Auto-encoding Variational Bayes

lruczu

These notes are fully based on the great research paper: "Auto-encoding Variational Bayes" (Diederik P. Kingma, Max Welling). Just like in classic variational inference problem, we would like to compute posterior distribution of latent variables:

$$p(\mathbf{z}|\mathbf{x}) \tag{1}$$

In CAVI algorithm we posit a family of distributions D, from which we select one that is the closest to the real posterior distribution. This family consists of factorized functions, each factor for each latent variable. Such approach not always is possible. Putting aside when it is and when it isn't, let's introduce another algorithm. First, let's recall the definition of elbo

$$ELBO(q) = E_{q(\mathbf{z})}[log(p(\mathbf{z}, \mathbf{x}))] - E_{q(\mathbf{z})}[q(\mathbf{z})] \tag{2}$$

Now, imagine the complete joint distribution which when factorized, describes the generative process.

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{3}$$

Latent variable $\mathbf{z}$ is taken from some prior distribution and then, conditionally on z, x is produced. Let's assume that the above distributions are parameterized by some vector θ. We can write

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

Besides, let's assume that pdfs are differentialbe w.r.t. θ. We can also assume that $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable and we posit a variational family of distributions parameterized by ϕ.

$$q_\phi(\mathbf{z}|\mathbf{x}) \tag{5}$$

Now, we can rewrite the definition of elbo as:

$$ELBO(q_\phi(\mathbf{z}|\mathbf{x})) = E_{q_\phi(\mathbf{z}|\mathbf{x})}[log(p_\theta(\mathbf{x}, \mathbf{z})) - log(q_\phi(\mathbf{z}|\mathbf{x}))] \tag{6}$$

We have two kinds of variables we want to optimize (to make elbo the biggest possible), the generative parameters θ and variational parameters ϕ. This is somewhat fishy, because we not only choose variational distribution to be as close to the real posterior as possible, we also tweak real posterior parameters to move it in the direction of the former. This is because we don't know real

posterior (θ^*) and here by "real", we mean the generative process that is also optimized. In order to see that elbo is useful in our context, let's rewrite elbo.

$$\begin{aligned} ELBO(q_\phi(\mathbf{z}|x_i)) = E_{q_\phi(\mathbf{z}|x_i)}[log(p_\theta(x_i,\mathbf{z})) - log(q_\phi(\mathbf{z}|x_i))] = \\ E_{q_\phi(\mathbf{z}|x_i)}[log(p_\theta(\mathbf{z}|x_i)) + log(p_\theta(x_i)) - log(q_\phi(\mathbf{z}|x_i))] = \\ E_{q_\phi(\mathbf{z}|x_i)}[log(p_\theta(x_i))] - KL(q_\phi(\mathbf{z}|x_i)||p_\theta(\mathbf{z}|x_i)) = \\ log(p_\theta(x_i)) - KL(q_\phi(\mathbf{z}|x_i)||p_\theta(\mathbf{z}|x_i)) \end{aligned} \tag{7}$$

Above x_i is fixed. It is just one point from our data.

$$\begin{aligned} \sum_{i=1}^{N} log(p_\theta(x_i)) = \sum_{i=1}^{N} ELBO(q_\phi(\mathbf{z}|x_i)) + KL(q_\phi(\mathbf{z}|x_i)||p_\theta(\mathbf{z}|x_i)) \geq \\ \sum_{i=1}^{N} ELBO(q_\phi(\mathbf{z}|x_i)) \end{aligned} \tag{8}$$

By maximizing elbo, which is the lowerbound of likelihood function, we hopefully maximize the latter. Let's rewrite elbo in the form that will be convenient later on.

$$\begin{aligned} \sum_{i=1}^{N} ELBO(q_\phi(\mathbf{z}|x_i)) = \sum_{i=1}^{N} E_{q_\phi(\mathbf{z}|x_i)}[log(p_\theta(x_i,\mathbf{z})) - log(q_\phi(\mathbf{z}|x_i))] = \\ \sum_{i=1}^{N} E_{q_\phi(\mathbf{z}|x_i)}[log(p_\theta(x_i|\mathbf{z})) + log(p_\theta(\mathbf{z})) - log(q_\phi(\mathbf{z}|x_i))] = \\ \sum_{i=1}^{N} E_{q_\phi(\mathbf{z}|x_i)}[log(p_\theta(x_i|\mathbf{z}))] - KL(q_\phi(\mathbf{z}|x_i)||p_\theta(\mathbf{z})) \end{aligned} \tag{9}$$

