CS571 Final Deliverable

April 23, 2022

1 Press Release

Duke Students Empowers Duke Hospital and beyond with AI-driven tool in Launching startup OpenCluster

NEWS | University

April 23, 2022

ML is helping clinical research at Duke Hospital, thanks to a ML clustering service provided by a group of Duke students.

Danny Luo, Trinity '23, along with four other co-founders, founded the startup OpenCluster.Inc, a company that offers a powerful clustering algorithm service for a plethora of use cases. Two projects at Duke hospital is benefiting from this service; one is pattern detection in Genome Sequencing, and the other is brain MRI segmentation. Both of the two projects are currently under contracts with OpenCluster, leveraging the company's core algorithm in unsupervised clustering of complex data.

There has been an increasing interest in the medical community to use machine learning techniques for various tasks such as disease diagnosis and pattern recognition. OpenCluster's specialization certainly took part in and contributed to this growing trend.

"In this day and age, we increasingly deal with huge amounts of very complex data," Boxuan Li, CTO of OpenCluster, said that the popular demand of effective clustering algorithm from industry and research communities helped him conceptualize the idea. "More often than not, people want to group or segment the data for various use cases and that is where our algorithm came to rescue."

Li explained that one of the key components of their algorithm is the Dirichlet Process Gaussian Mixture Model, an unsupervised, non-parametric model without the need to specify number of clusters ahead of time,

which is extremely helpful when domain experts have few prior beliefs on the underlying data distribution. "You don't need to make many assumptions on the data," said Li," you simply let the model do the work."

Noticeably, the pipeline that the group designed also incorporated a dimension-reduction component, which gives another kick to their product. Nianli Peng and Wei Wu, two other co-founders at OpenCluster, emphasized the difficulty of the current methods in dealing with high dimensional data. "ML community suffered a lot from the curse of dimensionality," said Peng, "that is why we think about ways to reduce dimensions from day one of our business." Wu disclosed several methods currently being deployed in the product, including Principle Component Analysis, Linear Discriminate Analysis and even neural network.

"OpenCluster really aims at solving our pain points", said Professor Eleanor Arroway, who directs research at Duke Clinical Research Institute. "We've never used such a tool that effectively and efficiently groups large volume of complex data. It works great as a first step onto lots of our key research problems, and it already starts to yield valuable insights to our work."

OpenCluster is not just about clinics and medicine, however. "Providing our service to clinical research and Duke hospital is just our first step out there to expand the growing market," said Luo, "We have tons of potential customers. They are interested in our products for fast, insightful pattern recognition and segmentation. It is about uncovering patterns that facilitate important insights and drive business decisions."

Jingyu Peng, another co-founder of the company, disclosed that OpenCluster has already established partnerships with several other businesses. One of the contractor is Duke Dining, who will be using the company's service for customer segmentation based on daily spending. With 4000 academic staff and around 15000 students, Duke Dining is in critical demand of an efficient clustering tool to better understand customer behaviors.

Beyond providing data processing and clustering services, the team plans to extend their business to matchmaking service based on their own algorithm. "Those who have the pleasure of swiping on dating apps for hours on end know how difficult and tiring the process can be", said Peng, "our product aims to make finding an ideal match relatively swift and easy. Just answer several questions, and the perfect date comes." The team is currently building their own dating app which aims to find the perfect match for users based on their personality, habits, and interests. "It's all about finding matches," says Luo, "just another clustering problem."

Eric Laber, an advisor to the founding members as well as Professor of Statistics at Duke University, is very glad to see that his very own course incubated the idea and this rising startup." I am truly amazed by their own initiative and their ability to transform what they learned into something useful to the society," said Laber.

Despite the group's expanding business, Luo admitted that the product itself still has a long way to go. "We did sacrifice some domain specialization since our current target is making our product applicable in a wide range of scenarios," he explained. "Also, clustering algorithm is only part of our expertise. There are many more potential directions we could take, but this is a very good starting point for us."

2 FAQs

1. **What exactly does your product do?**

   Our product is a non-parametric clustering tool for users to perform data analysis. It involves automatically discovering natural grouping in data. Unlike supervised learning, where a labeled training set must be provided, clustering algorithms only interpret the input data and find natural groups or clusters in feature space, which means we just need the data itself and there is no need to know how many groups there are or which sample belongs to which group. Our model will automatically group samples into clusters based on how 'similar' they are.

2. **What methods do you use?**

   For high dimensional data, We use dimension reduction techniques such as principle component analysis (PCA) to reduce high-dimensional data to a reasonably low dimension that Dirichlet Process Gaussian Mixture Model (DPGMM) could run efficiently and have good performances. Then we will apply DPGMM to find the latent distribution of the data. For large datasets, we will apply k-means to classify the whole dataset after the DPGMM identifies the latent variable on a subset of data to improve efficiency.

3. **What are some advantages of your model?**

   First, our product employs a non-parametric model so no prior knowledge of the data is required, which means we just need the data itself and there is no need to know how many groups there are or which sample belongs to which group. Second, compared with traditional state-of-the-art clustering algorithms, our product does some clever tricks to combine dimension-reduction techniques with Dirichlet Process Gaussian Mixture Model that yields a significant reduction in computation cost while retaining decent accuracy.

