Assignment 3. Deep Generative Models

Maurice Frank
11650656
maurice.frank@posteo.de
Code: github

1 Variational Auto Encoders

We have:

$\mathcal{D} = \{x_n\}_{n=1}^{N}$ the dataset

$x_n \in \{0,1\}^M$ one datapoint, from Bernoulli distribution

1.1 Variational autoencoders vs standard autoencoders

Let us first note down the two objectives of the VAE and a standard sparse autoencoder [1], respectively:

$$\mathbb{E}_{q(z|x_n)}[\log p(x_n|Z)] - D_{KL}(q(Z|x_n)||p(Z))] \tag{1}$$

$$||D_{AE}(E_{AE}(\boldsymbol{X})) - \boldsymbol{X}||^2 + \lambda||E_{AE}(\boldsymbol{X})||_0 \tag{2}$$

For the sparse autoencoder we use D_{AE}, E_{AE} for the Decoder model and Encoder model and λ is our regularization hyperparameter. The main similarity between both is that they have the encoder and decoder pair. The encoder encodes the data into a smaller latent space and the decode generates samples from this space back into our datasets space X.

a) How do they differ in their main function?

We see that both methods vary considerably in their main optimization objective. The VAE objective includes modeling the intractable encoding distribution $p(Z|X)$ through the surrogate $q(Z|X)$. The standard autoencoder on the other side does not actually model the encoding distribution but only has the reconstruction capabilities of those encodings as its objective.

b) Is the standard autoencoder a generative model?

The standard autoencoder is not a generative model as one cannot sample from the implicit latent space. We do have the latent representation and thus can vary the input into the decoder but this does not relate to sampling from the latent manifold of all possible encodings. We can sample from the space the encodings live in but as we do not model the actual distribution this is not a generative model.

c) Can the VAE be a drop-in replacement for a normal autoencoder?

The VAE can be a drop-in replacement for the standard autoencoder in respective applications. Autoencoders are used for compression as they can reduce the input data into the

smaller informative latent space. The VAE does do they same and can be used for the same. Further autoencoders can denoise data by mapping the noisy data close to the noise-less data in the latent space. Again the VAE can do the same. Generally the addition of the VAE is enforcing the continuity of the latent space by introducing $q(\cdot)$. While this opens up new (see generative) applications it does not restrictive the former.

d) Why does VAE generalize better?

The standard autoencoder is missing the is not modelling the encoding distribution. In the VAE our $q(Z|X)$ forces the latent space to be continuos. As such interpolating in the latent space does correspond to interpolating in the data space. During training we therefore build a latent space that is better capable to handling unseen data and can therefore also generalize beyond the trainings data.

1.2 How to sample from $p(Z)$?

We sample using ancestral sampling. First we sample $z_i \sim p(z_n) = \mathcal{N}(0, \mathbb{1}_D)$. We put this sample into the Decoder $f_\theta(\cdot)$ to get our parameters for the bernoulli distribution. Than we sample each pixel from each bernoulli $pixel_m \sim \text{Bern}(x_n^{(m)}|f_\theta(z_n)_m)$ to get the sample from the joint distribution.

1.3 Why is the assumption of $p(Z)$ not restrictive?

Our assumption that $p(Z)$ is a standard-normal distribution does not hinder us from learning a complex manifold for X as we transform it using $f_\theta(\cdot)$. Thus the complexity lies in the mapping from latent z-space and x-space.

1.4 Evaluating $\log p(x_n)$

a) Approximating using Monte-Carlo

We can sample from our distribution of z_n to compute the integral for one x_n:

$$\log p(x_n) = \log \int p(x_n|z_n)p(z_n)\,dz_n \tag{3}$$

$$\approx \log \frac{1}{n} \sum_{i=1}^{n} p(x_n|z_i), \quad z_i \sim p(z) \tag{4}$$

b) Why is Monte-Carlo integration here inefficient?

