ECE598PV: Representation Learning: Algorithms and Models Fall 2017

Lecture 11: Variational Auto-Encoders

Lecturer: Pramod Viswanath *Scribe: Sujan Gonugondla, Oct 21, 2018*

11.1 Introduction

Generative modeling is one of the important problems in Machine learning. It deals with estimating a probability distributions $P(X)$, defined over in high dimensional space $\mathcal{X}$. In this lecture we look at the problem where we need to construct this *probabilistic model* ($P(X)$) given some example data $\{X_1, X_2, .., X_i\}$ of the in the high-dimensional space $\mathcal{X}$. Here each data point X_i could be an image with millions of pixels, and the task is to capture all the dependencies and the frequency of these pixels, so that one can sample a entirely new image significantly different from any of the provided input images X_i. Primarily, we are interested in approximating this distribution using a neural network.

There are two popular techniques proposed today that attempt to solve this problem, which are:

- Variation auto-encoders (VAEs)[KW13, RMW14]
- Generative adversarial networks (GANs)[GPAM$^+$14]

Both the techniques target to repentant the probability distribution $P(X)$ in the form of a neural network, which takes a random latent variable Z with a probability $P(Z)$ as input and output a generated data-point $X_Z = f(Z)$ (see Fig. 11.1). Typically $P(Z)$ is a zero mean Gaussian random variable $\mathcal{N}(0, 1)$. The goal of the above algorithms is to approximate a function f using neural nets so that $P(f(Z)) \approx P(X)$.

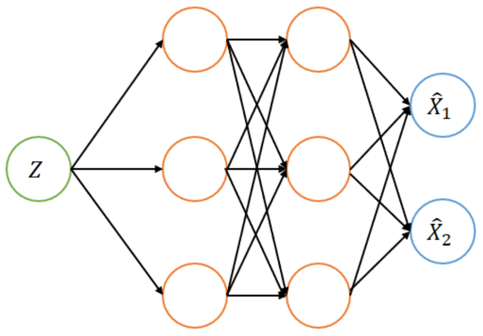


Figure 11.1: Image generation using neural network , by latent space modeling

11.2 Background

11.2.1 Latent variable models

Training a generative model with dependencies between dimensions in probability distributions are typically hard to train. This problem is much more difficult as the number of dimensions are increases. This problem can be simplified by introducing a random variable in a lower dimensional space that can be later mapped to the high dimensional space deterministically.

Suppose the we need to approximate a function $P(X)$ defined over a high dimensional space $\mathcal{X}$. It will be useful to introduce a latent variable in a lower dimensional space $\mathcal{Z}$ with a certain probability distribution $P(Z)$, to map it using a family of deterministic functions $f_\theta(Z)$, which are parameterized by θ in some space Θ, where $f : \mathcal{Z} \times \Theta \mapsto \mathcal{X}$. Using this notion of latent space $\mathcal{Z}$. The generative model learning will be simplified as finding the parameters θ, and the distribution $P(Z)$ such that $P(X) \approx P_\theta(X)$, where

$$P_\theta(X) = \int f_\theta(Z)P(Z)dZ. \tag{11.1}$$

For more details refer to [Doe16].

11.2.2 Auto-encoders

Auto-encoders were proposed as a compression/denoising [VLBM08] technique that uses neural network to map a high dimensional input $X \in \mathcal{X}$ to a low dimensional space $Z \in \mathcal{Z}$, such that it can be decoded (again using neural networks) to obtain the original image.

The encoder network estimates a low dimensional latent variable $Z \in \mathcal{Z}$ given an input $x \in \mathcal{X}$ represented by $Z = f_{\theta_e}(X)$. The quality of the encoder can be determined by the ability of a decoder to recover the input data from its representation in its latent space. This is solved by using an other neural network that maps Z to the higher dimensional space $\mathcal{X}$, represented by $f_{\theta_d}(Z) \in \mathcal{X}$ Auto-encoders try to train both the network encoders and decoders together by concatenating them as shown in Fig. 11.2, and by setting the target outputs of this concatenated network to the inputs. Thus resulting in the overall target function that needs to learned to be $h_{\theta_e,\theta_d}(X) \approx X$.

