

Project 5

Generative Models (GANs & VAEs)

Group Members

Mohammed Abukalam : mabukalam@aimsammi.org

Alioune Diankha : adiankha@aimsammi.org

Raissa Manfouo : rfeuyo@aimsammi.org

John Ndolo : jndolo@aimsammi.org

Yvan Pimi : ypimi@aimsammi.org

29. April 2022

African Institute for Mathematical Sciences – Senegal (AIMS Senegal)

OUTLINE

- ➊ Generative Models
- ➋ Variational Autoencoders(VAEs)
- ➌ Generative Adversarial Networks (GANs)
- ➍ Implementation of VAEs and GANs
- ➎ Comparison of VAEs to GANs

OBJECTIVES

- 1 Define generative models and their usefulness
- 2 To implement generative models, understand their losses and the intuition behind them.
- 3 Explain how variational auto encoder works, and the difference between it and the normal auto-encoder
- 4 Quantify the distance between a learned distribution and the normal distribution.
- 5 Describe what WGAN is
- 6 The research idea behind WGAN.

GENERATIVE MODELS

Definitions

Generative models: Models which sample from a learned distribution

- Models the distribution of individual classes
- Learn the joint probability distribution i.e $P(X, Y)$

$$P(X, Y) = P(X|Y) * P(Y)$$

Why generative Models useful?

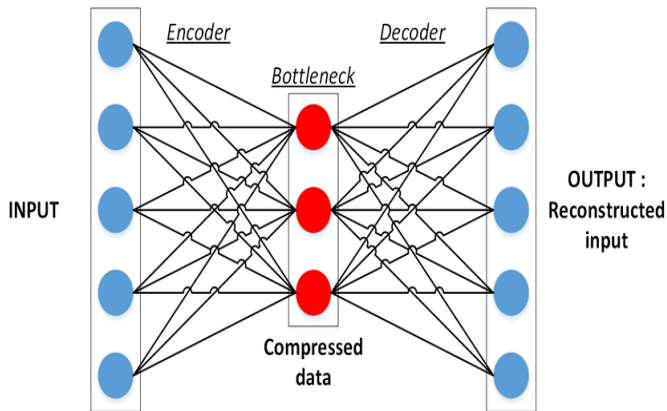
- 1 Data augmentation (Generate more data to increase model accuracy)
- 2 Increase images resolution,colorization
- 3 In time series data, the models can be used in simulation (Common in reinforcement learning applications)

Variational Autoencoders(VAEs)

NORMAL AUTO-ENCODERS

- ➊ It is unsupervised approach for learning a lower dimensional feature representation from unlabeled training data
- ➋ The objective in auto-encoders is to train input data such that the the learned features can be used to reconstruct the original data
- ➌ The auto-encoder is solely trained to encode and decode with as few loss as possible, no matter how the latent space is organised
- ➍ Features capture the meaningful factors of variation in the training data
- ➎ Let x be input data, z be the latent factor(variable), \hat{x} be the reconstructed data

SIMPLE AUTO-ENCODER REPRESENTATION



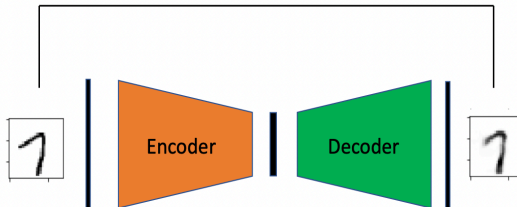
WORKED EXAMPLE

The objective is to minimize the reconstruction loss. MSE is used as the metric for the loss.

