(x|z) \Big\rangle_{Q(z|x)} - \texttt{KL-divergence}(Q(z|x)||P(z)) \approx \frac{1}{K} \sum_{z_i \in \{\mu_1 + \Sigma_1 \times y_k : y_k \sim \mathcal{N}(0,\mathcal{I})\}_{k=1}^K} \log P(x|z_i)$$
$$- \frac{1}{2} \sum_{i=1}^{N} - \log \sigma_{i,1}^2 - 1 + \sigma_{i,1}^2 + \mu_{i,1}^2$$

Note IV Having learned the parameters, each component of autoencoder can stand alone. The decoder component can be seen as a generative model, which given the latent variable z generates x. While the encoder is seen as inference model, which given x infers the latent variable z.

[3] Noting that we assumed the $Q(z|x)$ distribution to have diagonal covariance, then the sampling is essentially decomposed to independent samples along each dimensions, via univariate Gaussians, $\mathcal{N}(\mu_{i,1}, \sigma_{i,1})$ where $\sigma_{i,1}$ denotes the value of (i,i) element of the covariance matrix Σ_1, and $\mu_{i,1}$ denotes the ith element of mean vector μ_1.).

[4] In practice, mini-batching is done in training so this is done only for a subset of training dataset, but still quite costly

Appendix A

On non-negativity of KL-divergence

Considering the Jensen's inequality,

$$\left\langle f(g(x)) \right\rangle_{P(x)} \leq f\left(\left\langle g(x) \right\rangle_{P(x)} \right) \quad \text{when } f \text{ is strictly concave: } f'' < 0$$
$$\left\langle f(g(x)) \right\rangle_{P(x)} \geq f\left(\left\langle g(x) \right\rangle_{P(x)} \right) \quad \text{when } f \text{ is strictly convex: } f'' > 0$$

given that $\log_c$ is a concave/convex function depending on the value of c:

$$\frac{d \log_c(x)}{dx} = \frac{1}{x \ln c}$$
$$\frac{d^2 \log_c(x)}{d^2 x} = \frac{-\ln c}{(x \ln c)^2} = \frac{-1}{x^2 \ln c} \quad \text{strictly concave if } c > 1, \text{ and strictly convex if } 0 < c < 1$$

very often when we use logarithm we mean the natural logarithm (ln), hence:

$$\frac{d^2 \ln(x)}{d^2 x} = \frac{-1}{x^2}$$

which is strictly concave. Applying Jensen's inequalities to (−KL-divergence):

$$\begin{aligned}
-\texttt{KL}(Q(z|x)||P(z|x)) &= -\int Q(z|x) \log \frac{Q(z|x)}{P(z|x)} dz = \int Q(z|x) \log \frac{P(z|x)}{Q(z|x)} dz \\
&\Rightarrow \int Q(z|x) \log \frac{P(z|x)}{Q(z|x)} dz \leq \log \int Q(z|x) \frac{P(z|x)}{Q(z|x)} dz \\
&\Rightarrow \int Q(z|x) \log \frac{P(z|x)}{Q(z|x)} dz \leq \log 1 \\
&\Rightarrow \int Q(z|x) \log \frac{P(z|x)}{Q(z|x)} dz \leq 0 \\
&\Rightarrow \int -Q(z|x) \log \frac{P(z|x)}{Q(z|x)} dz \geq 0 \\
&\Rightarrow \int Q(z|x) \log \frac{Q(z|x)}{P(z|x)} dz \geq 0 \Rightarrow \texttt{KL}(Q(z|x)||P(z|x)) \geq 0
\end{aligned}$$

proves that KL-divergence is always non-negative.

Appendix B

Monte Carlo Gradient for the Variational Distribution

To maximize the `ELBO`, the gradient with respect to the parameters of the variational distribution Q is,

$$
\begin{aligned}
\nabla\left(\left\langle \log\frac{P(x,z)}{Q(z|x)}\right\rangle_{Q(z|x)}\right) &= \int\left(\nabla Q(z|x)\log P(x,z) - \nabla Q(z|x)\log Q(z|x) - Q(z|x)\frac{\nabla Q(z|x)}{Q(z|x)}\right)dz \\
&= \int\left(\nabla Q(z|x)\log P(x,z) - \nabla Q(z|x)\log Q(z|x)\right) - \int Q(z|x)\frac{\nabla Q(z|x)}{Q(z|x)}dz \\
&= \int\left(\nabla Q(z|x)\log P(x,z) - \nabla Q(z|x)\log Q(z|x)\right) - \int \nabla Q(z|x)dz \\
&= \int\left(\nabla Q(z|x)\log P(x,z) - \nabla Q(z|x)\log Q(z|x)\right) - \nabla\int Q(z|x)dz \\
&= \int\left(\nabla Q(z|x)\log P(x,z) - \nabla Q(z|x)\log Q(z|x)\right)dz - \nabla 1 \\
&= \int\left(\underbrace{\nabla Q(z|x)}_{Q(z|x)\nabla\log Q(z|x)}\log P(x,z) - \underbrace{\nabla Q(z|x)}_{Q(z|x)\nabla\log Q(z|x)}\log Q(z|x)\right)dz \\
&= \left\langle \nabla\log Q(z|x)\Big(\log P(x,z) - \log Q(z|x)\Big)\right\rangle_{Q(z|x)} \\
&= \left\langle \nabla\log Q(z|x)\Big(\log\frac{P(x,z)}{Q(z|x)}\Big)\right\rangle_{Q(z|x)}
\end{aligned}
$$