11.3 Variational Auto-Encoders

The mathematical basis of Variational Auto-encoders (VAEs) have little to do with the classical auto encoders discussed in Section 11.2.2. VAEs are a learning algorithm that uses the latent variable model to approximate the target probability distribution $P(X)$. Specifically, it is attempting to find the parameters θ such that the probability distribution $P_\theta(X)$ from Equation 11.1 approximates the *real* probability distribution $P(X)$.

The VAEs are called auto encoders as the objective function that needs to minimized is inspired from the traditional auto encoders (Section 11.2.2). The two major problems that must be deal with, in order to solve the generative problem defined by Equation 11.1. is to define a latent variable Z and to deal over the integral over Z. VAE tackle this problems by assuming that the there is no

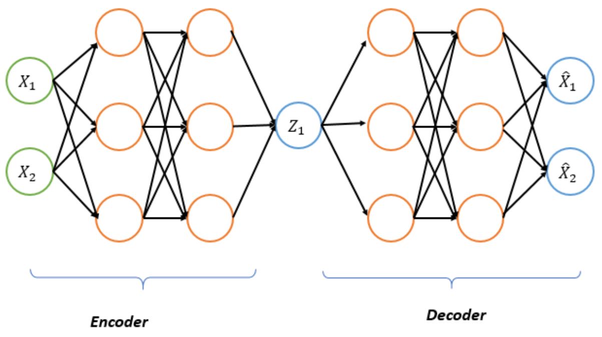


Figure 11.2: A toy example of a variational autoencoder. Here the input image is mapped to a low-dimensional space using *encoder* and a *decoder* is trained to recover the original image by concatenating both as shown in the figure

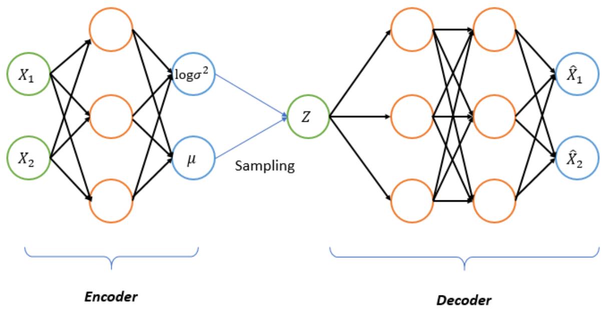


Figure 11.3: Variational auto encoders is used to generate image based on a a latent random variable Z. Encoder provides with the distribution of Z that is likely to generate the required example

simple interpretation of Z, and hence it can be modeled as a simple Gaussian distribution $\mathcal{N}(0, I)$, where I is an identity matrix with dimensions of the space $\mathcal{Z}$. By modeling Z as an uncorrelated Gaussian random variable, VAEs relax the burden of finding the structure of the latent variable.

Now, the big questions that are to be answered are: 'How to optimize for θ?', and 'How to find the function $f_\theta(Z)$?'. In order to find the distribution $P_\theta(X|Z)$, we may need to know the distribution $P(Z|X)$, so that we can sample Z that are likely to generate X. This is required to train a neural net that performed : $X = f_\theta(z)$. The way VAEs apporch this issue it to define a *encoder* $Q_\phi(Z|X)$ that provides the distribution of Z that is likely to generate X using the *decoder*(generator), $f_\theta(z)$.

In VAEs *encoder* ($Q_\phi(Z|X)$) is modeled using a neural network that computes the mean and variance,

$$Q_\phi(Z|X) = \mathcal{N}(\mu_\phi(X), \sigma_\phi(X)) \tag{11.2}$$

While the *decoder* (generator) estimates data point based on a Z sampled from $Q_\phi(Z|X)$. VAEs by imposing this structure, has enabled the training of the encoder and decoder together, using back-propagation (See Fig. 11.3).

