Deep Learning- Assignment 3: Deep Generative Models

YaChu Yang
11571276
ya-chu.yang@student.uva.nl

October 17, 2018

Abstract

This assignment is aimed to implement and discuss VAE and GAN.

1 Variational Auto Encoders

Question 1.1

PCA is restricted to a linear map, while VAE can have nonlinear encoder and decoders.

Question 1.2

First, starting by sampling from p(z_n), then we can get a vector z_n, which will be fed into the neural network(f_θ) to get the output($f_\theta(z_n)$). Then we sample from the next generation's distribution, it will be Bernoulli distributions in this case, by conditioning the output generated by the neural network.

Question 1.3

Because Bernoulli distribution is a discrete distribution, which only has value when the random variable equals to 1 or 0, we can't use gradient descent or other optimization techniques to increase $p(x_n|z_n)$ by learning parameters in Equation 4 and p(Z). Since the parameters in p(Z) can't be learned, there is no need to have trainable parameters in it. Thus, we just assume a simple assumption about it, $p(z_n) = N(0, I_D)$.

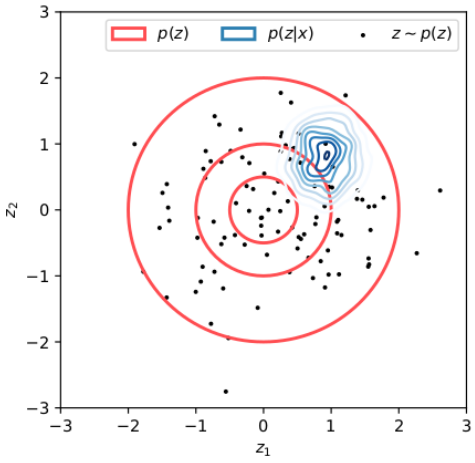


Figure 1: Plot of 2-dimensional latent space and contours of prior and posterior distributions.

Question 1.4

a

$$logp(x_n) = log\mathbb{E}_{p(z_n)}[p(x_n|z_n)] \simeq \frac{1}{L}\sum_{l}^{L} logp(x_n|z_{(n,l)}) \tag{1}$$

b From Figure 2, we can see that if we want to get good estimation on p(x) then we need to get most z being sampled from the blue area(posterior). However, p(z) is high variance in every dimension comparing to p(z|x)'s variance, which means that we need to take a very large number of samples from p(z) to make sure that we also get samples inside the blue area. Thus using Monte-Carlo Integration becomes inefficient especially when the dimensional of z gets larger because the variance of every dimension in z is larger than of p(z|x), then in every dimension, we all need to take a lot of samples to obtain good estimation[1].

Question 1.5

a when $q = N(5, 0.5)$, $D_{KL}(q||p)$ is large
when $q = N(0, 1)$, $D_{KL}(q||p)$ is 0

b $D_{KL}(q||p) = \frac{log(\frac{\sigma_q}{\sigma_p})^2}{2} + \frac{\sigma_q^2 + (\mu_p - \mu_q)^2}{2\sigma_p^2} - \frac{1}{2} = \frac{log\sigma_q^2}{2} + \frac{\sigma_q^2 + \mu_q^2}{2} - \frac{1}{2}$

Question 1.6

First, we can arrange the log probability($logp(x_n)$) to this formula:

$$logp(x_n) = \left(\mathbb{E}_{q(z|x_n)}[logp(x_n|Z)] - D_{KL}(q(Z|x_n)||p(Z))\right) + D_{KL}(q(Z|x_n)||p(Z|X_n)) \tag{2}$$

The quantities on the right-hand side, only the ones with red text are computable, and Kullback–Leibler divergence is always ≥ 0.

$$\therefore logp(x_n) \geq \left(\mathbb{E}_{q(z|x_n)}[logp(x_n|Z)] - D_{KL}(q(Z|x_n)||p(Z))\right) \tag{3}$$

$\therefore$ the quantities on the right hand side are the lower bound of the log probability. [3]

Question 1.7

We can't do optimize on log probability because$D_{KL}(q(Z|x_n)||p(Z|X_n))$, in Formula 2's right-hand side, are not computable due to $p(Z|X_n)$. However, we can do the optimization on the lower bound to make sure that $logp(x, z)$ is also optimized to a certain degree.

