Generative Models for High Dimensional Density Estimation

Taorui Wang

May 2022

1 Introduction

One of the major goal of the statistics and machine learning is estimating the probability density from given samples. Traditional density estimation method such as histograms and Kernel density estimation works well for low dimensional case. However, the problem becomes tricky as it is about the high dimensional data. With the application of deep learning, some techniques have been developed. This article will focus on generative model which addresses this problem.

The generative models are important because they can produce samples and learns the probability density functions.

The generative models can be divided into four families: 1. generative adversarial network(GAN), 2.variational auto-encoders(VAE). 3. autoregressive encoder. 4. normalizing flow. Sometimes, these methods can be intertwined. All of models from the four families can be used to do the density estimation even if some models can not give an explicit log likelihood.

The structure of methods use generative models for density estimation can be divided into three parts:1. generative model structure. 2. density estimator. 3.objective function(estimator of the objective function).

I will mainly focus on four models which build the foundations for the density estimation of the generative model. They are: Generative Adversarial Network, Variational Autoencoder, Masked Autogressive Autoencoder and Normalizing Flow.

These models address the problems arisen from the density estimation by modifying generative model structure, density estimator, or objective function(estimator of the objective function).

This review will give a general description of these methods. It will explain the motivation and the basic idea of all of those methods and some variants.

2 Generative Adversarial Network

2.1 GAN architecture

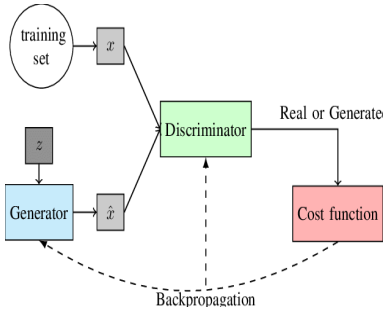
The Generative Adversarial Network [3] establishes a framework which makes a generative model compete with its adversary.

Given data $x \in R^m$ whose distribution is unknown and is denoted as p_x. Prior distribution $z \in R^m$ whose distribution is known and is denoted p_z.

A Generative Adversarial Network is composed of two parts- a generative model, G, and a discriminative model, D. The discriminative model $D(x)$, is a neural network model which outputs a scalar given the data point x. $D(x)$ represents the probability. And the generator G is a neural network model whose input is z which is sampled from prior distribution p(z). And $G(z)$ sends z to the data space. D is trained to maximize the probability to distinguish the true data. G is trained to make the discriminator to believe that $G(z)$ is from the data distribution. And this can be expressed in the following way:

$$\min_G \max_D E_{x\sim p_x}[logD(x)] + E_{z\sim p_z}[log(1 - D(G(z))]$$

The discriminator D and G are trained alternatively in two stages: 1.Train the discriminator D from data x to distinguish the fake sample generated by G. 2.Train the generator G to fool the discriminator.



2.2 GAN discussion

The Generative Adversarial Networks are good at generating data and is flexible. However, GANs are an implicit generative model. It can not compute the log likelihood function explicitly as the flow-based model. Training GANs are also unstable and delicate.

2.3 Wasserstein Generative Adversarial Network (WGAN)

Since training GAN models can be unstable, this article[9] introduce Wasserstein Generative Adversarial Network(WGAN). WGANs wants to minimize the

distance between the distribution of the training dataset and the distribution the generated.

Assume the data distribution over the real sample is p_x and the data distribution from the generator G is p_g.