In-order to train this large network, requires us to choose an objective function to minimize over. We can solve this problem by considering a probability distribution of the $P_\theta(X)$ defined over

$\mathcal{X}$. The goal of the training is to maximize the log likelihood $P_\theta(X)$. That is, to maximize the probability that the decoder regenerates X. However, it is not clear on how one could practically use back-propagations using this. We need to find an other objective function that can be empirically to maximize $\log P_\theta(X)$

$$\log P_\theta(X) = \int_Z Q_\phi(Z|X) \log P_\theta(X) dZ \tag{11.3}$$
$$= \int_Z Q_\phi(Z|X) \log \frac{P_\theta(X,Z)}{P_\theta(Z|X)} dZ \tag{11.4}$$
$$= \int_Z Q_\phi(Z|X) \log \left(\frac{P_\theta(X,Z)}{Q_\phi(Z|X)} \frac{Q_\phi(Z|X)}{P_\theta(Z|X)} \right) dZ \tag{11.5}$$
$$= \int_Z Q_\phi(Z|X) \log \frac{P_\theta(X,Z)}{Q_\phi(Z|X)} dZ + \int_Z Q_\phi(Z|X) \log \frac{Q_\phi(Z|X)}{P_\theta(Z|X)} dZ \tag{11.6}$$
$$= \mathcal{L}(P_\theta, Q_\phi) + D_{KL}(Q_\phi, P_\theta) \tag{11.7}$$

The second term $D_{KL}(Q_\phi, P_\theta)$ can be assumed to be small as we expect $Q_\phi(Z|X)$ to approximate well. While the first term $\mathcal{L}(P_\theta, Q_\phi)$ forms the empirical lower bound of the log likelihood $\log P_\theta(X)$. In order to maximize the log likelihood, $\log P_\theta(X)$ we need to maximize $\mathcal{L}(P_\theta, Q_\phi)$.

$$\log P_\theta(X) \geq \mathcal{L}(P_\theta, Q_\phi) = \int_{\mathcal{Z}} Q_\phi(Z|X) \log \frac{P_\theta(X,Z)}{Q_\phi(Z|X)} dZ \tag{11.8}$$
$$= \int_{\mathcal{Z}} Q_\phi(Z|X) \log \frac{P_\theta(X|Z)P(Z)}{Q_\phi(Z|X)} dZ \tag{11.9}$$
$$= \int_{\mathcal{Z}} Q_\phi(Z|X) \log P_\theta(X|Z) dZ + \int_{\mathcal{Z}} Q_\phi(Z|X) \log \frac{P(Z)}{Q_\phi(Z|X)} dZ \tag{11.10}$$
$$= \mathbb{E}_{q_\phi}[\log p_\theta(X|Z)] - D_{KL}(Q_\phi, P) \tag{11.11}$$

Here since $Q_\phi(Z|X)$ and $P(Z)$ are Gaussian random variables,

$$Q_\phi(Z|X) = \mathcal{N}(\mu_\phi, \sigma_\phi), \tag{11.12}$$
$$P_\phi(Z) = \mathcal{N}(0, I), \tag{11.13}$$

we can analytically estimate that $-D_{KL}(Q_\phi, P)$.

$$D_{KL}(Q_\phi, P) = \int_{\mathcal{Z}} Q_\phi(Z|X) \log \frac{P(Z)}{Q_\phi(Z|X)} dZ \tag{11.14}$$
$$= \int_{\mathcal{Z}} Q_\phi(Z|X) \log P(Z) dZ - \int_{\mathcal{Z}} Q_\phi(Z|X) \log Q_\phi(Z|X) dZ \tag{11.15}$$

$$= \int_{\mathcal{Z}} \mathcal{N}(\mu_\phi, \sigma_\phi) \log \mathcal{N}(0, I) dZ - \int_{\mathcal{Z}} \mathcal{N}(\mu_\phi, \sigma_\phi) \log \mathcal{N}(\mu_\phi, \sigma_\phi) dZ \tag{11.16}$$

$$= \Big(-\frac{N_z}{2\pi} - \frac{1}{2} \sum_{j=1}^{j=N_Z} (\mu_{\phi,j}^2 + \sigma_{\phi,j}^2) \Big) + \Big(-\frac{N_z}{2\pi} - \frac{1}{2} \sum_{j=1}^{j=N_Z} (1 + \log \sigma_{\phi,j}^2) \Big) \tag{11.17}$$

$$= \frac{1}{2} \Big(\sum_{j=1}^{j=N_Z} (1 + \log \sigma_{\phi,j}^2 - \mu_{\phi,j}^2 - \sigma_{\phi,j}^2) \Big) \tag{11.18}$$

