Nonparametric Manifold Clustering for Arbitrary Shaped Clusters

Sarthak Mittal 150640
Gurpreet Singh 150259
Debabrata Ghosh 13817226

Abstract

We study a number of existing techniques that allow clustering of data by jointly representing the data in a low dimensional latent form and performing clustering over this latent space. The key idea in such models is that the given data can be generated from a simpler pattern in a possibly lower-dimensional manifold through some non-linear mapping. Using a nonparametric clustering model over a latent space and a good approximation function that maps the latent space to our data space, it is possible to infer both, the number of clusters that the data lies in, and the varied shapes of the clusters in the data. Moreover, we try to come up with a model that not only performs this task efficiently but is also able to scale to large amounts of data.

1. Problem

In this course project, we aim to solve the problem of uncovering both the number of clusters and their arbitrary shapes. Moreover, we aim to propose a model that not only solves the above problem efficiently, but also scales up with the amount of data without requiring massive computational costs. To successfully come up with a solution to the problem, we start with a background reading of the concepts required and the existing work done in this context. Further, we propose the general framework of our model and provide reasons for its probable effectiveness over the previous models.

2. Introduction

Humans are capable of recognizing cluster patterns even when the shapes and sizes of the clusters are immensely varied. The intrinsic shortcomings of K-Means and Gaussian Mixture Models are that the model needs the number of clusters to be specified before-hand and is able to cluster the data in only Gaussian shaped clusters. Though Dirichlet Process Mixture Models alleviate the former problem of the model, it still tries to capture the data as a cluster 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 function.

One way to do 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 or through 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. 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 of an infinite mixture) and then learn a complex non-linear function that maps the latent space to data space.

3. Relevant Background

3.1 Gaussian Process

Gaussian Processes or GPs define non-parametric distributions over non-linear functions. A GP, denoted as $\mathrm{GP}(\boldsymbol{\mu}, \Sigma)$ is defined as by a *mean function* $\boldsymbol{\mu}$ and a *covariance function* Σ. A function $f : \mathbb{R}^D \to \mathbb{R}$, sampled from a Gaussian Process is written as

$$\mathbf{f} \quad \sim \quad \mathrm{GP}(\boldsymbol{\mu}, \Sigma)$$

We can imagine a function to be an infinite dimensional vector. Therefore, essentially GP samples infinite dimensional vectors, *i.e.* random function $\mathbf{f}$.

More formally, a GP can be thought of as a Stochastic Process defined in such a way that any finite-dimensional subset $\{x_n \in \mathbb{R}\}_{n=1}^N$ of the infinite-dimensional vector is distributed as a Gaussian Distribution. Thus, a GP assumes that the distribution $\mathbb{P}[\, f(x_1), f(x_2) \dots f(x_N)\,]$ is jointly a multivariate gaussian, with some mean $\boldsymbol{\mu}(\mathbf{X}) \in \mathbb{R}^N$ and some $\Sigma(\mathbf{X}) \in \mathbb{R}^{N \times N}$, where $X = \{\mathbf{x}_n\}_{n=1}$ (Murphy, 2012).

Note. For the ease of notation, we define $\mathbf{f} = [\, f(\mathbf{x}_1), f(\mathbf{x}_2) \dots f(\mathbf{x}_N)\,]^{\mathrm{T}}$

Therefore, a Gaussian Process generalizes the underlying distribution of a function $f : \mathbb{R}^D \to \mathbb{R}$. That is, given a set of points $\{x_n \in \mathbb{R}^D\}_{n=1}^N$ and their respective mappings $\{f^*(x_n)\}_{n=1}^N$, it is possible to model probable continuous functions representing f^* using a GP. We denote this as following

$$\mathbf{f} \quad \sim \quad \mathrm{GP}(\boldsymbol{\mu}(\mathbf{X}), \Sigma(\mathbf{X}))$$

where

$$\begin{aligned} \boldsymbol{\mu}(\mathbf{X}) &= \mathbb{E}[\,\mathbf{f}\,] \\ \Sigma(\mathbf{X}) &= \mathbb{E}\left[\, (\mathbf{f} - \boldsymbol{\mu}(X))^{\mathrm{T}} (\mathbf{f} - \boldsymbol{\mu}(X)) \,\right] \end{aligned}$$

