Research Paper Presentation

Lanka Prasannna

CS20BTECH11029

Topic

GAN-based Gaussian Mixture Model Responsibility Learning

Authors

- Wanming Huang
- Richard Yi Da Xu
- Shuai Jiang
- Xuan Liang
- Ian Oppermann

Year of Publication

2021.

Abstract

- Mixture model (MM) -dataset containing K different modes. When each of the modes is associated with a Gaussian distribution, we refer to it as Gaussian MM or GMM.
- Modern large datasets- GMM unable to apply to compute conditional likelihood and responsibility probability.
- We utilize the generative adversarial network (GAN) and we compute them at the data's latent space z instead of x.
- However, this process of $z \rightarrow x$ is irreversible under GAN which renders $\Pr(k|x,\theta)$ infeasible.
- Our paper proposed a novel method to solve it by using posterior consistency module (PCM).
- PCM acts like a GAN, except its generator does not output the data, but instead it outputs a distribution to approximate $\Pr(k|x,\theta)$.

Introduction

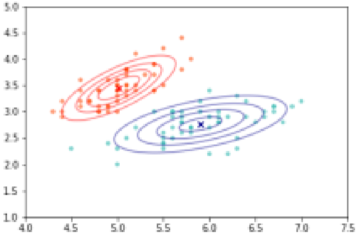
Gaussian mixture model (GMM) assumes that all data points come from a mixture of a finite number of Gaussian distributions. The density function of the GMM is defined below

$$\Pr(X = x) = \sum_{k=1}^{K} \alpha_k N(x; u_k, \Sigma_k) \tag{1}$$

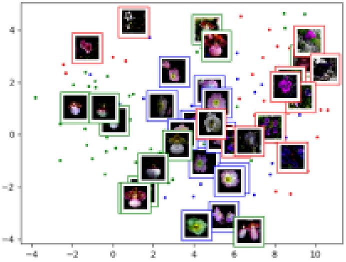
where K is the total number of Gaussians in the mixture, and k^{th} component is characterized by a Gaussian distribution with weight α_k , mean u_k and covariance matrix Σ_k.

Introduction

When Euclidean distance can be meaningfully defined over the data x when it has a low dimensionality, one can compute both $\Pr(x|k,\theta)$ and $\Pr(k|x,\theta)$ directly, see Figure a. However, it will be difficult to do so when dealing with complex and high data dimensional data, such as images, shown in Figure b.



(a) Low dimensional data



(b) High dimensional data

Figure: Illustration of clustering

Introduction

A. Posterior Consistency Module (PCM)

- PCM has allowed us to efficiently changing $\Pr(z)$ in GAN from single to a mixture of Gaussian densities while still be able to compute the responsibility $\Pr(k|x,\theta)$.
- Functions by assuming $\Pr(k|x,\theta)$ to have an equivprobability latent counterpart $\Pr(k|z,\theta)$. It is straightforward to compute $\Pr(k|z,\theta)$ analytically:

$$\Pr(k|z,\theta) = w \equiv (w_1, w_2, \cdots, w_K) = \tag{2}$$

$$\left(\frac{\mathcal{N}(z|\mu_1,\sigma_1)}{\sum_{k=1}^{K}\mathcal{N}(z|\mu_k,\sigma_k)}, \cdots, \frac{\mathcal{N}(z|\mu_K,\sigma_K)}{\sum_{k=1}^{K}\mathcal{N}(z|\mu_k,\sigma_k)}\right) \tag{3}$$

- The same cannot be achieved by substituting z with x. Therefore, we applied a neural network $C_\theta(x)$ with softmax output $\hat{w} = (\hat{w}_1, \cdots, \hat{w}_K)$. $\hat{w}$ is then compared with w for consistency

Introduction

B. Closely related works

1. Probabilistic GAN (PGAN)
2. GM-GAN
3. GMVAE
4. VaDE
5. AAE
6. ClutsterGAN
7. VAE

Architecture

The proposed architecture consists of three networks:

1. PCM as well as the loss function that matches it with $\Pr(k|z, \theta)$.
2. A generator G which produces synthetic samples from GMM random vectors.
3. A discriminator D which encodes samples to feature vectors and discriminates between real and synthetic samples.

