On Unifying the Autoregressive Model and the Vector Transformation Method

Aodong Li

al5350@nyu.edu

1 Introduction

The density estimation serves a bunch of applications related to machine learning technique. However, due to the mixed correlations between variables and the curse of dimensionality, density estimation is much more complex than common downstream tasks, e.g., classcification task, which only requires the model find striking features. Although the problem is tough, recent development in deep learning aids the solution.

To model the data distribution, two mainstream methods are explored in this field–autoregressive models and vector transformation methods. The autoregressive models[1][2][3][4][5] utilize the probability chain rule $p(\mathbf{x}) = \prod p(x_i|x_{<i})$ and model in a sequential way the conditional factors $p(x_i|x_{<i})$. On the other hand, the vector transformation methods[6][7][8][9] exploits the change of variable formula and a tractable determinant of Jacobian matrix to directly transform a random vector into another random vector of interest.

Most of the time, the two methods are discussed separately. But in this project, we try to unify them by the observation that both these methods involve a lower triangular Jacobian matrix of the transformation. The main content of this project may involve,

1. Show that both the autoregressive model and vector transformation have the lower triangular Jacobian matrix. Multiple vector transformations can be seen as a stacked autoregressive model.

2. Try to do inference on autoregressive model like Dinh et al.[7] since once we have a well-trained autoregressive model, namely, all the parameters

are obtained, the random noise $\boldsymbol{\epsilon} = g(\mathbf{x})$ can be transformed through the model in theory (the Jacobian matrix is still lower triangular).

3. Try to do a block vector transformation model combining vector transformation and autoregressive property,

$$(\epsilon_1, \epsilon_2, ..., \epsilon_d) \mapsto (x_{(1)}, x_{(2)}, ..., x_{(d)}),$$
$$(x_{(1)}, x_{(2)}, ..., x_{(d)}) \mapsto (x_{(d+1)}, x_{(d+2)}, ..., x_{(2d)}),$$
$$.....$$

   Note that the order of the subscript of x is not important. So it can be utilized to model local or global distribution.

4. The Variational Lossy Autoencoder[10] encodes less bits by $\mathrm{KL}(q_\phi(z|x)||p(z))$ than a normal VAE. So an intuitive way to encode a lossy latent representation is rendering small weight to the reconstruction error term in ELBO objective function, which encourages a larger negative $\mathrm{KL}(q_\phi(z|x)||p(z))$, i.e., less bits of $\mathrm{KL}(q_\phi(z|x)||p(z))$. But this does not specify what kind of information is stored in the latent variable. Another downstream classification task on the latent variable or reconstructed $\mathbf{x}$ can be added to do a joint training to encourage global representation learning.

5. The autoregressive generation model in the Variational Lossy Autoencoder can be replaced by the block vector transformation model in part 3.

2 Related Work

Germain et al.[4] noticed that a model satisfying autoregressive property has a transformation matrix of lower triangular structure if we view $x_i = f_\theta(x_{<i})$.

Kingma et al. [9] noticed the lower triangular Jacobian of the autoregressive model and they used the autoregressive model to perform vector transformation to improve the work of Rezende et al. [8]. The highlighted Jacobian matrix $\partial\mathbf{y}/\partial\boldsymbol{\epsilon}$ is already a tractable lower triangular form.

Recent work [11] utilized the change of variable formula flexibly and extended this formula to model the distribution with both vector transforma-

tion and an autoregressive model,

$$-\log p(x_1, ..., x_d) = -\sum_{t=1}^{T} \log \left| \det \frac{dq^{(t)}}{dq^{(t-1)}} \right| - \sum_{i=1}^{d} \log p(q_i(x)|h_i)$$

where $q_i(\cdot)$ is a series of transformation modeled by vector transformation whereas $p(q_i(x)|h_i)$ is a conditional factor modeled by autoregressive model.

3 Unified Framework

3.1 Vector transformation = autoregressive model

1. Vector Transformation

   (a) Use the change-of-variable formulation. This can be done by transforming a vector of simple distribution into a complex distribution variables.

   $$p_X(x) = p_\epsilon(F^{-1}(x)) \left| \det \frac{\partial F^{-1}(x)}{\partial x} \right| = p_\epsilon(F^{-1}(x)) \left| \det \frac{\partial F(\epsilon)}{\partial \epsilon} \right|^{-1}$$

   where $x = F(\epsilon)$ and F is bijective.

   (b) If the Jacobian matrix is lower triangle, its determinant is easy to compute.