WGANs replace the "discriminator" D with the $1-$ Lipschitz functions $f : R^m \to R$. The loss function becomes the Wasserstein distance between the real data distribution and the generated data distribution:

$$W(p_x, p_g) = \max_{||f||_L \leq 1} E_{x \sim p_x}[f(x)] - E_{z \sim p_z} E[f(g(z)]$$

2.4 Generative Adversarial Density Estimator

Although the flow-based models can give us the exact formula of the log likelihood, they are computationally demanding, and they perform worse than GAN when it comes to the generation of data. GANs perform well when they generate data but they can not give an explicit log likelihood

To address these two problems. This article [11] gives us an density estimator with the use of the GANs. It proposes a generative adversarial density estimator modifies the learning objective to ensure the estimator to achieve good log likelihood. The article wants to maximize the normalizer for the probability density function of x:

$$A_q(w) = \sup_q \{\int \phi(w, x) q(x|\theta) dx - q(x|\theta) log(q(x|\theta))\} = \sup_q \{E(\phi(w, x)) - H_x(q)\}.$$

w is the parameter which specifies the density function of the sample x and $\phi(w, x)$ is the energy function for the Boltzmann distribution. When q equals the probability density function of p_x. $A_q(w)$ has the optimal value.

To compute the entropy of q,$H_q(x)$, $H_q(x) = H(p_z) + H_g$. $H(p_z)$ is the entropy of the noise and $H_g = -E_{p_z}[log\,|det(J_z^T J_z)|^{\frac{1}{2}}]$ and J_z is the Jacobian matrix of the generator g.

Another method, from this article[13], uses the Roundtrip as the framework for GAN models which makes the evaluation of the probability density feasible.

3 Variational Autoencoder

3.1 Motivations

Assume a dataset $D = x^{(i)}{}_{i=1}^{N}$ are samples from random variable x whose distribution is unknown and and z is an unobserved continuous random variable whose distribution is p_z. Assume that x is generated from some random process which involves z. x^i is generated from $p_{x|z}$. Assume that the parameter of p_z and $p_{x|z}$ are θ. With these assumptions, the posterior density $p_{z|x} = p_{x|z}p_z/p_x$

We want to efficiently approximate the maximum likelihood with respect to parameter θ, $p_{z|x}$ and p_x.

This article[1] tries to achive this goal with the framework of autoencoder.

3.2 Architecture of Variational Autoencoder

3.2.1 Generalization of the Variational Autoencoder

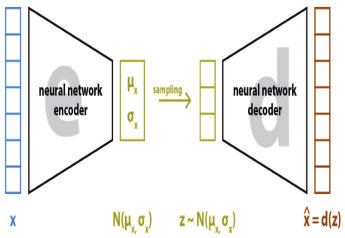
Variational Autoencoder method is an example of the Autoencoding Variational Bayesian(AEVB) algorithm with the Stochastic Gradient Variational Bayes(SGVB) Estimator. AEVB algorithm with SGVB estimator approximate the posterior distribution $p_{z|x}$ and gives a practical estimator of the variational lower bound with the reparameterization trick.

3.2.2 Variational Autoencoder

For Variational Autoencoder(VAE), first, we assume that the distribution of z is known as p_z and it is a multivariate standard Gaussian. We also assume that $x|z$ is also a multivariate Gaussian random variable. And $z|x$ is also assumed to be approximate multivariate Gaussian with diagonal variance.

We use $q_\phi(z|x)$ to approximate $p(z|x)$ and $q_\phi(z|x)$ has parameter ϕ. And $p_\theta(x|z)$ has the parameter θ.

With the framework of auto encoder $q_\phi(z|x)$ is the encoder, and z is the code, and $p_\theta(x|z)$ is the decoder. As the figure below:



The encoder gives us the random variable $z|x$. It is z conditioned on x. And we sample z from the conditional random variable $z|x$ and use the sample and decoder to generate $\widehat{x}$.

