Deep Learning Homework assignment

Nzuanzu Jier
10223258
jier.nzuanzu@student.uva.nl

1 Variational Auto Encoders

1.1 Latent Variable Models

- Question 1.1
  1. It both relates in terms of input but differ in terms of how the input is processed. The Autoencoder maps the input to a smaller hidden layer,thus compressing the input to extract the salient features and reproducing the input. However, the Variational Autoencoder focusses on how the output is generated from the hidden layer, which in terms acts, each node respectively, as a distribution with a mean and variance. The element wise multiplication and addition from these mean and variances with a different distribution z will be fed to the output layer of the network to reproduce the input.
  2. An Autoencoder is not generative as its focus is to focus on salient features which is inherently in the input and thus will be able to only reproduce that input; irrespective of the input the latent space in the hidden layer is input dependant thus generating different new images from this input is not feasible as it may well be that the latent space is not continuous nor separable -think of MNIST input numbers where every input is different- where a sample can be taken from.

1.2 Decoder: The Generative Part of VAE

- Question 1.2 So from every $n_t h$ image we sample every $m_t h$ pixel of it by condtioning it of the neural network output of the $m_t h - 1$ pixel of it until all pixels are passed. So after all n images we have a joint distribution of $m_t h$ pixels of every $n_t h$ images.
- Question 1.3 Due to the neural network output of every $m_t h$ pixel which in itself are complex and parametrised by θ means that z_n or rather $p(Z)$ could be very simplistic.
- Question 1.4

  (a)

$$\log p(x_n) = \log \frac{1}{n} \sum_{i=1}^{N} \mathrm{E}_{p(z_i)}[p(x_n|z_i)]$$

  (b) The sample $p(x_n)$ space will be likely bigger than the actual region of interest $p(x_n|z_i)$, therefore lot of computational power will be lost.

1.3 The encoder

- Question 1.5

  (a) Minimal difference ($\mu_q \approx 0$) and ($\sigma_q \approx 1$) and maximal difference is all other possible options for which μ_q, σ_q are different from distribution p.

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

(b)

$$
\begin{aligned}
D_{KL}(q\|p) &= -\int p(x)\log q(x)dx + \int p(x)\log p(x)dx \\
&= \frac{1}{2}\log\left(2\pi\sigma_2^2\right) + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}\left(1+\log 2\pi\sigma_1^2\right) \\
&= \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}
\end{aligned}
$$

We know $p \sim \mathcal{N}(0,1)$ with $\mu_2 = 0, \sigma_2 = 1$, so our expression becomes, $\mu_1 = \mu_q, \sigma_1 = \sigma_q$ the following:

$$
\begin{aligned}
D_{KL}(q\|p) &= \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \\
&= \log\frac{1}{\sigma_1} + \frac{\sigma_1^2 + \mu_1^2}{2} - \frac{1}{2}
\end{aligned}
$$

- Question 1.6
  It is called the lower bound as if $D_{KL} = (q(Z|x_n)\|p(Z|X_n)) \approx \vee = 0$ *right hand side* will have to be the minimum otherwise *the left hand side* will always be greater than or equal to it.
- Question 1.7
  We must do so because we are able to sample from the prior of the latent vector(s) z and from the variational input of the data. Effectively we are solving in parts the given data. Why not do from the right hand side, is just that the log probability of the data, see equation 5 from the assignment, is not known beforehand. Moreover, knowing the KL divergence requires knowing the posterior of $p(Z|x_n)$ which is not yet known.
- Question 1.8
  If and when the right hand side goes up, we obtain first a better description of the probability over the data and second we can now evaluate how closely the KL divergence of our encoding and decoding step is close to the real data.

1.4 Specifying the encoder

- Question 1.9
  So if we will just use one sample to approximate the expectation of the generated data, that will mean we are trying to reconstruct what has been sample/encoded from the data input. That is why we have a reconstruction step. To optimise our reconstruction we want a better latent space, which needs to be regularized given the input data that needs to be sampled from. This KL divergence term does fit this purpose as it can be used as an objective function.
- Question 1.10

$$
\begin{aligned}
\mathcal{L}^{recon} &= \frac{1}{B}\sum_{i}^{B}\log\left(p(x^i|z^i)\right) \\
&= \frac{1}{B}\sum_{i}^{B}\sum_{j}^{M}\log Bern\left(x_j^i|\underbrace{f_\theta(z^i)_j}_{=k}\right) \\
&= \frac{1}{B}\sum_{i}^{B}\sum_{j}^{M} k\cdot\log x_j^i + \log\left(1-x_j^i\right)(1-k)
\end{aligned}
$$

$$\begin{aligned}\mathcal{L}^{reg} &= D_{KL}(q_\phi(Z|x_n)\|p_\theta(Z)) \\ &= D_{KL}\left(\mathcal{N}(Z|\mu_\phi(x_n), diag\Sigma_\phi(x_n))\|\mathcal{N}(0, 1_D)\right) \\ &= \sum_i^D \log\frac{1}{\Sigma_{i,\phi}(x_n)} + \frac{\Sigma_{i,\phi}^2 + \mu_{i,\phi}^2(x_n)}{2} - \frac{1}{2}\end{aligned}$$

1.5 The Reparametrisation Trick

- Question 1.11
  (a) We need $\nabla_\phi \mathcal{L}$ to apply stochastic gradient descent over different values of X sampled from the dataset $\mathcal{D}$, this means we will be able to update and draw arbitrary many samples of X, z.
  (b) Because of sampling we cannot do a back-propagation as the generator depends on the sampled input of the encoder and back-propagation cannot handle stochastic units within the network.
  (c) So the *reparametrization trick* is to split the sampling of the encoder output by first evaluating with a continuous function on the input X with a **noise** that is not to be learned. The restriction on the continuity of the function over the input in the encoding part and non-learnable noise alleviate the dependance of the generator on the encoding part by making both function deterministic and continuous. This solves the back-propagation issue.

