Deep Learning - Assignment 3

Davide Belli
11887532
University of Amsterdam
davide.belli@student.uva.nl

1 Variational Auto Encoders

1.1

VAE, pPCA and Factor analysis are all techniques that can be used for dimensionality reduction. The main difference between the first and the other two is that VAE compresses the input data in form of parameters of a distribution (enforced through a prior distribution), which means that we can sample from this distribution to generate new data. On the other hand, pPCA and FA are linear methods that aim to maximize the variance in the reduced space, meaning they aim to keep the maximum amount of information characterizing the data discarding the features which are not discriminative (e.g. noise).

1.2

- sample $\mathbf{z}_n$ from the Normal distribution
- feed $\mathbf{z}_n$ into f_θ
- sample $\mathbf{x}_n$ from the Bernoulli distributions parameterized by the output of f

1.3

The considered assumption is that by fixing a type of distribution for the latent space does not limit the effectiveness of the model. This is true, because the decoder network following the encoding in the latent space is able to learn a mapping from this representation to the reconstruction of the autoencoder, overcoming possible limits in the latent space representation. It is however true that changing the prior distribution results in different performances for the model. By enforcing the posterior to match the form of the prior, we can easily sample from the standard normal distribution which is the prior.

1.4

(a) In this case, we can approximate the intractable integration in the following equation by sampling a certain number of z_n from the latent space and summing over them. We use S to refer to the set of samples for z_n obtained using MC sampling.

$$\log p(x_n) = \log E_{p(z_n)}[p(x_n|z_n)] = \log \int p(x_n|z_n)p(z_n)dz_n \approx \log \frac{1}{|S|} \sum_{z_n}^{S} p(x_n|z_n) \quad (1)$$

(b) Using sampling to train VAE is not efficient because, to meaningfully approximate the distribution over the prior, we would have to have a relevant number of samples at every iteration. This problem increase drastically with higher dimensionality of z_n, since it is much more rare to sample from space intervals where at least one of the dimension of the variable assumes low probability values.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

1.5

(a) For $\mu = 0, \sigma = 0.5$ the KL-divergence is small (if the std is 1, the divergence is actually null). For $\mu = 100, \sigma = 0.1$ the divergence is very large

(b) $D_{KL}(q||p) = \log \frac{1}{\sigma_2} + \frac{\sigma_2^2 + \mu_2^2}{2} - \frac{1}{2}$

1.6

Because, thanks to derivations provided in Eq. 7-10, it is proven that this quantity cannot be larger than than the log probability of x_n. In particular, the distance between the lower bound and that probability amounts to the KL-divergence between the approximate (q) and real (p) posterior distributions, which, by definition, cannot be negative.

1.7

Because the log probability of x_n contains an intractable integral, while the lower bound itself can be efficiently computed as difference between the expectation and the divergence between the approximate posterior and the true prior.

1.8

In the maximization step, the lower bound is pushed up by optimizing the loss by updating the parameters θ for the approximate distribution q. When this happens, one case is that the lower bound reaches the current value of the log probability, meaning that the KL divergence between the approximate and real distribution is zero (and the distributions are the same). The other possibility is that the lower bound increases to the previous log probability but the divergence is still positive. In this case, the distributions are different, and since the KL-divergence describes the gap between the lower bound and the new log probability, this last element will increase in comparison to its value before the maximization step.

1.9

The reconstruction term describes how good is the decoder (p_θ) into reconstructing the original x given its latent representation z. The regularization term is useful to enforce the encoding part of the autoencoder to assume the form of the distribution represented by the prior, avoiding the autoencoder to overfit and perfectly learn to map the training data only. This is particularly useful because it allow us to sample from the latent space, since the posterior will assume the form of the known prior.

1.10

When deriving the reconstruction loss for the VAE, we can approximate the expectation of the likelihood for the element n by sampling z_n from the posterior distribution parametrized by the outputs (μ_f, σ_f) of the encoder f which is fed as input x_n:

$\mathbf{z}_n \sim q(\mathbf{x}_n|\mathbf{z}_n) = N\Big(\mathbf{z}_n|\mu_\phi(\mathbf{x}_n), diag(\Sigma_\phi(\mathbf{x}_n))\Big)$

Then, it follows (using f_θ to represent the decoder):

$$\begin{aligned} L_n^{recon} &\approx -\log p(\mathbf{x}_n|\mathbf{z}_n) \\ &= -\log \prod_m^M Bernoulli(\mathbf{x}_n^{(m)}|f_\theta(\mathbf{z}_n)_m \\ &= -\sum_m^M \mathbf{x}_n^{(m)} \log(f_\theta(\mathbf{z}_n)_m) + (1 - \mathbf{x}_n^{(m)}) \log(1 - f_\theta(\mathbf{z}_n)_m) \end{aligned} \tag{2}$$

