Generative Auto Encoder

Yongdai Kim, Dongha Kim and Jaesung Hwang
Speaker : Dongha Kim

Department of Statistics, Seoul National University, South Korea

July 5, 2018

1. Introduction
2. Proposed method
3. Experiments

Introduction

- Estimation of deep generative models have received much attentions.
- There are two popular approaches, one is called variational auto encoder (VAE, Kingma and Welling (2013)) and the other is called generative adversarial networks (GAN, Goodfellow et al. (2014)).
- Based on the auto encoder, we propose a simple and novel approach to generative model.

Basic structure of deep generative model

- In many studies of deep generative model, the marginal distribution of observation x is assumed to be a mixture of latent variables z given as:

$$P(x;\theta) = \int_z P(x|z;\theta)P(z)dz$$

where $P(\cdot|z;\theta)$ is a decoder parametrized by θ.

- They model the marginal distribution of latent variable z, $P(z)$, to normal or uniform distribution.
- In this assumption, it requires many calculations to transform latent variable into real data, thus decoder and encoder have to be deep structures.

Our contributions

- We propose a simple but efficient algorithm to estimate the generate model for given data based on the auto-encoder, which is called generative auto encoder (GAE).
- Especially, we do not design specific form of the marginal distribution of latent variable, for instance $\mathcal{N}(0, I)$, and let the distribution be **determined by complexity of networks and input data.**
- By doing this, we expect that our method achieve similar or superior performance **with more compact structure** than other generative methods.

1. Introduction
2. Proposed method
3. Experiments

Model description

- We model the marginal distribution of latent variable z to **mixture of train data** as follows:

$$\begin{aligned} P(z;\phi) &= \int_y P(z|y;\phi)d\hat{F}(y) \\ &= \frac{1}{n}\sum_{j=1}^{n} P(z|x_j;\phi) \end{aligned}$$

  where $P(\cdot|y;\phi)$ is a encoder parametrized by ϕ and $\{x_j\}_{j=1}^n$ is train data.

- Here, $P(\cdot|y;\phi)$ is designed to a multivariate normal distribution, that is, if $z \sim P(\cdot|y;\phi)$ then

$$z = \mu(y;\phi) + \sigma(y;\phi) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

  where $\mu(y;\phi)$ and $\sigma(y;\phi)$ are deep architectures based on NN.

Model description

- Then the marginal distribution of an observation x can be rewritten to the following:

$$P(x;\theta,\phi) = \frac{1}{n}\sum_{j=1}^{n}\int_{z} P(x|z;\theta)P(z|x_j;\phi)dz$$

- We estimate parameters θ and ϕ by maximizing the log likelihood function:

$$\sum_{i=1}^{n}\log\left[\frac{1}{n}\sum_{j=1}^{n}\int_{z} P(x_i|z;\theta)P(z|x_j;\phi)dz\right]$$

Regularization

- To avoid over-fitting, we give some regularization terms for $\mu(\cdot;\phi)$ and $\sigma(\cdot;\phi)$ as follows:

$$\begin{aligned} R(x,\phi,\lambda_1,\lambda_2) &= \lambda_1 \sum_{j=1}^{J} \{\mu(x;\phi)_j^2\} \\ &+ \lambda_2 \sum_{j=1}^{J} \{1 + \log \sigma(x;\phi)_j^2 - \sigma(x;\phi)_j^2\} \end{aligned}$$

  where $\lambda_1, \lambda_2 > 0$ are hyperparameters and J is dimension of the latent space.

- The above regularization term is motivated by the regularization term of VAE.
- Then the final objective function is given as:

$$\sum_{i=1}^{n} \log \left[\frac{1}{n} \sum_{j=1}^{n} \int_z P(x_i|z;\theta) P(z|x_j;\phi) dz \right] + \sum_{i=1}^{n} R(x_i,\phi,\lambda_1,\lambda_2)$$

Generation of samples

- Our proposed method has slightly different procedure to generate samples because we also model $P(z)$ to a mixture of train data.
- The procedure to generate samples is as follows:
  1. Sample y from $\hat{P}$ where $\hat{P}$ is empirical distribution.
  2. Given y, sample z from $P(\cdot|y;\phi)$.
  3. Given z, sample x from $P(\cdot|z;\theta)$, which a generated sample using our method.

