Variational Autoencoders

Assume we have the following *latent variable model*: $p(x|z)$ with a prior $p(z)$, corresponding to the following *graphical model*

$$z \to x$$

Variational autoencoders essentially establish a connection between graphical models and neural networks. In other words, this above model specifies how data, x, is generated from the hidden variables z. A good model would assign high probabilities $p(x|z)$ to observed data x. Learning such a model would mean maximizing $p(x)$. If $p(x|z)$ is parametrized by θ, this would mean

$$\max_\theta p_\theta(x) = \max_\theta \int_z p(z)p_\theta(x|z)dz$$

However, in practice, the integral is intractable. Furthermore, we cannot really compute the posterior $p(z|x)$, as this would also mean computing $p(x)$, according to Bayes, i.e.,

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x,z)}{p(x)} = \frac{p(x,z)}{\int_z p(x,z)dz}$$

In variational inference, we aim to approximate $p(z|x)$ by a tractable distribution $q_\phi(z|x)$ and aim to optimize

$$\min_\phi \mathrm{KL}\big(q_\phi(z|x)||p(z|x)\big)$$

where

$$\mathrm{KL}(p(x)||q(x)) = \int_x q(x)\log\frac{q(x)}{p(x)}$$

denotes the Kullback-Leibler (KL) divergence between q and p, i.e., a measure of similarity between distributions (note, this is not a distance).

However, this is also not so simple, as

$$\begin{aligned}
\mathrm{KL}(q_\phi(z|x)||p(z|x)) &= \int_z q_\phi(z|x)\log\frac{q_\phi(z|x)}{p(z|x)} \\
&= \int_z q_\phi(z|x)\log\frac{q_\phi(z|x)p(x)}{p(x,z)} \\
&= \int_z q_\phi(z|x)\log\frac{q_\phi(z|x)}{p(x,z)} + \int_z q_\phi(z|x)\log p(x) \\
&= -\int_z q_\phi(z|x)\log\frac{p(x,z)}{q_\phi(z|x)} + \log p(x)\int_z q_\phi(z|x) \\
&= -\int_z q_\phi(z|x)\log\frac{p(x,z)}{q_\phi(z|x)} + \log p(x) \\
&= -L(\phi) + \log p(x)
\end{aligned}$$

We see that minimizing the KL divergence is equivalent to maximizing $L(\phi)$ and $L(\phi)$ only includes $q_\phi(z|x)$ and $p(x,z) = p(x|z)p(z)$, i.e., no intractable integrals.

Noting that the KL divergence is non-negative, we also note the following:

$$\begin{aligned}
\mathrm{KL}(q_\phi(z|x)||p(z|x)) &= -L(\phi) + \log p(x) \\
L(\phi) &= \log p(x) - \mathrm{KL}(q_\phi(z|x)||p(z|x)) \\
&\Rightarrow L(\phi) \leq \log p(x)
\end{aligned}$$

Hence, $L(\phi)$ is a lower-bound on $\log p(x)$ and since $p(x)$ is called the evidence, this bound is called the evident lower bound (ELBO).

Lets turn this around now and assume we want to learn $p_\theta(x|z)$ and fix q (i.e., not doing posterior inference). L now depends on θ, as

$$L(\theta) = \int_z q(z|x) \frac{p(z)p_\theta(x|z)}{q(z|x)}$$

We know that L lower bounds $\log p(x)$, hence by maximizing L we also maximize $\log p(x)$. Actually, we can do both at the same time, i.e., maximize L w.r.t. θ and ϕ, i.e.,

$$\begin{aligned}
\theta^*, \phi^* &= \arg\max_{\theta,\phi} L(\theta, \phi) \\
&= \arg\max_{\theta,\phi} \int_z q_\theta(z|x) \frac{p(z)p_\theta(x|z)}{q_\phi(z|x)} \\
&= \mathbb{E}_q \left[\log \frac{p(z)p_\theta(x|z)}{q_\phi(z|x)} \right]
\end{aligned}$$

Estimating this quantity requires sampling from $q_\theta(z|x)$, i.e., obtaining $z_1, \ldots, z_N$ to compute

$$L(\theta, \phi) = \frac{1}{N} \sum_i \log p_\theta(x, z_i) - \log q_\phi(z_i|x)$$

For gradient computation, θ is no problem, as z_i is sampled and p_θ is inside the sum. However, ϕ is the problem here, as the samples are from q_ϕ and q_ϕ obviously depends on ϕ. We cannot simply compute the gradient w.r.t. ϕ as it appears in the distribution with respect to which we take the expectation. Fortunately, the re-parametrization trick will save us. What if we would have

$$z = g_\phi(\epsilon, x), \epsilon \sim p(\epsilon)$$

where $p(\epsilon)$ is some noise distribution without parameters and g_ϕ is differentiable? In that case, we have

$$\mathbb{E}_p(\epsilon)[\log p(x, z) - \log q_\phi(g_\phi(\epsilon, x)|x)]$$

and we are done.

