Deep Learning: Assignment 3

Dhruba Pujary
University of Amsterdam
11576200
dhruba.pujary@student.uva.nl

1 Variational Auto Encoders

1.1 Latent Variable Models

Question 1.1

Variational Auto Encoders and Factor Analysis uses a generative model with latent variable z_i distributed according to some prior distribution $p(z_i)$. The observed variable x_i is then distributed according to a conditional distribution $p(x_i|z_i, \theta)$ where θ are model parameters. Both perform probabilistic, non-linear dimensionality reduction. Probabilistic PCA is a special case of Factor Analysis.

1.2 Decoder: The Generative Part of the VAE

Question 1.2

To sample from this model

1. sample $\mathbf{z}_n \sim \mathcal{N}(0, \mathbf{I}_D)$
2. evaluate $f_\theta(\mathbf{z}_n)_m$ for each m where m is dimensions of x
3. sample $\mathbf{x}_n \sim p(\mathbf{x}_n|\mathbf{z}_n)$ where $p(\mathbf{x}_n|\mathbf{z}_n) = \prod_{m=1}^{M} Bern(\mathbf{x}_n^{(m)}|f_\theta(\mathbf{z}_n)_m)$

Question 1.3

This is not a restricted assumption because any d-dimensional distribution can be generated by a set of d variables that are normally distributed and mapping a complicated function. This mapping can be learned using neural networks. As long as we are able to use a transformation which maps the latent variable Z to any other latent variable that has a more appropriate distribution of the data, the choice of initial distribution doesn't matter that much.

Question 1.4a

$\log p(\mathbf{x}_n)$ can be approximated by sampling a large number of z,i.e M samples where
$\log p(\mathbf{x}_n) \approx \log(\frac{1}{M} \sum_{m=1}^{M} p(\mathbf{x}_n|\mathbf{z}_n^m))$

Question 1.4b

This is inefficient because the data would lie in a very small region in the high dimensional latent space. However, the sampling would consider all the dimension equally. Most of the region will have probability $p(x|z)$ will be nearly zero. Thus a lot of sample would be required to get an accurate estimate of $p(x)$. This problem worsen with increasing latent space dimensionality.

Question 1.5a

$q = \mathcal{N}(\mu_1, \sigma_1) = \mathcal{N}(\mu_q, \sigma_q^2)$
$p = \mathcal{N}(\mu_2, \sigma_2) = \mathcal{N}(0, 1)$

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

$D_{KL} = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$

1. $\mu_q = 0, \sigma_q^2 = 1, D_{KL} = 0$
2. $\mu_q = 0, \sigma_q^2 = \infty, D_{KL} \approx \infty$

Question 1.5b

$D_{KL} = \log \frac{1}{\sigma_q} + \frac{\sigma_q^2 + \mu_q^2}{2} - \frac{1}{2}$

Question 1.6

$\log p(\mathbf{x}_n) - D_{KL}(q(Z|\mathbf{x}_n)||p(Z|x_n)) = \mathrm{E}_{q(z|x_n)}[p(\mathbf{x}_n|Z)] - D_{KL}(q(Z|\mathbf{x}_n)||p(Z))$.

KL is a distance metric which is either zero of positive value. Thus the minimum value $\log p(\mathbf{x}_n)$ can have is the $\mathrm{E}_{q(z|x_n)}[p(\mathbf{x}_n|Z)] D_{KL}(q(Z|\mathbf{x}_n)||p(Z))$.

Question 1.7

We have to optimize the lower bound because we cannot compute analytically $p(z|x)$ term for evaluating $\log p(x)$ directly in the KL divergence term.

Question 1.8

The lower bound can be pushed up by either increasing the value of $\mathrm{E}_{q(z|x_n)}[p(\mathbf{x}_n|Z)]$ or by decreasing $D_{KL}(q(Z|\mathbf{x}_n)||p(Z))$. If we increase the first term, i.e by reducing the reconstruction error to produce x.

Another way is making $q(Z|\mathbf{x}_n)$ and $p(Z)$ similar and reducing the KL divergence to zero.

1.3 Specifying the encoder $q_\phi(z_n|x_n)$

Question 1.9

Reconstruction Loss: $\mathcal{L}_n^{recon} = -\mathrm{E}_{q_\phi(z_n|x_n)}[\log p_\theta(\mathbf{x}_n|Z)]$

This loss penalizes the model to train the decoder to generate or reconstruct $\mathbf{x}_n$ from the latent variables $z \sim q_\phi(z_n|x_n)$.

Regularization Loss: $\mathcal{L}_n^{recon} = D_{KL}(q_\phi(Z|\mathbf{x}_n)||p(Z))$

This loss forces the distribution of $q(Z|\mathbf{x}_n)$ to be very similar to $p(Z)$ which is usually $\mathcal{N}(0, 1)$.