Question 1.8

If the lower bound is pushed up, $D_{KL}(q(Z|x_n)||p(Z))$ becomes relative smaller. which means that:

1. Distributions of $p(z_n)$(prior) and $p(z_n|x_n)$(posterior) are similar to each other.
2. The variational distribution, $q(z_n|x_n)$, closely approximate $p(z_n|x_n)$.

Question 1.9

For the reconstruction loss, this term is just a typical maximum likelihood problem if we view q(z|x) as prior knowledge. Thus, it means that given a z as an input, the decoder needs to learn how to reconstruct the data to approximate the real image, which is the reason why it's called reconstruction loss.

For the regularization problem, if we don't include this term, the encoder could give the same digits the representations z in a different region of latent space, and we can still probably get good results on decoder, which is the problem of over-fitting. However, we would like the representation in latent space meaningful, meaning z of same digits would be in the same region of latent space. Thus, including this term, we can make sure the estimated probability distribution q trying to approximate p and avoid over-fitting, which is the reason why it's called regularization loss.

Question 1.10

According to Monte Carlo Integration,

$$\mathcal{L}_n^{recon} = -\mathbb{E}_{q_\phi(z|x_n)}[logp_\theta(x_n|Z)] \simeq \frac{-1}{L}\sum_{l=1}^{L} logp_\theta(x_n|Z_{(n,l)}, \quad where \quad Z_{(n,l)\phi}(z|x_n)$$

$$\mathcal{L}_n^{reg} = D_{KL}(q_\phi(Z|x_n)||p_\theta(Z|X_n)) = \frac{1}{2}\sum_{j=1}^{J}[1 + log\sigma_{(n,j)}^2 - \mu_{(n,j)}^2 - \sigma_{(n.j)}^2]$$

$where \quad J =$ number of dimension of z_n

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}_n^{recon} + \mathcal{L}_n^{reg}$$

$$\simeq \frac{1}{N}\sum_{n=1}^{N}\big(\frac{-1}{L}\sum_{l=1}^{L} logp_\theta(x_n|Z_{(n,l)})\big) + \big(\frac{1}{2}\sum_{j=1}^{J}[1 + log\sigma_{(n,j)}^2 - \mu_{(n,j)}^2 - \sigma_{(n.j)}^2]\big)$$

Question 1.11

First, we need to perform gradient on the loss function to learn parameters in q distribution, so that q(z|x) can approximately match p(z|x). However, because z are sampled from q(z|x) when forward pass, which makes z more like the input of the network and non-continuous, then we can't learn parameters in q distribution through doing gradient descent.

Thus, the authors use reparametrization trick to solve the gradient problem. They use a new distribution $\epsilon \sim N(0, I)$ as an input layer, and then combine with q distribution, $N(\mu_\phi(x), \Sigma_\phi(x))$. Thus, $z = \mu_\phi(x) + \epsilon\Sigma_\phi(x)$. Through this trick, z is still obtained by sampling and continuous, meaning we can do gradient descent to optimize the loss function.

Question 1.12

In class VAE, there are two networks: encoder and decoder. Encoder is a two layer net with ReLu as the activation function for the output layer, and outputs $\mu and \Sigma$, which would be used in reparametrize trick. For the decoder, it is a two layer net with sigmoid as the activation function for the output layer.

In the training loop, the loss function consists of two parts, construction loss and K-L divergence of the encoded distribution. Adam optimizer is used in the training.

Question 1.13

Estimated lower-bounds of your training and validation set in training (Figure 2).

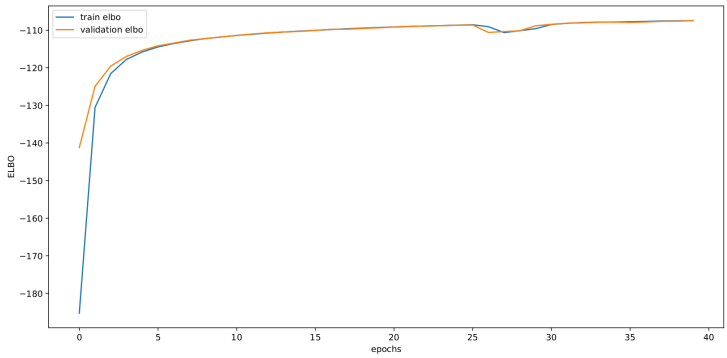


Figure 2: Estimated lower-bounds of your training and validation set in training.