In a classic variational autoencoder, the **prior** $p(z)$ is assumed to be unit Gaussian $N(0, I)$ and $p_\theta(x|z)$ depends on the data distribution. The **approximate posterior** is fixed to a Gaussian as well and we can apply the reparametrization trick easily. Assuming $p(\epsilon) \sim N(0, I)$, we can set $z = g_\phi(\epsilon, x) = \mu_\phi(x) + \epsilon \cdot \sigma_\phi(x)$, where $\mu_\phi : X \to \mathbb{R}^n$ and $\sigma_\phi : X \to \mathbb{R}^n$ are two parametrized functions mapping from x to the mean and standard deviation of a multivariate Gaussian. This mapping can be implemented by a neural network. Sampling z_i then simply amounts to sampling from $N(0, I)$, followed by adding the mean and multiplying by the standard deviation. In our previous notation

$$q_\phi(z|x) = N(z; \mu_\phi(x), \sigma_\phi(x))$$

and

$$L(\theta, \phi) \approx \frac{1}{N} \sum_i \log p_\theta(x, z_i) - \log q_\phi(z_i|x), z_i = \mu_\phi(x) + \epsilon \cdot \sigma_\phi(x), \epsilon \sim N(0, I)$$

with $z_1, \ldots, z_N$. A simple expansion of what we had before shows:

$$\begin{aligned}
\theta^*, \phi^* &= \arg\max_{\theta,\phi} L(\theta, \phi) \\
&= \arg\max_{\theta,\phi} \int_z q_\theta(z|x) \frac{p(z)p_\theta(x|z)}{q_\phi(z|x)} \\
&= \arg\max_{\theta,\phi} \mathbb{E}_q \left[\log \frac{p(z)p_\theta(x|z)}{q_\phi(z|x)} \right] \\
&= \arg\max_{\theta,\phi} \mathbb{E}_q \left[\log \frac{p(z)}{q_\phi(z|x)} \right] + \mathbb{E}_q[q_\phi(z|x)] \\
&= -\mathrm{KL}(q_\phi(z|x)||p(z)) + \mathbb{E}_q[p_\theta(x|z)] \\
&= \frac{1}{2} \sum_d (1 + \log \sigma^2_{\phi,d}(x) - \mu^2_{\phi,d}(x) - \sigma^2_{\phi,d}(x)) + \mathbb{E}_q[p_\theta(x|z)]
\end{aligned}$$

as both $p(z)$ and $q_\phi(z|x)$ are Gaussian (in fact, $p(z)$ is unit Gaussian), for which the KL divergence has a closed form expression. As before, the second expectation term is approximated via N random samples z_i (in reality, this is not done as we train with mini-batches and it has been observed that if the batch size is large enough, sampling more than one point does not make a huge difference empirically).

Conceptually, we implement $\mu_\phi(x)$ and $\sigma_\phi(x)$ with a neural network, i.e., the encoding part of the autoencoder. This encoder produces, given x, a mapping to the mean and standard deviation of a Gaussian. $p_\theta(x|z)$ is also implemented as a neural network, i.e., the decoder. The decoder takes a sample z_i (computed as above) and decodes it into $\hat{x}$. For MNIST (with binary outputs), we can simply use **binary cross entropy (BCE)** to maximize

$$\frac{1}{N}\sum_i \log p_\theta(x|z_i)$$

i.e., the BCE between x and the decoded $\hat{x}$. So, practically, in an implementation (for MNIST) we have

- $-\frac{1}{2}\sum_d(1+\log\sigma^2_{\phi,d}(x)-\mu^2_{\phi,d}(x)-\sigma^2_{\phi,d}(x))$, added to
- BCE between x and $\hat{x}$ (to minimize $\mathbb{E}_q[p_\theta(x|z)]$)

If the outputs are real-valued (and assuming Gaussian outputs), we can use the mean-squared error (MSE), instead of BCE. In other words, the BCE measures **reconstruction** error (as in a classic autoencoder) and the KL-div. loss sits on top of the latent codes.