Cluster Analysis of Multivariate Data Using Scaled Dirichlet Finite Mixture Model

Eromonsele Samuel Oboh

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

December 2016

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Eromonsele Samuel Oboh**

Entitled: **Cluster Analysis of Multivariate Data Using Scaled Dirichlet Finite Mixture Model**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

____________________ Chair
Dr. Ben Hamza

____________________ External Examiner
Dr. Hassan Rivaz

____________________ Examiner
Dr. Jamal Bentahar

____________________ Supervisor
Dr. Nizar Bouguila

Approved by ____________________
Chadi Assi, Chair
Department of Concordia Institute for Information Systems Engineering

__________ 2016 ____________________
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Cluster Analysis of Multivariate Data Using Scaled Dirichlet Finite Mixture Model

Eromonsele Samuel Oboh

We have designed and implemented a finite mixture model, using the scaled Dirichlet distribution for the cluster analysis of multivariate proportional data. In this thesis, the task of cluster analysis first involves model selection which helps to discover the number of natural groupings underlying a dataset. This activity is then followed by that of estimating the model parameters for those natural groupings using the expectation maximization framework.

This work, aims to address the flexibility challenge of the Dirichlet distribution by introduction of a distribution with an extra model parameter. This is important because scientists and researchers are constantly searching for the best models that can fully describe the intrinsic characteristics of the observed data and flexible models are increasingly used to achieve such purposes.

In addition, we have applied our estimation and model selection algorithm to both synthetic and real datasets. Most importantly, we considered two areas of application in software modules defect prediction and in customer segmentation. Today, there is a growing challenge of detecting defected modules early in complex software development projects. Therefore, making these sort of machine learning algorithms crucial in driving key quality improvements that impacts bottom-line and customer satisfaction.

Acknowledgments

"Seest thou a man diligent in his business? he shall stand before kings; he shall not stand before mean men."

- Proverbs 22:29

First and foremost, I would like to thank my supervisor Prof. Nizar Bouguila. He has made the most significant impact on my life since I embarked on my master's program. Providing me with guidance, mentorship and motivation during the course of my research work. I am also grateful for the opportunity to work with you as your teaching assistant on several occasions. This experience can not be over emphasized and it was a pleasure working with you. I am also thankful to Dr. Ben Hamza for his time, advice and insights during the course of my work.

I would also like to extend my gratitude to my research colleagues in the lab. They include, Muhammed, Walid , Eddy, Elise, Toufik, Dr. Cheng, Basim, Jai, Huda, Hend. I am indeed grateful for your patience and support during the program. On a personal note, my gratitude extends to my friends and community of support outside of my research work. I appreciate your time and sacrifice and hospitality. I can't over emphasize the importance of your support to the success of my work.

And very importantly, I want to express my gratitude to my parents Mr James Oboh and Mrs Janet Oboh. I appreciate your sacrifice and the opportunity to pursue my master's degree. You provided the enabling environment financially and emotionally in my pursuit of the degree and not to mention my siblings, Oye, Uyi, Ese, Joe, Ebehi, Eghon. You are simply the best, I am so glad you were part of this journey and I miss you all.

Finally, all the thanks belongs to God Almighty. I owe it all to Him, for His grace and protection through out my journey.

