Models for Clustering with Arbitrary Shaped Clusters

Sarthak Mittal 150640
Gurpreet Singh 150259
Debabrata Ghosh 13817226

Abstract

Most real world data can be represented in simpler patterns in a possible lower dimensional manifold and the data can be generated through some non-linear mapping from this latent space to the observed space. We study a number of existing techniques that allow clustering of data by jointly representing the data in a lower dimensional latent form and performing clustering over this latent space. We present a generic generative story for such models and look at various number of ways to perform inference in different settings. We also present some discussion regarding mixture of experts and propose two models extending the latent clustering to form gate networks for better mixtures.

1. Introduction

Humans are capable of recognizing cluster patterns even when the shapes and sizes of the clusters are immensely varied. The intrinsic shortcoming of K-Means and Gaussian Mixture Models is that these models are able to properly cluster the data only when it can be represented in gaussian shaped clusters, and we need to specify the number of clusters. Though Dirichlet Process Mixture Models alleviate the latter problem of the model, it still tries to capture the data as a mixture of Gaussians.

One of the most popular ways to solve this problem is by constructing a latent representation of the data and a complex non-linear function that can efficiently map the latent space representations to the data space representations. The latent representation can be thought of as a simple distribution like a mixture of Gaussians which is warped by a complex non-linear function to represent the data. Now, clustering on this latent representation by models that only allow gaussian-like cluster shapes still allow for arbitrary shaped clusters in the data space because of the non-linear warping functions.

One way to approach the above problem is to first uncover a good latent representation of the data through some non-linear transformation techniques like a Gaussian Process variant of Probabilistic Principal Component Analysis (Lawrence, 2004) or using a Variational Autoencoder. Both of these constitute powerful non-linear function approximators and thus solve the problem well. Once we have the latent representation of the data, we perform clustering on the latent representation through some standard clustering techniques like Gaussian Mixture Model or Dirichlet Process Mixture Model. Thus, we get clusters of arbitrary shapes.

The above technique, however, is not optimal because in doing so, we have made the task of latent

representation and clustering mutually independent. Therefore, our model will be focused on finding a good latent representation which can be reconstruct our observations with high precision, but would not care about the inherent differences in the observations, *i.e.* we would loose the inherent clustering present in the observations.

However, we can see that these tasks are actually quite related to each other and any model would be able to construct more disambiguated latent representations if it performs the task of latent representation and clustering jointly. Thus, to efficiently solve the problem, we define a Generative Model where a vector in the latent space is sampled from a mixture of Gaussians (either a finite mixture or an infinite mixture) and then learn a complex non-linear function that maps the latent space to data space.

2. Relevant Background

2.1. Variational Autoencoders

The key idea behind Variational Autoencoders (VAEs, Kingma and Welling, 2013) is that any distribution in D dimensions can be generated by taking a set of D variables that are normally distributed and mapping them through a sufficiently complicated function (Doersch, 2016; Kingma and Welling, 2013). Hence, provided powerful function approximators, we can simply learn a function which maps independent, normally-distributed variables to the latent variables needed for our model, and these latent variables are then used to model the observations $\mathbf{X}$, or rather the data distribution $\mathcal{D}$ not known to us, from which the observations are assumed to be sampled.

Therefore, the task is to find a strong non-linear mapping to approximate the latent variables needed to model the data distribution $\mathcal{D}$. This mapping, in VAEs, is modelled using a neural network, and the mapping from these latent variables to the observations is through a distribution (generally exponential family) depending on the observed space $\mathcal{X}$. For example, if $\mathcal{X} = \mathbb{R}$, then the estimated probability distribution for $\mathbf{x} \sim \mathcal{D}$ can be given as

$$\mathbb{P}\left[\mathcal{D} \,\middle|\, \mathbf{z}\right] \approx \mathcal{P}(\mathbf{X} \,|\, \mathbf{z}; \boldsymbol{\theta}) = \prod_n \mathcal{N}\left(\mathbf{x}_n \,\middle|\, f(\mathbf{z}; \boldsymbol{\theta}), \sigma^2 \mathbf{I}\right)$$

for some non-linear mapping $f : [\,0, 1\,] \to \mathbb{R}$ (in this case) modelled using a neural network.

The objective now is to compute the posterior over $\mathbf{z}$ given the observations $\mathbf{X}$. This would allow us to model the predictive posterior using samples of $\mathbf{z}$ from this posterior. Since the posterior is intractable, we approximate the posterior $\mathbb{P}\left[\mathbf{z} \,\middle|\, \mathcal{D}\right]$ using a proposal distribution $\mathcal{Q}(\mathbf{z}; \boldsymbol{\phi})$. We use the standard VI strategy to estimate the posterior, by maximizing the ELBO bound, which is given as

$$\begin{aligned} \mathcal{L}(\mathcal{Q}) &= -\mathrm{KL}\left(\mathcal{Q}(\mathbf{z}; \boldsymbol{\phi}) \,||\, \mathbb{P}\left[\mathbf{z} \,\middle|\, \mathcal{D}\right]\right) \\ &= \log \mathbb{P}\left[\mathcal{D} \,\middle|\, \mathbf{z}\right] - \mathrm{KL}\left(\mathcal{Q}(\mathbf{z}; \boldsymbol{\phi}) \,||\, \mathbb{P}\left[\mathbf{z}\right]\right) \end{aligned}$$

This is similar to a Variational Inference (VI) objective. In fact, this is the VI procedure. However one problem with VI (with mean-field assumption) is that it models the parameters of each latent variable independently, and therefore we need an iterative procedure to model the proposal with respect to each variable. VAE tackles this problem by assuming that the latent variable is dependent on the input, and the mapping from the input to the parameters of the proposal distribution is modeled using a neural network. Since looking at the ELBO, one can say we wish to minimize the KL Divergence between the proposal and the prior for $\mathbf{z}$ which is a standard Normal distribution, we can assume the proposal to be of the form $\mathcal{N}\left(\mathbf{z} \,\middle|\, \mu(\mathbf{x}; \boldsymbol{\phi}), \sigma^2(\mathbf{x}; \boldsymbol{\phi})\, \mathbf{I}\right)$. The negative ELBO is then considered as our loss, and the parameters (or weights) of the Neural Networks are learned using Backpropogation.

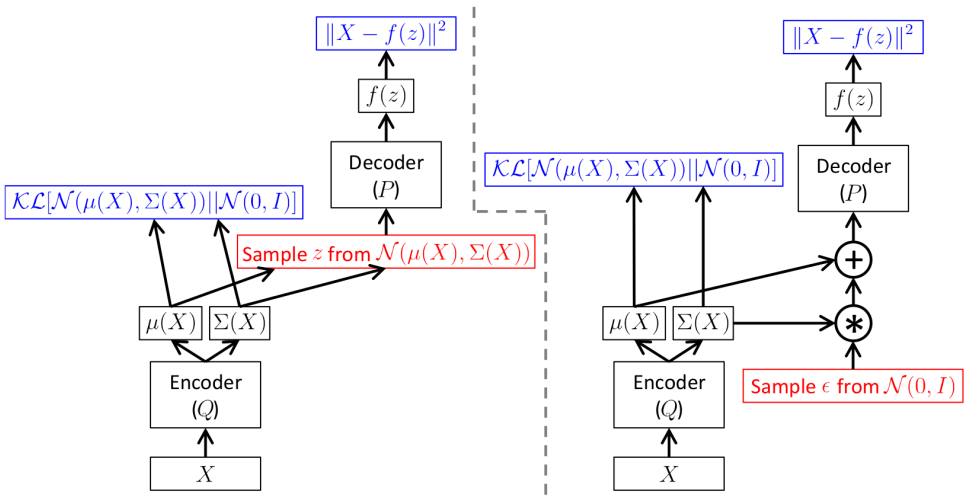

Figure 1: Left: Without the reparameterization trick, Right: With the reparameterization trick

This structure is similar to that of a standard autoencoder, where the network which maps the inputs to the parameters of the proposal is the encoder and the network mapping the sampled $\mathbf{z}$'s to the parameters of the conditional likelihood distribution is the decoder.

However, since we are assuming the variables $\mathbf{z}$ to be sampled from the proposal, we cannot apply back propogation across the decoder and the encoder networks. This is handled using the reparameterization trick, $\mathbf{z} = \epsilon \cdot \boldsymbol{\sigma}^2(\mathbf{x}; \boldsymbol{\phi}) + \mu(\mathbf{x}; \boldsymbol{\phi})$. The working of a standard VAE is given in Figure 1.

2.2. Dirichlet Process Mixture Models

Similar to Dirichlet Processes or DPs are a family of stochastic processes. A Dirichlet process defines a distribution over probability measures $G : \Theta \to \mathbb{R}^+$, where, for any finite partition of Θ, say $\{\boldsymbol{\theta}_k\}_{k=1}^K$, the random vector $(G(\boldsymbol{\theta}_1), G(\boldsymbol{\theta}_2) \dots G(\boldsymbol{\theta}_K))$ is jointly generalized under a Dirichlet Distribution (Murphy, 2012) [1], written as

$$(G(\boldsymbol{\theta}_i), G(\boldsymbol{\theta}_2) \dots G(\boldsymbol{\theta}_K)) \quad \sim \quad \text{Dir}\left(\alpha G_0(\boldsymbol{\theta}_1), \alpha G_0(\boldsymbol{\theta}_2) \dots \alpha G_0(\boldsymbol{\theta}_K)\right)$$

where α is called the concentration parameter and G_0 is the base distribution. αG_0 collectively is called the base measure.