Example

Minimize squared error loss:

$$\mathcal{L} = ||\mathbf{x} - Dec(Enc(\mathbf{x}))||_2^2$$



LIMITATION OF AUTO-ENCODERS

Limitation of Auto-encoders

- 1 It can not be used to generate data since it does not define a sampling distribution
- 2 The major issue with normal auto-encoders is that the latent space(z) that the inputs are converted to are discrete(not continuous) hence not easy to allow interpolation
- 3 The generative part of the auto encoder works by randomly picking a sample from the latent space which is challenging if its discontinuous or has gaps.
- 4 To solve the issue we settle for Variational Auto-encoders

VARIATIONAL AUTO-ENCODERS

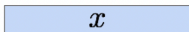
- 1 Built on the architecture of Auto-encoders
- 2 A variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.
- 3 In order to introduce some regularisation of the latent space, we proceed to a slight modification of the encoding-decoding process: **instead of encoding an input as a single point, we encode it as a distribution over the latent space**
- 4 Given input data, the encoder generates two vectors which represent the parameters of the sampling distribution:
 - Vector mean μ
 - vector σ

TRAINING THE MODEL

Given data $\{X_i\}_{i=1\dots n}$, train a generative model, to maximize the likelihood of the observed data

Sample from
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$



Sample from
true prior

$$p_{\theta^*}(z)$$

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

where we maximize the likelihood $P_{\theta}(x, z)$ by estimating the true parameters θ of the generative model.

INTRACTABILITY

We choose the prior $P(z)$ to be simple e.g Gaussian $N(0, 1)$. Learn the model to determine the maximum likelihood of data;

$$P_{\theta}(x) = \int P_{\theta}(x, z) dz \quad (1)$$

$$= \int P_{\theta}(z) P_{\theta}(x|z) dz \quad (2)$$

Which is difficult to compute for all latent variable(z). Posterior density is also intractable because of intractability of the data likelihood;

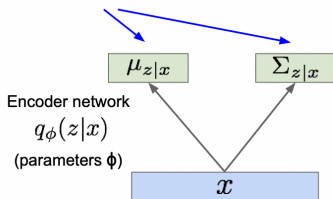
$$P_{\theta}(z|x) = \frac{P_{\theta}(x|z)P_{\theta}(z)}{P_{\theta}(x)} \quad (3)$$

To solve the issue of intractability, we approximate the sampling distribution of $P_{\theta}(z|x)$ by $q_{\phi}(z|x)$ in the encoder in addition to the decoder network modelling $P_{\theta}(x|z)$.

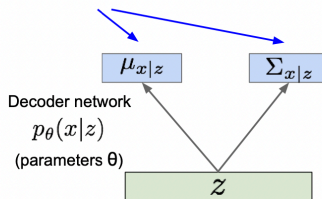
VARIATIONAL AUTOENCODERS

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic

Mean and (diagonal) covariance of $\mathbf{z} | \mathbf{x}$



Mean and (diagonal) covariance of $\mathbf{x} | \mathbf{z}$



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

VARIATIONAL AUTOENCODERS

Intractability

The introduction of $q_\phi(z|x)$ allow us to derive a lower bound on the data likelihood that is tractable, which we can optimize.

Thus the log data likelihood can now be calculated as follows;

$$\begin{aligned}\log p_\theta(x^{(i)}) &= \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} \left[\log p_\theta(x^{(i)}) \right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\&= \mathbf{E}_z \left[\log \frac{p_\theta(x^{(i)} | z) p_\theta(z)}{p_\theta(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\&= \mathbf{E}_z \left[\log \frac{p_\theta(x^{(i)} | z) p_\theta(z)}{p_\theta(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\&= \mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbf{E}_z \left[\log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] && (\text{Logarithms}) \\&= \underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z | x^{(i)}))}_{\geq 0}\end{aligned}$$

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

VARIATIONAL AUTOENCODERS

Now; we maximize the likelihood lower bound (The tractable part).
The first term represents the reconstruction term and the second term is the **Kullback-Leibler(KL) divergence**.

$$\mathcal{L}(x^{\{i\}}, \theta, \phi) = \mathbb{E}_z \left[\log p_{\theta}(x^{\{i\}}|z) \right] - D_{KL} \left(q_{\phi}(z|x^{\{i\}}) || P_{\theta}(z) \right) \quad (4)$$

The first term, the conditional probability of x given z , maximize likelihood of original input being constructed while the second term, we try to make the posterior distribution close to the prior.