For the regularization loss, considering the derivations in 1.5:

$$\begin{aligned} L_n^{reg} &= D_{KL}(q(Z|\mathbf{x}_n)||p(Z)) \\ &= \sum_i^D D_{KL}\Big(N(\mathbf{z}_i|\mu_\phi(\mathbf{x}_n)_i, diag(\Sigma_\phi(\mathbf{x}_n)_i))||N(0,I)\Big) \\ &= \frac{1}{2}\sum_i^D \Sigma_\phi(\mathbf{x}_n)_i^2 + \mu_\phi(\mathbf{x}_n)_i^2 - \log(\Sigma_\phi(\mathbf{x}_n)_i^2) - 1 \end{aligned} \tag{3}$$

1.11

(a) We need $\nabla_\phi L$ to be able to update the weights ϕ in order to reduce the loss L (the gradient tells us how a change in the parameter space affect the final loss).

(b) When we sample from the latent space we cannot compute the gradient with respect to the parameters of the encoder because the sampling operation is not differentiable.

(c) The reparametrization trick enables us to avoid passing through the (Gaussian) sampling step before decoding the latent space. Instead, we deterministically represent the latent encoding as a multiplication between σ^2 and a random noise following a standard normal distribution summed to μ (where μ and σ are the encoder's outputs). In this case, the sampling is not dependent on the mean and sigma and, thus, we do not need derivate with respect to that function.

1.12

Our VAE model consists of three main components. First, we have the encoder network, which consists of a hidden layer followed by a non-linearity and then two separate hidden layers to output the latent arrays for mean and std parameters of the Gaussian distribution modeling the latent space. These values are then used to sample from the multidimensional latent distribution. Actually, we use the reparametrization trick to avoid using the non-differentiable sampling function on the parameters, resulting in $z = \mu + \epsilon * \sigma^2$, where $\epsilon \sim N(0,1)$. Next, the latent vector z is fed to the decoder network, consisting of a hidden layer followed by a linear layer which outputs the reconstructed image. The elbo is then computed for every batch (and epoch) as the sum of the reconstruction error (using binary cross entropy loss) and the KL-divergence between the posterior and prior, as shown in Answer 1.5.

1.13

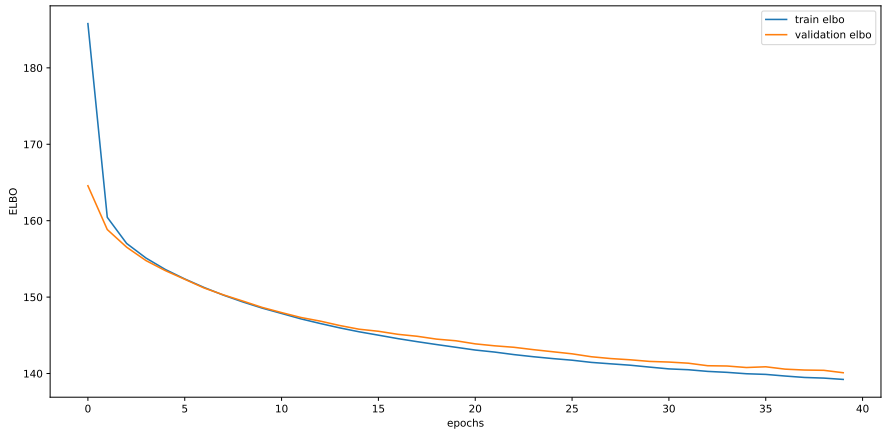
In Fig. 1 we show the elbos for training and validation sets using latent space dimensions of 2 and 20.

1.14

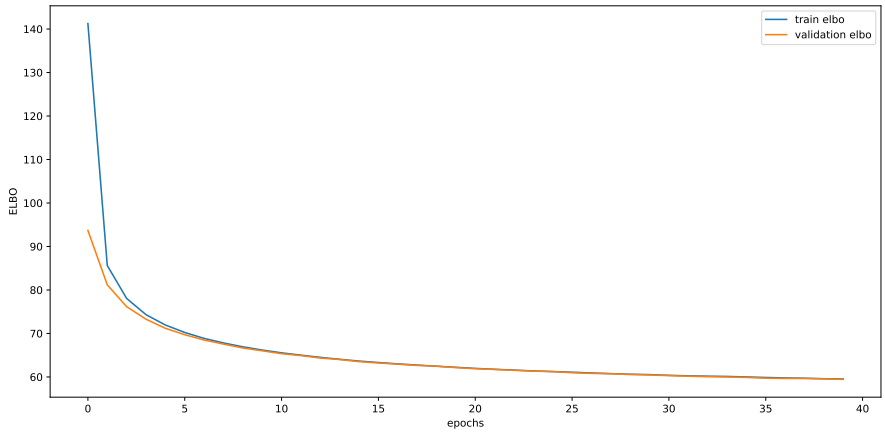
In Fig. 2 and 3 we show samples generated using our VAE at different timesteps (before, halfway and at the end of training). We notice how results improve drastically from initial random noise to (almost always) meaningful digits. The results halfway through the training are comparable to the one at the end of it, meaning that the model has almost converged around 20 epochs. The difference between the first row and the second stands in the usage or not of the Bernoulli distribution on the outputs of the decoder. If the means are used as means of the Bernoulli, the final image will only contain white or black pixels, as in the original BMNIST dataset. Otherwise, we can visualize the means themselves to see smoother digits with continuous pixel values ranging between zero and one.