Dirichlet processes are really useful in the task of non-parametric clustering, using a mixture of Dirichlet Processes (Ferguson, 1973; Antoniak, 1974) (commonly DP Mixture Models or Infinite Mixture Models). In fact, a DP Mixture Model can be seen as an extension of Gaussian Mixture Models over a non-parametric setting.

The basic DP Mixture Model follows the following generative story

$$\begin{aligned}
\textbf{Likelihood:} \quad & \mathbf{y}_n \,|\, \boldsymbol{\theta}_n \quad \sim \quad F(\mathbf{y}_n \,|\, \boldsymbol{\theta}_n) \\
\textbf{Conditional Prior:} \quad & \boldsymbol{\theta}_n \,|\, G \quad \sim \quad G \\
\textbf{Hyperparameter:} \quad & G \quad \sim \quad \text{DP}\left(G_0, \alpha\right)
\end{aligned}$$

where $G \sim$ DP denotes sampling from a Dirichlet Process given a base measure.

[1] $\text{G}(\boldsymbol{\theta})$ is a random variable since G itself is a random measure and is sampled from the Dirichlet Process

When we are dealing with DP Mixture Models for clustering, it helps to integrate out G with respect to the prior on G. (Neal, 2000). Therefore, we can write the clustering problem in an alternate representation, although the underlying model remains the same.

$$\begin{aligned}
\textbf{Likelihood:} \quad & \mathbf{y}_n \,|\, c_n, \Phi &\sim\ & F(\mathbf{y}_n \,|\, \boldsymbol{\phi}_{c_n}) \\
\textbf{Latent Distribution:} \quad & c_n \,|\, \mathbf{p} &\sim\ & \text{Discrete}\,(p_1, p_2 \dots p_K) \\
\textbf{Priors:} \quad & \boldsymbol{\phi}_k &\sim\ & G_0 \\
& \mathbf{p} &\sim\ & \text{Dir}\,(p_1, p_2 \dots p_K)
\end{aligned}$$

where c_n is the cluster assignment for the n^{th} point and $\Phi = \{\boldsymbol{\phi}_k\}_{k=1}^{K}$ are the likelihood parameters for each cluster. K denotes the number of clusters, and being a non-parametric model, we assume $K \to \infty$.

If the likelihood and the base distribution are conjugate, we can easily derive a posterior representation for the cluster assignments or the latent classes, and use inference techniques such as Mean Field VB (Blei and Jordan, 2004) and Monte Carlo Markov Chain (Escobar and West, 1995; Neal, 2000). *Neal* also describes various inference methods in case of non-conjugate base distribution.

Dirichlet processes are extremely useful for clustering purposes as they do not assume an inherent base distribution, and therefore it is possible to apply Dirichlet Process priors over complex models.

2.3. Stick-Breaking VAE

Stick-Breaking Variational Autoencoder (Nalisnick and Smyth, 2016) is a recent model that aims to combine the Variational Autoencoder structure with Bayesian Nonparametrics. Fairly novel and ingenious, it combines Stick-Breaking process with the Variational Autoencoder structure by making the latent representations of data-points an infinite dimensional probability vector. This latent representation is accomplished by having a stick-breaking process that generates the latent representation $\boldsymbol{\pi}$, given as

$$\begin{aligned}
\pi_{n,1} &= \beta_1 \\
\pi_{n,l} &= \beta_l \prod_{l'=1}^{l-1} (1 - \beta_{l'}) \\
&\text{and so on...} \\
\beta_i &\sim \text{Beta}(1, \alpha)
\end{aligned}$$

This makes the latent representation unbounded and hence we get a nonparametric model combined with the Variational Encoder. There are a couple of problems that need to be addressed before the model can work. The problems are related mainly to inference since the latent dimension has unbounded dimension, performing inference would be a highly non-trivial task. Moreover, there is also the problem that we cannot use Beta distribution to model the posterior in the inference as we know that Beta distribution is not a location-scale distribution and thus cannot be reparameterized into a distribution independent of the parameters (unlike a Gaussian Distribution). More precisely, a Beta distribution cannot be represented in the form of a Differentiable, Non-centered parameterization.

We first look at the general problem of inference. To tackle the infinite dimensionality problem, the paper proposes to use an inference procedure similar to Variational Inference for Dirichlet Process Mixture Models, proposed by Blei et al. (2006). To accomplish this, there is a truncation level enforced on the variational distribution. Note that this truncation of the variational posterior distribution does not imply a finite dimensional latent representation since the truncation level is a parameter of the variational distribution and not the model specification. Hence the latent representation of the model still has unbounded dimensions.

Having solved the problem of general inference in the model, we look at the problem of back-propagating through the Beta distribution, that is making this distribution free of the parameters emitted by the neural network so that variance is controlled in training and backpropagating does not involve a non-differentiable layer that will prohibit gradient descent. This is achieved by using approximate distributions in place of the standard Beta distribution. The paper uses the Kumaraswamy Distribution, which is defined as follows:

$$\text{Kumaraswamy}\left(x \,|\, a, b\right) \quad = \quad a\, b\, x^{a-1}\left(1-x^{a}\right)^{b-1} \qquad x \in (0,1) \quad a, b > 0$$

The paper mentions that for $a = 1$ or $b = 1$ or both, the Kumaraswamy and Beta distributions are equivalent and for equivalent parameter settings, the Kumaraswamy Distribution resembles the Beta distribution but has a higher entropy. Moreover, Kumaraswamy Distribution can be written in a differentiable, non-centered parameterization as

$$x \sim (1-u^{1/b})^{1/a}$$
$$u \sim \text{Uniform}(0,1)$$

Hence, sampling from this distribution is easy and also its KL-divergence with a Beta distribution can be obtained in closed form. We further note that the paper also mentions a Gaussian-Probit Parameterization with a logistic function as another possibility.

Having dealt with all the necessities, this model can now be trained using Stochastic Gradient Variational Bayes (Kingma and Welling, 2013), just like a standard Variational Autoencoder is trained but with the above modifications. There is one more small caveat though, the KL-Divergence term of the Kumaraswamy-Beta distribution reveals a Digamma function and an infinite taylor series sum. During implementation, the infinite taylor series sum needs to be approximated properly using required number of terms and the Digamma function's derivative, the polygamma function, is difficult to implement and hence the digamma function is further approximated using a taylor series expansion.

3. Survey of Models for Clustering with Arbitrary Shaped Clusters

The task at hand is to model the latent variables such that there is inherent clustering with the latent variables, and which can effectively generate samples from the real distribution $\mathcal{D}$, represented by a set of observations $\mathbf{X}$. We first present a generic generative story for such models, and then look at various implementations of the generative story.

Since we want the latent variables to be clustered, we can assume a Finite or Infinite Gaussian Mixture Prior over the latent variables, where the infinite case is handled using Dirichlet Process. Therefore, we assume that the mixture weights are sampled either from a Dirichlet Distribution or a Stick-Breaking Process. We represent a generalized prior, say MM which could mean either a Dirichlet or Stick-Breaking Distribution.

Since we are assuming a mixture of gaussians for the prior, a natural prior on the parameters of each gaussian is a Normal Inverse-Wishart Prior. The last step is to sample $\mathbf{x}$, which is done through a non-linear mapping from the latent variables to the parameters of the distribution of $\mathbf{x}$ which we represent by $\mathcal{P}$ and will depend on the domain that $\mathbf{x} \sim \mathcal{D}$ belongs to.

Using these steps, the complete generative story is given in 1.

We first take a look at a model which allows us to have a non-parametric (Infinite Gaussian Mixture) prior on the latent variables, however suffers from slow inference and no notion of reconstruction.

Algorithm 1: Generative Story for a Clustering Model

1. Draw mixture weights $\boldsymbol{\pi} \sim \mathrm{MM}(\alpha)$
2. For each component $k = 1 \dots K$
   (a) Draw gaussian parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \sim \mathrm{NIW}(\boldsymbol{\lambda})$
3. Draw model parameters $\boldsymbol{\theta} \sim \mathbb{P}[\boldsymbol{\theta}]$
4. For each observation $n = 1 \dots N$
   (a) Draw latent cluster assignment $c_n \sim \mathrm{Multinoulli}(\boldsymbol{\pi})$
   (b) Draw latent coordinates $\mathbf{z}_n \sim \mathcal{N}\left(\boldsymbol{\mu}_{c_n}, \boldsymbol{\Sigma}_{c_n}\right)$
   (c) Draw sample $\mathbf{x}_n \sim \mathcal{P}(\mathbf{x} \,|\, f(\mathbf{z}_n; \boldsymbol{\theta}), \boldsymbol{\gamma})$

The given generative story is a plausible generative model for any data as it only assumes that even though data exists in a complex manifold there exists a disentangled simpler representation of the data that explains it well, which is often true, even for real-world data. The question now boils down to constructing expressive variational distributions and performing efficient inference such that the data is most expressively represented in a latent space.

3.1. Infinite Warped Mixture Model

Following the generic story (Algorithm), the Infinite Warped Mixture Model (IWMM, Iwata et al., 2012) assumes that the observations are sampled has coordinates in a latent space and was generated through some non-linear functional mapping from the latent space to the data space.