   (c) The requirement can be satisfied by the following transformation,

   $$\begin{aligned} x_1 &= f_1(\epsilon_1) \\ x_2 &= f_2(\epsilon_1, \epsilon_2) \\ &... \\ x_n &= f_n(\epsilon_1, ..., \epsilon_n). \end{aligned}$$

   It can be shown that the Jacobian matrix of the transformation $(x_1, ..., x_n) = F(\epsilon_1, ..., \epsilon_n)$ is

   $$\begin{bmatrix} \frac{df_1(\epsilon_1)}{d\epsilon_1} & & \\ \frac{\partial f_2(\epsilon_1,\epsilon_2)}{\partial \epsilon_1} & \frac{\partial f_2(\epsilon_1,\epsilon_2)}{\partial \epsilon_2} & \mathbf{0} \\ \cdots & & \\ \frac{\partial f_n(\epsilon_1,...,\epsilon_n)}{\partial \epsilon_1} & \cdots & \frac{\partial f_n(\epsilon_1,...,\epsilon_n)}{\partial \epsilon_n} \end{bmatrix}$$

which is lower triangular matrix and its determinant is the multiplication of the diagonal entries $\frac{df_1(\epsilon_1)}{d\epsilon_1}\frac{\partial f_2(\epsilon_1,\epsilon_2)}{\partial \epsilon_2}\frac{\partial f_n(\epsilon_1,...,\epsilon_n)}{\partial \epsilon_n}$.

(d) This technique has been exploited to model various distributions.

2. Autoregressive Model

(a) The autoregressive model directly models the data distribution by the chain rule of probability

$$p(x) = p(x_{(1)}) \prod_{i=2}^{n} p(x_{(i)}|x_{<(i)}).$$

(b) If we set a start random variable ϵ to begin with, then we can define a recursive method to generate new variables one by one.

$$\begin{aligned} x_{(1)} &= f_\theta(\epsilon_1) \\ x_{(2)} &= f_\theta(\epsilon_2, x_{(1)}) \\ &... \\ x_{(n)} &= f_\theta(\epsilon_n, x_{(1)}, ..., x_{(n-1)}) \end{aligned}$$

The corresponding transformation and Jacobian matrix are

$$(x_1, ..., x_n) = F(\epsilon_1, \epsilon_2, ..., \epsilon_n)$$

and

$$\begin{bmatrix} \frac{\partial f_\theta(\epsilon_1)}{\partial \epsilon_1} & & \\ \frac{\partial f_\theta(\epsilon_2,x_1)}{\partial \epsilon_1} & \frac{\partial f_\theta(\epsilon_2,x_1)}{\partial \epsilon_2} & \mathbf{0} \\ \cdots & & \\ \frac{\partial f_\theta(\epsilon_n,x_1,...,x_{n-1})}{\partial \epsilon_1} & \cdots & \frac{\partial f_\theta(\epsilon_n,x_1,...,x_{n-1})}{\partial x_{n-1}} \end{bmatrix}$$

(c) Interestingly, if we set the transformation as a neural network such as RNN, due to the definition, the same computation path will be triggered any time a new variable is inserted. Under this reasoning, all the entries on the diagonal are the same derivatives, thus the determinant is greatly simplified as long as we can compute one of the derivatives.

3.2 Inference on autoregressive model

In this project, we will focus on the Gaussian autoregressive models as Kingma et al. [9],

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$
$$\boldsymbol{\epsilon} = \frac{\mathbf{y} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are functions of $\mathbf{y}$, and $\partial\mu_i/\partial y_j = 0, \partial\sigma_i/\partial y_j = 0$ for $j \geq i$.

$$\boldsymbol{\mu} = f(\mathbf{y})$$
$$\boldsymbol{\sigma} = g(\mathbf{y})$$

and

Such a transformation immediately results in a lower triangular Jacobian matrix. In theory, once we know $\mathbf{y}$, we can infer the corresponding point $\boldsymbol{\epsilon}$ in latent space.

3.3 Examples

RealNVP[7] uses multiple coupling layers and each layer can be seen as an autoregressive model. Thus it performs a stacked autoregressive model, i.e., forming an autoregressive flow.

At the first glance, Normalizing flow[8] does not use a lower triangular Jacobian matrix. But due to the property of determinant $\det(AB) = \det(A)\det(B)$, as long as we use LU decomposition on the Jacobian matrix, we have a two step transformation, which is also a stacked transformation. Note that the autoregressive model does not care the order of inputs, so the upper triangular matrix can be re-arranged into lower triangular matrix if we re-order the input again.