ALGORITHM

Data:

\mathcal{D} : Dataset

$q_{\phi}(\mathbf{z}|\mathbf{x})$: Inference model

$p_{\theta}(\mathbf{x}, \mathbf{z})$: Generative model

Result:

θ, ϕ : Learned parameters

$(\theta, \phi) \leftarrow$ Initialize parameters

while *SGD not converged* **do**

$\mathcal{M} \sim \mathcal{D}$ (Random minibatch of data)

$\epsilon \sim p(\epsilon)$ (Random noise for every datapoint in \mathcal{M})

 Compute $\tilde{\mathcal{L}}_{\theta, \phi}(\mathcal{M}, \epsilon)$ and its gradients $\nabla_{\theta, \phi} \tilde{\mathcal{L}}_{\theta, \phi}(\mathcal{M}, \epsilon)$

 Update θ and ϕ using SGD optimizer

end

Generative Adversarial Networks(GANs)

Structure : How it's works

GAN has two parts :

- Generator : Generate the new instances of images from random images
- Discriminator : Discriminate between different kinds of data instances
- And both of them are neural Networks

GENERATIVE MODEL (GAN:GENERATIVE ADVERSARIAL NETWORKS.)

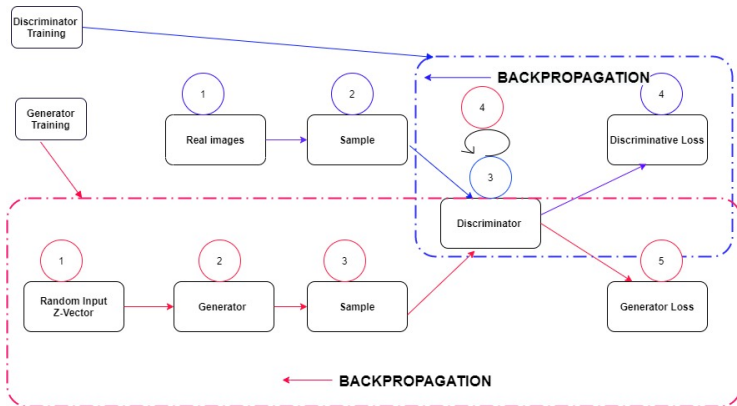
Structure : How it's works

More Formally , given a set of data instances X and a set of labels Y :

- Generator Models capture the joints probability $P(x,y)$ or just $p(x)$ there are no labels .
- Discriminator Models capture the conditional probability .

GENERATIVE MODEL (GAN:GENERATIVE ADVERSARIAL NETWORKS.)

Structure : How it's works



GENERATIVE MODEL (GAN:GENERATIVE ADVERSARIAL NETWORKS.)

Problem With GAN Model

GAN Problem come from LBCE (Binary Cross Entropy Loss) Problem .

- 1 No Stability which converge during the training
- 2 Overfitting and vanishing gradient problem ,
- 3 Mode Collapse due to the underlying cost function of the whole architecture , No high-quality images also based on Mode collapse

WGAN:WASSERSTEIN GAN

- The Wasserstein GAN algorithm is a variation of generative adversarial networks (GANs),
- proposes a new cost function using Wasserstein distance that has a smoother gradient everywhere,
- Resolve the problem from GAN ,
- Minimax Game (Generator \Rightarrow Minimize , Distributor(Critic) \Rightarrow Maximize)

WGAN:WASSERSTEIN GAN

Equation of The Earth-Mover (EM) distance or Wasserstein-1 Distance

$$W(\mathbb{P}_r, \mathbb{P}_g) = \underbrace{\sup}_{\|f\|_L \leq 1} \underbrace{\mathbb{E}_{x \sim \mathbb{P}_r} [f(x)]}_{\text{Real data}} - \underbrace{\mathbb{E}_{x \sim \mathbb{P}_g} [f(x)]}_{\text{Generator}} \quad (5)$$

Contraint on discriminator

- where sup is the least upper bound and f is a 1-Lipschitz function following this constraint $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$, $K = 1$ and x come from the same distribution .
- and Maximizes $\mathbb{E}_{x \sim \mathbb{P}_r} [f(x)] - \mathbb{E}_{x \sim \mathbb{P}_g} [f(x)]$ where \mathbb{P}_r and \mathbb{P}_g are two probability distribution.