Question 1.14

(a) Samples from model before the training.

(b) Samples from model in the training.

(c) Samples from model after the training.

Figure 3: Visualization of samples from the model at three different points in the training.

2 Generative Adversarial Networks

Question 2.1

Generator: The input of it would be a vector z where z_i is a randomly generated number from latent space. The output of it would be a vector which can be reshaped into a image. Thus, the element of the output represents as a pixel value in the output image.

Discriminator: The input would be an image. The output will be a value indicating whether the input image is real(1) or fake(0).

Question 2.2

For the discriminator($max_D V(D,G)$), the first term is aimed to maximize the chance to recognize the real images as real and the objective of the second term is to recognize generated ones as fake.

For the generator($min_G V(D,G)$), its objective function is the second term ($\mathbb{E}_{p_z(z)}[log(1 - D(G(Z)))]$), which is aimed to optimize G that can fool the discriminator. When the discriminator can't recognize the generated image as fake, the value of the this term would decrease. Thus, the generator would like to minimize the second term in the objective function.

Question 2.3

The value of V(D, G) would be -log(4) after reaching the convergence. The optimal discriminator is $D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$[2]. When the generator is also optimized, $p_g(x)$ would reach to 1, since the generated image is too real to be recognized as fake by the discriminator. Then $D^*_G(x) = \frac{1}{2}$. Thus, $V^*(D,G) = log\frac{1}{2} + log\frac{1}{2} = -log(4)$

Question 2.4

In early training, when the performance of the generator is poor, the discriminator can clearly differentiate between fake images and real images, which leads to vanishing gradient on the second term. ($\nabla_{\theta_g}\mathbb{E}_{p_z}[log(1 - D(G(Z)))] \to 0$)

Vanishing gradient can make the learning of the generator so hard. Thus, rather than training G to minimize the log(1 D(G(Z))) term, we can train G to maximize the log(D(G(Z))) term in early training to solve the vanishing gradient problem. Therefore, the objective function becomes this:
$max_G max_D \;\; \mathbb{E}_{p_{data}(x)}[log(D(x))] + \mathbb{E}_{p_z(z)}[log(D(G(z)))]$

Question 2.5

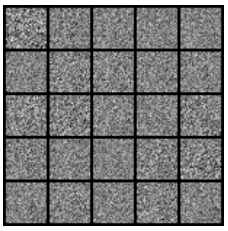
First, the input latent vector into the generator is generated by using a normal distribution with mean = 0, std = 1. This input vector is fed into the generator with 4 hidden layers and 5 linear maps, which generates a vector which can be reshaped into an image. The discriminator is a network with 2 hidden layers. Finally, the training loop alternates between two models. D is first trained with recognizing real or fake data. Then G is learned to fool D.

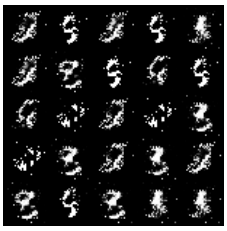
Question 2.6

Visualization of samples from the model (Figure 4).

Question 2.7

Interpolate between two digits in latent space (Figure 5).


(a) Before the training.


(b) During the training.

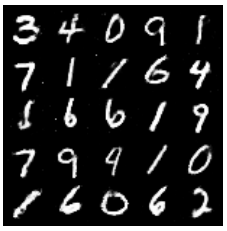

(c) After the training.

Figure 4: Visualization of samples from the model.

Figure 5: Interpolate between two digits in latent space.

3 Conclusion

VAE and GAN all incorporate the idea of latent space. However, for VAE, it learns the distribution of the latent space, GAN doesn't. Thus, in VAE, we can have more understanding on latent space not just a black box, which can make us generate our desired own data more easily.

Second difference is how they evaluate the generated images during the training. For VAE, the decoder learns by comparing its input to its output. Thus, the variety of the generated images depends on how various the input data is. For GAN, the discriminator only learns whether the generated image is real or fake. Therefore, GAN is good at generating various new data.

To conclude it, there are differences between these two methods. They both can generate good data in this case.

References

[1] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.