Mixture of GANs for Clustering*

Yang Yu and **Wen-Ji Zhou**
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
{yuy,zhouwj}@lamda.nju.edu.cn

Abstract

For data clustering, Gaussian mixture model (GMM) is a typical method that trains several Gaussian models to capture the data. Each Gaussian model then provides the distribution information of a cluster. For clustering of high dimensional and complex data, more flexible models rather than Gaussian models are desired. Recently, the generative adversarial networks (GANs) have shown effectiveness in capturing complex data distribution. Therefore, GAN mixture model (GANMM) would be a promising alternative of GMM. However, we notice that the non-flexibility of the Gaussian model is essential in the expectation-maximization procedure for training GMM. GAN can have much higher flexibility, which disables the commonly employed expectation-maximization procedure, as that the maximization cannot change the result of the expectation. In this paper, we propose to use the *ϵ-expectation-maximization* procedure for training GANMM. The experiments show that the proposed GANMM can have good performance on complex data as well as simple data.

1 Introduction

Clustering is a fundamental machine learning task that divides a data set into clusters, such that instances are similar in the same cluster but dissimilar in different clusters [Duda *et al.*, 2012]. Clustering is often employed to analyze unlabeled data set, and has been widely applied, such as in recommendation systems [Li and Kim, 2003], computer version [Frigui and Krishnapuram, 1999], speech processing [Chen and Gopalakrishnan, 1998], etc.

Among numerous clustering methods, Gaussian mixture model (GMM) [McLachlan and Basford, 2004; McLachlan and Peel, 2004] is a very classical and elegant one, which has still being consistently improved (e.g., [Li *et al.*, 2015; Raghunathan *et al.*, 2017]). GMM is a linear combination of a set of Gaussian distributions. Its training process finds the best parameters of each Gaussian distribution as well as the combination weights, such that the mixture distribution best fits the data. The training process commonly follows the expectation-maximization (EM) procedure, which solves the parameters efficiently. While GMM has gained successful applications, it has a fundamental assumption is that the data is composed by Gaussian distributions. However, real data sets rarely satisfy this assumption, on which GMM can result in poor clusters. Therefore, mixture of models with more flexibility is appealing for clustering complex data sets.

Recently, the generative adversarial networks (GANs) [Goodfellow *et al.*, 2014; Arjovsky *et al.*, 2017; Chen *et al.*, 2016] have attracted a lot attentions for its outstanding ability of capturing complex data distribution. GAN trains a generative network and a discriminative network simultaneously. The discriminative network is trained to separate the training data set from the generated data, while the generative neural network is trained to generate data indistinguishable from the training data. This adversarial process converges such that the generated data has a distribution close to the training data. GANs have achieved extraordinary performance in generating images, implying their ability to capture high-dimensional complex distributions.

The ability of modeling complex distributions making GAN a potential replacement of the Gaussian distribution for learning mixture models. However, we find that the replacement is not straightforward. The expectation-maximization (EM) procedure, which is commonly employed to train GMM, cannot be used on GAN directly. A key issue is that GAN models can be quite flexible, such that the training by the EM procedure immediately converges.

In this paper, we propose to use the ϵ-expectation-maximization (ϵ-EM) procedure to train the GAN mixture model (GANMM). The ϵ-EM procedure differs from the EM procedure in that the expectation is not fully accurate but with an ϵ error. The ϵ-expectation is implemented by learning a classifier to separate the clusters. Meanwhile, a controlling of the ϵ error can still guarantee the convergence of the procedure, and it can effectively train the GANMM. Experiments on MNIST as well as some UCI data sets show that the proposed GANMM with the ϵ-EM procedure can result in superior performance to GMM and some other state-of-the-art clustering approaches.

*This work is supported by National Key R&D Program of China (2017YFB1002201), Jiangsu SF (BK20170013), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

2 Background

2.1 Gaussian Mixture Model

Gaussian mixture model (GMM) is a popular unsupervised learning technique. A Gaussian mixture model consists of N Gaussian components. The Gaussian components are combined using linear combination to form a distribution $p(x)$,

$$\begin{aligned} p(x) &= \sum\nolimits_{i=1}^{N} \alpha_i \mathcal{N}(x; \mu_i, \Sigma_i) \\ &= \sum\nolimits_{i=1}^{N} \frac{\alpha_i}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)}{2} \right) \end{aligned}$$

where $\boldsymbol{x}$ is a d-dimensional random vector, the i-th Gaussian component has the mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and the covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d\times d}$, and α_i is the weight for the i-th Gaussian component satisfying $\sum_{i=1}^{N} \alpha_i = 1$ and $\forall i, \alpha_i \geq 0$.

Learning GMM requires to find the best parameters $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$ and α_i, such that the data is best fitted. The expectation-maximization procedure is commonly employed to solve this problem.