Contents

List of Figures **ix**

List of Tables **xi**

1 Introduction **1**

1.1 Background 1

1.2 Objectives 2

1.3 Contributions 3

1.4 Thesis Overview 4

2 Background **5**

2.1 Model Based Framework for Clustering 5

2.1.1 Dirichlet Model 6

2.1.2 Finite Mixture Models (FMM) 7

2.2 Maximum Likelihood Estimation 7

2.2.1 Expectation Maximization Algorithm 8

2.2.2 Initialization and Convergence Criterion 9

2.2.3 Newton-Raphson Method 9

2.3 Model Selection 10

2.4 Cluster Validation 11

2.5 Generalization of the Dirichlet Model 12

3 Proposed Model **13**

3.1 Scaled Dirichlet Distribution . . . 13

3.1.1 Shape Parameter . . . 14

3.1.2 Scale Parameter . . . 15

3.2 Scaled Dirichlet Mixture Model . . . 16

3.3 Finite Scaled Dirichlet Mixture Model Estimation . . . 17

3.3.1 Newton-Raphson method . . . 20

3.3.2 Initialization and Estimation Algorithm . . . 23

3.4 MML Approach for Model Selection . . . 24

3.4.1 Fisher Information for a Mixture of Scaled Dirichlet Distributions . 25

3.4.2 Prior Distribution . . . 28

3.4.3 Complete Learning Algorithm . . . 32

4 Experimental Results **33**

4.1 Overview . . . 33

4.2 Synthetic Data . . . 33

4.2.1 Results . . . 34

4.3 Real Dataset . . . 39

4.3.1 Iris Dataset . . . 39

4.3.2 Haberman Dataset . . . 41

4.4 Software Modules Defect Prediction . . . 43

4.4.1 Complexity Metrics of Software Modules . . . 44

4.4.2 Datasets Information . . . 46

4.4.3 Performance Criteria . . . 46

4.4.4 Results . . . 48

4.4.5 Assessing Quality of Prediction . . . 49

4.4.6 Challenges Encountered . . . 50

4.5 Cluster Analysis Application in Retail 50
4.5.1 Results 52
4.5.2 Discussion 53

5 Conclusion **54**

Appendix A **57**

Appendix B **59**

Appendix C **60**

Appendix D **62**

Appendix E **64**

List of Figures

Figure 3.1 Artificial histogram plots when D = 2 describing the properties of the shape parameter. 15

Figure 3.2 Artificial histogram plots when D = 2 describing the properties of the scale parameter. 16

Figure 4.1 Artificial histogram plots for one-dimensional data. 35

Figure 4.2 Message length plot for the 3-components generated dataset. The X axis represents the number of clusters and the Y axis represents the value of the message length. 36

Figure 4.3 3-D surface plot for generated dataset 2 ($4.3a$), dataset 3 ($4.3b$), dataset 4 ($4.3c$) and dataset 5 ($4.3d$) with 2, 3, 4 and 5 components, respectively. 37

Figure 4.4 Message length plot for generated dataset 2 ($4.4a$), dataset 3 ($4.4b$), dataset 4 ($4.4c$) and dataset 5 ($4.4d$) with 2, 3, 4 and 5 components, respectively. The X axis represents the number of clusters and the Y axis represents the value of the message length. 39

Figure 4.5 Message length plot for the Iris flower dataset. The X axis represents the number of clusters and the Y axis represents the value of the message length. 40

Figure 4.6 Message length plot for the Haberman dataset. The X axis represents the number of clusters and the Y axis represents the value of the message length. 42

Figure 4.7 Message length plot for the Haberman dataset. The X axis represents the number of clusters and the Y axis represents the value of the message length. 52

List of Tables

Table 4.1 Real and estimated parameters for the generated one-dimension dataset 1 with 3 clusters. 36

Table 4.2 Real and estimated parameters for generated dataset 2, dataset 3, dataset 4, dataset 5 with 2, 3, 4 and 5 components, respectively. 38

Table 4.3 Confusion matrix using SDMM, DMM and GMM on the Iris dataset. 41

Table 4.4 Confusion matrix using SDMM on Haberman dataset. 43

Table 4.5 Test results for the SDMM, DMM and GMM classification of the Haberman dataset. 43

Table 4.6 Software modules defect prediction datasets. 46

Table 4.7 A Typical Confusion Matrix. 46

Table 4.8 Confusion matrix for software defect prediction using SDMM on JM1 dataset. 48

Table 4.9 Confusion matrix for software defect prediction using SDMM on PC1 dataset. 48

Table 4.10 Confusion matrix for software defect prediction using SDMM on Datatrieve dataset. 48

Table 4.11 Confusion matrix for software defect prediction using SDMM on MIS dataset. 48

Table 4.12 Test results for the SDMM, DMM and GMM classification of the MIS, JM1, PC1 and Datatrieve Software defect prediction datasets. 49

Table 4.13 Confusion matrix using SDMM, DMM and GMM on the Wholesale Customer dataset. 52

Table 4.14 Test results for SDMM, DMM and GMM of the Wholesale Customer dataset. 53

Chapter 1

Introduction

1.1 Background

Over the past couple of decades, the world has experienced vast growth and advancement in information technology systems including the internet. These technologies have made data generation easy (resulting in large amount of data stored in our digital universe) as well as the growing challenge of how to harness these data. Recently, a number of top technology influencers described data as the new crude oil [1]. This simply means that just as refining crude oil produces a number of petroleum products, we can also derive potential benefits from harnessing data. Moreover, the value from several data analysis applications today results in companies earning great financial returns. These applications are seen in various industries such as finance, retail, manufacturing, transportation, health services, etc.