Appendix C

KL-divergence of two N-dimensional Multivariate Gaussians with Diagonal Covariance

Two N-dimensional multivariate Gaussian with diagonal covariance matrix could be viewed simply as a collection of N independent Gaussian-distributed random variables (hence the joint distribution turns into a product of N independent Gaussian-distributed random variables) with means and variances μ_i and σ_i^2. This means, for multivariate $Q(z|x)$ and $P(z)$ distributions with this property, we can rewrite their KL-divergence as,

$$\int Q(z|x) \log \frac{Q(z|x)}{P(z)} dz = \sum_{i=1}^{N} \int Q_i(z|x) \Big(\log \frac{Q_i(z|x)}{P_i(z)} \Big) dz \tag{C.1}$$

So, let us consider the univariate case, $Q(z|x) = \mathcal{N}(\mu_1, \sigma_1)$ and $P(z) = \mathcal{N}(\mu_2, \sigma_2)$, where KL-divergence$(Q(z|x)||P(z))$ is computed as,

$$\begin{aligned}
&\int Q(z|x) \log \frac{Q(z|x)}{P(z)} dz = \int Q(z|x) \Big(\log Q(z|x) - \log P(z) \Big) dz \\
&= \int Q(z|x) \Big(\log \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp - \Big(\frac{z-\mu_1}{\sqrt{2}\sigma_1} \Big)^2 - \log \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp - \Big(\frac{z-\mu_2}{\sqrt{2}\sigma_2} \Big)^2 \Big) dz \\
&= \int Q(z|x) \Big(\log \frac{\sqrt{2\pi\sigma_2^2}}{\sqrt{2\pi\sigma_1^2}} - \Big(\frac{z-\mu_1}{\sqrt{2}\sigma_1} \Big)^2 + \Big(\frac{z-\mu_2}{\sqrt{2}\sigma_2} \Big)^2 \Big) dz \\
&= \int Q(z|x) \Big(\log \frac{\sigma_2}{\sigma_1} - \Big(\frac{z-\mu_1}{\sqrt{2}\sigma_1} \Big)^2 + \Big(\frac{z-\mu_2}{\sqrt{2}\sigma_2} \Big)^2 \Big) dz \\
&= \Big\langle \log \frac{\sigma_2}{\sigma_1} - \Big(\frac{z-\mu_1}{\sqrt{2}\sigma_1} \Big)^2 + \Big(\frac{z-\mu_2}{\sqrt{2}\sigma_2} \Big)^2 \Big\rangle_{Q(z|x)} \\
&= \log \frac{\sigma_2}{\sigma_1} + \Big\langle - \Big(\frac{z-\mu_1}{\sqrt{2}\sigma_1} \Big)^2 \Big\rangle_{Q(z|x)} + \Big\langle \Big(\frac{z-\mu_2}{\sqrt{2}\sigma_2} \Big)^2 \Big\rangle_{Q(z|x)} \\
&= \log \frac{\sigma_2}{\sigma_1} - \frac{1}{2\sigma_1^2} \Big\langle (z-\mu_1)^2 \Big\rangle_{Q(z|x)} + \frac{1}{2\sigma_2^2} \Big\langle (z-\mu_2)^2 \Big\rangle_{Q(z|x)} \\
&= \log \frac{\sigma_2}{\sigma_1} - \frac{1}{2\sigma_1^2} \sigma_1^2 + \frac{1}{2\sigma_2^2} \Big\langle (z-\mu_2)^2 \Big\rangle_{Q(z|x)} \\
&= \log \frac{\sigma_2}{\sigma_1} - \frac{1}{2} + \frac{1}{2\sigma_2^2} \Big\langle (z+\mu_1-\mu_1-\mu_2)^2 \Big\rangle_{Q(z|x)}
\end{aligned}$$