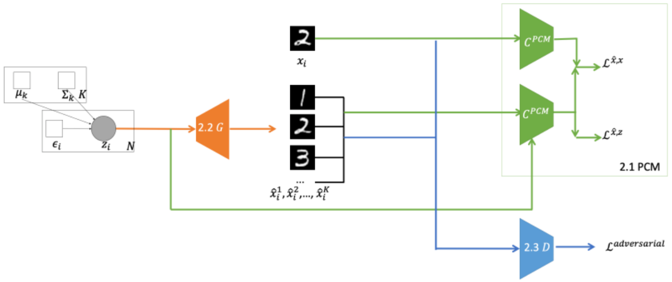

μ_k
Σ_k K
ϵ_i
z_i N
2.2 G
x_i
$\hat{x}_i^1, \hat{x}_i^2, \ldots, \hat{x}_i^K$
C^{PCM}
C^{PCM}
$\mathcal{L}^{\hat{x},x}$
$\mathcal{L}^{\hat{x},z}$
2.1 PCM
2.3 D
$\mathcal{L}^{adversarial}$

Figure: The overall architecture. The feed-forward logic of the classifier C, the generator G and the discriminator D are marked with different colours.

Architecture

A. PCM

The PCM is comprised of a classifier/ generator C_{PCM} which outputs $\Pr(k|z,\theta)$ given x. When a synthetic data $\hat{x}$ is used as its input, the output approximates $\Pr(k|\hat{x},\theta)$ and it needs to be matched against:

1. $\Pr(k|z,\theta)$, this is to ensure that the responsibility probability condition on its latent variable z is similar to the responsibility distribution dependent on x. The corresponding loss is named as:

$$\begin{aligned}\mathcal{L}^{\hat{x},z} =& \mathbb{E}_{\{z_1\sim\mathcal{N}(\mu_1,\Sigma_1),\cdots,z_K\sim\mathcal{N}(\mu_K,\Sigma_K)\}} \\ & \left[\frac{1}{K}\sum_{k=1}^{K} I(p(k|z_k,\theta),p(k|\hat{x}^k,\theta))\right] \\ =& \mathbb{E}_{\{z_1\sim\mathcal{N}(\mu_1,\Sigma_1),\cdots,z_K\sim\mathcal{N}(\mu_K,\Sigma_K)\}} \\ & \left[\frac{1}{K}\sum_{k=1}^{K} I(p(k|z_k,\theta),C^{\theta}_{\mathrm{PCM}}(\hat{x}^k))\right]. \qquad (4)\end{aligned}$$

Architecture

2. $\Pr(k|x,\theta)$, this is to ensure that the distribution generated is similar to those generated by the real data x. The corresponding loss is named as:

$$\begin{aligned}\mathcal{L}^{\hat{x},x} &= \mathbb{E}_{x_i\sim p_{data}}\sum_{k=1}^{K} I\big(p(k|x_i,\theta), p(k|\hat{x},\theta)\big)\\ &= \mathbb{E}_{x_i\sim p_{data}}\left[\sum_{k=1}^{K} I\big(C^{\theta}_{\mathrm{PCM}}(x_i), C^{\theta}_{\mathrm{PCM}}(\hat{x})\big)\right],\end{aligned}$$

where

$$I(X;Y) = \sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p_{(X,Y)}(x,y)\log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)\,p_Y(y)}\right). \quad (5)$$

Architecture

B. Generator

- We sample one random vector from each Gaussian as $z_i = [z_i^1, \cdots, z_i^K]$. These vectors are used to generate K synthetic images $[\hat{x}_i{}^1, \cdots, \hat{x}_i{}^K]$.
- The classification output is used to weigh the adversarial loss calculated from each pair of a generated sample $\hat{x}_i{}^k$ and the real image x_i.
- As our design expects the training to be performed in an “end-to-end” fashion, the reparameterization trick is applied to the z_i sampling process so that the backpropagation can be used to update the parameters μ_k and Σ_k of each Gaussian.
- Instead of sampling $z_i^k \sim \mathcal{N}(\mu_k, \Sigma_k)$, we define $z_i^k = \Sigma_k \epsilon + \mu_k \forall k \in [1, K]$, where ϵ is sampled as $\epsilon \sim \mathcal{N}(0, I)$.

Architecture

C. Discriminator

The design of the discriminator has two options:

1. Whether the image encoding layers are shared with the classifier or not.
2. The image encoding network is followed by a standard linear logistic regression to identify the given image to be real or fake.