4. **What are the potential applications of your product?**

   Our product can be helpful as long as our client wants to learn more about the problem domain, so-called pattern discovery or knowledge discovery. For example:

(a) The phylogenetic tree could be considered the result of a manual clustering analysis.

(b) Separating normal data from outliers or anomalies may be considered a clustering problem.

(c) Separating clusters based on their natural behavior is a clustering problem, referred to as market segmentation.

Several of the projects that we have embarked on include pattern detection in Genome Sequencing and brain MRI segmentation in collaboration with Duke Hospital as well as customer segmentation in collaboration with Duke Dining. Part of our future agenda also include modifying our product to perform match-making, which saw its wide application in geo-social networking and job search services.

5. **If your model are unsupervised, how are they tested?**

Our test is based on generated data which we know the ground truth distribution and clusters. The data generated by our expertise simulates the complexity of real-world data. Our next step is to test the algorithm on real world data that are labeled by human.

6. **What standards do you use in the tests?**

There are three standards we use:

(a) silhouette score: Score calculated by the mean intra-cluster distance a and the mean nearest-cluster distance b for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b).

(b) random score: Score computes a similarity measure between two clusters by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusters. It is calculated by RI = (number of agreeing pairs) / (number of pairs)

(c) adjusted random score: Random score adjusted by chance in order to ensured to have a value close to 0.0 for random labeling independently of the number of clusters and samples and exactly 1.0 when the clusters are identical (up to a permutation). The adjusted random score ARI= (RI - Expected RI) / (max(RI) - Expected RI)

7. **How long do I need to wait to get the results**

Our algorithm returns results in a timely manner. For large datasets, we use dimension reduction methods followed by a mixture model of DPGMM and K-means model to speed up calculation. Our test shows that we can find the clusters in a 100,000 by 100 dimensional data in 3 minutes with a high

ARI score (as compared to hours using solely DPGMM method). So the result should be available in minutes to hours after it is submitted to our server.

8. **How do you find clusters? Do I need to provide samples/labeled data?**

   Our algorithm finds the latent distribution of the data via DPGMM. It is a non-parametric unsupervised learning algorithm that does not require any labeled data. With no samples with ground truth labeled, we use the PCA algorithm to find the most important features for clustering. However, if the customer provides us with a labeled set of samples, we will be able to train an LDA model to speed up our dimension reduction algorithm and use less features in DPGMM, which will speed up the whole algorithm.

9. **What data formats does your algorithm support?**

   We accept all kinds of data formats–Excel charts, .txt file, csv files, json file, and even images.

10. **How did you test your method?**

    We generate testing data to test. Additionally, we test on the real-world dataset. We searched for labeled dataset, then cluster the dataset using our product and compared the results with the labels to judge our product's performance.

11. **What are some pitfalls of the product?**

    We are currently in the testing stage. At this point, our model requires the data to be generally Gaussian distributed. We are working on algorithms using neural-networks to fit data that do not follow a normal distribution.

12. **I am not familiar with data science. How do I know if my data is normally distributed so that your current algorithm could run?**

    Our expertise will perform normality test for our clients with their consents. Our testing methods include multivariate analysis of variance (MANOVA), principal component analysis (PCA), canonical correlation, and discriminant analysis, which will guarantee the data is suitable for the model.

13. **What will the results be like?**

    The result will include the labeled dataset, a summary of the data which includes the estimated center, variance, size of the clusters. We will also offer three evaluation criteria (silhouette score, random score, adjusted random score) of the model.

14. **What are contingency plans if the model malfunctions? How do we address bias in models?**

    We will cross validate the model via different dimension reduction and clustering methods. Our expertise will perform tests to make sure the model fits well. If the model performs poorly based on the evaluation criteria, a full refund will be issued.

15. **Are there privacy concerns about clients giving you all of customers' data?**

    Since our product automatically groups data into clusters, if our clients simply want their data to be divided into groups, we can simply return the output of our product without accessing it. On the other hand, if our clients want us to identify certain clusters of specific interests, ex. groups of customers with high spending, we will only access attributes of the clusters such as means and variances instead of those of individual observations. Hence data privacy is well-protected with our model.

3 Technical Details/Results

3.1 Motivation

The problem we focus on is clustering of large amount of data with unknown number of clusters. We assume that our data comes from a mixture Gaussian distribution with unknown number of components K in the mixture. Then, solving this clustering problem entails inference on the number of clusters, cluster means and covariance, as well as the cluster labels for each data points.

The inherent difficulty of the problem is unknown number of clusters. We adopt a Dirichlet process Gaussian mixture model (DPGMM) as the backbone of our algorithm (See [Ferguson, 1973] for the first proposal of Dirichlet Process and see [Escobar and West, 1995] for first proposal of DPGMM) .The key motivation behind our adoption of this model is that Dirichlet Process Gaussian mixture is non-parametric, in that it assumes a non-fixed number of clusters K. Thus compared with traditional parametric clustering technique such as K-means, it has following key advantages:

- it learns the number of clusters K from the data, without having to explicitly specify a fixed number. This is especially important when we do not have a strong prior belief on the number of clusters
- it saves computation to run the model selection procedure if we were to use parametric models. For example, if K-means is used, optimal number of clusters would have to be selected via multiple runs

of model with different K; this is, however, rather hard given that in certain cases we are not aware of even a range of the meaningful number of clusters.

