E6: Generative AI

Variational Autoencoder

D. Kingma, M. Welling, “Auto-Encoding Variational Bayes”, 2013

https://arxiv.org/abs/1312.6114

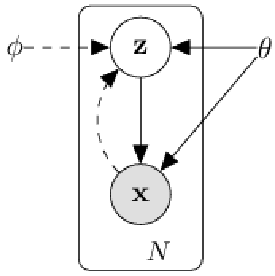

Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$. The variational parameters ϕ are learned jointly with the generative model parameters θ.

2.2 The variational bound

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints $\log p_\theta(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}) = \sum_{i=1}^N \log p_\theta(\mathbf{x}^{(i)})$, which can each be rewritten as:

$$\log p_\theta(\mathbf{x}^{(i)}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \tag{1}$$

The first RHS term is the KL divergence of the approximate from the true posterior. Since this KL-divergence is non-negative, the second RHS term $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$ is called the (variational) *lower bound* on the marginal likelihood of datapoint i, and can be written as:

$$\log p_\theta(\mathbf{x}^{(i)}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\theta(\mathbf{x}, \mathbf{z})\right] \tag{2}$$

which can also be written as:

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] \tag{3}$$

- ➔ Logarithm of the probability of the individual datapoint $\mathbf{x}^{(i)}$ in unknown posteriori distribution $\mathbf{p}_\theta$ which is also intractable as optimization objective. Optimize likelihood of the datapoint to be in the posteriori distribution.
- ➔ KL-Divergence as loss for the encoding part:
  - **$D_{KL}(q_\Phi(z|x^{(i)}) \,||\, p_\theta(z|x^{(i)}))$**
  - Our model **Φ** should learn an inner distribution **$q_\Phi(z)$** conditioned by datapoint **$x^{(i)}$**
  - The latent variable **z** should also fit to the posteriori distribution **$p_\theta(z)$**
- ➔ Loss **L** as the difference between the variational model **Φ** and the data distribution **θ** from each individual datapoint **$x^{(i)}$**

- ➔ Expected Value $\mathbb{E}$ of the probability of datapoint $\mathbf{x}^{(i)}$ conditioned by $\mathbf{z}$ with respect to the learned distribution $\mathbf{q}_\Phi$ conditioned by the datapoint $\mathbf{x}^{(i)}$
- ➔ Variational Lower Bound, because it is an abstraction (the latent space **Z** of the Autoencoding part has smaller dimensions than the real output **X**)

Further reading: https://mbernste.github.io/posts/elbo/

Auto-Encoding Variational Bayes (AEVB) algorithm

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

Reparameterization Trick:

For backpropagation of the loss, we need a differentiable variable.

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)} | \mathbf{z}^{(i,l)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \tag{10}$$

- ➔ Encoder: KL-Divergence with a multivariate normal distribution (Gaussian)
  - Reparameterization Trick for differentiability of the Gaussian
- ➔ Decoder: Negative Log-Likelihood from data point sampled from latent space for each number l of distributions

“We try to fit a multivariate normal distribution p to the real data, and a multivariate normal distribution q from latent space Z of our model to distribution p.”

Application (use FashionMNIST instead of MNIST):

https://github.com/ethanluoyc/pytorch-vae/blob/master/vae.py

Try to interpolate between different samples to generate new variations

```
Z1 = Encoder(x1)        # latent variable 1
Z2 = Encoder(x2)        # latent variable 2
Z = z1 + (z2-z1)/2      # interpolate between the two images (here: center between z1 and z2)
Image, _ = Decoder(z)   # decode latent variable to new image
```