Regularization loss might force the distribution of $q_\phi(z_n|x_n)$ to spread and lie in different regions that optimizes the objective. However, regularization loss forces the distribution to have zero mean and unit variance (similar to $p_\theta(Z)$).

Question 1.10

$$\mathcal{L}^{recon} = -\mathrm{E}_{q_\phi(\mathbf{z}|\mathbf{x}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z})] \tag{1}$$

$$= -\frac{1}{L}\sum_{l=1}^{L} \log(p_\theta(x_n|z_n^l)) \qquad \text{where } \mathbf{z}_n^l \sim q_\phi(z|X = x_n), l \in 1, 2, L \tag{2}$$

$$\mathcal{L}^{reg} = \frac{1}{L}\sum_{l=1}^{l} p_\theta(\mathbf{x}_n|\mathbf{z}) D_{KL}(q_\phi(\mathbf{z}|x_n)||\log p_\theta(\mathbf{z})) \tag{3}$$

$$= \frac{1}{2}\sum_{j=1}^{z_{dim}} (1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \tag{4}$$

$$\tag{5}$$

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(-\frac{1}{L}\sum_{l=1}^{L}\log(p_\theta(x_n|z_l)) + \frac{1}{2}\sum_{j=1}^{z_{dim}}(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2)) \tag{6}$$

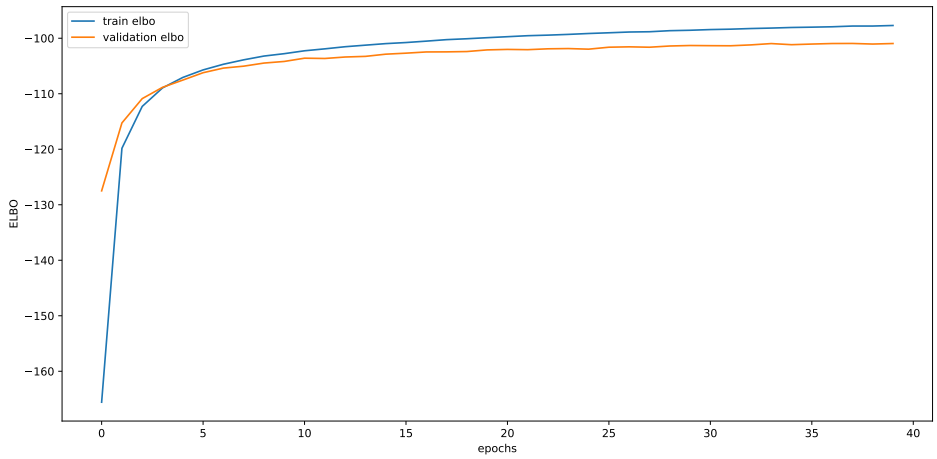


Figure 1: Lower-bounds of training and validation

1.4 The Reparametrizaion Trick

Question 1.11a

$\bigtriangledown_\phi \mathcal{L}$ is required to update the parameters of encoder using gradient decent. Trained encoder produces the parameters of latent distribution to sample z that is likely to have produced x.

Question 1.11b

In the forward pass, input x goes into encoder and output mean and variance of latent distribution. From this distribution z is sampled and passed through decoder to get an output. In the backward pass, the loss get backpropagated cannot go beyond the sampling of z because it is stochastic unit, i.e there is no fixed predefined path to apply chain rule.

Question 1.11c

Reparameterization trick is to introduce the randomization as a addition function by redefining the form of latent variable and allow the flow of gradient backward. In other words, previously sampled output from Gaussian $\mathcal{N}(\mu, \sigma)$ can computed as $z = \mu + \epsilon * \sigma$ where $\epsilon \sim \mathcal{N}(0, 1)$.

1.5 Putting things together: Building a VAE

Question 1.12

The VAE consist of encoder and decoder and both are amortized using neural networks. The encoder consist of three fully connected layers, one for mapping input($784dim$) to a hidden layer($500dim$) with ReLU non-linear activation, and rest of the two fully connected layers to map from hidden layer to output($zdim$) for mean and variance with linear and softplus non-linearity, respectively. The decoder network consist of two fully connected layer with ReLU and sigmoid non-linearity each.

