Deep Learning Assignment 1: MLPs, CNNs and Backpropagation

Paul ten Kaate

10743367

paultenkaate@outlook.com

December 17, 2019

1 Variational Auto Encoders

Question 1.1

The VAE differs from a normal auto-encoder in the way it maps to a a different kind of hidden layer. The purpose of the normal auto-encoder is to create a encoding network and a decoding network so that data can be encoded - reducing the dimensionality - and decoded again - bringing it back to the original shape - while trying to reproduce the original data as much as possible. An auto-encoder likely overfits to the data, and for similar data samples the latent variables often do not look similar. Due to this, taking a latent variable gathered by encoding a data sample, slightly changing the latent variable, and decoding the latent variable might give a completely different output than decoding the original latent variable. Generating new data that makes sense is therefore not what an standard autoencoder is used for.

The goal of a GAE however is not only to encode and decode data as good as possible, but also to generate new data when feeding the decoding algorithm. The encoding part maps the data samples to latent variables that form a gaussian distribution, where similar data samples have similar latent variables. A slight change in a encoded data sample will generate a slightly changed output. Given this fact, new outputs can be generated by sampling from the gaussion distribution.

Question 1.2

As can be deduced from the formulas in the assignment, First the latent variables z have to be drawn from a normal distribution, and from there every time a calculation is always made, the result of which is passed on for a subsequent calculation. These calculations can be found in the assignment under in the formulas (3) and (4). Each calculation therefore has an ancestor, except for the latent variables. This method of first sampling from the distribution without

parent and then performing subsequent calculations on this outcome, is called ancestor sampling.

Question 1.3

As explained in Doersch (2016), as long as you have a sufficiently complicated function, the values from a normal distribution can be transformed to the preferred outcome. Therefore, we do not need to train the p(z), retrieving the needed function is enough. As a neural network is capable of forming complex functions, training the neural network is enough.

Question 1.4

a) To calculate $\log p(\mathcal{D})$ we have to calculate the integral through z.

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log \int p\left(x_n|z_n\right) p\left(z_n\right) dz_n$$

However, we can also approximate the log probability by taking M samples of z's from the prior probability distribution for every x, and taking the average of the dataset probabilities for those z's. There formula for this is given hereunder:

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \frac{1}{M} \sum_{m=1}^{M} \log p\left(x_n|z_{nm}\right)$$

b) When we look at figure 2 in the assignment, we see the distribution of a two dimensional latent space. Furthermore, we see the distribution of the posterior distribution given that latent space. What can be seen is that a great part of the possible values for z result in a posterior probability of zero. This means we have to sample a lot of z's to find a few z's that result in a posterior probability that is not zero, making it able to estimate the posterior. Furthermore, with more latent dimensions, the probability of finding a z that does not result in a posterior of 0 decreases. Therefore using this method to estimate the posterior is inefficient.

1.1 Question 1.5

a) The closer a distibution lies to $\mathcal{N}(0,1)$, the smaller the KL-divergence will be. Therefore $\mathcal{N}(8,34)$ will result in a very large KL-divergence, and $\mathcal{N}(0.12,1.1)$ will result in a very small KL-divergence.

b) The closed form KL divergence for gaussian distributions:

$$KL(q\|\mathcal{N}(0,I)) = \frac{1}{2}\sum_{j=1}^{J}\left(1+\log\left((\sigma_j)^2\right)-(\mu_j)^2-(\sigma_j)^2\right)$$

Question 1.6

Equation 11 in the assignment states

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

We call the right part the *lower bound* in advance. Bringing the KL-divergence to the right side of the equation gives

$$\text{data log-probability} = \textit{lower bound} + \text{KL-divergence}$$

where KL-divergence is always larger than zero. Therefore

$$\text{data log-probability} \geq \textit{lower bound}$$

so the right part of the equation 11 is the lower bound of the data log-probability, which is reached when the KL-divergence is zero.

Question 1.7

As discussed in Question 1.4, directly calculating the log probability of the data required calculating integrals, or sampling a great amount z using the Monte-Carlo method. Both of these approached turned out to be very complex, therefore we first calculate the lower bound, from which is mentioned in the assignment consists only of directly computable quantities.

Question 1.8

The loss function

$$\mathcal{L}(\theta, \phi) = -\frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{q_\phi(z|x_n)} \left[\log p_\theta \left(x_n|Z\right)\right] - D_{\mathrm{KL}} \left(q_\phi \left(Z|x_n\right) \| p_\theta(Z)\right)$$

consists of two parts, the mean data log probability and the mean KL-divergence. If the loss goes up, deriving from the equation, at least one of the following thing has happened.