3.2 DPGMM Formulation

Here we gave the detailed mathematical formulation of Dirichlet Process and its application to Gaussian mixture model. We know a typical Gaussian Mixture model assumes the following sampling model:

$$\mathbf{x}_i|z_i = j \sim N(\mathbf{x}|\mu_i, \Sigma_j),$$
$$P(z_i = j) = w_j,$$

, where we first sample a latent membership variable z_i according to a mixture proportion vector $\mathbf{w}$ and then make a draw from the corresponding Gaussian model. The marginal likelihood of x is thus:

$$p(\mathbf{x}) = \sum_{k=1}^{K} w_k N(x|\mu_k, \Sigma_k) \tag{1}$$

Note that the above sampling procedure assumes a fixed K, the number of clusters. And for Bayesian inference, we usually place a Dirichlet prior on $\mathbf{w}$ so that $\sum_k w_k = 1$. To remedy that inflexibility, we introduce a "stick-breaking construction" of $\mathbf{w}$ such that we can have a countably infinite number of clusters. (See [Sethuraman, 1994] for the first proposal of this interpretation) Let α be concentration parameters that controls the distribution of mixing proportions, we have the following recursive process for generating $\mathbf{w}$:

$$\beta_k \sim Beta(1, \alpha)$$
$$w_k = \beta_k \prod_{l=1}^{k-1}(1 - \beta_k) \tag{2}$$

Intuitively, we can imagine that with a stick of length 1, the first cluster proportion will be of length β_1, and we broke it off. Then we broke off β_2 from the rest of the stick, so on and so forth to generate $\mathbf{w}$ such that $\sum_{k=1}^{\infty} w_k = 1$. Notice now we are able to have an infinite mixture model.

Now define a base distribution G_0 over $\theta = \{\mu, \Sigma\}$, the space of cluster parameters (mean and covariance

matrix in this case). We say $G \sim DirichletProcess(\alpha, G_0)$ if it satisfies the following:

$$\theta_k \sim G_0$$
$$G = \sum_{k=1}^{\infty} w_k \delta_{\theta_k}$$

, where $\delta_{\theta_\mathbf{k}}$ is a point mass centered on $\theta_\mathbf{k}$ and $\mathbf{w}$ is constructed as in (2).

Putting it all together, the sampling model of DPGMM has the form below:

$$x_i \sim N(x|\theta_\mathbf{i}),$$
$$\theta_\mathbf{i} = \{\mu_\mathbf{i}, \Sigma_i\} \sim G,$$
$$G \sim DP(\alpha, G_0)$$

Intuitively, we model set of observations $\{x_i\}_{i=1}^n$ through latent variables $\{\theta_i\}_{i=1}^n$, where each θ_i is drawn i.i.d from a discrete distribution G generated by the stick-breaking process. Because G is discrete and most probability mass is concentrated on a few values (as seen from the stick-breaking construction), it is very likely that multiple θ_i takes on the same values (i.e., falling into the same histogram bins in G), thus belong to the same cluster. Thus those few discrete values from G being taken becomes our cluster parameters. Now, analogous to (1), we have a sample draw from DPGMM having the following marginal density:

$$p(\mathbf{x}) = \sum_{k=1}^{\infty} w_k N(x|\mu_k, \Sigma_k) G(w_k, \mu_k, \Sigma_k) \tag{3}$$

For Bayesian inference, we follow the convention in literature to place the following priors on the parameters α and G_0:

$$\alpha \sim Gamma(a, b)$$
$$G_0(\mu, \Sigma) \sim N(\mu|\mu_\mathbf{0} = 0, \Sigma_0) IW(\nu_0, \Phi_0)$$

The usual inference task is to obtain the posterior predictive distribution:

$$p(\mathbf{x}|\mathbf{x}_1, .., \mathbf{x}_N, \alpha, G_0) = \int p(x|\theta) p(\theta|\mathbf{x}_1, .., \mathbf{x}_N, \alpha, G_0) d\theta \tag{4}$$

However, the posterior distribution of θ in (4) is usually complicated and not in closed form so that two approaches have been proposed to solve this problem: MCMC (See [Görür and Edward Rasmussen, 2010] for detailed description of Gibbs sampler for both conjugate model and conditionally conjugate model) and Variational inference (See [Blei and Jordan, 2006] for a detailed derivations, also implemented by scikit-

learn [Pedregosa et al., 2011]). While MCMC sampling can achieve better theoretical guarantees , variational inference is usually faster and gives approximation to a wider range of complex, non-closed-formed posteriors.

3.3 Dimension Reduction Techniques

In this section, we outlined our dimension reduction techniques.