The covariance matrix specifies the smoothness of the function sampled using a GP. In fact, each entry of the covariance matrix is the similarity between the respective pair of points, *i.e.* $\Sigma_{(i,j)} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ where κ is a kernel (see Souza and R., 2010) or similarity function.

We can alternatively say that $\mathbf{f}$ is sampled from a Gaussian Process $\text{GP}(\boldsymbol{\mu}, \Sigma)$, if its finite dimensional marginal is jointly a multivariate distribution. More formally, $\mathbf{f} \sim \text{GP}(\boldsymbol{\mu}, \Sigma)$ if for all $N \in \mathbb{N}$ and any arbitrary set of points $\{\mathbf{x}_n\}_{n=1}^{N}$, we have

$$\begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}(\mathbf{x}_1) \\ \boldsymbol{\mu}(\mathbf{x}_2) \\ \vdots \\ \boldsymbol{\mu}(\mathbf{x}_N) \end{bmatrix}, \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \dots & \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \dots & \kappa(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \kappa(\mathbf{x}_N, \mathbf{x}_2) & \dots & \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \right)$$

It is noteworthy that all marginals or conditionals for the random vectors are Gaussian distributions as per Gaussian properties [1].

This makes Gaussian processes very interesting to study, and also easily applicable in a wide variety of tasks, such as non-linear regression and classification. Another application of Gaussian Processes is in latent representation of high-dimensional vectors for clustering purposes (Iwata et al., 2012), which is discussed in 4.1.

3.2 Dirichlet Process Mixture Models

Similar to Gaussian Processes, 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) [2], 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 particularly useful in the task of non-parametric approach 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

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

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

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

[1] Any subset of $\mathbf{f}$ will be a random vector, which in case of Gaussian Processes will be a Gaussian Distribution itself
[2] $\text{G}(\boldsymbol{\theta})$ is random since G itself is random and is sampled from the Dirichlet Process

representation, although the underlying model remains the same.

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

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.

3.3 Variational Autoencoder

Variational autoencoders (VAEs) are essentially a deep learning technique for learning latent representations. We can consider a directed latent-variable model of the form

$$\mathbf{p}(\mathbf{x}, \mathbf{z}) = \mathbf{p}(\mathbf{x}|\mathbf{z})\mathbf{p}(\mathbf{z})$$

with observed $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ can be continuous, discrete, or latent. Suppose that given a dataset $\mathbb{D} = \{x^1, \dots, x^n\}$, we may be interested in the following inference and learning tasks:

- Learning the parameters θ of $\mathbf{p}$.
- Approximate posterior inference over $\mathbf{z}$.
- Approximate marginal inference over $\mathbf{x}$.

Auto-encoding variational Bayes (AEVB) is an algorithm that can efficiently solve these three inference and learning tasks; the variational auto-encoder being one instantiation of this algorithm. In variational inference, we are interested in maximizing the evidence lower bound (ELBO)

$$\mathcal{L}(\mathbf{p}_\theta, \mathbf{q}_\phi) = \mathbb{E}_{\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})}[\log\, \mathbf{p}_\theta(\mathbf{x}, \mathbf{z}) - \log\, \mathbf{q}_\phi(\mathbf{z}|\mathbf{x})]$$

over the space of whole $\mathbf{q}_\phi$ and evidently satisfies the equation

$$\log\, \mathbf{p}_\theta(\mathbf{x}) = \mathbb{KL}[\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})||\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})] + \mathcal{L}(\mathbf{p}_\theta, \mathbf{q}_\phi)$$

Since $\mathbf{x}$ is fixed, we can define $\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$ to be conditioned on $\mathbf{x}$ meaning that we are effectively choosing a different $\mathbf{q}(\mathbf{z})$ for every $\mathbf{x}$, which will produce a better posterior approximation than always choosing the same $\mathbf{q}(\mathbf{z})$. Now the question is how we optimize over $\mathbf{q}(\mathbf{z}|\mathbf{x})$?