1.6 Putting things together: Building VAE

- Question 1.12
- Question 1.13
- Question 1.14
- Qeustion 1.15

2 Generative Adversarial Networks

- Question 2.1
  For a generator we have random noise sampled from a distribution of choice which will generate a noisy image. This image together with the image of the training set will serve as input for the discriminator which will output labels , real or fake. Effectively the discriminator should discriminate well between the generator input and the data input.

2.1 Training objective: A minimax Game

- Question 2.2
  We want to deceive as well as possible the discriminator while maximize our classification power of the input data. So the generator wants to minimize its effect of the discriminator, while the discriminator wants to maximize its discriminative power over the generator. So the training objective is the sum of the classification power of the discriminator and its discriminative power of the generator.
- Question 2.3
  The value that will result after convergence with the minimax game between the generator and discriminator is $-\log 4$ as when the discriminator cannot differentiate between **real** and **fake** it will have $\frac{1}{2}$ time to discern the labels. Combining this knowledge with equation 15, we obtain the sum of two times $-\log 2$ which is $-\log 4$ as stated before.
- Question 2.4
  This is the case because the discriminator is fairly certain that the generator is not the real data, the distribution generated from this generator is far from the underlying target distribution which is the image input. To solve this we rather maximize $D(G(Z))$ as this will give stronger gradients during back-propagation and thus learn faster how to fool the discriminator, so the aim is to increase the log probability of the discriminator to mistaken the generator input as the data input, rather to improve the generator impprove at discerning real from

fake. The only difference with the previous emphasis is that $\mathrm{E}_{p_z(z)}\left[\log\left(1 - D(G(Z))\right)\right]$ not quickly becomes 0 thus not taking the adversiality into account.

2.2 Building GAN

- Question 2.5
- Question 2.6
- Question 2.7

3 Generative Normalizing Flows

3.1 Change of variables for Neural Networks

- Question 3.1
  If the mapping of f is an invertible smooth mapping and $x \in \mathbb{R}^m$ is a multivariate random variable then we will require the determinant of **Jacobian** of the input data instead of just the derivatives, formally:

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

$$p(x) = p(z) \det \left| \frac{\partial f}{\partial x} \right|^{-1} \qquad \text{change of variables}$$

  where $\frac{\partial f}{\partial x}$ is the Jacobian. Equivallently for our objective we will obtain the following:

$$\begin{aligned} \log p(x) &= \log p(z) + \sum_{l=1}^{L} \log \det \left| \frac{\partial h_l}{\partial h_{l-1}} \right| \\ &= p(z) + \sum_{l=1}^{L} \log \det \left| J^T J \right| \end{aligned}$$

  with $h_0 = x$ and $h_L = z$.
- Question 3.2
  This equation is computable if for each preceding dimensions the Jacobian is easily computed. Formally put, each preceding h_{l-1} has to have a defined determinant and be invertible.
- Question 3.3
  Calculating $N \times N$ Jacobians are expensive operations which have complexity $O(N^3)$ which is not feasible for higher dimensional data. When we sample data depending of the complexity of the original data, the sampling may be fast or slow.
- Question 3.4
  The consequence might be that the model will encourage degenerate solutions that puts probability mass on discrete data points, so the representation of the true distribution of the original data is not entirely generated. To solve this **FLOW++** offered to add noise from an uniform distribution in the input data such that no probability mass is gathered around specific values. This change the discrete integers input to continuous random variables.

3.2 The coupling-layers of Real NVP

- Question 3.5
  1. Bijective mapping of input data
  2. Rate of transformation of data input depending on a part of the data, in other words the Jacobian of some input of the data.
  3. This is the scaling of the input image conditioned on only previous data input and that in fact may be complex and this is the part of the neural network output.
  4. This is the determinant of every previous input data transformation, so this means it contains on its diagonal elements of scaling of a part of input data.

5. Because of the presence of the $\exp$ in the diagonal entries of the Jacobian and the restriction to be invertible the determinant of the Jacobian is positive indeed.
6. Taking the $\log$ is equivalent of taking the $\exp$ thus the diagonal entries of the Jacobian contains exponential term.
7. This is sampled input from the latent space and due to the fact that the set-up of the bijective function the backward computation of this statement is the reverse order of the original bijective function.

3.3 Building a flow-based model

- Question 3.6
  During training data input is conditioned, from a multivariate distribution to the product of one-dimensional conditional densities and are fed to the network, a part of the total image. In a recursive way the current input is fed through successive coupling layers with each layer alternating its input source from the previous conditioned data input which was left unchanged. After K layers the latent vector is obtained and its log-probability calculated from the latent distribution. This result is added to the sum of logged of the determinants to compute the negative log-likelihood of the input data, this will be taken as a loss and back-propagated. After training the the input may be another image or larger data set and the coupling layers will have classification power as each contains information of loss at a finer scale. The output will be a distribution of the representation of the input data.
- Question 3.7
  See code

  `a3_nf_template.py`
- Question 3.8

4 Conclusion

- Question 4.1