- The mean data log-probability has increased
- The mean KL-divergence has decreased

Question 1.9

Putting input x through the endocer giver output z. With the reconstruction loss, we then estimate the log probability of finding an output equal to the input x. A smaller probability entails the model is worse in reconstructing an output equal to the input.
The regularization loss calculated how similar (regular) the probability distribution of Z given input x is, compared to a normal Gaussian distribution. The more similar, the lower the loss. With the regularization loss we therefore try to make the distribution as standard (regular) as possible

Question 1.10

The loss is calculated with

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

substituting the losses gives

$$\mathcal{L} = \sum_{n=1}^{N} -\mathbb{E}_{q_\phi(z|x_n)} \left[\log p_\theta \left(x_n|Z\right)\right] + D_{\mathrm{KL}} \left(q_\phi \left(Z|x_n\right) \| p_\theta(Z)\right)$$

$$= \sum_{n=1}^{N} - \sum_{m=1}^{M} \log \text{Bern}\left(x_n^{(m)} | f_\theta\left(z_n\right)_m\right) + \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log\left(\left(\sigma_j\right)^2\right) - \left(\mu_j\right)^2 - \left(\sigma_j\right)^2\right), z \sim \mathcal{N}\left(z_n | \mu_\phi\left(\boldsymbol{x}_n\right), \text{diag}\left(\Sigma_\phi\left(\boldsymbol{x}_n\right)\right)\right)$$

1.2 Question 1.12

An encoder is created that maps the input to a hidden layer, and then with seperate weight matrices maps the hidden layer to a mean and a std output. Also a Decoder is created that takes a vector **z** as input and generates an image as output. A forward function combining these two functions is created, putting images into the Encoder, which gives a mean and a (log squared) std. Log squared in this case to avoid working with really small numbers. **z** is sampled from the mean and std and put in the Decoder. With this output and the mean and std a loss is calculated summing the regularization and reconstruction error. The gradients are calculated and the weights are updated. A seperate *sample()* function is written to sample **z** from a standard normal distribution, which can be put in the Decoder in order to generate images.

1.3 Question 1.13

The elbo over epochs can be seen in figure 1.

Question 1.14

The generated images over epoch 0, 20 and 39 can be seen in figure 2

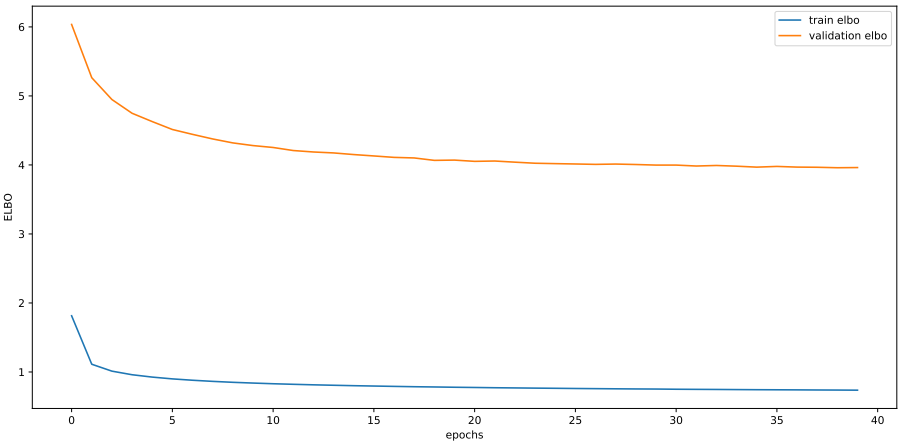

Figure 1: Estimated lower bound over epochs

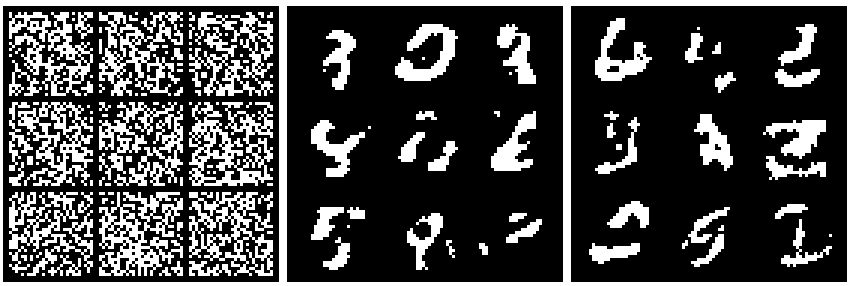

Figure 2: Generated samples over epoch 0, 20 and 39.