Now that $D_{KL}(Q_\phi, P)$ is estimated analytically $\mathbb{E}_{q_\phi}[\log p_\theta(X|Z)]$ can be estimated only empirically. Intuitively, the KL divergence term can be interpreted as a regularization while the $\mathbb{E}_{q_\phi}[\log p_\theta(X|Z)]$ is the term estimating reconstruction quality. Observe that the $\mathbb{E}_{q_\phi}[\log p_\theta(X|Z)]$ is an estimate of the likelihood of the reconstruction via the *decoder* when generated with Z sampled from the distribution obtained by the encoder. This can be empirically approximated as

$$\mathbb{E}_{q_\phi}[\log p_\theta(X|Z)] = \frac{1}{L} \sum_{i=1}^{L} \log p_\theta(X|Z_i) \tag{11.19}$$

Using this empirical cost functions gradients can be taken estimated for techniques such as stochastic gradient descent algorithm to train a network as described in Fig. 11.3. The empirical cost function is given by

$$\tilde{\mathcal{L}}(P_\theta, Q_\phi) = \frac{1}{2} \Big(\sum_{j=1}^{j=N_Z} (1 + \log \sigma_{\phi,j}^2 - \mu_{\phi,j}^2 - \sigma_{\phi,j}^2) \Big) + frac1L \sum_{i=1}^{L} p_\theta(X|Z_i) \tag{11.20}$$

However in-order to use the standard tools for stochastic gradient descent is is good to avoid the sampling step, where Z is sampled from the distribution obtained from the *encoder* $Q_\phi(Z|X)$. This is achieved by sampling $\hat{Z}$ from a a zero mean normal distribution $\mathcal{N}(0, I)$. And using this as in input of the network, were the sampling step is replaced by a multiplication and addition using,

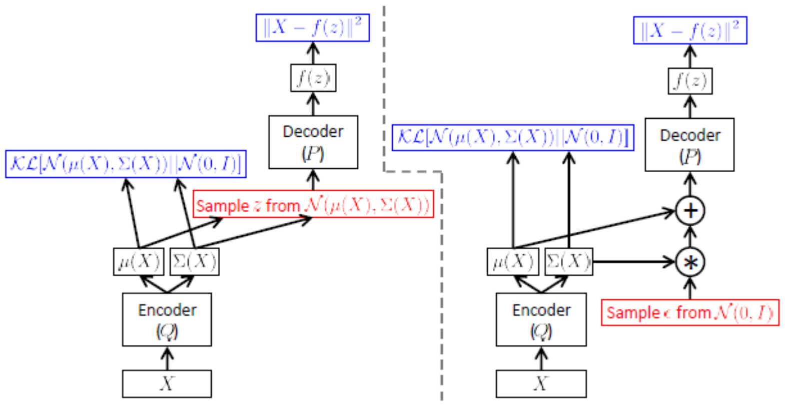
$$Z = \mu_\phi + \sigma_\phi \hat{Z}. \tag{11.21}$$

This *re parametrization trick* is described in the Fig. 11.4.