And We want to minimize the (variational) lower-bound on the log-likelihood of x:

$$E_x[E_{q_\phi(z|x)}[log(p_\theta(x|z)]] - E_x[KL(q_\phi(z|x)||p_z(z))]$$

To give an estimator of the lower bound, we want to use the samples we generate from the conditional distribution $z|x$ with $q_\phi(z|x)$. To generate samples, the model uses the reparameterization trick which assumes that $z|x$ is a function of some random variable ϵ and x, $z = g_\phi(\epsilon, x)$. g is a function which is also parameterized by ϕ. ϵ is some random variable with the distribution p_ϵ

The estimator for the formula above is:

$$\frac{1}{L}\sum_{l=1}^{L}(logp_\theta(x^{(i)}|z^{(i,l)})) - D_{KL}(q_\phi(z|x)||p_z(z))$$

$z^{(i,l)} = g_\phi(\epsilon^{(l)}, x^{(i)})$, ϵ is a random variable with distribution p_ϵ

In the case of Variational Autoencoder. Given the minibatch data $\{x^{(i)}\}_1^N$. $z|x^{(i)}$ is Gaussian distributed with $N(\mu^{(i)}, \sigma^{2(i)})$. We sample $z^{(i,l)}$ from the posterior distribution. $z^{(i,l)} = g_\phi(\epsilon^{(l)}, z^{(i)}) = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)}$. Note we use the MLP(Multi-layered perceptrons)(MLP) as the decoder for the calculation of $logp_\theta(x^{(i)}|z^{(i,l)})$

3.3 Discussion

The main contribution of the model is: 1. Using the reparameterization trick to give an estimator of the variational lower bound. 2. Using q_ϕ to approximate $p_{z|x}$ efficiently.

One of the advantage of Variational Autoencoder is flexibility. We can derive some new methods from it by generalizing the prior distribution of the random variable z and the training process of the encoder. We will see two examples of these changes.

4 Masked Autoencoder Distribution Estimator(MADE)

4.1 Motivation

This model[6] wants to make the autoencoder neural networks a tractable distribution estimator and work faster. The model enable the autoencoder to be a distribution estimator by imposing the masks. The maskss ensure the output

has the autoregressive property.

Considering the data which is generated from $D-$ dimensional random variable x, we can observe that for the random vector x, the probability function $p(x)$ can be factored into a product of one-dimensional distribution for any order:

$$p(x) = \prod_{d=1}^{D} p(x_d|x_{<d})$$

.

If we define an autoencoder whose input is the data $x = (x_1, ...x_D)$ and output is $\widehat{\mathbf{x}}_d = p(x_d|\mathbf{x_{<d}})$ for the component at d-th entry. $\widehat{\mathbf{x}}_d$ only depends on its previous units. This is the autoregressive property.

However, the fully connected network of autoencoder can not ensure the conditional dependency of different units. So this model implements masks to ensure the conditional dependency.

4.2 Architecture of MADE

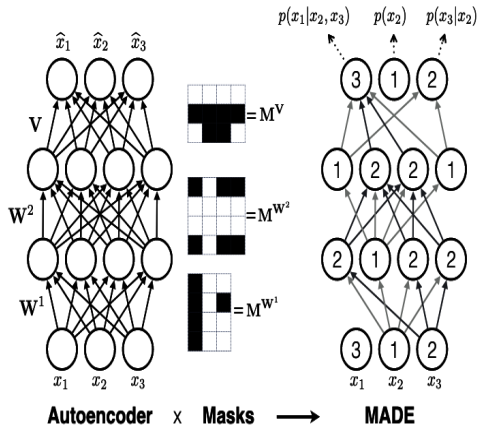
Given the input of the data $x = (x_1, ...x_D)$. With the autoencoder model and masks to ouput is a vector $\widehat{\mathbf{x}}$ with each component $x_d = p(x_d|\mathbf{x_{<d}})$ at d-th entry.

We use the negative log likelihood as our loss function:

$$-log\, p(\mathbf{x}) = \sum_{\mathbf{d=1}}^{\mathbf{D}} -\mathbf{log}\, \mathbf{p}(\mathbf{x_d}|\mathbf{x_{<d}}) = \sum_{\mathbf{d=1}}^{\mathbf{D}} -\mathbf{x_d log}\, \mathbf{p}(\mathbf{x_d} = \mathbf{1}|\mathbf{x_{<d}}) - (\mathbf{1} - \mathbf{x_d})\mathbf{log}\, \mathbf{p}(\mathbf{x_d} = \mathbf{0}|\mathbf{x_{<d}})$$

The loss function is in fact the cross entropy loss.