3.3.1 PCA

High-dimensional random vectors can be expressed as a linear combination of basis vectors plus noise:

$$\mathbf{x} = \sum_{j=1}^{k} \mathbf{h}_j w_j + \mathbf{m} + \mathbf{e} = \mathbf{H}\mathbf{w} + \mathbf{m} + \mathbf{e}$$

$$p(e) \sim N(0, \mathbf{V})$$

where $\mathbf{x}$ has length d and $\mathbf{w}$ has a smaller length k. The vector $\mathbf{m}$ defines the mean of $\mathbf{x}$ while $\mathbf{H}$ and $\mathbf{V}$ define its variance. For PCA, the noise variance $\mathbf{V} = v\mathbf{I}_d$, and $p(\mathbf{w}) \sim N(\mathbf{0}, \mathbf{I}_k)$ [Tipping and Bishop, 1999]. The goal of PCA is to estimate the basis vectors $\mathbf{H}$ and the noise variance v. According [Minka, 2000], the probability of observing a vector $\mathbf{x}$ is

$$p(\mathbf{x}|\mathbf{w}, \mathbf{H}, \mathbf{m}, v) \sim N(\mathbf{H}\mathbf{w} + \mathbf{m}, v\mathbf{I})$$

$$p(\mathbf{x}|\mathbf{H}, \mathbf{m}, v) = \int_{\mathbf{w}} p(\mathbf{x}|\mathbf{w}, \mathbf{H}, \mathbf{m}, v)p(\mathbf{w}) \sim N(\mathbf{m}, \mathbf{H}\mathbf{H}^T + v\mathbf{I})$$

the probability of observing the dataset

$$p(\mathbb{X}|\mathbf{H}, \mathbf{m}, v) = \prod_i p(x_i|\mathbf{H}, \mathbf{m}, v) = (2\pi)^{-Nd/2}|\mathbf{H}\mathbf{H}^T + v\mathbf{I}|^{N/2} exp(-\frac{1}{2} tr((\mathbf{H}\mathbf{H}^T + v\mathbf{I})^{-1}\mathbf{S}))$$

$$\mathbf{S} = \sum_i (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T$$

Hence the MLE for $\mathbf{m}$ is given by

$$\hat{\mathbf{m}} = \frac{1}{N}\sum_i x_i$$

, the mean of all observed data points. Suppose the covariance matrix of $\mathbf{x}$ can be decomposed to $\mathbf{U}_d\hat{\mathbf{\Lambda}}\mathbf{U}_d^T$ where $\mathbf{U}$ contains the top k eigenvectors of $\frac{\mathbf{S}}{N}$ and $\hat{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_k & 0 \\ 0 & v\mathbf{I}_{d-k} \end{bmatrix}$, then $\hat{\mathbf{H}} = \mathbf{U}(\Lambda_k - v\mathbf{I}_k)^{\frac{1}{2}}\mathbf{R}$ where,

the diagonal matrix $\mathbf{\Lambda}_k$ is the corresponding eigenvalues and $\mathbf{R}$ is an orthogonal matrix. Finally, v can be estimated by its MLE $\hat{v} = \frac{\sum_{j=k+1}^{d} \lambda_j}{d-k}$ where $\lambda_{j,j>k}$ is the left-out eigenvalues. Hence the likelihood of observing data $\mathbb{X}$ is maximized,

$$p(\mathbb{X}|\mathbf{H} = \hat{\mathbf{H}}, \mathbf{m}, v = \hat{v}) = (2\pi)^{Nd/2}(\prod_{j=1}^{k} \lambda_j)^{-\frac{N}{2}} \hat{v}^{\frac{-N(d-k)}{2}} exp(-\frac{Nd}{2})$$

3.3.2 LDA

Linear discriminant analysis (LDA) is a method used in statistics and other fields, to find a linear combination of features that characterizes or separates two or more classes of objects or events for dimension reduction [Wikipedia contributors, 2022]. LDA project the data with label into a low-dimensional space, so that the same type of data is as compact as possible, and different types of data are as scattered as possible. According to [Mika et al., 1999], we can define the mean of subclass:

$$\mu_i = \frac{1}{n_i} \sum_{x^{(i)} \in D_i} x^{(i)}$$

The mean of all samples:

$$\mu = \frac{1}{m} \sum_{x^{(i)} \in D_i} x^{(i)}$$

The within-class scatter matrix is defined as:

$$S_w = \sum_{i=1}^{n} (x_i - \mu_i)(x_i - \mu_i)^T$$

where μ_i is the sample mean of the i^{th} class.
The between-class scatter matrix is defined as:

$$S_b = \sum_{i=1}^{m} n_i(\mu_i - \mu)(\mu_i - \mu)^T$$

where m is the number of classes, μ is the all sample mean and n_i is the number of samples in the i^{th} class. LDA then can be an optimization problem. The loss function is:

$$J(w) = \frac{w^T S_b w}{w^T S_w w}$$

Because we would like the sample in every subclass is compact and the distance between two subclass is large, so our target is to maximize the loss function. Then it comes to:

$$\hat{w} = \text{argmax} \lim_{w} \frac{w^T S_b w}{w^T S_w w}$$

The column vector of the optimal projection matrix satisfies the following characteristic equation:

$$S_b w = \lambda S_w w$$

Then we select the biggest d eigenvalues' corresponding eigenvector to form as low dimension space.