2 Generative Adversarial Networks

Question 2.1

The GAN exists of two functions, a generator and a discriminator.

- The generator takes a input vector **x** that is sampled from a normal distribution. The function maps the vector **x** to a vector **o** that represents for example an image.
- The discriminator takes this input **o** or a real image as input and maps it to a scalar **p** that represents a probability ranging from 0 to 1. Here a higher probability means a higher change that the

2.1 Question 2.2

The loss function is defined by

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

- $\mathbb{E}_{p_{\text{data }}(x)}[\log D(X)]$ is the expectation of the log probability given by the discriminator for inputs that are real images. You would want to maximize the probability for the discriminator, so it gives a higher probability output when labeling real images.
- $\mathbb{E}_{p_z(z)}[\log(1 - D(G(Z)))]$ is the expectation of the log probability given by the discriminator for inputs that are generated from a normal distribution. You would want to maximize this for the discriminators, so it gives a lower probability output for generated image, and you want to minimize this for the generater, so it becomes better in generating images that have a high probability of being real.

Question 2.3

When the training converges, the discriminator cannot distinguish a real image from a generated image. In other words, it is up to chance whether a real or

generated image is being labeled correctly. Thus, D(X) and D(G(Z)) are both 0.5. Filling this in the function V(D,G) gives log(0.5) - log(1 - 0.5) ≈ -1.3863

Question 2.4

At the start of the training, the generator can be bad at generating close to real images. If this is the case, D(G(Z)) can lie at zero. log(1 - D(G(Z))) = log(1 - 0) = log(1) = 0. If the loss is 0, there is no gradient to update the weights. The other way around, if the discriminator is too bad at separating generated from real images, the D(G(Z)) can lie at one. log(1 - D(G(Z))) = log(1 - 1) = log(0) = undefined. Both of these problems can be added by adding a really small number to the probabilities.

Question 2.5

A Generator function is created that takes an input z and gives an image as output. A Discriminator function is created that takes an image as input and gives a number between 0 and 1 as output, representing the estimated probability that the image is real. Binary Cross Entropy is used as loss function. The Discriminator is trained by feeding it real images, and calculating the BCE between the predicted probability and a probability of 1, and by feeding it fake images (generated with the Generator) and calculating the BCE between the predicted probability and a probability of 0. These two losses can be added and used to calculate the gradients and update the weights for the Discriminator. The Generator is trained by sampling an input from a normal distribution using the output as input for the discriminator. Here the BCE is calculated of the predicted probability and a probability of 1, because you want the probability as high as possible for the Generator. With this loss, the weights for the Generator are updated.

Question 2.6

Figure 3 shows generated images at various stages of the training.

Figure 3: Generated images at the start, middle and end of training

Figure 4: Interpolation between two numbers.

Question 2.7

Figure 4 shows the interpolation between the numbers 0 and 1.

2.2 Question 3.1

Equation 16 can be rewritten as

$$z = f(x); \quad x = f^{-1}(z); \quad p(x) = p(z) \left| \det \frac{df}{dx} \right|$$

Equation 17 can be rewritten as

$$\log p(x) = \log p(z) + \sum_{l=1}^{L} \log \left| \det \frac{dh_l}{dh_{l-1}} \right|$$

References

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

2.3 Question 3.2

Because we have to calculate the determinant of $\frac{dh_l}{dh_{l-1}}$, this matrix has to be a squared matrix. This only happens when dh_l and dh_{l-1} are of the same size. Thus the constraint of f is that it has to map the input to an output of the same size.

Due to a lack of time I made the decision to stop the homework from here and to free up extra time to prepare for the exams. See below the conclusion which is mainly focused on the UAE and GAN models

2.4 Conclusion

In this report three different models are examined and two of these models are implemented. The Variational Auto-Encoder tries to map the data to a normal distribution, making it possible to generate new data by sampling from a normal distribution. A Generative Adversarial Network learns to produce images by getting feedback from a discriminatie model. The Normalizing Flow model predicts the probability of a pixel based on the previous pixels.

In this experiment, the VAE and the GAN are implemented. The VAE is faster than the GAN, and produces reasonable outputs earlier. Howver, looking at both results in figure 2 and 3, is can be said that the images produced by the GAN are more realistic. Each model has it's advantages in certain situations. The VAE, due to it's speed, could be useful in rapidly changing environments which need a lot of re-training. For photo-realistic results however, a GAN is the better choice.