$$
\begin{aligned}
&= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2} + \frac{\left\langle (z-\mu_1)^2 + (\mu_1-\mu_2)^2 + 2(z-\mu_1)(\mu_1-\mu_2) \right\rangle_{Q(z|x)}}{2\sigma_2^2} \\
&= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2} + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2 + 2(\mu_1-\mu_2)\left\langle z-\mu_1 \right\rangle_{Q(z|x)}}{2\sigma_2^2} \\
&= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2} + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2 + 2(\mu_1-\mu_2)(\left\langle z \right\rangle_{Q(z|x)} - \left\langle \mu_1 \right\rangle_{Q(z|x)})}{2\sigma_2^2} = \\
&= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2} + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2 + 2(\mu_1-\mu_2)(\mu_1-\mu_1)}{2\sigma_2^2} \\
&= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2} + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2}{2\sigma_2^2}
\end{aligned}
$$

which completes the proof for the univariate case. For the multivariate case, given Eq. C.1 we can rewrite it as,

$$
\begin{aligned}
\int Q(z|x)\log\frac{Q(z|x)}{P(z)}dz &= \sum_{i=1}^{N}\log\frac{\sigma_{i,2}}{\sigma_{i,1}} - \frac{1}{2} + \frac{\sigma_{i,1}^2 + (\mu_{i,1}-\mu_{i,2})^2}{2\sigma_{i,2}^2} \\
&= \frac{1}{2}\sum_{i=1}^{N} 2\log\frac{\sigma_{i,2}}{\sigma_{i,1}} - 1 + \frac{\sigma_{i,1}^2 + (\mu_{i,1}-\mu_{i,2})^2}{\sigma_{i,2}^2} \\
&= \frac{1}{2}\sum_{i=1}^{N}\log\sigma_{i,2}^2 - \log\sigma_{i,1}^2 - 1 + \frac{\sigma_{i,1}^2 + (\mu_{i,1}-\mu_{i,2})^2}{\sigma_{i,2}^2}
\end{aligned} \tag{C.2}
$$

where $\sigma_{i,1}$ denotes the value of (i,i) element of the covariance matrix Σ_1, and $\mu_{i,1}$ denotes the ith element of mean vector μ_1.

Appendix D

Reparameterization

In case that dealing with a particular distribution is inefficient, for many distributions we can simulate genuine samples by generating samples from other distributions and transforming them. In here, we list some of the popular transformations:

- Dirichlet Distribution: With a source of Gamma-distributed random variables, one can easily sample a random vector $x = (x_1, \ldots, x_K)$ from the K-dimensional Dirichlet distribution with parameters $(\alpha_1, \ldots, \alpha_K)$. First, draw "K" independent random samples $y_1, \ldots, y_K$ from Gamma distributions each with density:

$$\text{Gamma}(\alpha_i, 1) = \frac{y_i^{\alpha_i - 1} e^{-y_i}}{\Gamma(\alpha_i)},$$

  and then we normalize the draws

$$x_i = \frac{y_i}{\sum_{j=1}^{K} y_j},$$

  the resulting vector is Dirichlet distributed.

- Kumaraswamy Distribution (or Beta Distribution): samples can be drawn via the Kumaraswamy's closed-form inverse CDF transform by first drawing from a uniform distribution:

$$y \sim \mathcal{U}(0, 1),$$

  and then transforming,

$$x = (1 - y^{\frac{1}{b}})^{\frac{1}{a}}$$

  where a and b are Kumaraswamy's parameters. The resulting sample is Kumaraswamy (or Beta, if $a = 1$ or $b = 1$ or both) distributed (with these parameters).

- Gumbel Distribution: using its closed-form inverse CDF can generate samples, by first drawing drawing from a uniform distribution:

$$y \sim \mathcal{U}(0, 1),$$

  and then transforming,

$$\frac{\ln(\alpha) + \ln(\ln(-(x-1)^{-1}))}{\beta}$$

  where α and β are Gumbel's parameters. The resulting sample is Gumbel (with these parameters) distributed.

- Gaussian Distribution: using a simple transformation, samples from a standard Gaussian could be transformed to any other Gaussian distribution, by first:

$$y \sim \mathcal{N}(0, 1)$$

  and then transforming,

$$x = \mu + \sigma y$$

  where μ and σ are the parameters of the Gaussian distribution. The resulting sample is Gaussian (with these parameters) distributed.

In addition to above, we could generate samples from many distributions using the same tricks: Gamma distribution, Log-Normal, etc.

Bibliography

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

[Maddison et al., 2017] Maddison, C. J., Lawson, J., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. W. (2017). Filtering variational objectives. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6576–6586.

[Mnih and Gregor, 2014] Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1791–1799.

[Mnih and Rezende, 2016] Mnih, A. and Rezende, D. J. (2016). Variational inference for monte carlo objectives. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2188–2196.

[Paisley et al., 2012] Paisley, J. W., Blei, D. M., and Jordan, M. I. (2012). Variational bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.