Data clustering is one of the major techniques used in data analysis. Clustering is based on the concept of locating and separating data samples within a data set into different clusters. The data samples within a particular cluster are closely related and are widely different from data samples in a different cluster. This concept makes data clustering very useful in decision making, exploratory pattern analysis, and machine learning applications involving image segmentation, information retrieval, anomaly detection, object and character recognition [2]. There exist several approaches in data clustering and the model based approach is one that is beginning to get more attention from researchers because it provides a principled statistical way to solve data clustering problems [3].

The finite mixture model is increasingly becoming popular in data clustering. This technique is used to model distributions that represent a variety of random phenomena as well as to cluster large data sets [4]. Finite mixture is flexible and widely applicable in modeling different forms of data.

Different probabilistic methods have been implemented by researchers to model different kinds of data sets and this leads us to our research work which aims to propose a flexible probability distribution that can better find patterns within data sets without under or over-fitting.

1.2 Objectives

The major objective of this thesis is to further expand current research on finite mixture modelling by focusing on the scaled Dirichlet distribution. This involves developing a learning framework that is based on the principle of maximum likelihood estimation to infer the optimal parameters of our proposed mixture model. However, we intend to apply our proposed learning framework to address four important issues of cluster analysis:

- The challenge of choosing a flexible mixture density for model based cluster analysis.
- Estimation of the mixture model parameters.
- Model selection which estimates the number of clusters that exist within a data set.
- Evaluation and validation of the cluster analysis method.

For the purpose of this thesis, we explore the use of finite mixture model in cluster analysis. We consider an extension of the Dirichlet distribution called the scaled Dirichlet distribution to better model multivariate data vectors. As mentioned above, our proposed learning framework will employ the use of maximum likelihood estimation. Furthermore, for model selection we will implement a minimum message length (MML) criterion to estimate the optimal number of clusters inherent within our data set. We intend to validate our clustering approach on different problems. This will further elaborate the usefulness of our proposed approach in several real life applications. In particular, we will apply our data clustering algorithm to solve quality related problems and show how finite mixture modeling can be used to bring improvement in engineering systems.

1.3 Contributions

Our major contribution in this thesis is the application of a generalization of the Dirichlet distribution in finite mixture modeling. This distribution is called the scaled Dirichlet distribution. The scaled Dirichlet distribution introduces a new parameter called the scale parameter in addition to the shape parameter contained in the Dirichlet distribution. This allows for more flexibility in the modeling of engineering and natural phenomena as earlier mentioned. In our thesis, we decompose the density function of the scaled Dirichlet distribution to show this generalization as we will see in the Appendix B. We then develop a finite mixture modeling algorithm to cluster data sets as well as a model selection algorithm that tells us the optimal number of clusters that best describes a given dataset.

In addition, we will see in the experiments that our learning algorithm is tested with both synthetic and real datasets. For the purpose of our thesis, we will apply our algorithm to detect fault prone software modules and in customer segmentation.

1.4 Thesis Overview

In our thesis, we propose the scaled Dirichlet distribution as it contains an extra scale parameter to model multivariate proportional datasets. In chapter 2, we carry out a literature review on finite mixture modeling which is the foundation on which our thesis is built on. In addition, we consider the model based clustering framework, parameter estimation techniques as well as issues regarding model selection, cluster validation and generalization of the Dirichlet distribution. In chapter 3, we begin by proposing our scaled Dirichlet distribution and then provide a detailed explanation of estimating its model parameters. We also consider in depth, the algorithm for parameter estimation and model selection. Chapter 4 is dedicated to our experimental work. Here we present the results of our experiments in a clear and organized form. We also discuss the application of defect prediction within software modules and present some challenges experienced in the process. Finally in chapter 5, we conclude our work, highlight some challenges and suggest future work.

Chapter 2

Background

2.1 Model Based Framework for Clustering

Data clustering is one of the most common methods for unsupervised learning. The quality of a clustering activity usually depends on the criterion that defines the similarity between data samples. This similarity criterion can be thought of as a distance measure which can be represented as either a probabilistic model or a distance metric in an Euclidean space. We consider a model based framework for data clustering because of its statistical foundation. It stems from the assumption that our data population can be explained using a statistical model and each data cluster will contain data samples with unique model parameters different from data samples in other clusters. This is the concept of mixture modeling [5] and today it is increasingly used to solve cluster analysis problems. There are several applications of model based clustering techniques in text document classification, object identification, software quality prediction, image classification, protein sequencing [6, 7, 8, 9, 10], etc.