To tackle the problem of manifold clustering for arbitrary shaped clusters, the infinite Warped Mixture Model (iWMM) (Lawrence, 2004) assumes that the latent coordinates are generated from a Dirichlet Process Mixture Model and these coordinates are mapped to the observations in data space through some complex non-linear function which is modeled using a Gaussian Process.

As mentioned earlier, we use the stick-breaking process to generate mixture weights for a Dirichlet process with parameter $\boldsymbol{\alpha}$. Also, since the non-linear mapping is defined by a function sampled from a Gaussian Process, we need to modify the generative story where instead of sampling $\boldsymbol{\theta}$, we directly sample f from a gaussian process with a defined mean and a kernel function.

Since the model uses Gaussian Process and Dirichlet Process to jointly learn latent representations and arbitrary shaped clusters, the inference scheme in the procedure is non exact and non trivial. Inference in accomplished through a Sampling based approximate inference scheme, the general structure of which is:

1. For each observation n $= 1, ..., N$, sample the cluster assignment z_n by Collapsed Gibbs Sampling
2. Sample latent coordinates X and kernel parameters using Hybrid Monte Carlo

Thus, by constructing a latent manifold on which the data exists and learning a complex non-linear function, the model is able to both accurately uncover clusters and estimate density of the data even when the shapes of the clusters are very varied and non-Gaussian like. The disadvantage of this

problem, though, is that since the model uses Gaussian process and Sampling scheme for inference, the complexity of the model increases with the amount of data we have, and thus inference is performed with the comlpexity of each iteration being $\mathcal{O}(N^3)$. Thus, though the model is very powerful, it doesn't scale well and is slow to train.

The next models we discuss aim to solve this problem using VAEs, which allow for faster inference without losing on model complexity.

3.2. InfoGAN

InfoGAN (Chen et al., 2016) is an information-theoretic extension to Generative Adversarial Network (GAN) (Goodfellow et al., 2014), which is a type of implicit-density generative model. A vanilla GAN works by having two networks, a generator network G and a discriminator network D and the model optimization is modelled by a game between the generator and the discriminator. To be more precise, G maps some latent distribution like $\mathcal{N}(0, I)$ to the data distribution and D has the task of discriminating between the true data distribution and the data distribution to which G maps the latent distribution.

This is achieved by sampling from the latent distribution and the true data distribution and then using G to get the generated distribution and D to discriminate between the two. The optimization works in the way that G has to generate samples that can fool D while D has to learn to discriminate between the generated points and true points. This sort of game-theoretic objective has an equilibrium governed by the principle of Nash Equilibrium. Thus, D and G play the two-player minimax game with the value function:

$$\min_G \max_D V(G, D) \quad = \quad \mathop{\mathbb{E}}_{\mathbf{x} \sim p_{\text{data}}} \Big[\log D(\mathbf{x}) + \mathop{\mathbb{E}}_{\mathbf{z} \sim p_z} \left[\log \left(1 - D(G(\mathbf{z})) \right) \right] \Big]$$

Though this vanilla version of GAN is able to generate very realistic looking samples, it cannot be used to cluster the data. However, the information theoretic extension InfoGAN improves upon the vanilla GAN by using a latent code $\mathbf{z}$ that can be decomposed into an uncompressible noise and a latent code $\mathbf{c}$ which targets the salient structured semantic features of the data distribution. The InfoGAN works by maximizing the mutual information between the generated data and the latent code $\mathbf{c}$. This information theoretic regularization maintains high mutual information between the latent code $\mathbf{c}$ and the generator distribution. That is, $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$ should be high. Note that $I(\,\cdot\,)$ is the difference of the two entropy terms:

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

This modifies the objective value function, and the new value function could be seen to be the following:

$$\mathrm{V}_{new}(G, D) = \mathrm{V}(G, D) - \lambda I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$$

Denote the true posterior as $P(\mathbf{c}|\mathbf{x})$ and the approximating auxiliary distribution as $Q(\mathbf{c}|\mathbf{x})$. Since calculating the mutual information $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$ requires access to the posterior distribution, the paper aims to maximize a lower bound of the mutual information, which has been obtained to be the following:

$$\begin{aligned} I(\mathbf{c}; G(\mathbf{z}, \mathbf{c})) \quad &\geq \quad \mathop{\mathbb{E}}_{\mathbf{x} \sim G(\mathbf{z}, \mathbf{c})} \left[\mathop{\mathbb{E}}_{\mathbf{c}' \sim P(\mathbf{c}|\mathbf{x})} \left[log\, Q(\mathbf{c}'|\mathbf{x}) \right] \right] + H(\mathbf{c}) \\ &= \quad L_1(G, Q) \end{aligned}$$

Note that $L_1(G, Q)$ can be maximized w.r.t Q directly and w.r.t. G via the reparameterization trick. Hence, the final InfoGAN is defined as a minimax game with a variational regularization of the mutual information with the hyperparameter λ as:

$$\min_{G,Q} \max_{D} V_{\text{InfoGAN}}(G, D, Q) \quad = \quad V(D, G) - \lambda L_1(G, Q)$$

With this objective, by using a latent code that follows a Categorical (Multinoulli) distribution, we can obtain an implicit clustering on the data as well as a good generative model.

3.3. Structured Variational Autoencoder

Though the iWMM model is very efficient at uncovering the number of clusters and the cluster shapes, it does not scale well to large amounts of data. To solve this issue, GMM-SVAE (Johnson et al., 2016) defines the required non-linear functional mapping through a neural network, which have been proved to be universal function approximators. The model takes advantage of the fact that Variational Autoencoders work well in problems of density estimation and hence, on combining VAEs with GMM, the model is able to uncover arbitrary shaped clusters.

This model follows the exact generative story presented in Algorithm 1. The prior on the latent variables is given by a finite mixture model, and therefore the MM prior defines a Dirichlet prior for SVAE

The model performs inference by using recognition networks to produce local evidence potentials which are then incorporated in the general graphical model inference algorithm that sits on top. It uses a conditional random field (CRF) variational family and learns recognition networks that output conjugate graphical model potentials instead of complete variational distribution's parameters. These potentials are then used in the graphical model inference algorithms in place of non-conjugate likelihoods.

3.3.1. Inference

The inference scheme used in the model is fairly complex but we try to give the formal structure of the inference. We use different notation from the original paper so that the notation is consistent throughout the report.

We define a general conjugate exponential family model having global parameters $\boldsymbol{\gamma}$ and local latent variables $\mathbf{z} = \{\mathbf{z}_n\}_{n=1}^N$. We define $\mathbb{P}\left[\,\mathbf{z}\,\middle|\,\boldsymbol{\gamma}\,\right]$ as exponential family and $\mathbb{P}\left[\,\boldsymbol{\gamma}\,\right]$ as its conjugate prior. Hence, we have

$$\begin{aligned}
\mathbb{P}\left[\,\boldsymbol{\gamma}\,\right] &= \exp\left(\left\langle \eta_{\boldsymbol{\gamma}}^0,\ t_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})\right\rangle - \log Z_{\boldsymbol{\gamma}}\left(\eta_{\boldsymbol{\gamma}}^0\right)\right) \\
\mathbb{P}\left[\,x\,\middle|\,\boldsymbol{\gamma}\,\right] &= \exp\left(\left\langle \eta_z^0(\boldsymbol{\gamma}),\ t_z(\mathbf{z})\right\rangle - \log Z_z\left(\eta_x^0(\boldsymbol{\gamma})\right)\right) \\
&= \exp\left(\left\langle t_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}),\ [\,t_z(\mathbf{z}), 1\,]\right\rangle\right)
\end{aligned}$$

where the last equality is written by leveraging exponential family conjugacy structure. Now we model $\mathcal{P}\left(\mathbf{x}\,\middle|\,\mathbf{z}, \boldsymbol{\theta}\right)$ by a general distribution (e.g. Neural Network) and have the exponential family prior $\mathbb{P}\left[\,\boldsymbol{\theta}\,\right]$ on $\boldsymbol{\theta}$.

Now for inference, we consider a general mean field assumption with variational distribution

$\mathcal{Q}(\boldsymbol{\gamma})\mathcal{Q}(\gamma)\mathcal{Q}(x)$. The variational inference objective reduces to maximizing the following ELBO:

$$\mathcal{L}\left[\,\mathcal{Q}(\boldsymbol{\gamma})\mathcal{Q}(\boldsymbol{\theta})\mathcal{Q}(x)\,\right] \quad = \quad \mathop{\mathbb{E}}_{\mathcal{Q}(\boldsymbol{\gamma})\mathcal{Q}(\boldsymbol{\theta})\mathcal{Q}(x)}\left[\log\left(\frac{\mathbb{P}\left[\,\boldsymbol{\gamma}\,\right]\mathbb{P}\left[\,\mathbf{z}\,\middle|\,\boldsymbol{\gamma}\,\right]\mathbb{P}\left[\,\mathbf{x}\,\middle|\,\mathbf{z},\boldsymbol{\theta}\,\right]}{\mathcal{Q}(\boldsymbol{\gamma})\mathcal{Q}(\boldsymbol{\theta})\mathcal{Q}(x)}\right)\right]$$

Now, we take $\mathcal{Q}(\boldsymbol{\gamma})$ to be in the same exponential family as the prior $\mathbb{P}\left[\,\boldsymbol{\gamma}\,\right]$ and we denote its natural parameters by $\eta_{\boldsymbol{\gamma}}$. We restrict $\mathcal{Q}(\boldsymbol{\theta})$ to be in the same exponential family as $\mathbb{P}\left[\,\boldsymbol{\theta}\,\right]$ with natural parameters $\eta_{\boldsymbol{\theta}}$. We restrict $\mathcal{Q}(x)$ to be in the same exponential family as $\mathbb{P}\left[\,\mathbf{z}\,\middle|\,\boldsymbol{\gamma}\,\right]$ with natural parameters η_z.