We sample from $p(Z)$. As we nee the integral for each datapoint this is inefficient with any non trivial dimensionality in Z. The precision of our density estimation of $p(x_n|z_n)$ decreases exponentially as the dimension of x_n increases. As such we would need to increase the number of samples exponentially with higher dimension of z_n to keep the estimation precise. In the real world the hidden embeddings often has hundreds of dimensions. A density dimension that we would have to do for every single input in every inference step is too inefficient to be usable.

1.5 Kullback-Leibler divergence of Gaussians

We assume: $q = \mathcal{N}(\mu_q, \sigma_q^2), \quad p = \mathcal{N}(0, 1)$

a) Two examples of KL-divergence

Checking with the closed form solution in Section b) we see as expected that the divergence is zero for $\sigma_q = 1, \mu_q = 0$ and increases for any deviation from either of those values:

$$Big: \quad (1 + 10^5, 10^5) \implies D_{\text{KL}}(q||p) = 1.0 \times 10^{10} \tag{5}$$

$$Small: \quad (1 + 10^{-5}, 10^{-5}) \implies D_{\text{KL}}(q||p) = 1.49 \times 10^{-10} \tag{6}$$

b) Closed-form solution of the KL-divergence

$$q(x) = \frac{1}{\sqrt{2\pi}\sigma_q} \exp\left[-\frac{1}{2}\left(\frac{x - \mu_q}{\sigma_q}\right)^2\right] \tag{7}$$

$$p(x) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}x^2\right] \tag{8}$$

$$D_{\mathrm{KL}}(q||p) = -\mathbb{E}_{q(x)}\left[\log\frac{p(x)}{q(x)}\right] \tag{9}$$

$$= -\mathbb{E}_{q(x)}\left[\log\left(\frac{1}{\sqrt{2\pi}}\right) - \frac{1}{2}x^2 - \log\left(\frac{1}{\sqrt{2\pi}}\right) + \log\sigma_q + \frac{1}{2}\left(\frac{x - \mu_q}{\sigma_q}\right)^2\right] \tag{10}$$

$$= -\mathbb{E}_{q(x)}\left[\frac{1}{2}\left(\left(\frac{x - \mu_q}{\sigma_q}\right)^2 - x^2\right)\right] - \mathbb{E}_{q(x)}\left[\log\sigma_q\right] \tag{11}$$

$$= -\frac{1}{2}\mathbb{E}_{q(x)}\left[\left(\frac{x - \mu_q}{\sigma_q}\right)^2 - x^2\right] - \log\sigma_q \tag{12}$$

$$= -\frac{1}{2}\left(\frac{1}{\sigma_q^2}\mathbb{E}_{q(x)}[(x - \mu_q)^2] - \mathbb{E}_{q(x)}[x^2]\right) - \log\sigma_q \tag{13}$$

$$= -\frac{1}{2}\left(\frac{\sigma_q^2}{\sigma_q^2} - \sigma_q^2 - \mu_q^2\right) - \log\sigma_q \tag{14}$$

$$= \frac{\sigma_q^2 + \mu_q^2 - 1}{2} - \log\sigma_q \tag{15}$$

1.6 Why is ELBO a lower bound?

The evidence lower bound (ELBO) for our VAE is given by:

$$\mathbb{E}_{q(z|\boldsymbol{x}_n)}[\log p(\boldsymbol{x}_n|Z) - D_{\mathrm{KL}}(q(Z|\boldsymbol{x}_n)||p(Z))] \tag{16}$$

We want to maximize our data model $p(\boldsymbol{x}_n)$, or written including the ELBO:

$$p(\boldsymbol{x}_n) = \mathbb{E}_{q(z|\boldsymbol{x}_n)}[\log p(\boldsymbol{x}_n|Z) - D_{\mathrm{KL}}(q(Z|\boldsymbol{x}_n)||p(Z))] + D_{\mathrm{KL}}(q(Z|\boldsymbol{x}_n)||p(Z|\boldsymbol{x}_n)) \tag{17}$$

As KL divergence is strictly > 0 we directly see that the ELBO is a lower bound for our actual objective. Now the distance from the lower bound to the actual evidence is the divergence of our surrogate approximate posterior $q(Z|\boldsymbol{x}_n)$ from the actual posterior $p(Z|\boldsymbol{x}_n)$. The lower bound always stays a lower bound so maximizing the lower bound does also maximize the actual objective.