One approach called black-box variational inference consists of maximizing the ELBO using gradient descent over ϕ only assuming that $\mathbf{q}_\phi$ is differentiable in its parameters ϕ which involves computing the gradient

$$\nabla_{\theta,\phi}\mathbb{E}_{\mathbf{q}_\phi(\mathbf{z})}[\log\, \mathbf{p}_\theta(\mathbf{x}, \mathbf{z}) - \log\, \mathbf{q}_\phi(\mathbf{z})]$$

One way to estimate this gradient is via the so-called score function estimator:

$$\nabla_\phi \mathbb{E}_{\mathbf{q}_\phi(\mathbf{z})}[\log\ \mathbf{p}_\theta(\mathbf{x},\mathbf{z}) - \log\ \mathbf{q}_\phi(\mathbf{z})] = \mathbb{E}_{\mathbf{q}_\phi(\mathbf{z})}[(\log\ \mathbf{p}_\theta(\mathbf{x},\mathbf{z}) - \log\ \mathbf{q}_\phi(\mathbf{z}))\nabla_\phi \log\ \mathbf{q}_\phi(\mathbf{z})]$$

which follows from some basic algebra and calculus (**?**).

The above identity referred to as score function estimator places the gradient inside the expectation, which we may now evaluate using Monte Carlo. Unfortunately, the score function estimator has an important shortcoming: it has a high variance. The key contribution of the VAE paper (**?**) is to propose an alternative estimator that is much better behaved which is done in two steps: we first reformulate the ELBO so that parts of it can be computed in closed form (without Monte Carlo), and then we use an alternative gradient estimator, based on the so-called *reparametrization trick*.

By rearranging the ELBO we get the reformulation called SGVB estimator as follows:

$$\log\ \mathbf{p}(x) \geq \mathbb{E}_{\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})}[\log\ \mathbf{p}_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{KL}[\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})||\mathbf{p}(\mathbf{z})]$$

We may think of $\mathbf{x}$ as an observed data point. The left-hand side consists of two terms, both of which involve taking a sample $\mathbf{z} \sim \mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$, which we can interpret as a code describing $\mathbf{x}$. So we refer to $\mathbf{q}$ as the *encoder*.

In the first term, $\log\ \mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$ is the log-likelihood of the observed $\mathbf{x}$ given the encoding $\mathbf{z}$ that we have sampled. This term is maximized when $\log\ \mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$ assigns high probability to the original $\mathbf{x}$. So the attempt is to reconstruct $\mathbf{x}$ given the encoding $\mathbf{z}$. Hence we call $\log\ \mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$ the *decoder* and the term is called the *reconstruction error*.

The second term is the divergence between $\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$ and the prior $\mathbf{p}(\mathbf{z})$, which we will fix to be $\mathcal{N}(0,\mathbf{I})$. It encourages the encodings $\mathbf{z}$ to look Gaussian. We call it the regularization term. It prevents $\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$ from encoding the distribution in arbitrary space and instead forces it to stay close to the prior distribution, so that at test time when we sample from this prior, we indeed get good results.