During practical implementations the log likelihood is maximized, $frac1L \sum_{i=1}^{L} p_\theta(X|Z_i)$ using a L_2 distance metrics to minimize on. However there could be many different way in assessing similarity of the input and output image. Some of the generated images using VAEs are shown in Fig. **??**

11.4 Extensions to Variational Auto-encoders

Since its first inceptions there have been many works that have proposed variations over the original algorithm to address applications needs. I have restricted a couple of interesting works, though there are many more interesting works that can be talked about,



$\|X - f(z)\|^2$
$f(z)$
Decoder (P)
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) \| \mathcal{N}(0, I)]$
Sample z from $\mathcal{N}(\mu(X), \Sigma(X))$
$\mu(X)$
$\Sigma(X)$
Encoder (Q)
X
$\|X - f(z)\|^2$
$f(z)$
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) \| \mathcal{N}(0, I)]$
Decoder (P)
$+$
$\mu(X)$
$\Sigma(X)$
$*$
Encoder (Q)
Sample ϵ from $\mathcal{N}(0, I)$
X

Figure 11.4: The re-parameterization trick allows the variational auto encoder to behave like a standard neural network for training using the stranded tools. Here the input X is sampled from the set of training images while Z is sampled from a $\mathcal{N}(0, I)$. Figure obtained from [Doe16]

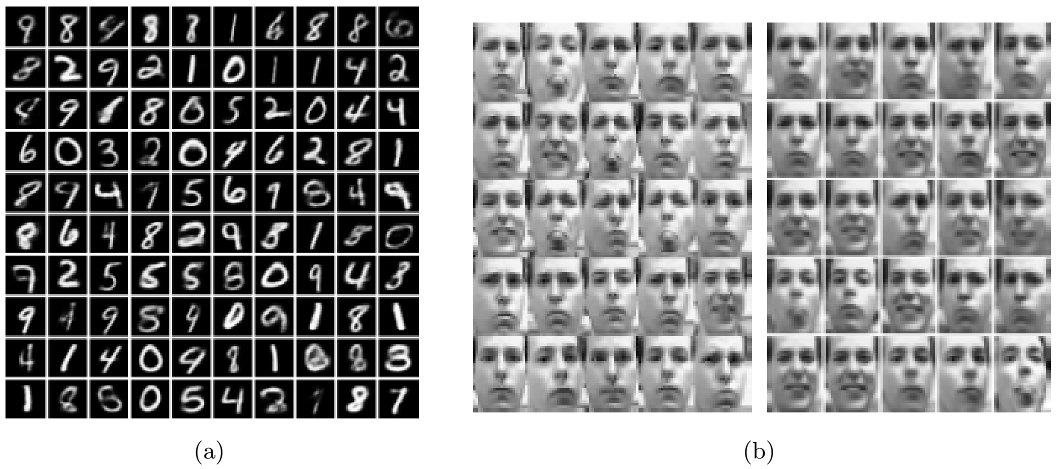


Figure 11.5: Images generated using Variational Auto Encoders: a) Using MNIST dataset [Doe16] b) Using XX dataset [RMW14]

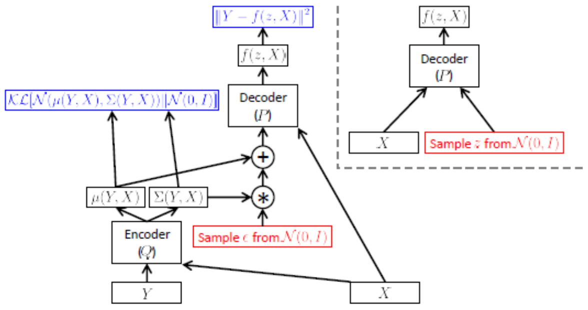

$\|Y - f(z,X)\|^2$
$f(z,X)$
$\mathcal{KL}[\mathcal{N}(\mu(Y,X),\Sigma(Y,X))\|\mathcal{N}(0,I)]$
Decoder (P)
+
*
$\mu(Y,X)$
$\Sigma(Y,X)$
Encoder (Q)
Sample ϵ from $\mathcal{N}(0,I)$
Y
X
$f(z,X)$
Decoder (P)
X
Sample z from $\mathcal{N}(0,I)$

Figure 11.6: Conditional Variational Auto-Encoders. Figure from [Doe16]