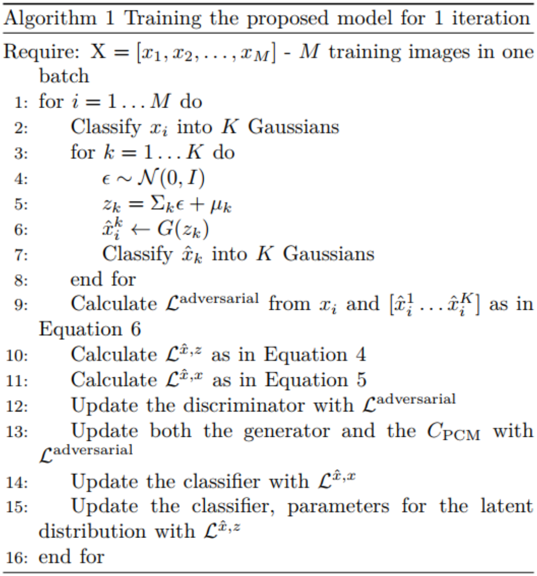
The adversarial loss, which is calculated over K pairs of real and synthetic samples as:

$$\mathcal{L}^{\text{adversarial}} = \tag{6}$$

$$\mathbb{E}_{x_i \sim p_{data}} \Big(\frac{1}{K} \sum_{k=1}^{K} p(k|x_i, \theta) \times (\log D(x_i) + \log(1 - D(\hat{x}^k))) \Big).$$

Architecture

D. Training parameters for the prior distribution



Algorithm 1 Training the proposed model for 1 iteration

```
Require: X = [$x_1, x_2, \ldots, x_M$] - $M$ training images in one batch
 1: for $i = 1 \ldots M$ do
 2:     Classify $x_i$ into $K$ Gaussians
 3:     for $k = 1 \ldots K$ do
 4:         $\epsilon \sim \mathcal{N}(0, I)$
 5:         $z_k = \Sigma_k \epsilon + \mu_k$
 6:         $\hat{x}_i^k \leftarrow G(z_k)$
 7:         Classify $\hat{x}_k$ into $K$ Gaussians
 8:     end for
 9:     Calculate $\mathcal{L}^{\text{adversarial}}$ from $x_i$ and $[\hat{x}_i^1 \ldots \hat{x}_i^K]$ as in Equation 6
10:     Calculate $\mathcal{L}^{\hat{x},z}$ as in Equation 4
11:     Calculate $\mathcal{L}^{\hat{x},x}$ as in Equation 5
12:     Update the discriminator with $\mathcal{L}^{\text{adversarial}}$
13:     Update both the generator and the $C_{\text{PCM}}$ with $\mathcal{L}^{\text{adversarial}}$
14:     Update the classifier with $\mathcal{L}^{\hat{x},x}$
15:     Update the classifier, parameters for the latent distribution with $\mathcal{L}^{\hat{x},z}$
16: end for
```

Experiments

A. Experiment setup
The datasets we use are the MNIST, Fashion-MNIST and Oxford-102 Flower datasets. In particular, we only select a subset of Oxford-102 to perform the training, which is the images that belong to the first 10 classes.

Experiments

B. Linear Interpolation across Gaussian

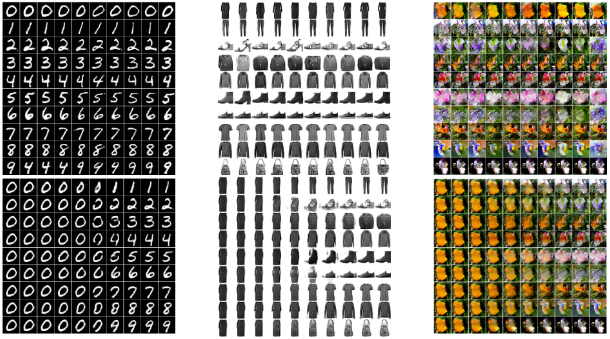


Figure: Samples generated by the proposed models trained on the MNIST (left column), Fashion-MNIST (middle column) and CIFAR-10 (right column) datasets. The top row contains images generated using random vectors sampled from a different Gaussian. The bottom row shows the linear interpolation result from one Gaussian to another

Experiments

.

- When we are performing the linear interpolation as in the right panels, the random vector z for each image generation is calculated as $z = \Sigma_k \epsilon + \mu_k \forall k \in [1, K]$, where ϵ is sampled as $\epsilon \sim \mathcal{N}(0, I)$. ϵ is kept the same for all images for each dataset.
- We can draw two conclusions from the results in above figure. First, a fully trained proposed method can learn to "allocate" each class of image to a Gaussian.
- Second, the trained model can be used to perform smooth linear interpolation between Gaussians and even among more than two Gaussians.