3.3.3 Autoencoder

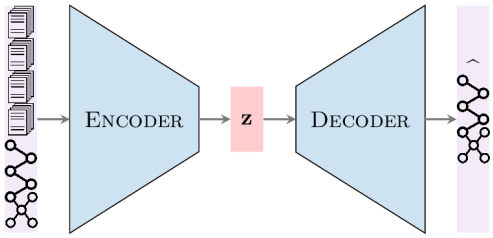

Figure 1: Encoder Decoder Architecture illustration

Since our assignment, ultimately, is to identify useful features for big data clustering problem, we have also turned our attention to autoencoders, an unsupervised deep learning technique in which we are able to leverage neural networks to extract most important features from the original high-dimension inputs [Goodfellow et al.,]. In particular, we are designing a feed-forward network architecture that extract most informational features by imposing a "bottleneck".

The concept of autoencoders has been introduced in literatures since decades ago [Rumelhart and McClelland, 1987]. Essentially, an autoencoder architecture consists of a feed-forward encoder structure that takes the input x and outputs feature representation. Then, a generative, top-down decoder structure will use the learned features to generate a reconstructed signal $\hat{x} = g(f(x))$ where f is the encoder function and g is the decoder function (See Figure 1 for an illustration). The network is trained aiming to minimize the reconstruction error $L(x, \hat{x})$.

In comparison, PCA essentially extracts linear, uncorrelated features, whereas autoencoders are able to model complex nonlinear features without a guarantee on correlation within features. Due to the good performance by a relatively simpler PCA model, for this product, we are proposing a simple, undercomplete au-

toencoder structure as the dimension of hidden features is less than the input dimension [Goodfellow et al.,].

The autoencoder architecutre is constructed as follows:

- Encoder: fully connected layers: input dimension $\rightarrow$ 256(ReLU) $\rightarrow$ 64(ReLU) $\rightarrow$ output dimension(Linear)
- Decoder: fully connected layers: output dimension $\rightarrow$ 64(ReLU) $\rightarrow$ 256(ReLU) $\rightarrow$ input dimension(Linear)

The training loss is derived from the mean squared error between the input signal x and the reconstructed signal $\hat{x}$:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x_i})^2$$

The neural network is constructed using PyTorch [Paszke et al., 2017].

3.4 Experiments

In this section, we conducted simulation experiments to test the effectiveness of dimension reduction + DPGMM model and showed that it is superior to parametric models.

3.4.1 Experiment Setup

We used synthetic data throughout our experiments. We generated the data drawn from multivariate Gaussian mixture models given dimension of data, number of clusters, variance of each dimension, cluster center bounds, ratio of data points in each cluster, and total data points. Cluster centers are drawn randomly from a bounded region. Variance, data dimension and cluster center bounds together controls how much each cluster mingles and how hard the clustering task is.

We implemented DPGMM using package from scikit-learn [Pedregosa et al., 2011] with default parameters set up for priors except explicitly stated. We used n_components=50 as the maximum number of clusters allowed to keep. (Since DPGMM is an infinite mixture model, we can suppress values drawn from discrete G distribution with lower probability mass) We used t-SNE (See proposal and a discussion of the method in greater details from [Van der Maaten and Hinton, 2008]) in order to visualize the high dimensional data structure mapped onto a 2-d plane. We used adjusted random score as similarity metric of the predicted clustering scheme and the ground truth for purpose of evaluation.