Having done this, we now present the novelty in the inference scheme followed by the authors. They chose the variational parameter η_x as a function of the parameters $\eta_{\boldsymbol{\gamma}}$ and $\eta_{\boldsymbol{\theta}}$. They claim that a natural choice is to have η_x as the local partial optimizer of $\mathcal{L}$. However due to lack of conjugacy structure, finding the local optimizer would be computationally expensive. Hence instead, η_x is chosen as the local partial optimizer of a surrogate objective $\hat{\mathcal{L}}$ which has conjugacy structure, given by:

$$\hat{\mathcal{L}}(\eta_{\boldsymbol{\gamma}},\eta_z,\boldsymbol{\phi}) = \mathop{\mathbb{E}}_{\mathcal{Q}(\boldsymbol{\gamma})\mathcal{Q}(\mathbf{z})}\left[\log\frac{\mathbb{P}\left[\,\boldsymbol{\gamma}\,\right]\mathbb{P}\left[\,\mathbf{z}\,\middle|\,\boldsymbol{\gamma}\,\right]\exp\left(\psi(\mathbf{z};\mathbf{x},\boldsymbol{\phi})\right)}{\mathcal{Q}(\boldsymbol{\gamma})\mathcal{Q}(\mathbf{z})}\right]$$

$$\psi(\mathbf{z};\mathbf{x},\boldsymbol{\phi}) \quad = \quad \langle r(\mathbf{z};\boldsymbol{\phi}),\ t_z(\mathbf{z})\rangle$$

where $\{r(\mathbf{z};\boldsymbol{\phi})\}$ is a parameterized class of functions obtained using a recognition model. Note that the potentials $\psi(\mathbf{z};vx,\phi)$ have a conjugate form to $\mathbb{P}\left[\,\mathbf{z}\,\middle|\,\boldsymbol{\gamma}\,\right]$. Hence, we obtain the variational distribution $\mathbf{Q}(\mathbf{z})$ and the variational parameter η_z as:

$$\eta_z^*(\eta_{\boldsymbol{\gamma}},\phi) \quad = \quad \arg\max_{\eta_z}\ \hat{\mathcal{L}}(\eta_{\boldsymbol{\gamma}},\eta_z,\boldsymbol{\phi})$$

$$\mathcal{Q}^*(\mathbf{z}) \quad = \quad \exp\Big(\big\langle\eta_z^*(\eta_{\boldsymbol{\gamma}},\phi),\ t_z(\mathbf{z})\big\rangle - \log Z_z(\eta_z^*(\eta_{\boldsymbol{\gamma}},\boldsymbol{\phi}))\Big)$$

Having obtained the local optimum value of η_z, we can update the other variational parameters $\eta_{\boldsymbol{\gamma}}$ and $\eta_{\boldsymbol{\theta}}$ using gradients of the following expression of $\mathcal{L}_{\text{SVAE}}$ (we have to maximize it):

$$\mathcal{L}_{\text{SVAE}} \quad = \quad \mathop{\mathbb{E}}_{\mathcal{Q}(\boldsymbol{\gamma})\mathcal{Q}(\boldsymbol{\theta})\mathcal{Q}^*(\mathbf{z})}\left[\log\frac{\mathbb{P}\left[\,\boldsymbol{\gamma}\,\right]\mathbb{P}\left[\,\mathbf{z}\,\middle|\,\boldsymbol{\gamma}\,\right]\mathbb{P}\left[\,\mathbf{x}\,\middle|\,\mathbf{z},\boldsymbol{\theta}\,\right]}{\mathcal{Q}(\boldsymbol{\gamma})\mathcal{Q}(\boldsymbol{\theta})\mathcal{Q}^*(\mathbf{z})}\right]$$

$$= \quad \mathop{\mathbb{E}}_{\mathcal{Q}(\boldsymbol{\theta})\mathcal{Q}^*(\mathbf{z})}\left[\log\mathcal{P}(\mathbf{x}\,|\,\mathbf{z},\boldsymbol{\theta})\right] - \mathrm{KL}\left(\mathcal{Q}(\boldsymbol{\gamma})\mathcal{Q}^*(\mathbf{z})\,||\,\mathbb{P}\left[\,\boldsymbol{\gamma},\mathbf{z}\,\right]\right) - \mathrm{KL}\left(\mathcal{Q}(\boldsymbol{\theta})\,||\,\mathbb{P}\left[\,\boldsymbol{\theta}\,\right]\right)$$

This can be obtained using Reparameterization trick and automatic differentiation. Note that we can obtain natural gradients in this case.

The disadvantage of the model is that it has been worked on only in the parametric setting where the number of clusters present in the data is provided to the network. Hence, though the model efficiently learns the shapes of the clusters, it doesn't give us the number of clusters present in the data. At the same time, it is worth pointing out that this model, through the use of Neural Networks, scales up to increasing amounts of data very efficiently.

3.4. Structured Inference Network

Structured Inference Network (SIN) (Lin et al., 2018) is a very recent work that aims to combine Probabilistic Graphical Models (PGMs) with Deep Neural Networks (DNNs). It builds up on the

work of Structured Variational Autoencoders and extends it by simplifying and generalizing the methodology of SVAE (Johnson et al., 2016). The main idea behind this model is that it incorporates the graphical model structure in the inference network of variational autoencoders, thereby giving us both the flexibility of Neural Networks along with an interpretable and powerful graphical model sitting behind it.

Assuming we have data $\mathbf{x}_1, ..., \mathbf{x}_N$ and we are modelling the data using latent variables $\mathbf{z}_1, ..., \mathbf{z}_N$ that have a probabilistic graphical model over them. Thus, the SIN uses the following joint distribution:

$$p(\mathbf{x}, \mathbf{z}, \theta) = \underbrace{[\prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\theta}_{NN})]}_{\text{DNN}} \underbrace{[p(\mathbf{z}|\boldsymbol{\theta}_{PGM})]}_{\text{PGM}} \underbrace{[p(\boldsymbol{\theta}_{PGM})]}_{\text{Hyperprior}}$$

We know that Structured Variational Autoencoders aim to solve similar problems using mean-field variational inference and a two-stage iterative procedure. The advantage is that if the variational family over $\mathbf{z}$ is chosen conjugate with the PGM structure sitting on $\mathbf{z}$, inference can be performed. However, the inference is equivalent to an implicitly constrained optimization which is difficult to solve. Moreover, this inference procedure does not work if the prior contains non-conjugate terms. SIN can, however, preserve some structure even if the graphical model contains non-conjugate factors.

Structured Inference Networks work by having a variational distribution with a certain structure. From below, we assume $\boldsymbol{\theta}$ to be deterministic, that is, it does not have any prior. This proposed structure consists of two types of factors, and can be written as:

$$q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) = \frac{1}{\mathcal{Z}(\boldsymbol{\phi})} \underbrace{[\prod_{n=1}^{N} q(\mathbf{z}_n|\mathbf{f}_{\phi_{NN}}(\mathbf{x}_n))]}_{\text{DNN Factor}} \underbrace{[q(\mathbf{z}|\boldsymbol{\phi}_{PGM})]}_{\text{PGM Factor}}$$

Note that the role of the DNN term is to enable flexibility while the role of the PGM term is to incorporate model structure. To perform efficient inference, there are certain conditions that need to be met. Firstly, the normalizing constant $\mathcal{Z}(\boldsymbol{\phi})$ is easy to evaluate and differentiate. Secondly, we can draw samples from the distribution, that is $\mathbf{z}^*(\boldsymbol{\phi}) \sim q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$. Moreover, an additional desirable though not required feature is the ability to compute the gradient of $\mathbf{z}^*(\boldsymbol{\phi})$ using the reparameterization trick.

Now, if these conditions are met, a stochastic gradient of the lower bound can be obtained in a way similar to VAEs (Kingma and Welling, 2013). The variational lower bound can be written as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \mathbb{E}_q[log\, \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})}] \\ &= \underbrace{\sum_{n=1}^{N} \mathbb{E}_q[log\, \frac{p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\theta}_{NN})}{p(\mathbf{z}_n|\mathbf{f}_{\phi_{NN}}(\mathbf{x}_n))}]}_{\text{Similar to VAE}} + \underbrace{\mathbb{E}_q[log\, p(\mathbf{z}|\boldsymbol{\theta}_{PGM})]}_{\text{Due to Structured Prior}} \underbrace{- \mathbb{E}_q[log\, q(\mathbf{z}|\boldsymbol{\phi}_{PGM})] + log\, \mathcal{Z}(\boldsymbol{\phi})}_{\text{Due to PGM term in inference}}\end{aligned}$$

Now, for our problem, the PGM required is a Gaussian Mixture Model. Hence, the model's PGM structure and the variational distribution can be defined as follows for efficient inference:

$$p(\mathbf{z}|\boldsymbol{\theta}_{PGM}) = \prod_{n=1}^{N} [\sum_{k=1}^{K} \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k]$$

$$q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) = \frac{1}{\mathcal{Z}(\boldsymbol{\phi})} \prod_{n=1}^{N} \underbrace{[\mathcal{N}(\mathbf{z}_n|\mathbf{m}_n, \mathbf{V}_n)]}_{\text{DNN Factor}} \underbrace{[\sum_{k=1}^{K} \mathcal{N}(\mathbf{z}_n|\bar{\boldsymbol{\mu}}_k, \bar{\boldsymbol{\Sigma}}_k)\bar{\pi}_k]}_{\text{GMM Factor}}$$

Having this setting of the model satisfies the conditions required for the Structured Inference Network and thus structured, amortized, efficient inference can be performed on the model. Thus we see that SIN is a very powerful and recent approach that combines PGM structure with Neural Networks very efficiently.