In model based clustering approach, we try to fit, infer and optimize what probability distribution our data sample is generated from. This process of fitting is currently a growing topic of research and researchers are exploring different probability distributions that can analyze complex forms of datasets. Researchers usually base their choice of model density used in clustering algorithms on flexibility, robustness, ease of use of the model as well as the application and the data structure of the dataset to be clustered.

2.1.1 Dirichlet Model

The Dirichlet distribution has a long history dating as far back as the 18th century. It is well known in statistics, Bayesian analysis, modeling of multivariate data, non-parametric inference, stochastic processes, reliability theory and other areas [11]. The Dirichlet distribution is the multivariate generalization of the Beta distribution. In addition, it is constrained on a simplex and has the ability to detect patterns within this constrained space [12].

Before the popularity of the Dirichlet distribution, the Gaussian distribution was used in most multivariate data clustering algorithms. The Gaussian distribution is symmetric which makes it difficult to detect asymmetric patterns in data as well as data generated from non-Gaussian sources [13]. The Dirichlet however, is very flexible and can assume several shapes depending on its parameter value [6]. It is known to have one parameter that describes the shape of the distribution. This has led to research work focused on introducing more parameters to the Dirichlet distribution that could enhance the flexibility of the model [14]. For example, if we keep the shape parameter of a Dirichlet distribution constant, we do not have another parameter that can affect the shape of the Dirichlet distribution. In addition, [15] cites the need to generalize the Dirichlet distribution because of its poor parameterization that limits its ability to better model variance and covariance.

It is also important to note that a single model is incapable of finding the entire patterns in data sets. This challenge led to a more robust technique that allows the use of more than one model to better fit and cluster data sets.

2.1.2 Finite Mixture Models (FMM)

From the previous section, we mentioned a robust technique for better modeling results. This technique is seen as a mixture model that can represent multi-component distributions underlying a dataset. By finite, we mean that the total number of components is indeed countable. In a clustering problem, these mixtures or component distributions will represent the entire pattern and number of clusters identified within the dataset. Finite mixture model (FMM) is useful in various application in computer vision, pattern recognition, signal processing, etc. In statistical pattern recognition, FMM is mostly used in an unsupervised learning approach called clustering [16]. For example, scenarios such as a company trying to define its unique customer segments or an engineer trying to summarize a collection of text documents into distinct topic categories explain some applications where we require clustering.

The parameters of the probability distribution that can fit the patterns associated with data samples are what we aim to infer using a finite mixture model. Once this fit is completed using the mixture model, the data samples are assigned to a cluster where they have the highest estimated posterior probability of belonging to.

2.2 Maximum Likelihood Estimation

To tackle the task of FMM parameter estimation, we consider the popular maximum likelihood method. For example, lets assume that our data are independent and identically distributed (IID). The maximum likelihood estimation method (MLE) helps us to find the optimal value of the mixture model parameters. It does this by selecting the optimal parameter value that maximizes the product of the likelihood function of each data sample.

In the case of estimating the parameters of a Dirichlet distribution, [17, 18] are early works that propose the MLE method. Moreover, the work of [19] explains in simple detail, an efficient iterative scheme for estimating parameters in a Dirichlet model. Though the MLE is effective in parameter estimation, it does not fully incorporate the Bayesian framework as may be seen in the maximum a posteriori estimation (MAP). The MAP estimation simply adds additional information called prior to the likelihood and then finds the parameter that maximizes the posterior probability.

In computing the MLE of a finite mixture model, we usually resort to the use of the Expectation Maximization (EM) framework. This is because the EM algorithm helps us to systematically compute the optimal model parameters while maximizing the complete data likelihood. However, the EM framework helps us to overcome the challenge of maximizing an intractable likelihood function amongst other benefits.

In addition to the MLE method for parameter estimation, in [6, 20] we observe also the use of method of moments (MOM) in the task of parameter estimation. The moments method relies on the moments equations of the model distribution that we intend to compute its parameters.

2.2.1 Expectation Maximization Algorithm

The early work by [21] presented another benefit of incorporating the EM framework in maximum likelihood estimation. Here, we see how EM is used to iteratively compute the maximum likelihood estimate of incomplete data. By incomplete data, we mean that the assignment variable that indicates the component that a particular data sample is generated from is unknown.