1.15

In Fig. 4 we show the manifold in a 2-dimensional latent space learned by our model. By visual inspection, we notice how all the 10 digits are present in the plot, meaning that every digit class is mapped to a particular part of the latent space. In particular, we notice that our latent space should be distributed as a standard normal, as enforced by the prior: $p(z) \sim N(0,1)$. Since we know how the probability mass is distributed in a Gaussian distribution depending on its parameter, we know which ranges of μ have a significant content in our latent space. For a Normal distribution with mean 0 and



(a) Training and Validation elbos for z = 2



(b) Training and Validation elbos for z = 20

Figure 1

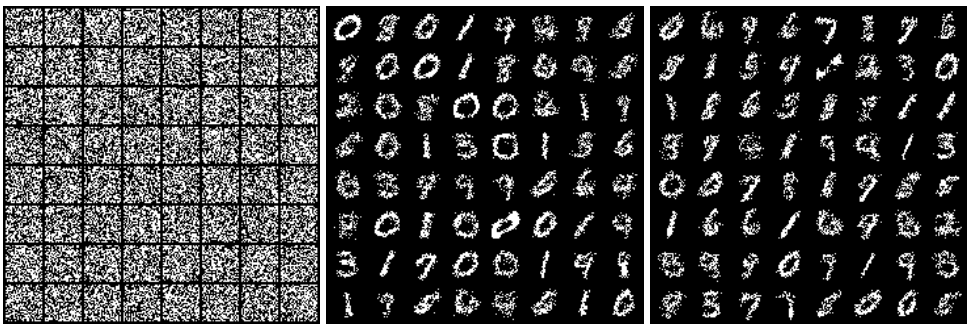


Figure 2: Sampling datapoints at 0, 20, 40 epochs using Bernoulli

std 1, around 95% of the probability mass is contained in the interval $[-2, 2]$. For this reason, we choose this range for our manifold plot.