Transformation Autoregressive Networks[11] are also a stacked autoregressive model. **(I still need to figure out how it transforms. Haven't test it strictly)**

For discrete distribution models, through Gumbel-Softmax, extra noise is added into the model, which also forms the foundation of tractable-Jacobian vector transformation.**(Haven't tested it strictly.)**

4 Experiment

Dataset: MNIST dataset.

Autoregressive model inference task: First train a simple autoregressive model, then perform inference on autoregressive model like realNVP.

Local distribution model task: design a model by block vector transformation and use log-likelihood objective. If successful, replace the autoregressive generative model in Variational Lossy Autoencoder (VLAE) with this model because such a limited generative model in VLAE does not have to be a strong autoregressive model.

Lossy representation task: Learn a lossy representation by relaxing the reconstruction error (set small weight) in the ELBO objective. Joint training with classification task to encourage capture a global representation. The generative model need not be strong.

Ensemble model: Ensemble of different input orders of RealNVP.

References

[1] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[2] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[3] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

[4] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.

[5] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[6] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[8] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[9] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[10] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

[11] Junier B Oliva, Avinava Dubey, Barnabás Póczos, Jeff Schneider, and Eric P Xing. Transformation autoregressive networks. *arXiv preprint arXiv:1801.09819*, 2018.

5 Personal notes

1. Vector Transformation

   (a) Use the change-of-variable formulation. This can be done by transforming a vector of simple distribution into a complex distribution variables.

   $$f_X(x) = f_\epsilon(F^{-1}(x)) \left| \det \frac{\partial F^{-1}(x)}{\partial x} \right| = f_\epsilon(F^{-1}(x)) \left| \det \frac{\partial F(\epsilon)}{\partial \epsilon} \right|^{-1}$$

   where $x = F(\epsilon)$ and F is bijective.

   (b) If the Jacobian matrix is lower triangle, its determinant is easy to compute.

   (c) The requirement can be satisfied by the following transformation,

   $$\begin{aligned} x_1 &= f_1(\epsilon_1) \\ x_2 &= f_2(\epsilon_1, \epsilon_2) \\ &... \\ x_n &= f_n(\epsilon_1, ..., \epsilon_n). \end{aligned}$$

   It can be shown that the Jacobian matrix of the transformation $(x_1, ..., x_n) = F(\epsilon_1, ..., \epsilon_n)$ is

   $$\begin{bmatrix} \frac{df_1(\epsilon_1)}{d\epsilon_1} & & \\ \frac{\partial f_2(\epsilon_1,\epsilon_2)}{\partial \epsilon_1} & \frac{\partial f_2(\epsilon_1,\epsilon_2)}{\partial \epsilon_2} & \mathbf{0} \\ \cdots & & \\ \frac{\partial f_n(\epsilon_1,...,\epsilon_n)}{\partial \epsilon_1} & \cdots & \frac{\partial f_n(\epsilon_1,...,\epsilon_n)}{\partial \epsilon_n} \end{bmatrix}$$

   which is lower triangular matrix and its determinant is the multiplication of the diagonal entries $\frac{df_1(\epsilon_1)}{d\epsilon_1} \frac{\partial f_2(\epsilon_1,\epsilon_2)}{\partial \epsilon_2} \frac{\partial f_n(\epsilon_1,...,\epsilon_n)}{\partial \epsilon_n}$.

   (d) This technique has been exploited to model various distributions.

2. Autoregressive Model

   (a) The autoregressive model directly models the data distribution by the chain rule of probability

   $$p(x) = p(x_1) \prod_{i=2}^{n} p(x_i | x_{<i}).$$

(b) If we set a start random variable ϵ to begin with, then we can define a recursive method to generate new variables one by one.

$$\begin{aligned} x_1 &= f_1(\epsilon) \\ x_2 &= f_2(\epsilon, x_1) \\ &... \\ x_n &= f_n(\epsilon, x_1, ..., x_{n-1}). \end{aligned}$$

The corresponding transformation and Jacobian matrix are

$$(x_1, ..., x_n) = F(\epsilon, x_1, ..., x_{n-1})$$

and

$$\begin{bmatrix} \frac{df_1(\epsilon)}{d\epsilon} & & \\ \frac{\partial f_2(\epsilon, x_1)}{\partial \epsilon} & \frac{\partial f_2(\epsilon, x_1)}{\partial x_1} & \mathbf{0} \\ \cdots & & \\ \frac{\partial f_n(\epsilon, x_1, ..., x_{n-1})}{\partial \epsilon} & \cdots & \frac{\partial f_n(\epsilon, x_1, ..., x_{n-1})}{\partial x_{n-1}} \end{bmatrix}$$

(c) Interestingly, if we set the transformation as a neural network such as RNN, due to the definition, the same computation path will be triggered any time a new variable is inserted. Under this reasoning, all the entries on the diagonal are the same derivatives, thus the determinant is greatly simplified as long as we can compute one of the derivatives.