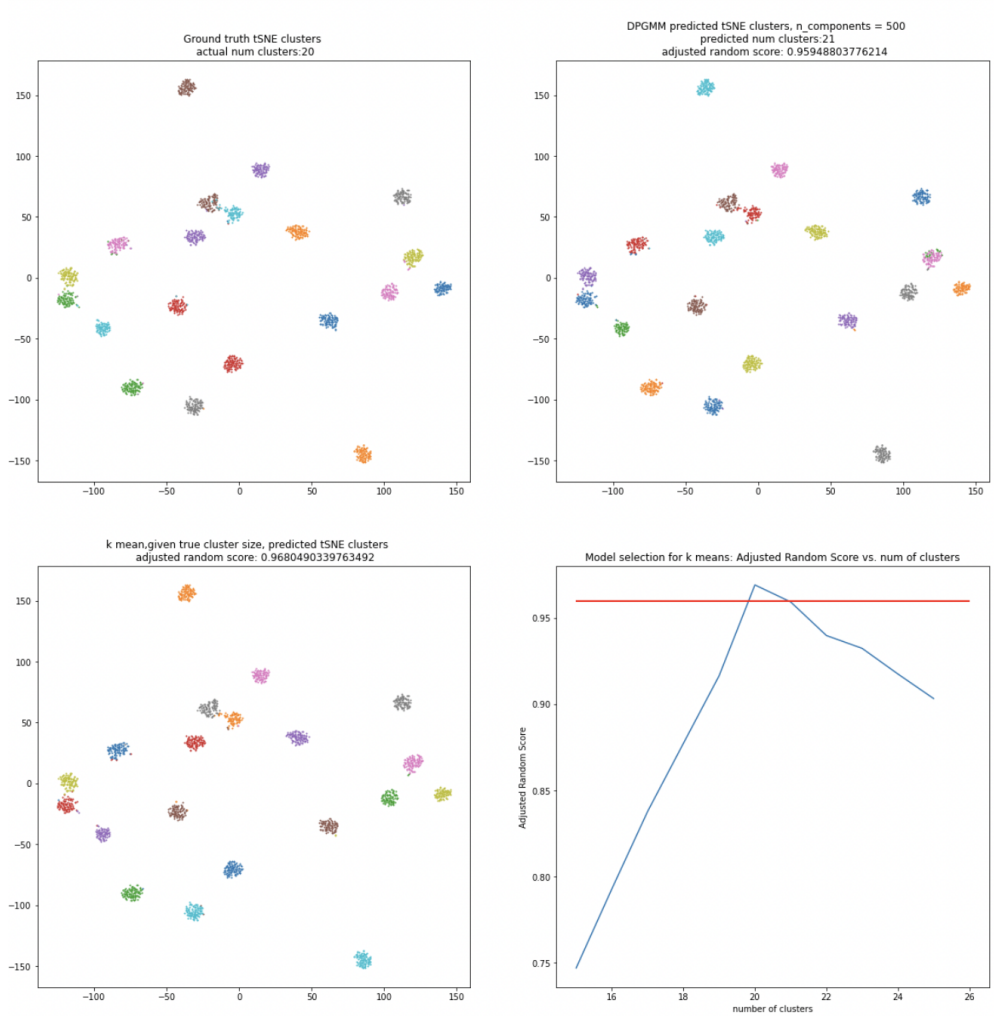

Figure 2: Performance of DPGMM versus K-Means. Top-left: Ground truth 20 clusters of 5-dimensional data with N = 10,000 with variance in each dimension 500, projected to 2d space by t-SNE with different color codes for each cluster. Top-right: DPGMM prediction of clustering. Bottom-left: K-means prediction of clustering given ground truth number of clusters. Bottom-right: Adjusted Random Score for K-means as we vary the mis-specified number of clusters. Red horizontal line indicates DPGMM performance.

3.4.2 DPGMM versus K-means

The clustering performance of our non-parametric DPGMM and traditional parametric K-means is shown in Figure 2, as we compared DPGMM clustering performance and K-means clustering performance both visually through t-SNE and quantitatively through adjusted random scores. DPGMM achieved an adjusted random score of 0.959 with inferenced number of cluster to be 21, very close to the ground truth number of clusters. For K-means, when the ground truth number of clusters is known, the performance is very good, achieving an adjusted random score of 0.968. However, when number of cluster K is miss-specified, as we can see from the bottom-right graph, the performance drops quickly below the DPGMM performance. This

shows that K-means is sensitive to model miss-specification, highlighting the strength of DPGMM when no strong prior belief above the number of clusters exists.