2.2 Expectation-Maximization Procedure

Expectation-Maximization procedure is a general method for parameter estimation. Generally, consider observable variables X with observed instances $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_M\}$, and hidden variables Z. Our probabilistic model with parameters $\boldsymbol{\theta}$ ($\boldsymbol{\theta} = \{\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$ for GMM) gives the joint probability $p(X, Z|\boldsymbol{\theta})$ to the variables. The parameter estimation of $\boldsymbol{\theta}$ is by maximizing the log-likehood

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} LL(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \ln p(D|\boldsymbol{\theta}).$$

For GMM, we know the log-likelihood is

$$LL(\boldsymbol{\theta}) = \sum_{i=1}^{M} log \left\{ \sum_{j=1}^{N} \alpha_j \mathcal{N}(\boldsymbol{x}_i | \mu_j, \Sigma_j) \right\}.$$

A simple derive of the EM procedure is by expanding the log-likelihood including a valid probability distribution $q(Z)$,

$$\begin{aligned} LL(\boldsymbol{\theta}) &= \sum\nolimits_Z q(Z) \ln p(D|\boldsymbol{\theta}) \\ &= \sum_Z q(Z) \Big(\ln(p(D, Z|\boldsymbol{\theta})/q(Z)) - \ln(p(Z|D, \boldsymbol{\theta})/q(Z)) \Big) \\ &= \sum_Z q(Z) \ln(p(D, Z|\boldsymbol{\theta})/q(Z)) + KL(q||p) \\ &= \mathcal{L}(q, \boldsymbol{\theta}) + KL(q||p). \end{aligned} \tag{1}$$

where $KL(q||p)$ is the KL-divergence between the two distributions. Due to the expansion of the log-likelihood into two terms, the maximization can be carried in two alternative E-steps and M-steps.

The E-step, given the current parameters $\boldsymbol{\theta}^{(t)}$, matches the distribution $q(Z)$ to $p(Z|D, \boldsymbol{\theta}^{(t)})$, so that $KL(q||p)$ reaches its minimum at zero and

$$\begin{aligned} LL(\boldsymbol{\theta}) &= \mathcal{L}(q, \boldsymbol{\theta}) = \sum\nolimits_Z q(Z) \ln(p(D, Z|\boldsymbol{\theta})/q(Z)) \\ &= \sum\nolimits_Z p(Z|D, \boldsymbol{\theta}^{(t)}) \ln p(D, Z|\boldsymbol{\theta}) - q(Z) \ln q(Z). \end{aligned}$$

For GMM, the E-step finds a guess of $q(Z)$ according to the posterior

$$p(z_{ij}|\boldsymbol{x}_j, \boldsymbol{\theta}^{(t)}) = \frac{p(\boldsymbol{x}_j, z_{ij}|\boldsymbol{\theta}^{(t)})}{p(\boldsymbol{x}_j|\boldsymbol{\theta}^{(t)})}$$

in which $z_{ij} = 1$ if and only if $\boldsymbol{x}_j$ is generated by the i-th Gaussian model.

Then the M-step is to maximize $LL(\boldsymbol{\theta})$ by maximizing the the non-constant term

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = E_{Z\sim p(Z|D,\boldsymbol{\theta}^{(t)})} \ln(p(D, Z|\boldsymbol{\theta})).$$

For GMM, we know the probability that an instance $\boldsymbol{x}_i$ is generated by the k-th Gaussian model is

$$\gamma(i,k) = \frac{\alpha_k \mathcal{N}(\boldsymbol{x}_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{N} \alpha_j \mathcal{N}(\boldsymbol{x}_i|\mu_j, \Sigma_j)}.$$

Letting $M_k = \sum_{i=1}^{M} \gamma(i,k)$, we can update α, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ as

$$\begin{aligned} \boldsymbol{\mu}_k &= \frac{1}{M_k} \sum\nolimits_{i=1}^{M} \gamma(i,k) \boldsymbol{x}_i \\ \boldsymbol{\Sigma}_k &= \frac{1}{M_k} \sum\nolimits_{i=1}^{M} \gamma(i,k) (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu_k})^T \\ \alpha_k &= \frac{M_k}{M}. \end{aligned}$$

The E-steps and M-steps iterate until the convergence. The convergence of the EM procedure has been well addressed. Including the convergence properties of EM on finite mixture models [Jordan and Xu, 1995], the link between EM procedure and gradient methods via the projection matrix [Xu and Jordan, 1996], and more recently, the increase of the likelihood along any EM sequence [Bouveyron and Brunet, 2012], and the statistical consistency of parameter estimation for GMM [Xu *et al.*, 2016].

2.3 Generative Adversarial Network

Generative adversarial network (GAN) [Goodfellow *et al.*, 2014] is a class of approaches that train neural network models to generate data very similar to the training data. GAN commonly consists of two networks competing with each other in a zero-sum game framework. The objective function of GAN is presented as [Goodfellow *et al.*, 2014]

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

where G is the generator that takes input of a noise z from a noise distribution p_z and outputs a sample, D is the discriminator, and p_r is the distribution of the training data.