3.5. Variational Deep Embedding

Variational Deep Embedding (VaDE) was proposed by Jiang et al. (2016). It is a simplistic model than the previous models discussed, and greatly simplifies the generic generative story by assuming a uniform prior on the cluster means, variances, and misture proportions. Also, it assumes a uniform prior on the model parameters $\boldsymbol{\theta}$.

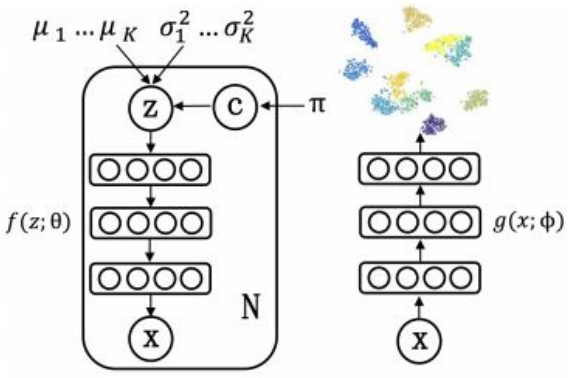

Figure 2: (Left) Plate Notation for VaDE and (Right) an encoder network $g(\mathbf{x};\phi)$

Assuming whether the observations $\mathbf{x}$ is binary or real-valued we compute $\mu_x = f(\mathbf{z},\boldsymbol{\theta})$ and choose a sample $\mathbf{x} \sim Ber(\mu_x)$ or compute $[\mu_x; \log(\sigma)_x^2] = f(\mathbf{z},\boldsymbol{\theta})$ and choose a sample $\mathbf{x} \sim \mathcal{N}(\mu_x, \sigma_x^2\mathbf{I})$. From the generative story, the joint probability can be written as

$$\mathbb{P}\left[\mathbf{x},\mathbf{z},c\right] \quad = \quad \mathbb{P}\left[\mathbf{x}\,\middle|\,\mathbf{z},\boldsymbol{\theta}\right]\mathbb{P}\left[\mathbf{z}\,\middle|\,c\right]\mathbb{P}\left[c\right]$$

An instance of VaDE is tuned to maximize the likelihood of the given data points. Given the generative process described above we have

$$\begin{aligned}\log \mathbb{P}\left[\mathbf{x}\right] \quad &= \quad \log\left(\int_{\mathbf{z}}\sum_{c}\mathbb{P}\left[\mathbf{x},\mathbf{z},c\right]\,\mathrm{d}\mathbf{z}\right)\\ &\geq \quad \mathop{\mathbb{E}}_{\mathcal{Q}(\mathbf{z},c\,|\,x)}\left[\log\frac{\mathbb{P}\left[\mathbf{x},\mathbf{z},c\right]}{\mathcal{Q}(\mathbf{z},c\,|\,\mathbf{x})}\right] \quad = \quad \mathcal{L}_{\text{ELBO}}\end{aligned}$$

where $\mathcal{Q}(\mathbf{z},c\,|\,\mathbf{x})$ is the proposal distribution.

Mean field is assumed on the proposal distribution, giving:

$$\mathcal{Q}(\mathbf{z},c\,|\,\mathbf{x}) \quad = \quad \mathcal{Q}(\mathbf{z}\,|\,\mathbf{x})\mathcal{Q}(c\,|\,\mathbf{x})$$

Similar to VAE, the distribution $\mathcal{Q}(\mathbf{z}\,|\,\mathbf{x})$ is modelled using a neural network g as follows

$$\begin{aligned}\tilde{\boldsymbol{\mu}}, \log\tilde{\boldsymbol{\sigma}}^2 \quad &= \quad g(\mathbf{x},\boldsymbol{\phi})\\ \mathcal{Q}(\mathbf{z}\,|\,\mathbf{x}) \quad &= \quad \mathcal{N}\left(\mathbf{z}\,|\,\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\sigma}}^2\mathbf{I}\right).\end{aligned}$$

Further, a decoder model is added giving the following:

$$\mathbb{P}\left[\,\mathbf{x}\,\middle|\,\mathbf{z}\,\right] \quad = \quad \mathbb{P}\left[\,\mathbf{x}\,\middle|\,f(\mathbf{z},\boldsymbol{\theta})\,\right]$$

The authors use an interesting approach to "approximating" the proposal distribution $\mathcal{Q}(c\,|\,\mathbf{x})$, which although looks absurd, in practice works surprisingly well. First, one can realize, the ELBO can be re-written in the following manner,

$$\begin{aligned}
\mathcal{L}_{\text{ELBO}} &= \mathop{\mathbb{E}}_{\mathcal{Q}(\mathbf{z},c\,|\,\mathbf{x})}\left[\log\frac{\mathbb{P}\,[\,\mathbf{x},\mathbf{z},c\,]}{\mathcal{Q}(\mathbf{z},c\,|\,\mathbf{x})}\right] \\
&= \int_{\mathbf{z}}\sum_{c}\mathcal{Q}(c\,|\,\mathbf{x})\mathcal{Q}(\mathbf{z}\,|\,\mathbf{x})\left\{\log\frac{\mathbb{P}\left[\,\mathbf{x}\,\middle|\,\mathbf{z}\,\right]\mathbb{P}\,[\,\mathbf{z}\,]}{\mathcal{Q}(\mathbf{z}\,|\,\mathbf{x})}+\log\frac{\mathbb{P}\left[\,c\,\middle|\,\mathbf{z}\,\right]}{\mathcal{Q}(c\,|\,\mathbf{x})}\right\} \\
&= \int_{\mathbf{z}}\mathcal{Q}(\mathbf{z}\,|\,\mathbf{x})\log\frac{\mathbb{P}\left[\,\mathbf{x}\,\middle|\,\mathbf{z}\,\right]\mathbb{P}\,[\,\mathbf{z}\,]}{\mathcal{Q}(\mathbf{z}\,|\,\mathbf{x})}\,\mathrm{d}\mathbf{z}-\int_{\mathbf{z}}\mathcal{Q}(\mathbf{z}\,|\,\mathbf{x})\mathrm{KL}\left(\mathcal{Q}(c\,|\,\mathbf{x})\,||\,\mathbb{P}\left[\,c\,\middle|\,\mathbf{z}\,\right]\right)\,\mathrm{d}\mathbf{z}
\end{aligned}$$

In order to maximize the above ELBO, it seems we need to minimize the aberage KL Divergence over the latent variables, when sampled using the posterior. *Jiang et al.* approximated this simply using only one sample of $\mathbf{z}$ and therefore, we can write

$$\mathcal{Q}(c\,|\,\mathbf{x}) \quad = \quad \mathbb{P}\left[\,c\,\middle|\,\mathbf{z}=\hat{\mathbf{z}}\,\right] \quad = \quad \frac{\mathbb{P}\,[\,c\,]\,\mathbb{P}\left[\,\mathbf{z}=\hat{\mathbf{z}}\,\middle|\,c\,\right]}{\sum_{c'=1}^{K}\mathbb{P}\,[\,c'\,]\,\mathbb{P}\left[\,\mathbf{z}=\hat{\mathbf{z}}\,\middle|\,c'\,\right]}$$

where $\hat{\mathbf{z}}$ is a sample from the posterior proposal $\mathcal{Q}(\mathbf{z}\,|\,\mathbf{x})$.

Given the simplicity of the model, we performed some experiments on it. We implemented the complete model on Tensorflow from scratch, and performed experiments on a couple of datasets. This is discussed in detail in the following section.

3.6. Experiments on Clustering with VaDE

3.6.1. MNIST Dataset

We firstly performed a set of experiments on the MNIST Dataset, which is a collection of hand-written digits providing 55000 training examples and 5000 test examples, with each digit an image of dimensions 28x28. We further binarized the images with a threshold of 0.5.

We performed unsupervised clustering on the model using VaDE, and compared the results with a plugin clustering model, where a latent representation is learned using a VAE and a GMM is learned over the latent variables for the observations.

We first describe the training process and then show the results of our experiments. Since we want to maximize the ELBO defined previously, our loss is simply the negative of the ELBO. Also, since we are working with binary data, the likelihood is given by a multivariate bernoulli.