(d) IMPORTANT QUESTIONS: At least for RNN units like LSTM and GRU, the next generation is a function of two new inputs instead of one. So the dimension of the matrix are not the same:

$$\begin{aligned} x_1 &= f_\theta(\epsilon_1, \epsilon_2), h_1 = g_\theta(\epsilon_1, \epsilon_2) \\ x_2 &= f_\theta(x_1, h_1), h_2 = g_\theta(x_1, h_1) \\ &\cdots \\ x_n &= f_\theta(x_{n-1}, h_{n-1}), h_n = g_\theta(x_{n-1}, h_{n-1}) \end{aligned}$$

How to represent the corresponding Jacobian matrix?

3. These two models are essentially the same under the technique of change-of-variable formulation, but authors use these two models separately most of the time although sometimes they implicitly use the change-of-variable formulation on autoregressive model.

[Improved Variational Inference with Inverse Autoregressive Flow, VARIATIONAL LOSSY AUTOENCODER]

(a) Come up with a framework based on change-of-variable technique to synthesize vector transformation methods and autoregressive models.

(b) Try to put the following models under the framework.

[Variational Inference with Normalizing Flows][NICE- NON-LINEAR INDEPENDENT COMPONENTS ESTIMATION][transformation autoregressive network][DENSITY ESTIMATION USING REAL NVP][Improved Variational Inference with Inverse Autoregressive Flow, VARIATIONAL LOSSY AUTOENCODER]

(c) The trade-off between sequential generation and parallel generation.

Note that if the dimension of original vector is the same as the dimension of generated vector without a recursive way, this be fully parallel. Will this speed up the generation if we add more and more independent variables? In other words, can we generate more than one variable at a time to relax the strict sequential generation? Experiment?

Interesting question: what form of Jacobian matrix does variational lossy auto-encoder correspond to? Long distance dependency and short distance dependency correspond to different 0-entry positions in the Jacobian matrix.

(d) Generalize the approach to directed graphical model with a proper structure.

(e) Pay attention to the skip connection and think about can it mimic the variational lossy autoencoder. At the same time, variational lossy autoencoder is not well defined, it is only intuitive.

(f) **The more input random variables, the more parallization?** If there is only one noise random variable at the beginning, the generation is sequential since the model has to generate a valid density or discrete distribution on the fly for the next generation (This situation is well appropriate in the discrete distribution like language model). If the model's output distribution is continuous, like Gaussian (linear Gaussian model), then the model's workload

is alleviated to predict the parameters. In all, without the help of the noise, the network has to infer a valid distribution from the observed variables. If an assumption is exerted, like Gaussian, then the model's work is much less.

4. Properties

   (a) NICE: Non-linear Independent Components Estimation

   - Based on the idea: A good representation is one in which the distribution of the data is easy to model, the authors transform independent latent variables into data distribution.
   - Consider the change of variable $h = f(x)$, the authors assume that f is invertible and the dimension of h is the same as the dimension of x.
   - Useful components include diagonal, lower triangular, or upper triangular Jacobian matrix.
   - Design coupling layers which have the form

$$y_I = x_I$$
$$y_{II} = x_{II} + m(x_1)$$

     where I and II are different block partitions of the vector.
   - The diagonal of the Jacobian of the transformation is unit, so it is called **volume preserving**. To counteract this, add a final diagonal layer S.
   - The coupling function m has no constraints. It can be neural networks, such as auto-regressive models.
   - The prior distribution can be logistic distribution or Gaussian distribution. The authors use **logistic distribution** in order to provide a better behaved gradients.

   (b) Variational Inference with Normalizing Flows

   - Use vector transformation technique to enhance the approximate posterior distribution.
   - By the chain rule[1] and the property of Jacobians of invertible

[1] https://en.wikipedia.org/wiki/Inverse_functions_and_differentiation

functions[2].

$$q_Z(z) = q_\epsilon(\epsilon) \left| \det \frac{\partial f^{-1}(z)}{\partial z} \right| = q_\epsilon(\epsilon) \left| \det \frac{\partial f(\epsilon)}{\partial \epsilon} \right|^{-1}$$

- The density $q_k(z)$ is obtained by successively transforming z_0 through a chain of k transformations.

$$z_k = f_k \circ ... \circ f_2 \circ f_1(z_0)$$

$$\ln q_k(z_k) = \ln q_0(z_0) - \sum_{i=1}^{k} \ln \left| \det \frac{\partial f_i(z_{i-1})}{\partial z_{i-1}} \right|$$

- Like the reparametriation trick, the expectation $\mathbb{E}_{q_k(z)}[h(z)]$ can be written as the expectation with respect to z_0.

$$\mathbb{E}_{q_k(z)}[h(z)] = \mathbb{E}_{q_0(z_0)}[h(f_k \circ ... \circ f_2 \circ f_1(z_0))]$$

- Instead of using lower-triangle Jacobian transformation, the authors used the matrix determinant lemma[3], which results in a family of transformations of a particular form.