Experiments

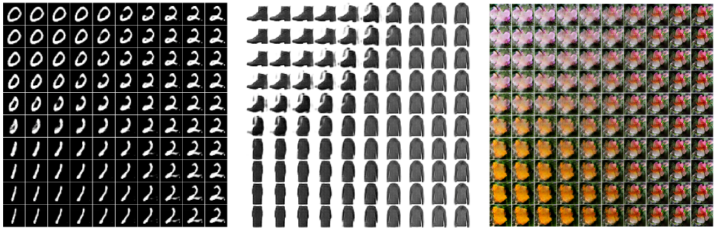


Figure: Linear interpolation over 3 Gaussians on the MNIST, Fashion-MNIST and Oxford-102 dataset

Experiments

C. Image Generation Quality

The generation performance is measured with two commonly used metrics: Inception score and Fréchet Inception Distance (FID) score.

1. Inception score is calculated as $I = exp(E_x D_{KL}(p(y|x)||p(y)))$, and a higher value generally indicates a better performance, where x is a generated image and y is the label predicted by the Inception model
2. FID score: lower value shows a better image quality and diversity. It calculates the difference between real images x and generated images g as $FID_{(x,g)} = ||\mu_x - \mu_g||_2^2 + Tr(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}})$.

Experiments

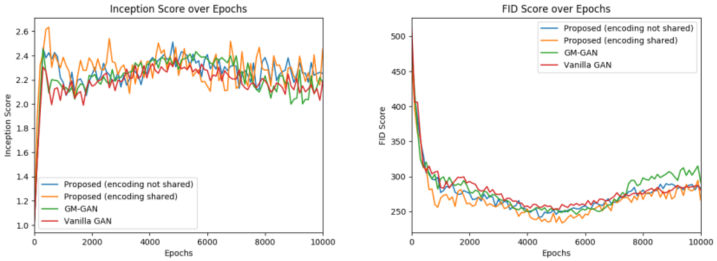


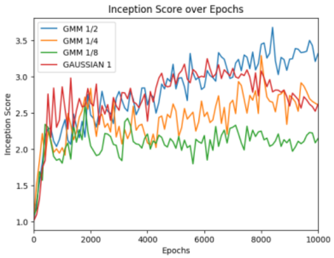
Figure: Inception score and FID score over training epochs on the Oxford dataset

From the above figure, it is clear to see that the proposed method is able to out-perform previous baseline models in terms of image generation quality. The shared feature encoding layers would further improve the performance and reduce the size of the network.

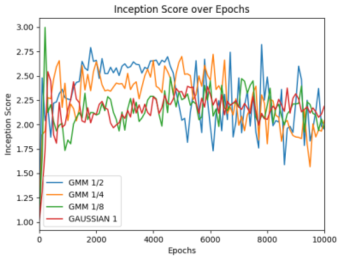
Experiments

D. Network Compression

Applying Gaussian mixture to GANs can significantly reduce the number of parameters while achieving close performance compared with the vanilla GAN.



(a) Inception scores on the CUB dataset.



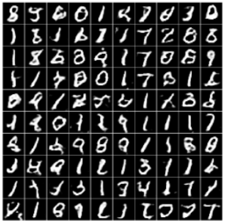
(b) Inception scores on the Oxford-102 dataset.

Figure: Inception scores over epochs for the proposed GMM-based GAN with various size of the network and the vanilla GAN.

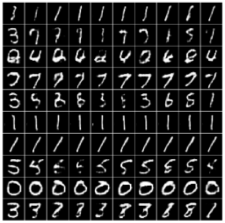
Experiments

E. Performance on Highly Imbalanced Dataset

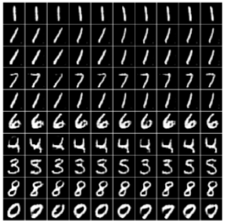
We also demonstrate the performance on the highly imbalanced dataset.



(a) Gaussian as the latent distribution



(b) GMM as the latent distribution



(c) The proposed framework

Figure: Performance on high imbalanced dataset.

Experiments

- The imbalanced dataset we used is a subset of the original MNIST dataset. From digit 0 to digit 9, we randomly choose digit d as the dominant category. 1000 samples are randomly selected from all other categories and the category *d* is kept the same. This becomes the imbalanced training dataset.
- While using both the standard Gaussian and GMM result in poor generation quality and collapsed output, the proposed framework is able to deliver digits with clear shape and categorization.

Conclusion

In addition to GMM modelling, we also demonstrate the multitude of benefits in our approach:

1. We demonstrated through experiments that our proposed method retains GAN's ability in sharper image generation compared with other GMM GAN methods.
2. The proposed approach can significantly save the computation cost as only half number of parameters is required to achieve the same image generation quality compared to classic DCGAN.
3. The proposed method surpasses previous baselines when dealing with highly imbalanced dataset.

THANK YOU