Further, the expectation with respect to the latent variables is approximated using Monte Carlo estimation with a single sample point, generated from the approximate posterior of the latent variables, $\mathcal{Q}(\mathbf{z}\,|\,\mathbf{x})$. Therefore, we can write the final term for the ELBO as

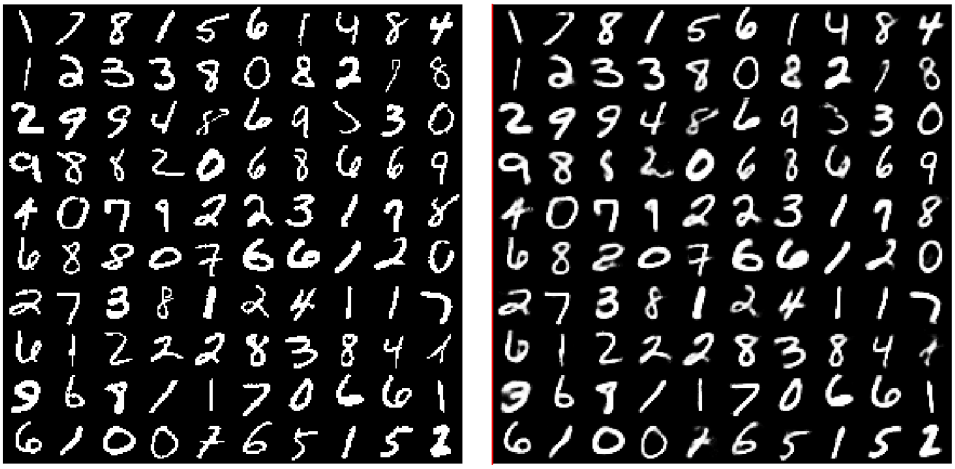

Figure 3: Plug-in Clustering – Left: Original digits, Right: Reconstructed Digits

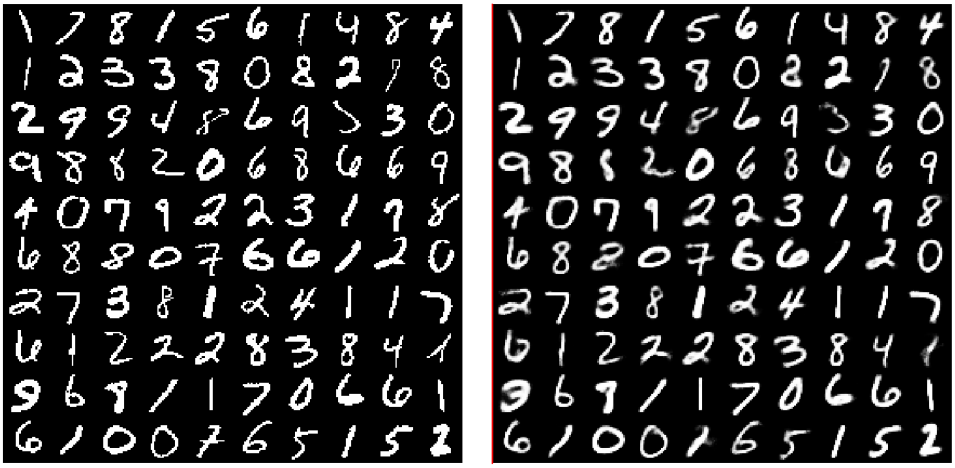

Figure 4: VaDE Clustering – Left: Original digits, Right: Reconstructed Digits

$$
\begin{aligned}
\mathcal{L}_{\text{ELBO}} &= \frac{1}{N}\sum_{n=1}^{N}\sum_{d=1}^{D}\left\{\mathbf{x}_d \log\left(\boldsymbol{\mu}_x^{(n)}|_d\right)\sum_{d=1}^{D}(1-\mathbf{x}_d)\log\left(1-\boldsymbol{\mu}_x^{(n)}|_d\right)\right\} \\
&- \frac{1}{2}\sum_{c=1}^{K}\gamma_c\sum_{j=1}^{J}\left\{\log\boldsymbol{\sigma}_c^2|_j + \frac{\tilde{\boldsymbol{\sigma}}^2|_j}{\boldsymbol{\sigma}_c^2|_j} + \frac{\left(\tilde{\boldsymbol{\mu}}|_j - \boldsymbol{\mu}_c|_j\right)^2}{\boldsymbol{\sigma}_c^2|_j}\right\} \\
&+ \sum_{c=1}^{K}\gamma_c \log\frac{\pi_c}{\gamma_c} + \frac{1}{2}\sum_{j=1}^{J}\left(1+\log\tilde{\boldsymbol{\sigma}}^2|_j\right)
\end{aligned}
$$

The loss / negative ELBO is minimized using an Adam optimizer. There was another interesting observation made by the authors, which we as well found in our experiments, that using Adam optimizer works on even the GMM parameters, including the mixture proportions, inspite of the fact that we are performing unconstrained optimization, when the GMM mixture components are clearly constrained.

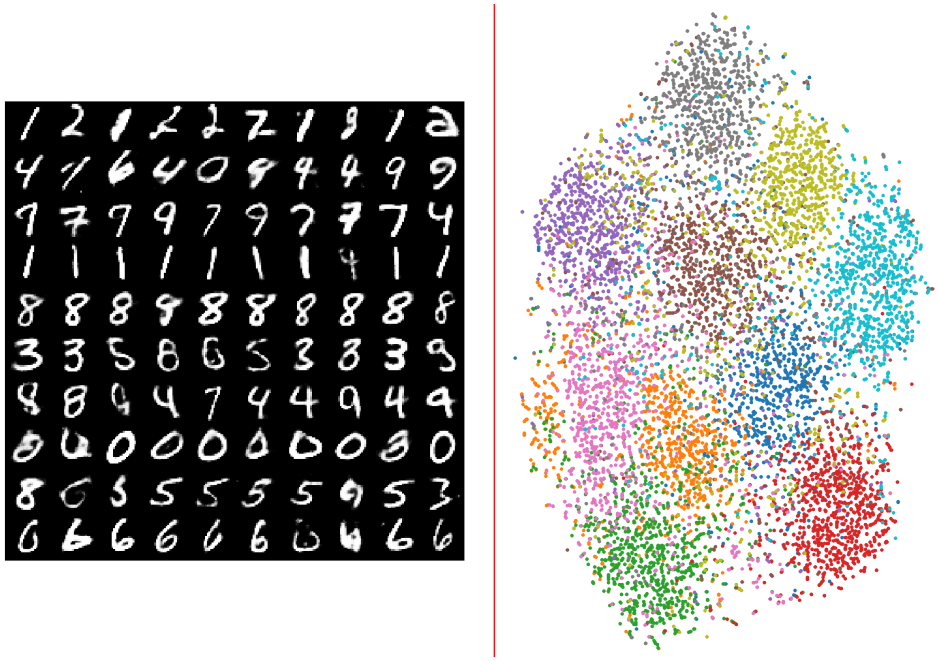

Figure 5: Plug-in Clustering – Left: Sampled Digits with row representing cluster, Right: TSNE Plot

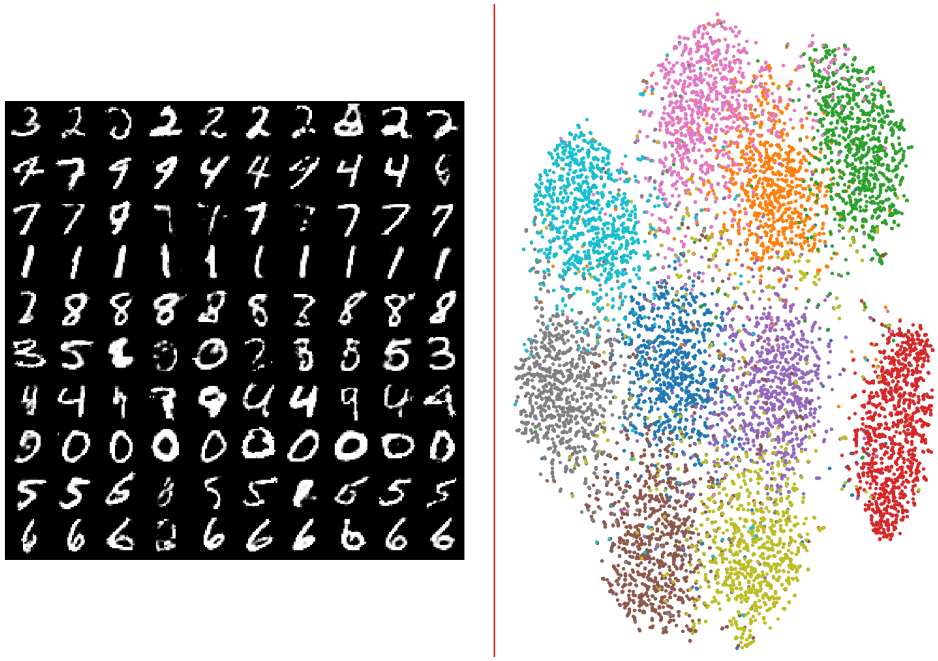

Figure 6: VaDE Clustering – Left: Sampled Digits with row representing cluster, Right: TSNE Plot

We first compare the reconstruction of the VAE for the plug-in case and the VaDE Case. The reconstruction plots are given in Figures 3 and 4 respectively for plug-in and VaDE case. One can see that the the reconstruction in case of VaDE is slighly lesser, however there is minimal difference and therefore conclude that inspite of adding a different prior to the latent variables, the reconstruction power of the VAE is not lost.