The second term of the RHS is not a problem, because very often it can be computed analytically (we choose prior on $\mathbf{z}$ to make it really easy). The first term is more difficult part. It is computed using Monte Carlo estimation. The recipe is quite simple. We have to compute gradients of adjustable parameters of variational and generative distribution and apply update rule on parameters. But first, it has to be shown that such procedure is feasible, because we have vector of random variable $\mathbf{z}$ to take care of (data is fixed, and inference on parameters are not taken into account in basic form of the algorithm). Let's start by concentrating on generative parameters θ.

$$\begin{aligned} \nabla_\theta ELBO(q_\phi(\mathbf{z}|x_i)) = \nabla_\theta E_{q_\phi(\mathbf{z}|x_i)}[log(p_\theta(x_i|\mathbf{z}))] = \\ \nabla_\theta \int q_\phi(\mathbf{z}|x_i) log(p_\theta(x_i|\mathbf{z})) d\mathbf{z} = \\ \int \nabla_\theta q_\phi(\mathbf{z}|x_i) log(p_\theta(x_i|\mathbf{z})) d\mathbf{z} = \int q_\phi(\mathbf{z}|x_i) \nabla_\theta log(p_\theta(x_i|\mathbf{z})) d\mathbf{z} = \\ E_{q_\phi(\mathbf{z}|x_i)}[\nabla_\theta log(p_\theta(x_i|\mathbf{z}))] \end{aligned} \tag{10}$$

To do the same for variational parameters ϕ, we have to apply reparametrization trick. Before doing so, let's specify a bit variational distribution. It is often assumed, that

$$q_\phi(\mathbf{z}|x_i) = \mathcal{N}(\mathbf{z}; \mathbf{mean}(\phi, x_i), \mathbf{var}(\phi, x_i)) \tag{11}$$

Variance is specified as diagonal matrix for simplicity. Both variance and mean are the result of neural network with weights ϕ applied on the point x_i. Given the above, posterior $\mathbf{z}$ can be written as (reparametrization trick):

$$\mathbf{z}|x_i = \mathbf{mean}(\phi, x_i) + \sqrt{\mathbf{var}(\phi, x_i))} * \epsilon = g(x_i, \phi, \epsilon) \tag{12}$$

On the RHS the only random part (given x_i) is ϵ which is generated according to standard normal (written here as p(ϵ)).

$$\begin{aligned} \nabla_\phi ELBO(q_\phi(\mathbf{z}|x_i)) = \nabla_\phi E_{g(x_i,\phi,\epsilon)}[log(p_\theta(x_i|\mathbf{z}))] = \\ \nabla_\phi \int g(x_i, \phi, \epsilon) log(p_\theta(x_i|\mathbf{z})) d\mathbf{z} = \\ \nabla_\phi \int p(\epsilon) log(p_\theta(x_i|\mathbf{z})) d\epsilon = \\ E_{p(\epsilon)}[\nabla_\phi log(p_\theta(x_i|\mathbf{z}))] \end{aligned} \tag{13}$$

The learning procedure is now more clear. Conceptually it looks like the following. Take a point x_i, produce variational mean and variance (first part of neural network, called encoder). Sample some number of ϵ, let's say S. Compute latent samples z_s and apply them to the generative model (second part of neural network, called decoder) and compute $p_\theta(x_i|z_s)$. Take a Monte Monte carlo estimation of it and optimize it through backpropagation algorithm. It can be illustrated as follows (for θ)

$$\nabla_\theta ELBO(q_\phi(\mathbf{z}|x_i)) = \nabla_\theta E_{q_\phi(\mathbf{z}|x_i)}[log(p_\theta(x_i|\mathbf{z}))] = \nabla_\theta \frac{1}{S} \sum_{s=1}^{S} log(p_\theta(x_i|z_s)) \tag{14}$$