The initial GAN could be hard to train as it was delicate and unstable. Soon following GAN, many variants have emerged. For examples, the DCGAN [Radford *et al.*, 2015] employs deep convolutional neural networks for generating high quality pictures; the f-GAN [Nowozin *et al.*, 2016] use the f-divergence to train GAN; and the Wasserstein GAN (WGAN) [Arjovsky *et al.*, 2017] choose to minimize an efficient approximation of the Earth-Mover distance to improve the performance of GAN.

GANs have shown very strong ability at capturing complex data distributions, such as images and audios from raw features. Therefore, GANs can be a useful replacement of the simple Gaussian model for complex data clustering.

3 GAN Mixture Model

Early Convergence of EM Procedure

We use GAN models instead of Gaussian models in learning mixture models for capturing clusters of complex data, while keeping the loss function unchanged. However, it is not straightforward to train GAN mixture models (GANMM) as that in GMM. To illustrate the problem, let us recall the EM procedure. As the log-likelihood is expanded in (1), the E-step matches the distribution of the hidden variable (i.e., clusters) with the posterior probability of the current model, i.e., $q(Z) = p(Z|D, \boldsymbol{\theta}^{(0)})$ from any initial model $\boldsymbol{\theta}^{(0)}$. Then the M-step solves the parameters $\boldsymbol{\theta}^{(1)}$ from the resulted log-likelihood

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(0)}) = \sum\nolimits_Z p(Z|D, \boldsymbol{\theta}^{(0)}) \ln p(D, Z|\boldsymbol{\theta})$$
$$= \sum\nolimits_Z p(Z|D, \boldsymbol{\theta}^{(0)})(\ln p(Z|D, \boldsymbol{\theta}) + \ln p(D|\boldsymbol{\theta})).$$

Note that, if our model can be arbitrarily capable, the model maximizing $\mathcal{L}(q, \boldsymbol{\theta})$ will have the extreme value that $p(\boldsymbol{z}^{(0)}|D, \boldsymbol{\theta}^{(1)}) = 1$ for $\boldsymbol{z}^{(0)} = \operatorname{argmax}_{\boldsymbol{z}} p(\boldsymbol{z}|D, \boldsymbol{\theta}^{(0)})$ and otherwise 0, and $p(D|\boldsymbol{\theta}) = 1$.

In the next iteration, we will have $q(Z) = p(Z|D, \boldsymbol{\theta}^{(1)})$ and solves $\boldsymbol{\theta}^{(2)}$ such that $p(\boldsymbol{z}^{(1)}|D, \boldsymbol{\theta}^{(2)}) = 1$ for $\boldsymbol{z}^{(1)} = \operatorname{argmax}_{\boldsymbol{z}} p(\boldsymbol{z}|D, \boldsymbol{\theta}^{(1)})$ and otherwise 0. Noticed that

$$\operatorname*{argmax}_{\boldsymbol{z}} p(\boldsymbol{z}|D, \boldsymbol{\theta}^{(1)}) = \boldsymbol{z}^{(0)} = \operatorname*{argmax}_{\boldsymbol{z}} p(\boldsymbol{z}|D, \boldsymbol{\theta}^{(0)}),$$

we have $\theta^{(1)} = \theta^{(2)}$, and thus the procedure has converged.

Therefore, the EM procedure will converge too early if the model is of very high capacity. That would be likely to happen for using GANs as the model.

ϵ-Expectation-Maximization Procedure

To avoid the early convergence of the EM procedure, one way is to avoid that the model fits the current guess of the hidden variables too well. Our idea is to introduce some error in the E-step, i.e., an ϵ-E-step with

$$KL(q^{(t)}||p^{(t)}) = \epsilon_t > 0,$$

By Eq.(1), we now have

$$LL(\boldsymbol{\theta}^{(t)}) = \mathcal{L}(q^{(t)}, \boldsymbol{\theta}^{(t)}) + \epsilon_t.$$

and thus

$$\forall q, \mathcal{L}(q, \boldsymbol{\theta}^{(t)}) \leq LL(\boldsymbol{\theta}^{(t)}) = \mathcal{L}(q^{(t)}, \boldsymbol{\theta}^{(t)}) + \epsilon_t.$$