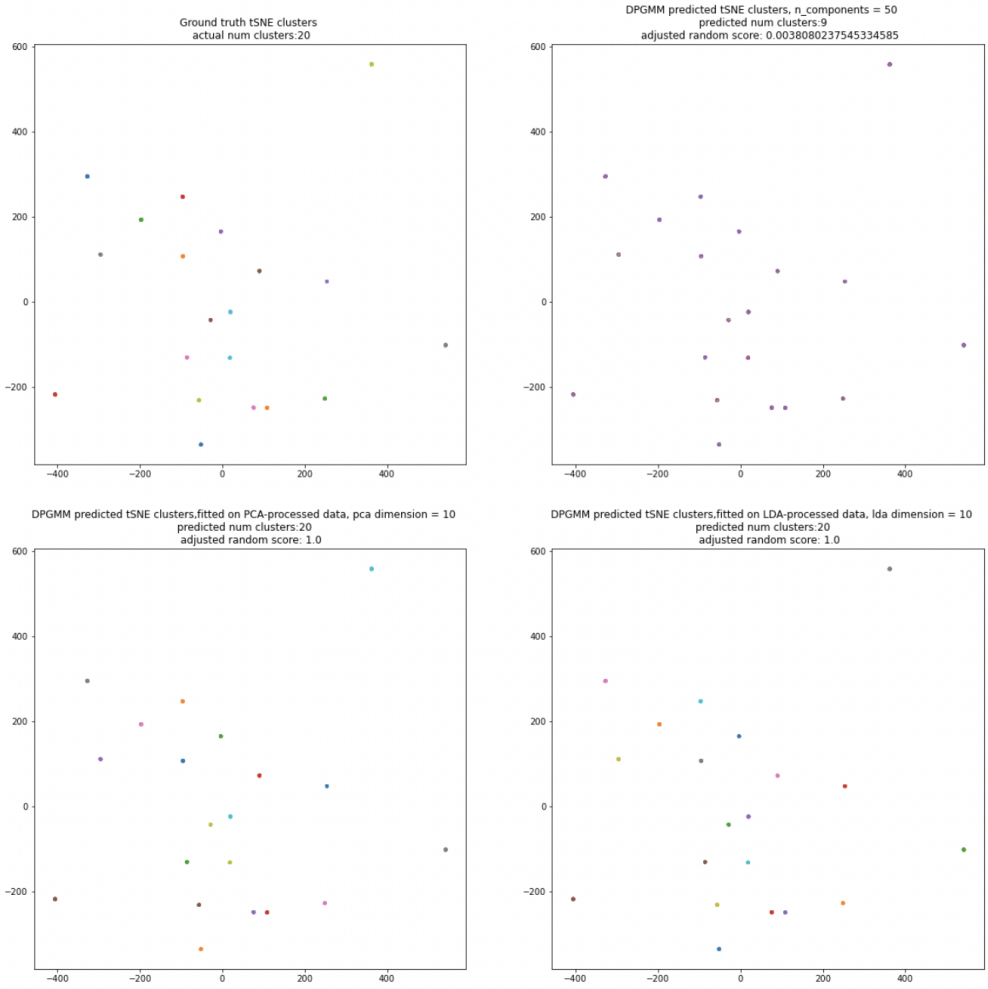

Figure 3: Performance of DPGMM with dimension reduction. Top-left: Ground truth 20 clusters of 1000-dimensional data with N = 10,000 with variance in each dimension 1000, with center range of [0,100] in each dimension, projected to 2d space by t-SNE. Top-right: DPGMM prediction of clustering without dimension reduction. Bottom-left: DPGMM prediction of clustering with PCA dimension reduction to dimension of 10. Bottom-right: DPGMM prediction of clustering with LDA dimension reduction to dimension of 10.

3.4.3 DPGMM with dimension reduction

We experimented with high-dimensional data and an additional stage of dimension reduction before feeding the reduced input data to DPGMM for clustering. Given 1,000 dimensional data, we compared the performance of plain DPGMM with added PCA and LDA stages and plot the results in Figure 3. As seen from the plot, plain DPGMM yields very poor adjusted random score while both of the dimension-reduced

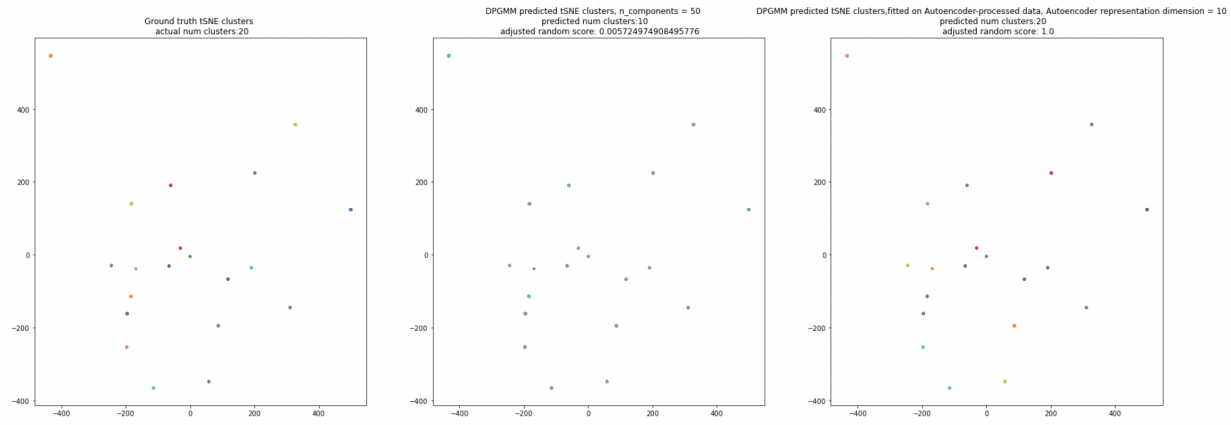

Figure 4: Performance of DPGMM with dimension reduction with Autoencoder. Left: Ground truth 20 clusters of 1,000-dimensional data with N = 10,000 with variance in each dimension 1,000, with center range of [0,100] in each dimension, projected to 2d space by t-SNE. Middle: DPGMM prediction of clustering without dimension reduction. Right: DPGMM prediction of clustering with Autoencoder dimension reduction to dimension of 10,trained with 10 epochs.

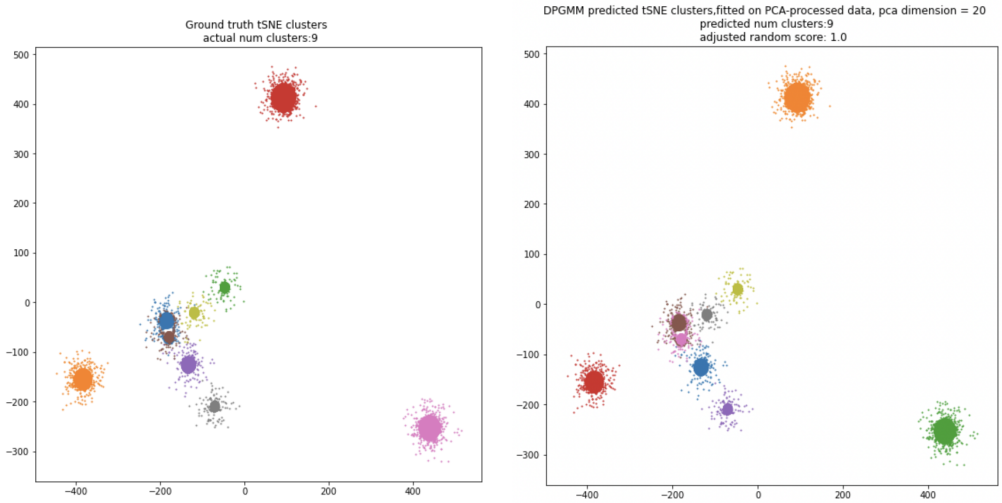

Figure 5: Performance of DPGMM with dimension reduction with large N = 100,000. Left: Ground truth 20 clusters of 100-dimensional data with variance in each dimension 500, with center range of [0,200] in each dimension, projected to 2d space by t-SNE. Right: DPGMM prediction after PCA projection to 20 dimensional space.