Estimation of parameters

- Note that the we can rewrite the log likelihood function as follows:

$$\sum_{i=1}^{n} \log \left[\int_y \int_z P(x_i | z; \theta) dF(z | x_j; \phi) d\hat{F}(y) \right]$$

- It is infeasible to calculate $P(x; \theta, \phi)$, while $P(x, z, y; \theta, \phi)$ is easy to calculate which is given as

$$P(x, z, y; \theta, \phi) = \hat{P}(y) \cdot P(z | y; \phi) \cdot P(x | z; \theta).$$

- So we treat y as well as z as latent variables and optimize the log likelihood using EM algorithm.

Experiments

- We conduct 3 numerical experiments comparing our method with other methods on multiple benchmark datasets.

1. First, we generate samples to confirm whether our method generate visually realistic and diverse images.
2. Secondly we visualize the marginal distribution of latent variable z. We expect that the more simple the architectures are the more complex the marginal distribution of z is.
3. Lastly we conduct quantitative analysis to measure the performance of our method. Two measures are used, KDE and approximated log likelihood.

Generated images

MNIST dataset

Figure : **(Left)** Generated samples using our method **(Right)** Generated samples using VAE. All samples are generated randomly. It seems that our method consistently generates visually realistic images.

Generated images

Toronto Face Dataset (TFD)

- We forgot to save the best model...

Visualization of latent space

MNIST dataset

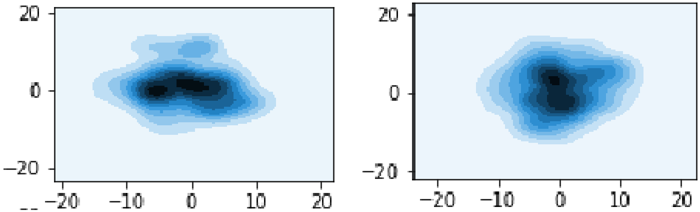

Figure : We sample 1000 samples of latent variable and conduct kernel density estimation using these samples. We use 2-dimensional latent space. **(Left)** Estimated kernel density with 1-layered dec. and enc. **(Right)** Estimated kernel density with 2-layered dec. and enc.

Visualization of latent space

MNIST dataset

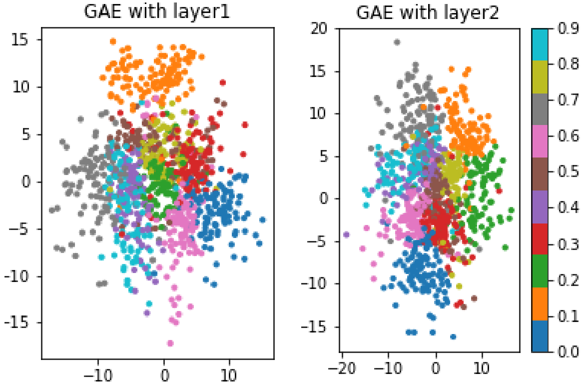

Figure : Using test dataset, we sample z from $P(\cdot|x;\phi)$ and plot these zs. zs are colored according to their true class label. **(Left)** Scatter plot with 1-layered dec. and enc. **(Right)** Scatter plot with 2-layered dec. and enc.

Quantitative analysis

Kernel density estimation (KDE)

- We generate 10,000 samples and conduct kernel density estimation using these samples.
- Then we calculate test log likelihood of test data using the estimated kernel density.

Method	MNIST	TFD
VAE(Kingma and Welling, 2013)	296.77	2572.59
GAN(Goodfellow et al., 2014)	300.331	2057
GMMN+AE(Li et al., 2015)	282	2294
AAE(Makhzani et al., 2015)	340	2252
GAE(1 layered)	**456.71**	**2815.76**
GAE(2 layered)	**460.73**	**2796.91**

Table : Test performances on MNIST and TFD datasets.

Quantitative analysis

Approximated log likelihood

- Approximate test log likelihood by sampling latent variable z as follows:

$$\log P(x) \approx \log \left[\frac{1}{S} \sum_{s=1}^{S} P(x|z_s;\theta) \right], \quad z_s \sim P(z)$$

Method	biMNIST
VAE(1 layered)(Kingma and Welling, 2013)	-107.18
VAE(2 layered)	-96.94
VAE(3 layered)	-97.62
VAE(4 layered)	-102.97
GAE(1 layered)	**-97.66**
GAE(2 layered)	**-96.91**
GAE(3 layered)	**-95.76**
GAE(4 layered)	**-94.66**

Table : Test performances on biMNIST dataset.

References

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Li, Y., Swersky, K., and Zemel, R. (2015). Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.