Now we can talk about the reparametrization trick to get a low-variance gradient estimator for optimizing our objective. Under certain mild conditions, we may express the distribution $\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$ as the following two-step generative process:

First, we sample a noise variable ϵ from a simple distribution like the standard Normal
$$\epsilon \sim \mathcal{N}(0,\mathbf{I})$$
Then, we apply a deterministic transformation $\mathbf{h}_\phi(\epsilon,\mathbf{x})$ that maps the random noise into a more complex distribution
$$\mathbf{z} = \mathbf{h}_\phi(\epsilon,\mathbf{x}).$$

Gaussian variables provide the simplest example of the reparametrization trick. Given μ and Σ as the mean and covariance of $\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$, we can sample from $\mathcal{N}(\mu,\Sigma)$ by first sampling $\epsilon \sim \mathcal{N}(0,\mathbf{I})$ and then computing $\mathbf{z} = \mu + \Sigma^{\frac{1}{2}}\epsilon$. The biggest advantage of this approach is that we many now write the gradient of an expectation with respect to $\mathbf{q}_\phi(\mathbf{z})$(for any f) as

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{x},\mathbf{z})] = \nabla_\phi \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I})}[f(x,\mathbf{h}_\phi(\epsilon,\mathbf{x}))] = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I})}[\nabla_\phi f(x,\mathbf{h}_\phi(\epsilon,\mathbf{x}))]$$

The gradient is now inside the expectation and we may use Monte Carlo to get an estimate of the right-hand term. Still, we have not specified the exact form of $\mathbf{p}$ or $\mathbf{q}$ besides saying that these could be arbitrary functions. The best $\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$ should be able to approximate the true posterior $\mathbf{p}(\mathbf{z}|\mathbf{x})$). Similarly, $\mathbf{p}(\mathbf{x})$ should be flexible enough to represent the richness of the data. Therefore, we are going to parameterize $\mathbf{p}$ and $\mathbf{q}$ by neural networks.

4. Related Work

4.1 Manifold Clustering through GPLVMs

The model assumes that each observation in some data space 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) (Iwata et al., 2012) 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.

More concretely, assume that we have a data space $\mathbb{R}^K$ and the corresponding latent space $\mathbb{R}^D$. Then the iWMM model defines the following generative story:

1. Draw mixture weights $\boldsymbol{\pi} \sim Gem(\eta)$
2. For each component c $= 1, ..., \infty$
   (a) Draw precision matrix $\boldsymbol{\Lambda}_c \sim \mathcal{W}(\mathbf{S}^{-1}, \nu)$
   (b) Draw mean vector $\boldsymbol{\mu}_c \sim \mathcal{N}(\mathbf{u}, (r\boldsymbol{\Lambda}_c)^{-1})$
3. For each observed dimension k $= 1, ..., K$
   (a) Draw function $f_k(x) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$
4. For each obsevation n $= 1, ..., N$
   (a) Draw latent cluster assignment $z_n \sim Mult(\boldsymbol{\pi})$
   (b) Draw latent coordinates $\mathbf{x}_n \sim \mathcal{N}(\boldsymbol{\mu}_{z_n}, \boldsymbol{\Lambda}_{z_n}^{-1})$
   (c) For each observed dimension d $= 1, ..., D$
      i. Draw feature $y_{nd} \sim \mathcal{N}(f_k(\mathbf{x}_n), \beta^{-1})$

Note that the $Gem(\pi)$ defines the stick-breaking process that generates mixture weights for a Dirichlet process with parameter η. Also note that $Mult(\pi)$ defines a Multinomial distribution with parameter π and $m(x)$ and $k(x, x')$ are the mean and kernel functions of the Gaussian Process respectively.

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.

4.2 Manifold Clustering through GMM-SVAE

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 (**?**) 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.

In more precise terms, assuming our latent representation of the data lies in $\mathbb{R}^K$ space and our data lies in $\mathbb{R}^D$ space, the model defines the following generative story:

1. Draw mixture weights $\boldsymbol{\pi} \sim Dir(\boldsymbol{\alpha})$
2. For each component k $= 1, ..., K$
   (a) Draw $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \sim NIW(\boldsymbol{\lambda})$
3. Draw parameters of encoder and decoder model $\boldsymbol{\gamma} \sim p(\boldsymbol{\gamma})$
4. For each observation n $= 1, ..., N$
   (a) Draw cluster assignment $c_n \sim Mult(\boldsymbol{\pi})$
   (b) Draw latent coordinates $\mathbf{x}_n \sim \mathcal{N}(\boldsymbol{\mu}_{c_n}, \boldsymbol{\Sigma}_{c_n})$
   (c) Draw a data point $\mathbf{y}_n \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}_n; \boldsymbol{\gamma}), \boldsymbol{\Sigma}(\mathbf{x}_n; \boldsymbol{\gamma}))$