K Lipschitz continuity

A real function $f: \mathbb{R}$ to \mathbb{R} is Lipschitz continuous if :

- $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$
- $\frac{|f(x_1) - f(x_2)|}{|x_1 - x_2|} \leq K$ for all real value of x_1, x_2
- Lipschitz constant equals the maximum absolute value of the derivatives.

Loss Function

- 1 Approximates the earth mover's distance between real and generated distribution but it has nicer properties than BCE
- 2 calculate difference between the expected values of the predictions of the discriminator , we called it a critic
- 3 W Loss prevents mode collapse and vanishing gradient problem

WGAN:WASSERSTEIN GAN

Loss Function

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(x^{(i)}) + \log (1 - D(G(z^{(i)}))) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (D(G(z^{(i)})))$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^m [f(x^{(i)}) - f(G(z^{(i)}))]$	$\nabla_{\theta} \frac{1}{m} \sum_{i=1}^m f(G(z^{(i)}))$

WGAN:WASSERSTEIN GAN

Algorithm

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.



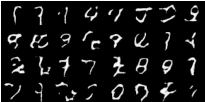
Require: : α , the learning rate. c , the clipping parameter. m , the batch size. n_{critic} , the number of iterations of the critic per generator iteration.

Require: : w_0 , initial critic parameters. θ_0 , initial generator's parameters.

```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while
```

IMPLEMENTATION RESULTS

Some Results

Model	original image	predicted or generated
VAE		
		

SUMMARY

GANs

Pros

- 1 Allows inference of $q(z|x)$, can be useful feature representation for other tasks
- 2 With the VAE model we can get an representation of the data and use it in a downstream application

Cons

- 1 Samples blurrier and lower quality

VAEs

Pros

- 1 Beautiful, state-of-the art sample!

Cons

- 1 Trickier/more unstable to train (but more stable training WGAN)
- 2 Can't be used for inference of $p(z|x)$ or $p(x)$

COMPARISON (VAE & (W)GAN)

- 1 Both are used to approximate probability distributions of datasets and generate complex synthetic pictures.
- 2 Although both use parameterized distributions to approximate the underlying data distribution, whose exact inference is intractable, their behaviors are very different.
- 3 Although WGAN avoids mode collapse, we found that the generated images still do not look very realistic, as there is no term in the objective function that encourages the synthetic data to look like the training data. This is encouraged implicitly in another type of deep generative model, VAE.¹

¹Lu Mi ,Macheng Shen and Jingzhao Zhang. A Probe Towards Understanding GAN and VAE Models. <http://arxiv.org/abs/1812.05676>

COMPARISON(VAE & (W)GAN)

- 1 the sample images generated by VAE are usually blurry and of lower quality compared to those from GAN models.²
- 2 One advantage of VAE models over GAN models is that it could map an input in the original dataset to latent factors and further to an image in the generator's approximation.
- 3 The process of WGAN is to minimize the loss, which is implicitly minimizing the Wasserstein distance between the generative distribution and the data distribution.
- 4 For VAE, the training process is to reduce the mean squared error between itself and the target and the KL divergence between the encoded latent variable and standard normal distribution.

²Lu Mi ,Macheng Shen and Jingzhao Zhang. A Probe Towards Understanding GAN and VAE Models. <http://arxiv.org/abs/1812.05676>

CONCLUSION

From the results of our experiments, we can said that each model has its own advantage, while sharing the same objectives.

THANK YOU, QUESTIONS?