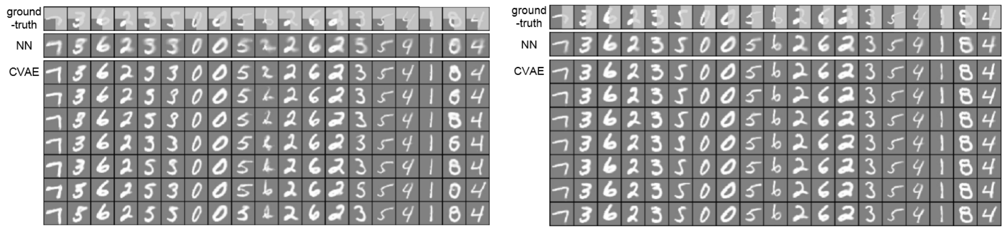


Figure 11.7: Image completion using the CVAEs, the shaded quarters on the top were provided while the unshaded part of the image was generated. Figure from [SLY15]

11.4.1 Conditional Variational Autoencoders

Conditional Variational Auto-encoders (CVAE) is proposed [SLY15] as an extension on VAEs that imposes some structural constraints on the output based on the priors X. The conditional generative process of the model is given in Fig. 11.6 as follows: for given observation X, Z is drawn from the prior distribution $Q_\phi(z|x)$, and the output y is generated. Here the output Y is generated conditioned on the input X and some latent variable Z. This can be used for image completion as shown for the MNIST in Fig. 11.7

11.4.2 Deep Convolutional Inverse Graphics Network

This work proposed by Tejas D. Kulkarni et, al. [KWKT15] attempts to model Z so as to obtain a specific output based on the latent variable Z. Here they train the VAE so that latent variables Z represent a specific transformation of the input (See Fig. 11.8). The Fig. 11.9 shows the outputs obtained by training using this method.

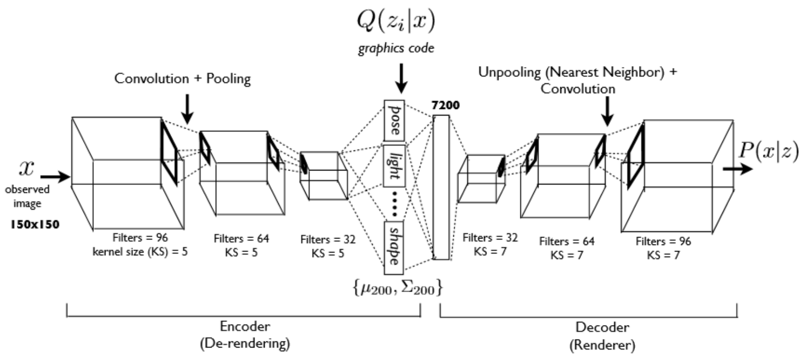


Figure 11.8: The Convolutional VAE (DCIGN) that is proposed to generate images with transformation given a provided image. Figure from [KWKT15].

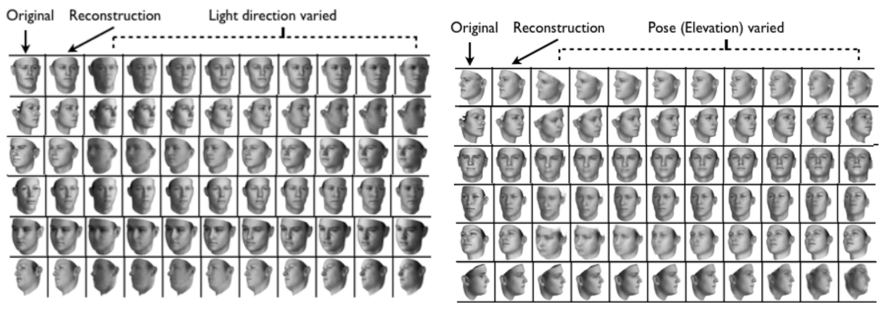


Figure 11.9: Images generated using the DCIGN. Figure from [KWKT15].

Bibliography

[Doe16] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[GPAM+14] 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.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[KWKT15] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

[VLBM08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.