1.7 Why do we optimize the lower bound instead of the log-probability?

We must optimize the lower bound as we cannot compute $D_{\mathrm{KL}}(q(Z|\boldsymbol{x}_n)||p(Z|\boldsymbol{x}_n))$. Computing this divergence would mean we need to compute $p(z|\boldsymbol{x}_N)$. If we could do this we would not have introduced $q(z|\boldsymbol{x}_n)$.

1.8 What can happen if we push ELBO up?

Basically the two things that can happen is either our data log probability $\log p(\boldsymbol{x}_N)$ rises or the divergence between $q(Z|\boldsymbol{x}_n)$ and $p(Z|\boldsymbol{x}_n)$ gets lower. Both ways are good for us. In the first case the likelihood of the trainings data rises meaning we get a better ML model. The reconstruction gets better. In the second case our surrogate gets more similar to the actual encoding which means we improve our embeddings.

1.9 How can we name the parts of the loss of an VAE?

We can write the loss of the VAE as the per-sample sum of the *reconstruction* and the *regularizer* loss:

$$\mathcal{L}_n^{\text{recon}} = -\mathbb{E}_{q_\phi(z|x_n)}[\log p_\theta(x_n|Z)] \tag{18}$$

$$\mathcal{L}_n^{\text{ref}} = D_{\text{KL}}(q_\phi(Z|x_n)||p_\theta(Z)) \tag{19}$$

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(\mathcal{L}_n^{\text{recon}} + \mathcal{L}_n^{\text{reg}}) \tag{20}$$

As written before $\mathcal{L}_n^{\text{recon}}$ gives us the negative data likelihood of the input sample x_n using its embedding from the encoder $q_\phi(\cdot)$. In other words this gives us the reconstruction probability of x_n using the decoder $p_\theta(\cdot)$ under an already set embedding for x_n, but negative (to get a loss).

As we have an VAE not an standard autoencoder we actually model the distribution $q_\phi(Z|X)$ to have a generalizing model in which we can interpolate inside the embedding space of z. Having the continuos z-space is the models regularization. Our metric for how good our projection from x to z is, is the KL divergence from the actual $p(Z|X)$. Therefore it is fair to say that the divergence is called the regularization loss.

1.10 Writing down the complete loss

We assume for our encoder:

$$q_\phi(z_n|x_n) = \mathcal{N}(z_n)\mu_\phi(x_n, \text{diag}(\Sigma_\phi(x_n))) \tag{21}$$

$$\mathcal{L}_n^{\text{recon}} = -\mathbb{E}_{q_\phi(z|x_n)}[\log p_\theta(x_n|Z)] \tag{22}$$

$$\approx -\log p_\theta(x_n|z_n) \quad \text{with } z_n \sim q_\phi(Z|x_n) \tag{23}$$

$$\mathcal{L}_n^{\text{ref}} = D_{\text{KL}}(q_\phi(Z|x_n)||p_\theta(Z)) \tag{24}$$

$$= \frac{\Sigma_\phi(x_n)^2 + \mu_\phi(x_n)^2 - 1}{2} - \log \Sigma_\phi(x_n) \tag{25}$$

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(\mathcal{L}_n^{\text{recon}} + \mathcal{L}_n^{\text{reg}}) \tag{26}$$

$$= \tag{27}$$

References

[1] Carl Doersch. Tutorial on Variational Autoencoders. URL http://arxiv.org/abs/1606.05908.