Figure 3: Sampling datapoints at 0, 20, 40 epochs **without** using Bernoulli

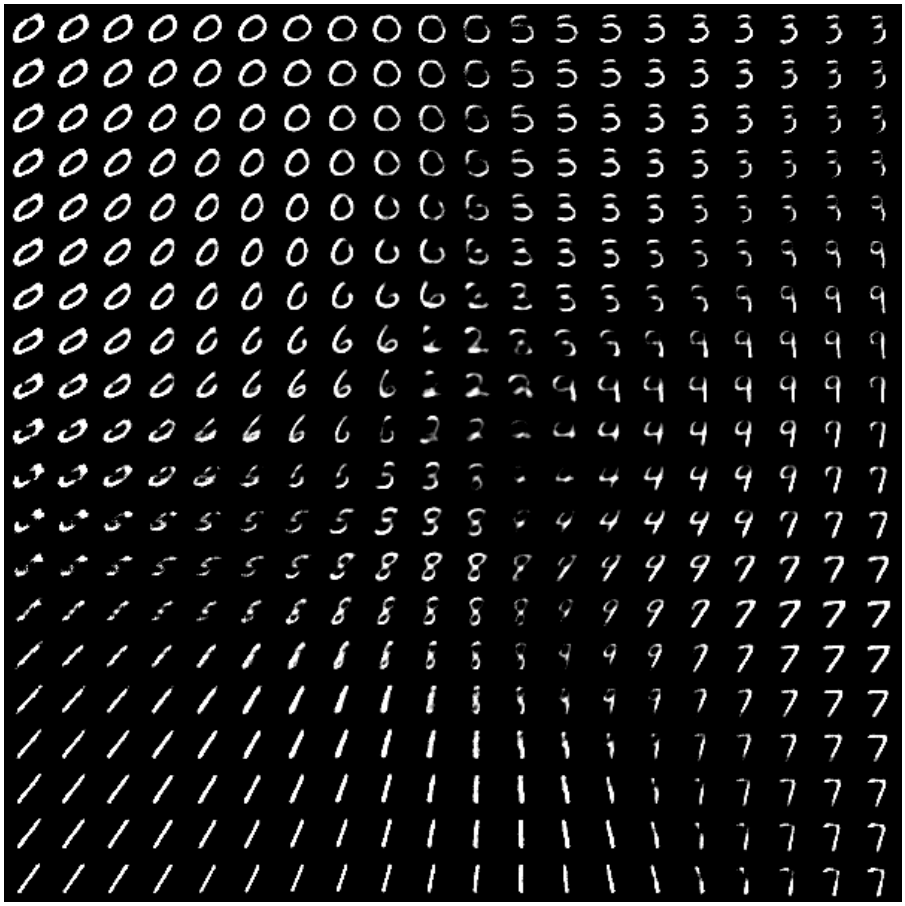


Figure 4: Manifold of the VAE latent space at the end of training

2 Generative Adversarial Networks

2.1

Let's consider as an example the case of image generation. In the generator network, the input is a latent vector representing a datapoint in the latent space of image features. Starting from this latent vector, the generator network reconstructs and outputs an image. The discriminator network takes as inputs images (or in general datapoints), coming either from the real distribution of images (dataset), or from the distributions of the reconstructed images (generated by the generator network). The discriminator then tries to understand from which distributions does the input come from, returning

as output a scalar probability value (where 1 means that the image comes from the true distribution, and 0 means that it comes from the reconstructed one).

2.2

The first term is the probability assigned by the discriminator to a batch (datapoint) of real inputs (images). To train the discriminator, we want to maximize this value. The generator is not affected by changes in this value. The second term is the probability assigned by the discriminator to a batch (datapoint) of reconstructions originated by samples from the latent space Z using the generator network. The discriminator is trained to minimize its prediction (and thus maximize the term, which equals to 1 less the output probability), while the generator is trained to minimize the term. The updates in generator and discriminator parameters are done by fixing their respective opponent network.

2.3

When the training has converged the value of V is $-\log 4$. This means that the real and reconstructed distributions are so similar (actually, identical) that the discriminator assigns probability 0.5 to both. Note that if the discriminator would predict values higher or lower than 0.5 when it cannot differentiate between the two distributions, its score wouldn't be maximized. In the worst case where it would output always 0 or 1 for both cases, the score would be $-\infty$.

2.4

The second term can be problematic since the discriminator is not yet able to recognize the true images from the fake one, and thus the signal it is going to provide to the generator can be wrong or inaccurate.

2.5

GANs consist of generator and discriminator networks. The generator networks takes as input a random "noise" 100-dimensional vector sampled from a standard normal distribution. This vector is fed into a sequence of feed-forward layers alternated with leaky relus and batch normalizations. The output of this network is a 784-dimensional vector, which, after reshaping, represents a 28×28 MNIST image. The discriminator takes as input a batch of real **or** reconstructed images, and assigns to each image the probability of it being sampled from the real distribution. At the beginning of the training, the averages over the batch for the probability score are very similar between real and reconstructed images. At the end of the training, the discriminator has improved enough to distinguish between the two distributions, with average scores at 0.95 for real images and 0.65 for reconstructed ones (convergence is not reached yet!). At the same time, the generator has learned to produce realistic reconstruction thanks to the signal from the discriminator.

2.6

Figure 5: Sampling datapoints at 0, 100, 200 epochs

In Fig. 5 we show reconstructed images sampled at the beginning, halfway and at the end of the training. The results appear realistic enough when compared to the original MNIST dataset, even if some images are very noisy and cannot be accurately assign to a class by a human evaluator. We can motivate this by thinking how some portions of the latent space might not be accurately modeled, either because few training samples in that space have been seen during training, or more likely, because two different classes collide in that part of latent space.

2.7

Figure 6: Interpolating samples between two points in the latent space

In Fig. 6 we show an example of interpolation from our model. To create this, we sample two points in the latent space and use *linspace* method to find other 7 points at the same distance laying on the segment spanning between the two points. We then sample the image with our GAN from the vectors representing the coordinates of all of the 9 points. In the visualization, we see how the digit gradually changes from a "1" to a "9", passing through other digit class in the manifold found by our model.

3 Conclusion

3.1

Both VAEs and GANs are powerful unsupervised learning models that can be used for generative tasks. Variational Auto-Encoders makes it possible to enforce an explicit distribution as prior for the latent space (commonly, a Gaussian). This enables us to express in a mathematical form the learned distribution, and can also be convenient in order to incorporate additional information about the data in the form of prior distribution. Although they are generally easier to train, often times the quality of the results is worse with respect to GANs due to the MSE reconstruction term in the loss. Generative Adversarial Networks benefit of more realistic results, since they optimize directly on the adversarial loss which is learned from the real data distribution through the discriminator network. On the other side, we do not have a clear quantitative understanding of how good the results are, since the easiest way to validate them is by visual inspection. Finally, GANs are more difficult to be train since this involves a convergence in the Nash equilibrium between discriminator and generator, and they also suffer from mode collapse.