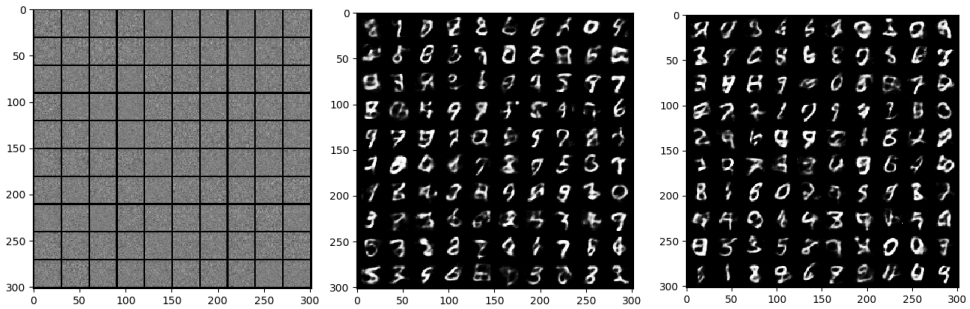
In the forward pass, input data is passes through the encoding network which return mean and standard deviation parameter of latent distribution. A random noise is sampled from unit normal and new latent variable is obtained using the reparametrization trick. The new latent variable is then passed through the decoder network to obtain the reconstructed input data. The reconstruction loss and KL loss are evaluated and error is propagated backward to update the network parameters. The optimizer used in training is $ADAM$ with default parameters.

Question 1.13

Figure 1

Question 1.14

Figure 2



(a) before training (b) Epoch 19(halfway) (c) Epoch 38

Figure 2: Samples from the model during training

Question 1.15

Figure 3

2 Generative Adversarial Networks

Question 2.1

The Generator network G takes a random noise $\mathbf{z}$ from $\mathcal{N}(0, \mathbf{I})$ of latent dimension as input. It maps this noise in the generative network to produce fake/newly generated data as output $\hat{x}$,i.e $G : \mathbf{z} \rightarrow \hat{\mathbf{x}}$. The output in case of image generation task would be new/fake images $\hat{\mathbf{x}}$ generated by the network.

The Discriminator network D takes the original training data $\mathbf{x}$ and generated data $\hat{\mathbf{x}}$, and outputs binary labels predicting whether the input data is fake or real, i.e $D : \mathbf{x} \rightarrow s$ where $s \in [0, 1]$. In context of GANs for images, the input would be real training images as well as generated images.

2.1 Training objective: A Minimax Game

Question 2.2

The objective function of GAN is

$$\min_G \max_D V(D, G) = \min_G \max_D \mathrm{E}_{p_{data}(x)}[\log D(X)] + \mathrm{E}_{p_z(z)}[log(1 - D(G(Z)))]$$

The first term is cost function associated with the decoder which takes the log of the prediction of true data by the decoder. The objective is to maximize the correct prediction of real data, i.e $D(X) \approx 1, X \sim p_{data}(x)$. The second term ensures that the prediction for the fake data are also correct by penalizing the false prediction, i.e $D(G(Z)) \approx 0, Z \sim p_z(z)$.

The network is trained to first maximize the value function for decoder to to be a strong discriminator and then minimize the generator loss given the decoding network.

Question 2.3

The minimum value of $V(D, G)$ is $-\log 4$

Question 2.4

In the early stages of training, if the discriminator is very good at discriminating the real data from the generated ones then loss saturates and there is small/no gradient flowing backwards. The $D(G(Z)) \approx 0$ and $\log(1 - D(G(Z))) \approx 0$, and this affect the generator network because it is yet to learn.

This can be avoided by changing the objective function for the generator to maximize the $\log(D(G(Z)))$ which makes the generator to generate more realistic samples and cheat the discriminator.