Since the M-step is reserved, we still have

$$\boldsymbol{\theta}^{(t)} = \arg\max\nolimits_{\boldsymbol{\theta}} \mathcal{L}(q^{(t-1)}, \boldsymbol{\theta}),$$

which means $\mathcal{L}(q^{(t-1)}, \boldsymbol{\theta}^{(t)}) \geq \mathcal{L}(q^{(t-1)}, \boldsymbol{\theta}^{(t-1)})$. Therefore, we have

$$LL(\boldsymbol{\theta}^{(t)}) = \mathcal{L}(q^{(t)}, \boldsymbol{\theta}^{(t)}) + \epsilon_t \geq \mathcal{L}(q^{(t-1)}, \boldsymbol{\theta}^{(t)})$$
$$\geq \mathcal{L}(q^{(t-1)}, \boldsymbol{\theta}^{(t-1)}) = LL(\boldsymbol{\theta}^{(t-1)}) - \epsilon_{t-1}$$
$$\geq LL(\boldsymbol{\theta}^{(0)}) - \sum\nolimits_{i=0}^{t-1} \epsilon_i,$$

which shows that the ϵ-E-step can lead to an amount of loss of the log-likelihood. However, if we keep $\lim\limits_{t\to+\infty} \sum_{i=0}^{t-1} \epsilon_i < \infty$, there exists $C > 0$ such that for all $t > C$, $LL(\boldsymbol{\theta}^{(t)}) \geq LL(\boldsymbol{\theta}^{(t-1)})$ and thus the procedure converges. The above procedure is call ϵ-EM.

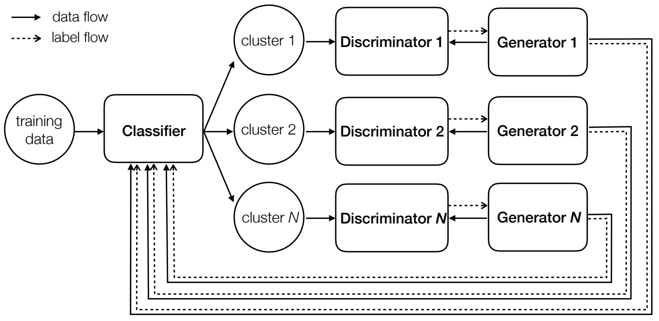

Figure 1: The GANMM Architecture

GAN Mixture Model

To implement the ϵ-EM procedure, we need an unfitted model for the expectation at some beginning steps. We restrict that the distribution of the clusters $q^{(t)}$ in a bounded hypothesis space H_t. Then, $\epsilon_t = \min_{q\in H_t} KL(q||p^{(t)})$. Given the power of the M-step, it is easy to see that as long as $\forall t : H_t \subseteq H_{t-1}$ and $|H_t| > 0$ (i.e., the ϵ-E-step cannot be quite adversarial, the M-step will fit the data and eliminate the error), the finite accumulated error that $\lim_{t\to+\infty} \sum_{i=0}^{t-1} \epsilon_i < \infty$ is guaranteed.

In the above, finding $q^{(t)}$ by the criterion $\min_{q\in H_t} KL(q||p^{(t)})$ inspires us to train a classifier with moderate complexity (i.e., to control the hypothesis space H_t) to fit the current guess of the cluster assignment. Therefore, we propose the GANMM training approach using the ϵ-EM procedure as follows.

ϵ-E-step: (1) From the current model $\boldsymbol{\theta}^{(t)}$, sample a data set $S^{(t)} = \{(\tilde{\boldsymbol{x}}_i, y_i)_{i=0}^n\}$ from the GAN generators, while $\tilde{\boldsymbol{x}}_i$ is the data point generate by k-th generator and $y_i = k$. (2) Train a (not so perfect) classifier h_q from S. (3) Assign the cluster of each training instance $\boldsymbol{x}_i$ by the training data by the classifier, i.e., $h_q(\boldsymbol{x}_i)$.

M-step: For the clustered data $D = \{D_1, D_2, \ldots, D_N\}$, train the i-th GAN model on D_i for several iterations.

We run the ϵ-E-step and the M-step alternatively until convergence. The GANMM architecture is shown in Figure 1.

Implementation Details

GAN model: Since the original GAN approach proposed in [Goodfellow *et al.*, 2014] has some limitations such as missing modes, we employ the Wasserstein GAN (WGAN) [Arjovsky *et al.*, 2017] to train our GAN models, which implemented from `https://github.com/igul222/improved_wgan_training`. WGAN optimize the two losses iteratively

$$loss_G = -E_{x\sim P_g}[f_w(x)]$$
$$loss_D = E_{x\sim P_g}[f_w(x)] - E_{x\sim P_r}[f_w(x)]$$

Initialization: GMM usually starts from a random parameter initialization. However, for a random parameter initialization of GANMM, it is highly possible that all training data is assigned to one cluster, resulting in a bad clustering. We therefore start from a random assignment of training instances to clusters, and then learn each GAN model from each cluster, and continue the ϵ-EM procedure.