In [6, 8, 20] the authors explain the use of EM in solving the MLE problem when the cluster assignment is unknown. The EM algorithm is first initialized with some random model parameters in order to work. And then it iteratively uses two steps. The expectation step in which the posterior probability is computed and the maximization step where the likelihood function is maximized until convergence.

2.2.2 Initialization and Convergence Criterion

From the above section, the choice of initialization and convergence criterion is very important. The work by [6] uses K-means and method of moments to initialize the model parameters in order to reduce the possibility of EM algorithm converging at local maxima. As seen in [21], convergence occurs at a stationary point on the likelihood function. During EM iterations, convergence results when the complete log likelihood function does not change significantly over a number of EM iterations. The authors in [22] review and show comparison with the works of [18, 23] regarding how they implement initialization. In addition, also considering their drawbacks as well as emphasizing the importance for efficient re-parameterization technique. This re-parameterization usually occurs when the parameters exceed a very large number or become negative, making convergence of the EM iteration difficult.

2.2.3 Newton-Raphson Method

The works of [6, 20] present the implementation of the Dirichlet and inverted Dirichlet distribution respectively in positive data clustering. However, in the cases above, we identified the inability of calculating a closed-form solution for the maximum likelihood estimate of their model parameters. This challenge led to the use of an iterative optimization technique called the Newton-Raphson method to find the MLE for the parameter of the Dirichlet distribution.

This iterative Newton-Raphson method is known to converge very fast as compared with other optimization techniques (e.g. gradient descent, fixed point iteration, etc.) [24]. Its major drawback lies in inverting the second derivative of the likelihood function, which is call the Hessian matrix. Inverting this matrix becomes a very difficult process when we have high-dimensional data. However, in [25] an approximation technique is introduced for inverting complex matrices like the Hessian matrix in [6, 20]. Approximations are very useful during EM iterations as they allow easy computation of inverting Hessian matrices.

Numerical methods like the Newton-Raphson, if not initialized properly, can also result in estimates that are outside the parameter range. In the case of the Dirichlet distribution, the parameters must be non-negative. The early work of [26] uses the methods of moment for initializing the Beta distribution. This method is also displayed in [6, 20] to initialize the Newton-Raphson method in the case of the Dirichlet and inverted Dirichlet mixtures.

In addition to good initialization using the methods of moments, [6] introduced an interesting method for re-parameterizing the Dirichlet parameters. In the course of our research, this re-parameterization technique would be useful when we experience negative parameter values during the EM iterations.

2.3 Model Selection

One fundamental problem in mixture modeling is model selection. Model selection describes the determination of the number of components. In our case, the number of clusters that best explains the data to be clustered. It is also important to note that the EM algorithm can not be implemented if we do not specify the number of clusters or mixture components in the model. This makes the activity of model selection very important.

There are several model selection techniques used by researchers today. Some approaches used are cross validation [27], hypothesis testing and resampling to find number of clusters. Deterministic methods of model selection can be divided into two main classes [23]. The first is based on the Bayesian approach e.g. Laplace empirical criterion (LEC) and Bayesian Information Criterion (BIC). The second class is based on information theory concepts. Examples include minimum message length (MML) [23, 28, 29], Akaikes information criterion (AIC) and minimum description length (MDL) [30].

The focus of our work is on the minimum message length. The MML is special because it has both the Bayesian and information theoretic interpretation in its principle. The Bayesian interpretation is that it infers the optimal cluster number by maximizing the product between the parameter likelihood and its prior probability [28]. The Information theoretic interpretation is that the model with minimum message length is that which describes the data with minimal error [28].

2.4 Cluster Validation

This presents yet another problem in cluster analysis. It is important for a researcher to be able to confirm or make claim with high level of confidence that his clustering algorithm has yielded the right cluster labels of the data samples. We highlight a few approaches to the issue of cluster validation as seen in the literature [31].

The first approach considers carrying out significance test for example the multivariate analysis of variance (MANOVA). The authors in [31], state that though this technique may be used in literature, it is not considered as a useful validation technique. The second approach involves estimating the degree to which the clusters can be replicated. By this we mean that we have a good clustering when we get similar cluster solution across different samples of our dataset [31]. The authors in [6, 20, 23] validate their clustering algorithm with a labeled dataset using a confusion matrix to calculate the following clustering performance criteria such as: overall accuracy, average accuracy, precision, recall, etc.