With masks, this model can be implemented as the figure below:



What is important about the MADE architecture is the training process with masks. After each batch, this model choose a different ordering of units of $x = [x_1, ...x_D]$ by the implementation of masks.

4.3 Discussion

With the implementation of masks and changing orders of units of x during the training, this model gives a good performance for high dimensional data.

Compared with the previous AutoRegressive Networks, Deep Autoregressive Network(DARN)[2], MADE has the similar training costs as them but evaluates the probability with the efficiency of autoencoder.

Compared with the Neural Autoregressive Distribution estimator(NADE)[7], the MADE only requires 1 feed-forward passes through the network to evaluate $p(x)$ but NADE requires D passes.

5 Normalizing Flow

5.1 Motivation

To represent the distribution of complex data. We want to know what is a "good" representation. Normalizing flow models[4] are derived from the view that "a good representation is one in which the distribution of the data is easy to model". We want to map the complex data to a latent space in which the distribution of the mapping of data can be factorized. Suppose x has dimension D. $z = f(x)$. z also has dimension D. $p(z) = \prod_d p(z_d)$. z_d is a component of the vector z.

5.2 Basic Architecture

Let $x \in R^D$ be a random vector with complex distribution from which we draw the sample of our data with unknown probability density function of p Let z be a random vector whose distribution is known with probability density function $p_z : R^D \to R$. We call z the base distribution. Given T is an invertible, differentiable and composable transformation. And suppose that $x = T(z)$. Then:

$$p_x(x) = p_z(z)|DetJ_T^{-1}(x)| \text{ where } z = T^{-1}(x)$$

And T is composable finitely such that $T = T_K \circ \cdots \circ T_1$. Assume that $z_0 = z$ and $z_K = x$. The forward evaluation is $z_k = T_k(z_{k-1})$ for $k = 1, ..., K$. The inverse evaluation is $z_{k-1} = T_k^{-1}(z_k)$ for $k = K, ..., 1$.

Then $log\,|detJ_T^{-1}(x)| = log\,|\prod_{k=K}^1 detJ_{T_k^{-1}}(z_k) = \sum_{k=K}^1 log\,|detJ_{T_k^{-1}}(z_k)|$.

We can construct the finite compositions of T with the methods in the table below[10]:

Autoregressive flows	Transformer type: – Affine – Combination-based – Integration-based – Spline-based	Conditioner type: – Recurrent – Masked – Coupling layer
Linear flows	Permutations Decomposition-based: – PLU – QR Orthogonal: – Exponential map – Cayley map – Householder	
Residual flows	Contractive residual Based on matrix determinant lemma: – Planar – Sylvester – Radial	

The flow structures I have researched for this semester are Autoregressive flows. One of the advantage of the autoregressive flow structures is that they have the triangular Jacobian . The triangular Jacobian matrix enables the fast computation.

However, some autoregressive flow structures like Real NVP do not have enough information exchange between the different dimensions of the data. To address this problem, KRNET[12][15] is proposed as the generalization of the real NVP which incorporates the triangular structure of the Knothe-Rosenblatt rearrangement into the definition of the transport.

The loss function we want to minimize for the Normalizing flow model is the Negative Log-Likelihood function. Assume T has the parameter ϕ and p_z has the parameter φ. We want to minimize the negative Log Likelihood of x. Given sample $\{x^i\}_{i=1}^N$:

$$\min_{\phi,\varphi} - \sum_1^N (log\, p_z(T^{-1}(x^i)) + log\, |DetJ_{T^{-1}}(x^i)|)$$

With the estimation of Monte Carlo, the KL divergence of between true target distribution of $p_x(x)$ and the flow-based distribution $p_z(T^{-1}(x))|DetJ_{T^{-1}}(x)|$ has the similar formula as the maximal likelihood above.