Algorithm 1 GAN mixture model learning algorithm

```
Require:
    $\alpha$: learning rate.
    $m$: training set size.
    $N$: cluster number.
    $n_{epoch}$: number of epoch for GANs.
    $\sigma_t$, number of augmented data points.
1:  randomly divide $D$ into $\{D_1^{(0)}, \ldots, D_N^{(0)}\}$
2:  $\boldsymbol{\theta}_{D_i^{(0)}}^{(0)}, \boldsymbol{\theta}_{G_i}^{(0)} \leftarrow$ trainWGAN($D_i^{(0)}$, $n_{epoch}$) for each $i = 1, \ldots, N$
3:  $t = 0$
4:  while $\boldsymbol{\theta}_E$ not converged do
5:      $t = t + 1$
6:      $S = \{(\tilde{\boldsymbol{x}}_i, y_i)\}_{i=1}^m$ is sampled from $\{G_j\}_{j=1}^N$ each with probability $|D_i^{(t-1)}|/|D|$.
7:      $g_{\theta_h} \leftarrow \nabla_{\theta_h}[\frac{1}{m}\sum_{i=1}^m cross\_entropy(f_{\boldsymbol{\theta}_h}(\tilde{\boldsymbol{x}}_i), y_i)]$
8:      $\theta_h \leftarrow \theta_h + \alpha\cdot$ RMSProp($\theta_h, g_{\theta_h}$)
9:      assign $D$ as $\{D_i\}_{i=1}^N$ by using $h_q(D; \theta_h)$
10:     for $i = 1$ to $N$ do
11:         add $\sigma_t$ instances from $D - D_i$ with highest posterior for cluster $i$ by $h_q$ to $D_i$
12:         $\boldsymbol{\theta}_{D_i}^{(t)}, \boldsymbol{\theta}_{G_i}^{(t)} \leftarrow$ trainWGAN($D_i^{(t)}$, $n_{epoch}$) for each $i = 1, \ldots, N$
13:     end for
14: end while
```

Dealing with imbalanced classification: Since we employed a ϵ-E-step classifier trained from GAN generated samples, the classifier is easy to assign clusters with imbalanced instances, particularly at the beginning of the procedure. The imbalance could get reinforced through the procedure. As a result, some GAN models receives more and more data and the remaining fewer and fewer.

To fix this issue, we first make sure that every GAN model generates the same amount of data for training the ϵ-E-step classifier. Moreover, we allow a GAN model explore new by augmenting their training data. After the assignment of clusters, $D = \{D^1, \ldots, D^N\}$, we augment each cluster data set D_i by adding an amount of instances from $D - D_i$ with the highest posterior probability of belonging to the i-th cluster according to the output of the classifier. The amount of the augmented data is reducing along the procedure for the convergence.

An implementation of GANMM can be found at `https://github.com/eyounx/GANMM`.

4 Experiments

We compare GANMM with GMM, a state-of-the-art deep clustering method Deep Embedded Clustering (DEC) [Xie *et al.*, 2016], and the degrade version of GANMM trained using EM procedure (where the E-step uses the discriminators to assign the clusters). Note that DEC was originally proposed to operate in the embedded space by an autoencoder. For a fair comparison, we have tested them both in the raw features space and the embedded space. The implementation of DEC is from `https://github.com/fferroni/DEC-Keras`. As for GANMM, we don't modify the WGAN implementation, and the classifier has two convolution layers and two dense layers on the raw feature data sets. Meanwhile, we use only the two dense layers in the generator, discriminator and the classifier on the embedded feature data sets.

Evaluate Metrics

Since clustering results are hard to be measured objectively, using the data with oracle labels (note that the original labels themselves usually not purposed for clustering), we employ three metrics: purity, Adjusted Rand index (*ARI*) and Normalized Mutual Information (*NMI*).

Purity is the average of the portion of the largest class in each cluster, i.e.,

$$purity = \sum_{i=1}^{N} \frac{m_i}{m} \max_j \frac{m_{ij}}{m_i},$$

where m_i is size of cluster i, and m_{ij} is the number of class j instances in cluster i. *Purity* lies in between 0 and 1. Higher purity indicates that each cluster is more concentrated.

ARI considers the number of instances that exist in the same cluster and different clusters.

$$ARI = \frac{m_{11} + m_{00}}{m},$$

where m_{11} is the number of pairs of instances that are in the same cluster and have the same oracle label, and m_{00} is the number of pairs of instances that are in the different cluster and have the different oracle labels. *ARI* lies in between 0 and 1. High *ARI* means that instances are clustered more correctly.

NMI can effectively measure the amount of statistical information shared by random variables representing the cluster assignments and the oracle labels [Fahad *et al.*, 2014].

$$NMI = \frac{\sum d_{h,l} log(\frac{|\Omega| \cdot d_{h,l}}{d_h c_l})}{\sqrt{(\sum_h d_h log(\frac{d_h}{d}))(\sum_l c_l log(\frac{c_l}{d}))}},$$

where d_h is the number of instances in class h. c_l denotes the number of instances in cluster l. And $d_{h,l}$ is the number of instances in cluster l which have ground-truth label h. *NMI* also lies in between 0 and 1 and high *NMI* means cluster assignments matches ground-truth label assignments well.

4.1 On MNIST Dataset

We first compare the methods on the classical MNIST dataset [LeCun *et al.*, 1998]. It is a handwriting digital dataset containing 60,000 images of size 28 by 28 pixels consist of 10 classes from digit '0' to '9'.