DPGMM yields a perfect adjusted random score of 1 and correctly identified number of clusters. This shows the effectiveness and necessity of dimension reduction techniques given a large dimensional dataset.

In Figure 4, we used the same setting and replaced PCA with auto encoder (decoder and encoder has 3 layer each) trained for 10 epochs and used the trained representation for downstream DPGMM clustering. The result indicates similar strong performance as PCA and LDA as we achieved perfect adjusted random score even when the clusters are close to each other and data from different clusters are mixed together. The number of clusters returned by the result also matches the truth.

We also experimented with large N, number of datapoints. We tried N = 100,000, and displayed the results in Figure 5. As we can see, DPGMM with PCA clearly did a great job in both identifying the true number of clusters as well as assigning membership labels to each data points.

3.4.4 DPGMM Prior Selection

We also briefly experimented with different weight concentration priors in Figure 6. As seen from the Figure, DPGMM achieves an adjusted random score of 0.86 when weight concentration prior is set to the default value of $\frac{1}{\text{number of component}} = 0.02$ and achieves an adjusted random score of 0.87 when setting prior to be 5. The discrepancy, however, is small, and we do not currently have a decisive conclusion on how that prior choice would impact our algorithmic performance.

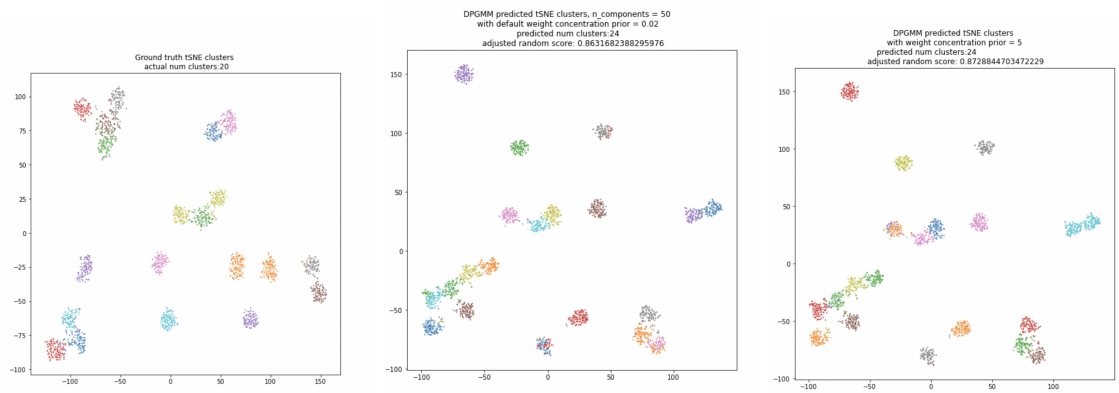

Figure 6: Performance of DPGMM with different weight concentration prior. Left: Ground truth 20 clusters of 5-dimensional data with N = 10,000 with variance in each dimension 500, with center range of [0,200] in each dimension, projected to 2d space by t-SNE. Middle: DPGMM prediction of clustering without default weight concentration prior of 0.02. Right: DPGMM prediction of clustering with weight concentration prior = 5

4 Code

See github repo.

File structure is as follows:

- experiment.ipynb is the main script conducting simulation experiment in Section **??**
- DataGenerator.py is the data generator used in this project,
- models/AutoEncoders.py is where autoencoder model is defined
- utils.py contains relevant helper methods used in the experiments

References

[Blei and Jordan, 2006] Blei, D. M. and Jordan, M. I. (2006). Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143.

[Escobar and West, 1995] Escobar, M. D. and West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the american statistical association*, 90(430):577–588.

[Ferguson, 1973] Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230.

[Goodfellow et al.,] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press.

[Görür and Edward Rasmussen, 2010] Görür, D. and Edward Rasmussen, C. (2010). Probabilistic principal component analysis. *Journal of Computer Science and Technology*, 25(4):653–664.

[Mika et al., 1999] Mika, S., Ratsch, G., Weston, J., Scholkopf, B., and Mullers, K. (1999). Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, pages 41–48.

[Minka, 2000] Minka, T. P. (2000). Automatic choice of dimensionality for pca. *M.I.T. Media Laboratory Perceptual Computing Section Technical Report*, (No. 514).

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Rumelhart and McClelland, 1987] Rumelhart, D. E. and McClelland, J. L. (1987). *Learning Internal Representations by Error Propagation*, pages 318–362.

[Sethuraman, 1994] Sethuraman, J. (1994). A constructive definition of dirichlet priors. *Statistica sinica*, pages 639–650.

[Tipping and Bishop, 1999] Tipping, M. E. and Bishop, C. M. (1999). Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Royal Statistical Society*, B(1999):611–622.

[Van der Maaten and Hinton, 2008] Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

[Wikipedia contributors, 2022] Wikipedia contributors (2022). Linear discriminant analysis — Wikipedia, the free encyclopedia. [Online; accessed 24-April-2022].