We further compare the effeciency of the clustering algorithms, which can be seen in Figures 5

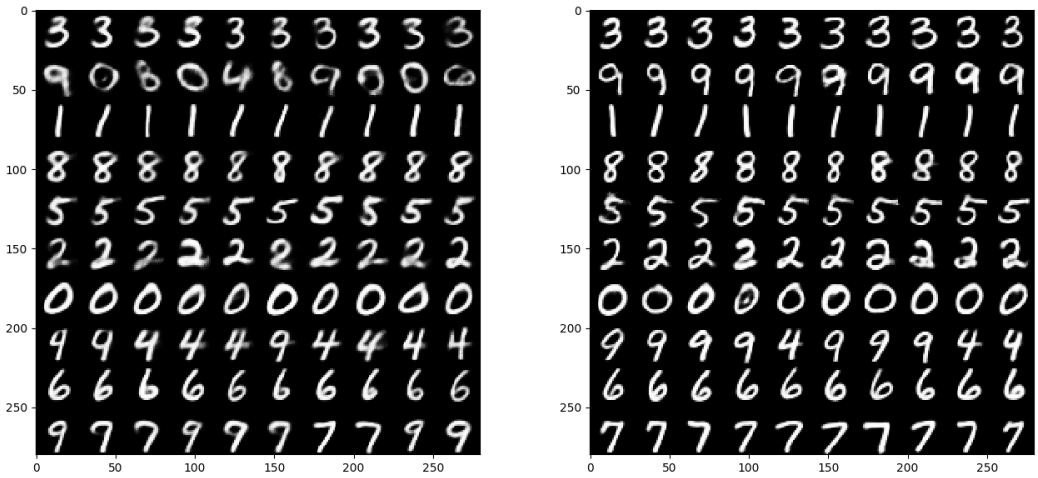

Figure 7: Left: Samples with Thresholding for plug-in model and Right: Samples with Thresholding for VaDE model

and 6 respectively for plug-in and VaDE models. It is clear that there is much lesser ambuigity in clustering when using VaDE model, and also the TSNE plot clearly shows there are larger gaps between clusters, and therefore, we can conclude VaDE outperforms the direct plug-in clustering model.

Remark. The lower quality digits in the sampled plot are due to no thresholding on the predictive posterior, and would be greatly improved if some thresholding is added. Also, the lower quality in case of VaDE is more profound here, however proper thresholding and longer training runs would get rid of such loss. We show, in Figure 7, samples generated by controlling the variance while sampling.

We thus see that by controlling the predictive variance, we get very good results where the clusters are representative of different digits. We note that the model still struggles with some disambiguations (e.g. between 4 and 9). Moreover, the model is quite sensitive to initializaiton as some runs lead to perfect disambiguations between the classes while some runs result in a particular class split into two different clusters while some classes are merged into one cluster.

We further look at the usability of VaDE for clustering where the observations are real valued.

3.6.2. Spiral Dataset

This is a toy dataset on which we run the VaDE model in order to analyze the performance of VaDE. Since the observations are real valued in this case, we can rewrite the loss, exactly as done in previous case, but with a gaussian likelihood, and for simplicity's sake, with a known variance.

In Figure 8, we have shown the training data after clustering, both using plug-in model (left) and VaDE model (right). It can be seen that there is better clustering in the case of VaDE model where the clusters seem to be pure, which is not the case with plug-in model. The superiority of VaDE in this case is even more clear when one looks at the TSNE plots in Figure 9.

With this, we conclude the models we surveyed for our report. In the next section, we present a model we propose, combining the power of VAEs and non-parametric models, and apply this for the purpose of

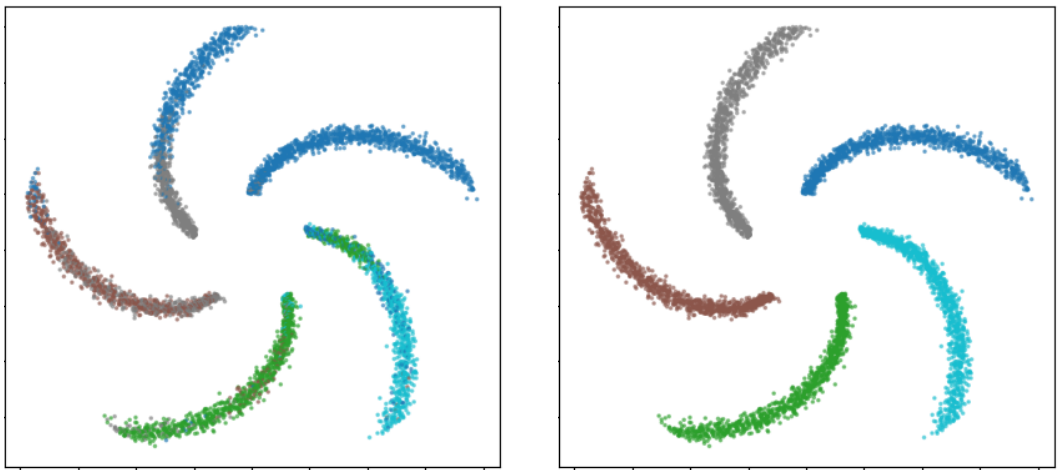

Figure 8: Left: Clustering with plug-in model, Right: Clustering with VaDE model

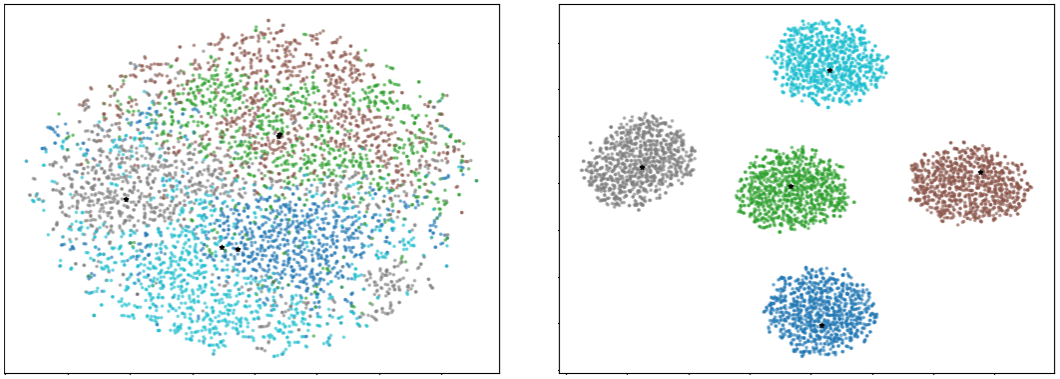

Figure 9: Left: TSNE of clusters with plug-in model, Right: TSNE of clusters with VaDE model

clustering, same as above.

4. Relevant Extensions

4.1. DPMM-SVAE

Our aim is to come up with a model that not only recovers the complex shaped clusters in the data, but also the number of clusters present. Moreover, it should scale well to large amounts of data. To solve this problem, we propose a model similar to the GMM-SVAE model described above. However, instead of having a finite mixture model at the prior like the GMM-SVAE model, we propose to have a Dirichlet Process Mixture Model sitting at the top. This allows creating of new clusters as more and more data is seen.

To achieve this aim, we describe the generative story as given in

Note that while computationally creating an infinite number of clusters is not possible, we solve this problem by using a $Gem(\cdot)$ distribution which describes a stick breaking scheme, where we generate new clusters on demand. Thus, our proposed model is theoretically able to capture both the number of clusters in the data as well as their arbitrary shapes, while being able to scale to large data. Thus,

Algorithm 2: Generative Story for DPMM-SVAE

1. Draw mixture weights $\boldsymbol{\pi} \sim Gem(\boldsymbol{\alpha})$
2. For each component $k = 1 \dots \infty$
   (a) Draw $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \sim NIW(\boldsymbol{\lambda})$
3. Draw parameters of the encoder and decoder model $\boldsymbol{\theta} \sim \mathbb{P}[\boldsymbol{\theta}]\text{kd}$
4. For each obsevation n $= 1 \dots N$
   (a) Draw latent cluster assignment $c_n \sim \text{Multinoulli}(\boldsymbol{\pi})$
   (b) Draw latent coordinates $\mathbf{x}_n \sim \mathcal{N}\left(\boldsymbol{\mu}_{c_n}, \boldsymbol{\Sigma}_{c_n}^{-1}\right)$
   (c) Draw a data point $\mathbf{y}_n \sim \mathcal{N}(\boldsymbol{\mu}(x_n; \boldsymbol{\theta}), \boldsymbol{\Sigma}(x_n; \boldsymbol{\theta}))$

it has the capacity to, in theory, surpass the existing methods mentioned above. We however were unable to perform inference on this model due to shortage of time, and we leave this model for future work in the subject.

5. Mixture of Experts

Mixture of Experts (MoE) was proposed by Jacobs et al. (1991). MoE is essentially a machine learning technique where multiple learners (referred to as experts) are used to divide the problem space into homogenous regions Masoudnia and Ebrahimpour (2014).

The MoE model essentially has three components Yuksel et al. (2012)

1. several experts - regressors or classifiers
2. a gate that makes soft partitions of the input space based on performance on the experts
3. a probabilistic model to combine the gate and the experts

Very similar to an ensemble, the MoE model allows us to learn simple models on parts of data and separately model a gating network to combine the results of each learner (competitive learning among the experts). The inference of MoE models is generally performed using EM algorithm, VI or Sampling. There are other methods available as well, such as IRLS, GEM, Netwon-Raphson, etc.

An interesting part of the MoE framework is the gating function. Popular examples are Softmax gate, Generative models for gating, etc. However most gating functions do not consider the inherent clustering or divisions among the data space, which could be leveraged to model a gating function which is dependent on the cluster to make the mixture of experts model more prominent where there is such inherent clustering within the input features.