Clustering in Raw Feature Space

In the raw 28 by 28 pixels feature space, we test the clustering performance of each method, which is a challenging task as the algorithm is required to extract features and find the clusters simultaneously. The experiments are repeated 10 times to report the means and standard deviations, as listed in Table 1.

We can observe that GANMM has significantly higher scores than all other methods on all three metrics. It achieves twice purity and triple ARI and NMI of the runner-up. DEC (in this experiment without a fine-tuned stacked auto-encoder) performs overall similar with GMM, which is slightly worse in purity and slightly better in ARI and NMI. Meanwhile, GANMM (EM) has the worst performance in purity and ARI.

To see more closely, we list the purity results of these methods on MNIST datasets in a single experiment in Table 2. First, counting the majority of the oracle label in each cluster, GANMM and GMM each loses one label ("9"), DEC loses three labels ("2", "5", "6"), and GANMM (EM) loses the most labels ("0", "4", "5", "8"). Then from the number of instances in each cluster, GANMM and GMM achieve related balanced clusters, DEC has small (<2,000) and large (>10,000) clusters, and GANMM (EM) is the worst with a cluster eliminated. These results show that GANMM achieves similar properties with the classical GMM. Meanwhile, GANMM achieves significantly higher purity in clusters than all the other methods, from 0.5666 to 0.9379.

This experiment shows that GANMM has an outstanding ability to deal with complex data directly, without any extra pre-process step. Comparing with the inferior performance of GMM and DEC, we can owe the ability of GANMM to the employment of the GANs that can capture complex distributions. Comparing with the degraded performanced of GANMM (EM), it shows that the ϵ-EM procedure is crucial to iterate the GANMM models for a good convergence.

Since GANMM is composed of generative models, we can easily observe GANMM by generating images from it. A set of generated images by GANMM and GANMM (EM) are shown in Figure 2. Images in the left two columns are generated by the ten clusters of GANMM and right two columns are generated by GANMM (EM). Each block is generated from one of the GAN model.

It can be observed that the digits generated by GANMM are obviously more regular than those generated by GANMM (EM). Although we observed from Table 2 that GANMM losses the digit "9", we can find that it actually generated "9" but mixed with digits "4" and "7". Meanwhile, some digits generated by GANMM (EM) are not in good shape, which is due to the very small clusters it receives.

	Purity	ARI	NMI
GANMM	**0.6430±0.0045**	**0.4924±0.0059**	**0.6159±0.0038**
GMM	0.3261±0.0006	0.0991±0.0003	0.1414±0.0004
DEC	0.3065±0.0003	0.1437±0.0005	0.1935±0.0003
GANMM(EM)	0.2786±0.0019	0.0665±0.0013	0.2414±0.0004

Table 1: Comparison of clustering performance in raw feature space on MNIST.

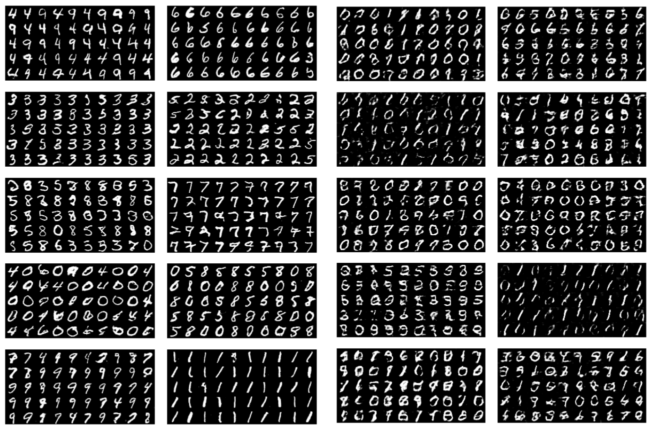

Figure 2: Images generate by GANMM (left two columns) and by GANMM (EM) (right two columns)

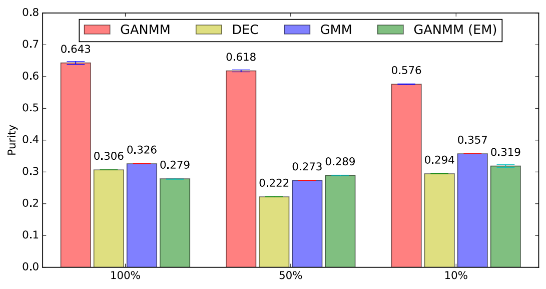

Figure 3: Comparison of purity on different data scales

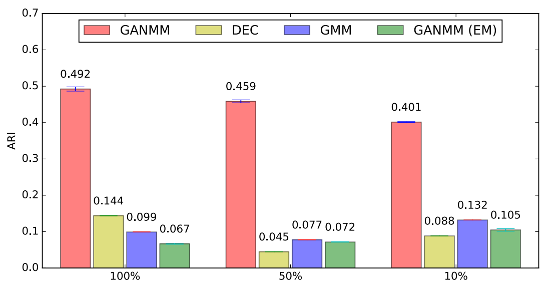

Figure 4: Comparison of ARI on different data scales

On Different Data Scales

We then investigate how sensitive GANMM is to the data scale. We experimented on MNIST dataset using different data scales. We train on 100%, 50%, 10% of MNIST dataset separately and assign the clusters on the whole dataset. In this experiment, we also directly run the clustering methods in the original raw feature space. We repeat the experiments five times to report the means and standard deviations. The results are shown in Figures 3, 4, and 5.