Note that $Dir(\boldsymbol{\alpha})$ defines a Dirichlet distribution over the mixing proportions and $NIW(\boldsymbol{\lambda})$ defines Normal-Inverse-Wishart distributions over the parameters associated with each cluster. Moreover, $p(\boldsymbol{\gamma})$ defines the distribution over the parameters of the encoder-decoder model and $\boldsymbol{\mu}(\mathbf{x}_n; \boldsymbol{\gamma})$ and $\boldsymbol{\Sigma}(\mathbf{x}_n; \boldsymbol{\gamma})$ are neural networks with parameters $\boldsymbol{\gamma}$ which map $\mathbf{x}_n$ to the observed data $\mathbf{y}_n$.

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.

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.

5. 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.

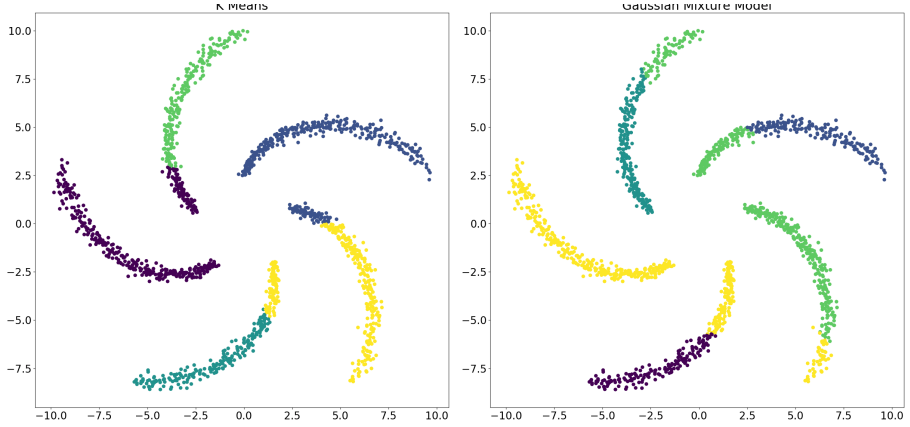
To achieve this aim, we describe the following generative story:

1. Draw mixture weights $\boldsymbol{\pi} \sim Gem(\eta)$
2. For each component c $= 1, ..., \infty$
   (a) Draw $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \sim NIW(\boldsymbol{\lambda})$
3. Draw parameters of the encoder and decoder model $\boldsymbol{\gamma} \sim p(\boldsymbol{\gamma})$
4. For each obsevation n $= 1, ..., N$
   (a) Draw latent cluster assignment $c_n \sim Mult(\boldsymbol{\pi})$
   (b) Draw latent coordinates $\mathbf{x}_n \sim \mathcal{N}(\boldsymbol{\mu}_{c_n}, \boldsymbol{\Lambda}_{z_n}^{-1})$
   (c) Draw a data point $\mathbf{y}_n \sim \mathcal{N}(\boldsymbol{\mu}(x_n; \boldsymbol{\gamma}), \boldsymbol{\Sigma}(x_n; \boldsymbol{\gamma}))$

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, it has the capacity to, in theory, surpass the existing methods mentioned above.