(c) MADE: Masked Autoencoder for Distribution Estimation

- By carefully designing the inner connections of an autoencoder, the authors represent the output as $\hat{x}_d = p(x_d = 1|x_{<d})$, which is the autoregressive property. This enables the utilization of probability product rules.
- The autoregressive property requires that the mask matrix satisfy some properties. Since $\hat{x}_d$ only depends on $x_{<d}$, there must be **no computational path** between $\hat{x}_d$ and inputs $x_d, ..., x_D$. In other words, at least one connection represented by the entry of the weight matrix must be zero.
- The multiplication of mask matrices represents the number of path from the inputs to the outputs. To enable the autoregressive property, the multiplication of mask matrices shall be **strict lower triangular**.
- Note that the order of conditionals is not important. So an ensemble of different ordering models might be powerful.

[2]https://en.wikipedia.org/wiki/Inverse_function_theorem
[3]https://en.wikipedia.org/wiki/Matrix_determinant_lemma

(d) Improved Variational Inference with Inverse Autoregressive Flow

- It is a member of a family of normalizing flows. It scales easily to high-dimensional latent space whereas the former models doesn't scale well.
- It aims to approximate the posterior distribution better by introducing a chain of Gaussian autoregressive models where each step is a full realization of autoregressive model.
- Introducing the noise ϵ at the first step enables an easily computational determinant of Jacobian matrix.

$$z_0 = \mu_0 + \sigma_0 \odot \epsilon$$
$$z_n = \mu_n + \sigma_n \odot z_{n-1}$$

where each autoregressive model's output can be computed immediately in a closed form once we know the structure of the model.

(e) Variational Lossy Autoencoder

- Starting from an information-theoretic view, it illustrates that when the decoder is powerful enough, the latent variable falls shorts of effects due to all the data distribution p_{data} is modeled by the decoder.
- Intuitively, the aoturegressive prior is more powerful than the inverse autoregressive posterior because the effect path is longer, thus might be beneficial to the density estimation.
- The author also conjectures that by putting local information into the decoder, the global information can be stored into the latent variables, thus controlling the fine-grained storable information.
- **However, the problem is not well-defined, it is only intuitive.**

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_{z\sim q(z|x)}[\log p(x,z) - \log q(z|x)] \\
&= \mathbb{E}_{z\sim q(z|x)}[\log p(x|globalfeature)p(globalfeature) - \log q(z|x)] \\
&= \mathbb{E}_{z\sim q(z|x)}[\sum \log p(x_i|window(x_i), globalfeature) \\
&\qquad + \log p(globalfeature) - \log q(z|x)]
\end{aligned}$$

Specifically, if the window is chosen to be $window(x_i) = \{x_{i-1}, x_{i-2}\}$ it can be seen that $global feature = z$ has to contain the information of $\{x_{<i-2}\}$, so does the latent representation. Spreading the product $p(x|z) = \prod p(x_i|windowx_i, z)$ shows the latent representation has to complement the information other than $windown(x_i)$. Such situation has to comply to all dimensions of x. In particular, for the last dimension x_d, z is required to contain $x_{-window(x_d)}$, which is almost all the other dimensions if the length of the window is small. Thus, latent variable involves part of the information of the data, **but may not desired to be the global information as suggested in the paper**.

Note the sum operation on generative models. This is equivalent to multiple generative models.

- What if the decoder is not autoregressive model? The latent variable has to encode all the information about the observed data.

(f) Transformation Autoregressive Networks

- The computation might be problematic.
- It widely uses RNN to model both autoregressive model and vector transformation method because RNN has good properties for tractable Jacobian matrix. But it has to be performed in an sequential manner, which restricts the performance a lot.
- It utilizes the change of variable formula to perform the composing transformation.

$$
\begin{aligned}
-\log p(x_1, ..., x_d) = &-\sum_{t=1}^{T} \log \left| \det \frac{dq^{(t)}}{dq^{(t-1)}} \right| \\
&-\sum_{i=1}^{d} \log p(q_i(x)|h_i)
\end{aligned}
$$