It can be observed that even on 10% of the data, i.e., 6,000 instances in all, GANMM still have significantly superior performance to the other methods, on all the three metrics. Meanwhile, we can still observe some performance drops of GANMM as the data size reduces. Since GANMM can fit complex data distribution, too few instances would result in worse fitted model. The other approaches are less effected by the data scales, as their performance has already been quite low compared with GANMM.

Clustering in Embedded Feature Space

We also compare the methods in an embedded feature spaced. We use a fine-tuned stacked auto-encoder (SAE) to transform the original pictures to 10 dimensional feature vectors. The SAE is the same as in the DEC's paper [Xie *et al.*, 2016]. The results of the compared methods are shown in Table 3.

cluster id	GANMM			GMM			DEC			GANMM(EM)		
	label	#data point	purity	label	#data point	purity	label	#data point	purity	label	#data point	purity
1	1	5940	0.9149	3	5894	0.3174	8	5914	0.1921	9	510	0.3941
2	7	2984	0.5666	1	6094	0.4641	7	2219	0.6237	3	9291	0.2797
3	7	5099	0.6607	4	4169	0.3118	9	1315	0.5148	9	6098	0.3296
4	6	4982	0.9379	2	5415	0.3162	1	6185	0.8791	1	15109	0.3395
5	8	4744	0.6072	5	2895	0.1565	3	8769	0.2127	7	327	0.9220
6	2	4958	0.9255	1	4692	0.4235	7	1649	0.3596	2	16207	0.2269
7	5	4128	0.6061	0	7773	0.3435	7	3069	0.6695	6	429	0.9161
8	4	5526	0.7144	6	5245	0.3399	4	4317	0.3141	-	0	0.0000
9	3	5852	0.6476	8	2016	0.2787	0	14282	0.3161	1	40	0.7250
10	0	5787	0.7665	7	5807	0.3675	9	2285	0.4512	6	1944	0.9460

Table 2: Purity of each cluster in one experiment

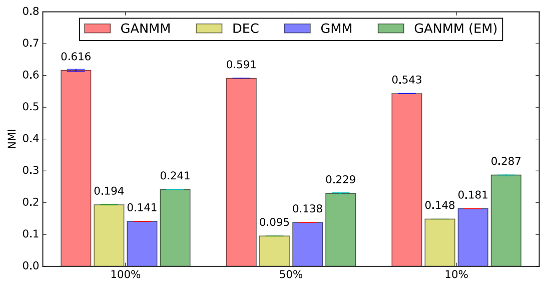

Figure 5: Comparison of NMI on different data scales

	Purity	ARI	NMI
GANMM	**0.8908±0.0015**	**0.8361±0.0025**	**0.8654±0.0008**
GMM	0.8617±0.0008	0.7933±0.0011	0.8451±0.0002
DEC	0.8673±0.0000	0.8091±0.0000	0.8457±0.0000
GANMM(EM)	0.5243±0.0024	0.3210±0.0034	0.4701±0.0019

Table 3: Comparison of clustering performance in embedded feature space on MNIST

GANMM (EM) has the worst performance in the embedded feature space, while the performance of GMM and DEC has been drastically improved from the raw feature space. This observation implies that GMM and DEC seriously rely on feature pre-process in applications, and would work in low dimensional spaces. Still, GANMM achieves better performances in all metrics. GANMM improves from DEC by around 0.02 on *purity* and *NMI*, and 0.03 on *ARI*. These improvements can be significant given the high values of the baseline scores. The result reveal that GANMM can work well not only in raw feature spaces but also in highly abstracted embedded spaces.

4.2 On More Datasets

We finally compare the clustering performance on two UCI datasets [Dua and Karra, 2017]. They are Image Segmentation dataset and Artificial Characters dataset. The Image Segmentation dataset is an image dataset described by high-level numeric-valued attributes. It contains 2310 instances with 19 features. It have 7 classes, which are *BRICKFACE, FOLIAGE, CEMENT, WINDOW, PATH, SKY, GRASS*. The Artificial Characters dataset is artificially generated by using first order theory which describes structure of ten capital letters of English alphabet. It contains 6000 instances with 6 features. It has 10 classes, which are *A, C, D, E, F, G, H, L, P, R*.