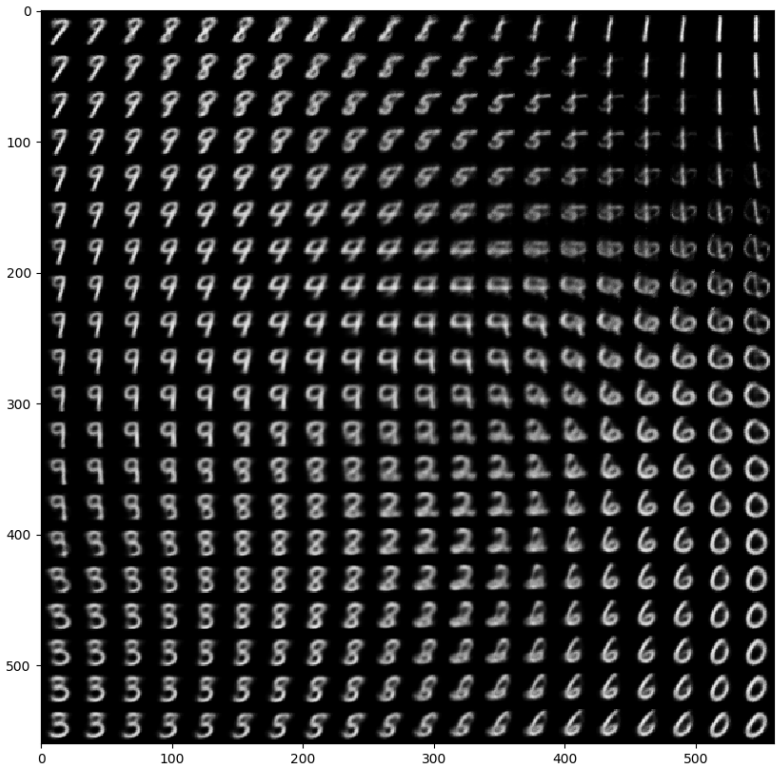


Figure 3: Lower-bounds of training and validation

2.2 Building a GAN

Question 2.5

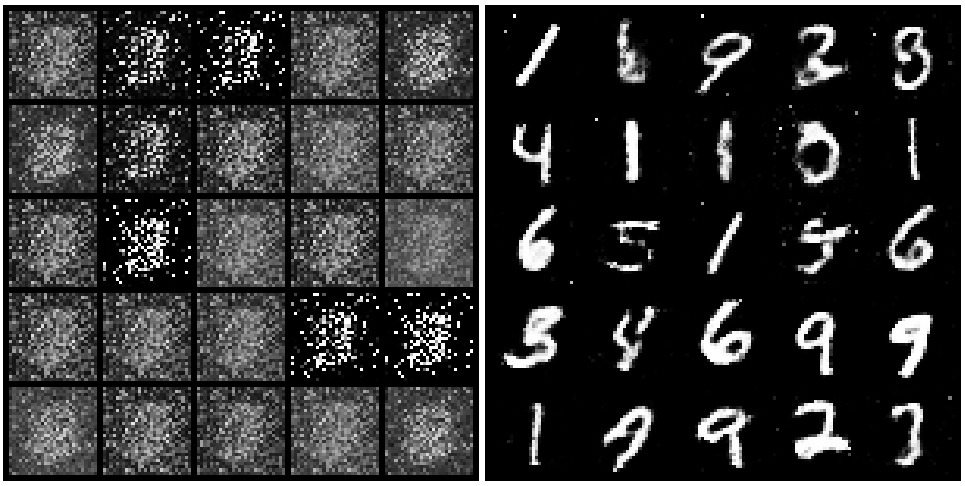
The generator network defined here consist of 3 blocks of consecutive linear, LeakyReLU and batch norm layers with increasing hidden units. The inputs of latent dimension are mapped to a hidden layer followed by these three blocks and ends with a linear and $\tanh$ non-linearity. The discriminator network consist of three blocks of linear and LeaklyReLU non-linearity, with exception in the last block which has $Sigmoid$ non-linearity.

For training, two tensors(variables) for real and fake data are initialized to $1, 0$ respectively for a fixed batch size. The generator network is trained first by sampling random noise (of fixed batch size and latent dimension) and doing a forward pass through the generator to generate fake images. The generated images are then passed through the discriminator network to get the prediction. The generator loss is then evaluated and weights are updated by the ADAM optimizer after backpropagation. The discriminator network is trained next passing a fixed batch of training data and newly generated sampled data to get the predictions. These predictions are used to evaluate the loss and optimized using ADAM optimizer after backpropagation. The loss criterion used is binary cross-entropy.

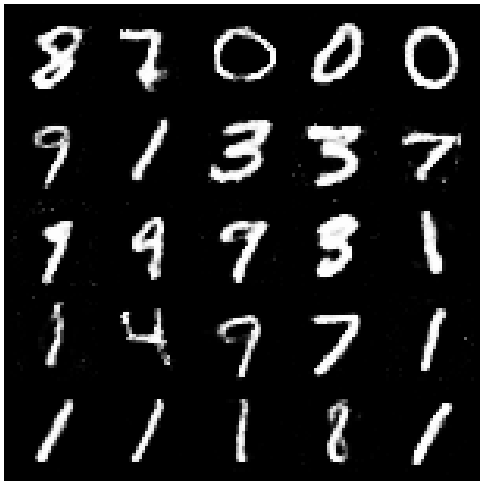
Question 2.6

Figure 4

Question 2.7



(a) after first update (b) middle



(c) final

Figure 4: Samples from the model during training

Figure 5

3 Conclusion

Question 3.1

VAE are generative models which optimizes the likelihood of data whereas GANS are not. The optimization objective of GANs is to generate samples that are very similar to training data. Both the models has differentiable generator network and a second network used for different purpose. The VAE network is a recognition model that performs approximate inference whereas in GANs it is the discriminator network. GANs require differentiation through the visible units, thus cannot model discrete data, while VAEs require differentiation through the hidden units, and cannot have discrete latent variables. For model evaluation, there are ways to evaluate the quality of the two VAE models(log-likelihood) but for GANs there are not such comparison other than visual inspection.

(a) Epoch 0

(b) Epoch 19

(c) Epoch 38

(d) Epoch 0

(e) Epoch 19

(f) Epoch 0

(g) Epoch 19

Figure 5: Interpolation sample after training