In the following sections, we propose two models which achieve the above, however comment little on the inference, again due to shortage of time.

5.1. Stick Breaking Mixture of Experts

We saw that Stick-Breaking VAE tries to construct a non-parametric latent representation of the data that performs quite well in practice. One of the important things to note in this respect is that the latent representation learned by the model for the data is always a probability vector since it is obtained using a Stick-Breaking scheme. One natural extension of this algorithm is to incorporate a Mixture of Experts type of setting since we already are getting a probability distribution with respect to the data.

Formally, we define the generative story as:

1. Draw $\pi_n \sim \text{Gem}(\eta)$
2. For each component $\text{c} = 1, ..., \infty$
   (a) Draw the expert $expert_c$
3. Draw the observation $y_n \sim \sum_{i=1}^{\infty} \pi_{ni}\, p(y_n | expert_i)$

Note however, that since the inference scheme is similar to the Stick-Breaking VAE model, we obtain $q_\phi(\pi_i|y_i)$ instead from a recognition model such that where ϕ are the parameters of the recognition model. This can formally be seen as:

$$\pi_i = \begin{bmatrix} \pi_{i,1} & \pi_{i,2} & \dots & \pi_{i,K} \end{bmatrix} = \begin{bmatrix} v_{i,1} & v_{i,2}(1 - v_{i,1}) & \dots & \prod_{j=1}^{K-1}(1 - v_{i,j}) \end{bmatrix}$$
$$v_{ij} = \text{Kumaraswamy}(v_{ij} | f_j(y_i; \phi), g_j(y_i; \phi))$$

Note that here K is the truncation factor for variational inference in Dirichlet Process Models as described in Blei's paper.

5.2. Variational Deep Embedding Mixture of Experts

We expand the simple idea of VaDE to a mixture model, where instead of finding modelling the mixture proportions using a softmax gate, we compute the cluster probabilities of each data point and use that as the mixture proportion. The generative story (Algorithm 3) for the same is given similarly as what was given for the clustering model.

Algorithm 3: Generative Story for VaDE-MoE

1. Choose a cluster / expert $c \sim \text{Multinoulli}\,(\boldsymbol{\pi})$
2. Choose a latent variable $\mathbf{z} \sim \mathcal{N}\left(\boldsymbol{\mu}_c, \sigma_c^2 \mathbf{I}\right)$
3. Draw sample $\mathbf{x} \sim \mathcal{P}(\mathbf{x} \,|\, f(\mathbf{z}_n; \boldsymbol{\theta}), \boldsymbol{\gamma})$
4. Draw label $y \sim \mathcal{P}(\mathbf{y} \,|\, g(\mathbf{x}; \boldsymbol{\phi}_c), \boldsymbol{\lambda}_c)$

We give brief steps regarding the inference. Although the inference is similar to that of VaDE, there is one key difference, we propose to model the proposal distribution for the cluster assignments as dependent on both $\mathbf{x}$ and y however do not model the involvement of labels on the proposal for the latent variable $\mathbf{z}$.

Similar to the VaDE case, the encoder is given using a neural network, and therefore there is no difference in this aspect. That is, similar to VaDE, we assume the mean field assumption, and therefore, we write the proposal (which is now also dependent on the labels) $\mathcal{Q}(\mathbf{z}, c \,|\, \mathbf{x}, \mathbf{y}) = \mathcal{Q}(\mathbf{z} \,|\, \mathbf{x})\mathcal{Q}(c \,|\, \mathbf{x}, \mathbf{y})$. The form of $\mathcal{Q}(\mathbf{z} \,|\, \mathbf{x})$ remains exactly the same as in the case of VaDE and therefore we skip those steps here.

For the inference of the second part, *i.e.* $\mathcal{Q}(c \,|\, \mathbf{x}, y)$, following the same steps as we did in the case of VaDE, one can realize that the KL divergence in the second term will be between the distributions $\mathcal{Q}(c \,|\, \mathbf{x}, y)$ and $\mathbb{P}\left[c \,|\, y, \mathbf{z} \right]$, under an integral over $\mathbf{z}$. However, with the same approximation as did the authors of the VaDE paper, we can approximate the proposal $\mathcal{Q}(c \,|\, \mathbf{x}, y)$ as

$$\begin{aligned}
\mathcal{Q}(c \,|\, \mathbf{x}, y) &\approx \mathbb{P}\left[c \,|\, y, \mathbf{z} = \hat{\mathbf{z}} \right] \\
&\propto \int_{\mathbf{x}} \mathbb{P}\left[y \,|\, c, \mathbf{x} \right] \mathbb{P}\left[\mathbf{x} \,|\, \mathbf{z} = \hat{\mathbf{z}} \right] \, \mathrm{d}\mathbf{x} \cdot \mathbb{P}\left[c \,|\, \mathbf{z} = \hat{\mathbf{z}} \right] \\
&\propto \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathbf{x} \,|\, \mathbf{z} = \hat{\mathbf{z}}} \left[\mathbb{P}\left[y \,|\, c, \mathbf{z} \right] \right] \cdot \mathbb{P}\left[c \,|\, \mathbf{z} = \hat{\mathbf{z}} \right]
\end{aligned}$$

The second term is computed as was done in the paper (exact). The first term is an expectation with respect to the the data point $\mathbf{x}$, and can be compute usign Monte Carlo Estimation, where sampling from $\mathbf{x} \,|\, \mathbf{z}$ is simple assuming we have a trained decoder. Since we are performing alternate maximization, we do have a sample of a decoder which can be used to generate the samples as required.

We have, above, briefly outlined the inference of the model, while proposing a method for approximating the proposal distribution. After completing inference, we can use the soft probabilities generated for choosing the cluster (via the encoder) as our mixture proportions for the mixture model during test time, and therefore have a soft mixture of experts model based on the structure of our input variables, as desired.

6. Conclusion

In this project, we surveyed different papers discussing several approaches to generate models for clustering data with arbitrary shapes. We observed distinctive differences in the generative models or inference procedure among these models. The Infinite Warped Mixture Model assumes the latent coordinates generating from a Dirichlet Process Mixture Model which in turn mapped into observation space via a nonlinear function modeled using GP to learn both cluster assignments as well as number of clusters. InfoGAN incorporates a simple modification to the generative adversarial network objective by maximizing the mutual information between a fixed small subset of the GAN's noise variables and the observations to discover highly semantic and meaningful hidden representations so that generative modelling augmented with a mutual information cost helps in learning disentangled representations.

As Opposed to this, Structured Variational Autoencoder, Structured Inference Network and Variational Deep Embeddings models inculcate Neural Networks for nonlinear mapping in parameterized setting and combine VAE to uncover arbitrary shaped clusters using GMM in the latent space. The inference in SVAE is however based on optimizing a surrogate objective and then the true objective, whereas, Structured Inference Network which is a more general form of SVAE, doesn't include any surrogate loss and the whole model is trained together using one loss only (ELBO). Finally in the VaDE model variational distribution is modeled using just decomposition of (latent, cluster) into $\mathcal{Q}$(latent) and $\mathcal{Q}$(cluster), both conditioned on x.

We also proposed an extension to the clustering model combining the strengths of non-parametric models and fast and easy inference of VAEs, namely the DPMM-SVAE model. Finally, we proposed applications of the clustering models as gating networks in mixture of experts, namely Stick Breaking MoE and VaDE-MoE, which allow for inherent arbitrary shaped clusters within the input features to be separated and therefore

inferred separately as a soft mixture of experts.

References

Charles E. Antoniak. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974. ISSN 00905364. URL http://www.jstor.org/stable/2958336.

David M. Blei and Michael I. Jordan. Variational methods for the dirichlet process. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 12–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015439. URL http://doi.acm.org/10.1145/1015330.1015439.

David M Blei, Michael I Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.

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*, pages 2172–2180, 2016.

C. Doersch. Tutorial on Variational Autoencoders. *ArXiv e-prints*, June 2016.

Michael D. Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995. doi: 10.1080/01621459.1995.10476550. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1995.10476550.

Thomas S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2): 209–230, 1973. ISSN 00905364. URL http://www.jstor.org/stable/2958008.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Tomoharu Iwata, David Duvenaud, and Zoubin Ghahramani. warped mixtures for nonparametric cluster shapes. *arxiv*, abs/1206.1846v2, 2012.

R. A. Jacobs, M. I. Jordan, S. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. 1991.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: A generative approach to clustering. *CoRR*, abs/1611.05148, 2016. URL http://arxiv.org/abs/1611.05148.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems 29*, pages 2946–2954. 2016.

D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. pages 329–336, 2004.

Wu Lin, Mohammad Emtiyaz Khan, and Nicolas Hubacher. Variational message passing with structured inference networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyH9lbZAW.

Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, Aug 2014. ISSN 1573-7462. doi: 10.1007/s10462-012-9338-y. URL https://doi.org/10.1007/s10462-012-9338-y.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

Eric Nalisnick and Padhraic Smyth. Deep generative models with stick-breaking priors. *arXiv preprint arXiv:1605.06197*, 2016.

Radford M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000. doi: 10.1080/10618600.2000.10474879. URL `http://amstat.tandfonline.com/doi/abs/10.1080/10618600.2000.10474879`.

S. E. Yuksel, J. N. Wilson, and P. D. Gader. 20 years of mixture of experts. 2012.