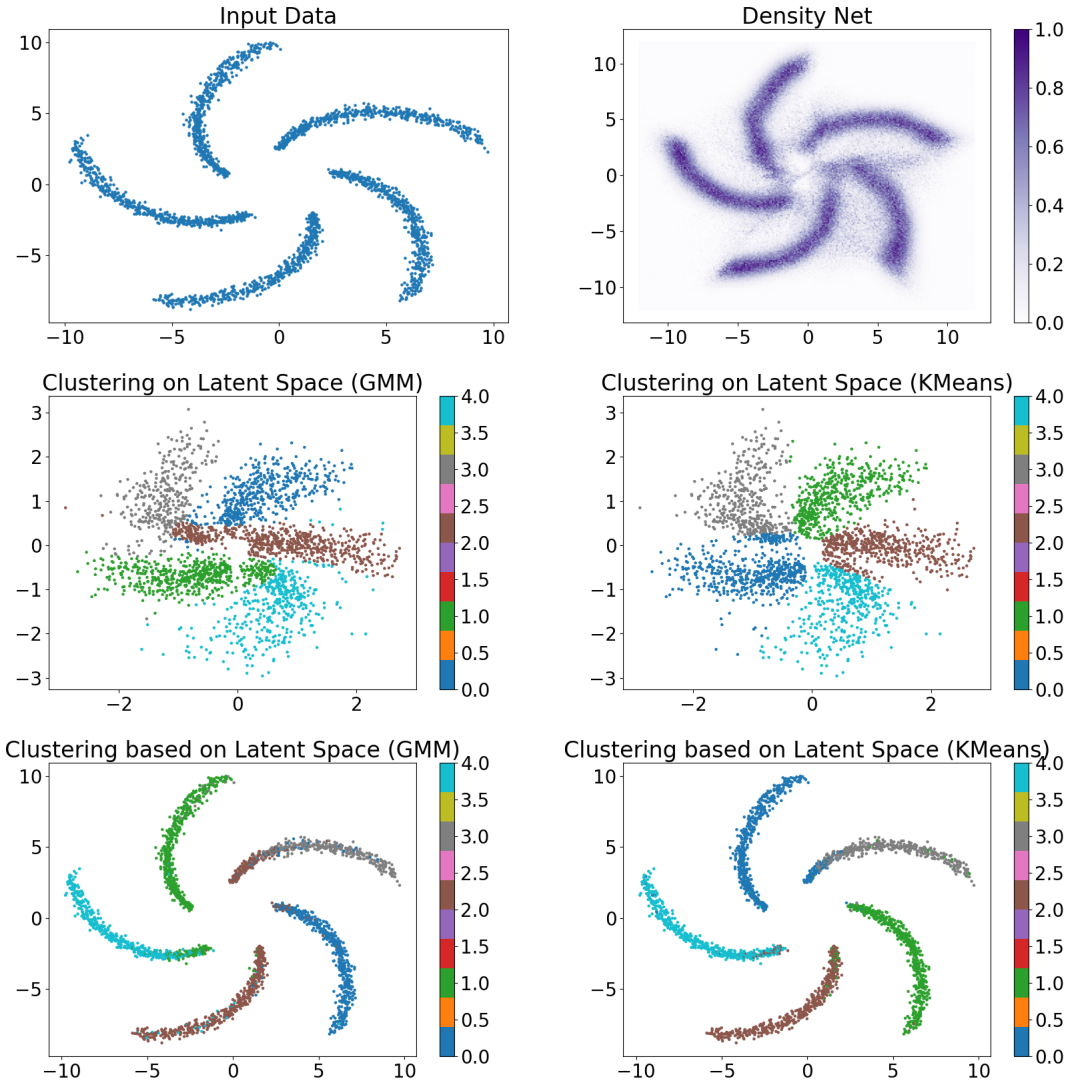
6. Experiments

6.1 Clustering using Standard GMM and K-Means



Clustering Spiral Dataset using K-Means and GMM

6.2 Clustering on Latent Representation in VAE



Clustering Spiral Dataset by independently clustering over VAE's latent space

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.

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.

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

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

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.

Souza and César R. Kernel functions for machine learning applications, 2010. URL http://crsouza.blogspot.com/2010/03/kernel-functions-for-machine-learning.html.