5.3 Discussion

Normalizing flow models are used widely. They can give a explicit formula of the Log Likelihood and it can generate different kinds of data.

And normalizing flow is very flexible. We can implement different kinds of flow structures to improve the estimation result. And it is very good at the density estimation.

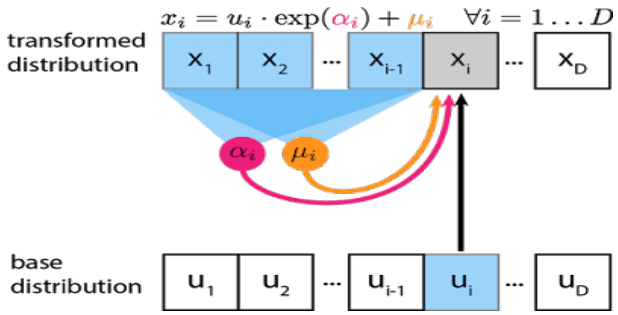
However, sometimes normalizing flow can be computationally demanding, like KRNET. And it performs worse than GAN and VAE with respect to generation of the data.

6 Model Combination

We can combine the four kinds of models above to increase the effectiveness of the density estimation model or generalize the model.

6.1 Masked Autoregressive Flow

As we see in the MADE, the $\widehat{\mathbf{x}}$ is generated from the data x with the autoregressive property. If the target distribution of our data can be transformed from some distribution invertibly and differentiablely with autoregressive property. We can view this kind of Autoregressive Model as the normalizing flow[8]. With the transformation of components from the base distribution to target distribution in this way.



Masked Autoregressive Flow(MAF) uses masks as the MADE to ensure the conditional dependency.

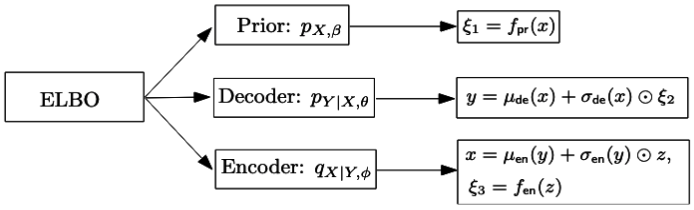
6.2 VAE with Normalizing Flow

A normalizing Flow model, is a good way to learning the distribution of complex data. However, if the dimension of the data with target distribution x is given. The dimension of the base distribution z is limited. Sometimes, we need to be more flexible about the dimension of the latent space. To address this issue, this article[14] incorporate the normalizing flow model KRNET into the Variational Autoencoder. In this way, we can cover a range of different dimensions of latent

space.

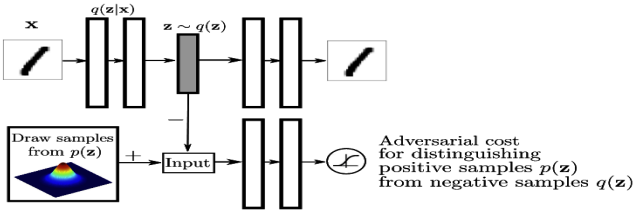
As this paper proposes, we can generalize the prior distribution of the "code" distribution in the structure of autoencoder with the KRNET. Instead of using the standard gaussian distribution, we can use the distribution generated from the KRNET. And we can also generalize the encoder by using the normalizing flow. The flow structures which is used in this article is the KRNET.

In the figure below, the latent random variable which is the code in the autoencoder structure is denoted as$X$, and original random variable which generates samples is denoted as Y. f_{pr} and f_{en} are two flow-based generative model:



VAE model has the problem of the underestimation of the variance. To alleviate this issue, this article wants to maximize the mutual information between the latent random variable and the original random variable and minimize KL divergence between the density model and the original distribution.

6.3 Adversarial Autoencoder(AAE)



The Adversarial Autoencoder(AAE)[5] wants to turn an autoencoder into an generative model. What this model wants to do is make the "code" part of the autoencoder match the distribution we imposed prior distribution.