	Purity	ARI	NMI
GANMM	**0.7241±0.0010**	**0.5523±0.0022**	**0.6428±0.0018**
GMM	0.4532±0.0048	0.2214±0.0095	0.4124±0.0047
DEC	0.4428±0.0078	0.2014±0.0104	0.4077±0.0159
GANMM(EM)	0.5791±0.0012	0.4101±0.0010	0.5386±0.0009

Table 4: Comparison of clustering performance in raw feature space on Image Segmentation dataset

	Purity	ARI	NMI
GANMM	**0.2568±0.0002**	**0.0887±0.0003**	**0.2027±0.0022**
GMM	0.2184±0.0001	0.0452±0.0000	0.1206±0.0001
DEC	0.2031±0.0000	0.0365±0.0000	0.0856±0.0000
GANMM(EM)	0.2495±0.0000	0.0645±0.0000	0.1309±0.0000

Table 5: Comparison of clustering performance in raw feature space on Artificial Characters dataset

Tables 4 and 5 present the experiment results. It can be observed that in all cases and all metrics, GANMM consistently has a better performance than the other methods. Note that the improvement of the metric scores can be quite significant in both data sets. These results imply that GANMM can be applied in various situations.

5 Conclusion

In this paper, we propose a mixture model with GAN as the components, i.e., GANMM. Mixture models such as Gaussian mixture models are very popular tools for data analysis. However, traditional distribution models, such as Gaussian distribution, is lack of flexibility for complex data. The recently developed GAN model has shown outstanding performance to model complex distributions. Therefore, GANMM can serve as an alternative mixture model to GMM for complex situations. We noticed that GANMM is hard to be trained by the classical EM procedure. We thus propose to use the ϵ-EM procedure, where the ϵ-E-step, implemented by a classifier learned from the clusters, introduces some controllable error so that the optimization of GANMM can continue. Several experiment results show that GANMM can drastically improve the clustering performance on complex data from GMM as well as some state-of-the-art approaches, in both of the raw feature space and abstracted embedded feature space.

References

[Arjovsky *et al.*, 2017] Martin Arjovsky, Soumith Chintala, and Leon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223, Sydney, Australia, 2017.

[Bouveyron and Brunet, 2012] Charles Bouveyron and Camille Brunet. Theoretical and practical considerations on the convergence properties of the Fisher-EM algorithm. *Multivariate Analysis*, 109:29–41, 2012.

[Chen and Gopalakrishnan, 1998] Scott Shaobing Chen and Ponani S. Gopalakrishnan. Clustering via the bayesian information criterion with applications in speech recognition. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 645–648, Seattle, Washington, USA, 1998.

[Chen *et al.*, 2016] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29*, pages 2172–2180, Barcelona, Spain, 2016.

[Dua and Karra, 2017] Dheeru Dua and Taniskidou Efi Karra. UCI machine learning repository, 2017.

[Duda *et al.*, 2012] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

[Fahad *et al.*, 2014] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y. Zomaya, Sebti Foufou, and Abdelaziz Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics of Computing*, 2(3):267–279, 2014.

[Frigui and Krishnapuram, 1999] Hichem Frigui and Raghu Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):450–465, 1999.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680, Montreal, Canada, 2014.

[Jordan and Xu, 1995] Michael Irwin Jordan and Lei Xu. Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, 8(9):1409–1431, 1995.

[LeCun *et al.*, 1998] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Li and Kim, 2003] Qing Li and Byeong Man Kim. Clustering approach for hybrid recommender system. In *Proceedings of the 2003 IEEE / WIC International Conference on Web Intelligence*, pages 33–38, Halifax, Canada, 2003.

[Li *et al.*, 2015] Baohua Li, Ying Zhang, Zhouchen Lin, and Huchuan Lu. Subspace clustering by mixture of gaussian regression. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2094–2102, Boston, MA, USA, 2015.

[McLachlan and Basford, 2004] Geoffrey John McLachlan and Kaye Enid Basford. *Mixture models: Inference and Applications to Clustering*. Marcel Dekker, 2004.

[McLachlan and Peel, 2004] Geoffrey John McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[Nowozin *et al.*, 2016] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems 29*, pages 271–279, Barcelona, Spain, 2016.

[Radford *et al.*, 2015] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv*, 1511.06434, 2015.

[Raghunathan *et al.*, 2017] Aditi Raghunathan, Prateek Jain, and Ravishankar Krishnaswamy. Learning mixture of Gaussians with streaming data. In *Advances in Neural Information Processing Systems 30*, pages 6608–6617, Long Beach, CA, 2017.

[Xie *et al.*, 2016] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 478–487, New York City, NY, 2016.

[Xu and Jordan, 1996] Lei Xu and Michael Irwin Jordan. On convergence properties of the em algorithm for Gaussian mixtures. *Neural computation*, 8(1):129–151, 1996.

[Xu *et al.*, 2016] Ji Xu, Daniel J. Hsu, and Arian Maleki. Global analysis of expectation maximization for mixtures of two Gaussians. In *Advances in Neural Information Processing Systems 29*, pages 2676–2684, Barcelona, Spain, 2016.