The encoder of the autoencoder performs as the generator of GAN. And the discriminator of the GAN tries to distinguish the data whose the distribution is imposed from the data which is generated by the encoder.

Suppose, we impose z with the distribution p_z and x has the distribution p_x, encoder $q_{z|x}$: And we want to define the aggregated posterior distribution

$q_z(z) = \int_x q_{z|x}(z|x)p_x(x)dx$. This model wants the GAN model guides q_z. So the model wants the encoder $q_{z|x}$ to fool the discriminator in GAN that samples from aggregated posterior q_z is from the distribution p_z.

The upper bound on the negative likelihood from VAE becomes:

$$E_x[E_{q_\phi(z|x)}[log(p_\theta(x|z)]] - E_x[KL(q_\phi(z|x)||p_z(z))] = \\ Reconstruction - Entropy + CrossEntropy(q_z, p_z)$$

Because of the modification of the loss function, One of the advantage of AAE over VAE is that we don't need to specify the exact functional form of the prior distribution.

References

[1] Diederik P Kingma, Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv* (2013). DOI: https://doi.org/10.48550/arXiv.1312.6114.

[2] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, Daan Wierstra. "Deep AutoRegressive Networks". In: *arXiv* (2013). DOI: https://doi.org/10.48550/arXiv.1310.8499.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. "Generative Adversarial Networks." In: *arXiv* (2014). DOI: https://doi.org/10.48550/arXiv.1406.2661.

[4] Laurent Dinh, David Krueger, Yoshua Bengio. "NICE: Non-linear Independent Components Estimation". In: *arXiv* (2014). DOI: https://doi.org/10.48550/arXiv.1410.8516.

[5] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, Brendan Frey. "Adversarial Autoencoders". In: *arXiv* (2015). DOI: https://doi.org/10.48550/arXiv.1511.05644.

[6] Mathieu Germain, Karol Gregor, Iain Murray, Hugo Larochelle. "MADE: Masked Autoencoder for Distribution Estimation". In: *arXiv* (2015). DOI: https://doi.org/10.48550/arXiv.1502.03509.

[7] Mathieu Germain, Karol Gregor, Iain Murray, Hugo Larochelle. "Neural Autoregressive Distribution Estimation". In: *arXiv* (2016). DOI: https://doi.org/10.48550/arXiv.1605.02226.

[8] George Papamakarios, Theo Pavlakou, Iain Murray. "Masked Autoregressive Flow for Density Estimation". In: *arXiv* (2017). DOI: https://doi.org/10.48550/arXiv.1705.07057.

[9] Martin Arjovsky, Soumith Chintala, Léon Bottou. "Wasserstein GAN". In: *arXiv* (2017). DOI: https://doi.org/10.48550/arXiv.1701.07875.

[10] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, Balaji Lakshminarayanan. "Normalizing Flows for Probabilistic Modeling and Inference". In: *arXiv* (2019). DOI: https://doi.org/10.48550/arXiv.1912.02762.

[11] M. Ehsan Abbasnejad, Javen Shi, Anton van den Hengel, Lingqiao Liu. "A Generative Adversarial Density Estimator". In: *CVPR Paper* (2019).

[12] Keju Tangb, Xiaoliang Wan, Qifeng Liao. " Deep density estimation via invertible block-triangular mapping". In: (2020).

[13] Qiao Liu, Jiaze Xu, Rui Jiang, Wing Hung Wong. "Roundtrip: A Deep Generative Neural Density Estimator". In: *arXiv* (2020). DOI: https://doi.org/10.48550/arXiv.2004.09017.

[14] Xiaoliang Wan, Shuangqing Wei. "VAE-KRnet and its applications to variational Bayes". In: *arXiv* (2020). DOI: https://doi.org/10.48550/arXiv.2006.16431.

[15] Xiaoliang Wan, Kejun Tang. "Augmented KRnet for density estimation and approximation". In: *arXiv* (2021). DOI: https://doi.org/10.48550/arXiv.2105.12866.