Another interesting approach to cluster validation makes use of some information theoretic interpretation called mutual information [32, 33, 34].

2.5 Generalization of the Dirichlet Model

First, we take a look at the concept of over fitting. In [35] over-fitting is defined as a situation that occurs when a learning algorithm is more accurate in fitting known data and less accurate in predicting new data. The question now is that can we generalize the Dirichlet model (find a model that can better model unseen data and give useful probability models) without over-fitting? This challenge is the basis for several research efforts [36, 37, 38, 39].

Another issue we consider that has a relationship with the challenge mentioned above is how we measure a situation of over-fitting in the context of data clustering. For example, if we use a clustering algorithm on a labeled dataset and we get a hundred percent accuracy of classification, can one conclude that the clustering model suffers from over-fitting? Moreover, it is important to note that in building a model that generalizes another model we have to be cautious of the number of extra parameters that we introduce to avoid over-fitting.

The works in [38, 39, 40, 41] provide useful background concerning the generalization of the Dirichlet distribution. However, in our case we present a generalization that introduces an extra parameter to the shape parameter of the Dirichlet called the scale parameter. This distribution is known as the scaled Dirichlet distribution. [41] Argues that this distribution is flexible and can be used to model different real-life situations and phenomena. This simply means that the Dirichlet distribution is a special case of the scaled distribution. We show the mathematical proof of this in appendix B. The works of [7, 21], provide implementations of mixture modeling using various generalizations of the Dirichlet distribution.

Chapter 3

Proposed Model

3.1 Scaled Dirichlet Distribution

The scaled Dirichlet distribution as described in the previous chapter is a generalization of the Dirichlet distribution. The Dirichlet distribution is widely known to model proportional data. However, as stated by [38] when the Gamma random variables are scaled equally, the scaled Dirichlet distribution can be reduced to a Dirichlet distribution. This means that the scaled Dirichlet is formed once this equal scaling constraint is relaxed or removed. For example, let us assume that our proportional data represents the outcomes of a random event. The scaled Dirichlet distribution helps us to model or find the probability that a particular event will occur based on the proportion of its outcome.

As part of our research, we will show that the scaled Dirichlet distribution can be used as well to model proportional multivariate data that is constrained on a simplex. This means that for our data vector $\vec{X}_n = (x_{n1}, ..., ..., x_{nD})$,$\sum_{d=1}^{D} x_{nd} = G$, where G is a constant. In our case this constant is equal to 1.

Assuming that $\vec{X}_n$ follows a scaled Dirichlet distribution with parameters $\vec{\alpha}$ and $\vec{\beta}$, then the density function of the scaled Dirichlet distribution is:

$$p(\vec{X}_n|\theta) = \frac{\Gamma(\alpha_+)}{\prod_{d=1}^{D}\Gamma(\alpha_d)} \frac{\prod_{d=1}^{D}\beta_d^{\alpha_d} x_{nd}^{\alpha_d-1}}{(\sum_{d=1}^{D}\beta_d x_{nd})^{\alpha_+}} \tag{1}$$

Where Γ denotes the Gamma function, $\alpha_+ = \sum_{d=1}^{D}\alpha_d$ and $\theta = (\vec{\alpha}, \vec{\beta})$ is our model parameter. $\vec{\alpha} = (\alpha_1, ..., \alpha_D)$ is the shape parameter and $\vec{\beta} = (\beta_1, ..., \beta_D)$ is the scale parameter of the scaled Dirichlet distribution.

If we assume that a set $\mathcal{X} = \{\vec{X}_1, \vec{X}_2, ..., \vec{X}_N\}$ composed of data vectors is independent and identically distributed (I.I.D), the resulting likelihood is

$$p(\mathcal{X}|\theta) = \prod_{n=1}^{N}(\frac{\Gamma(\alpha_+)}{\prod_{d=1}^{D}\Gamma(\alpha_d)} \frac{\prod_{d=1}^{D}\beta_d^{\alpha_d} x_{nd}^{\alpha_d-1}}{(\sum_{d=1}^{D}\beta_d x_{nd})^{\alpha_+}}) \tag{2}$$

3.1.1 Shape Parameter

The shape parameter simply describes the form or shape of the scaled Dirichlet distribution. The flexibility of this parameter is very important in finding patterns and shapes inherent in a dataset. In fig 3.1 we see, in a $2D$ density plot, that when we have a shape parameter less than 1, it results in a convex density plot while higher shape parameter values result in concave plots of varying shapes.

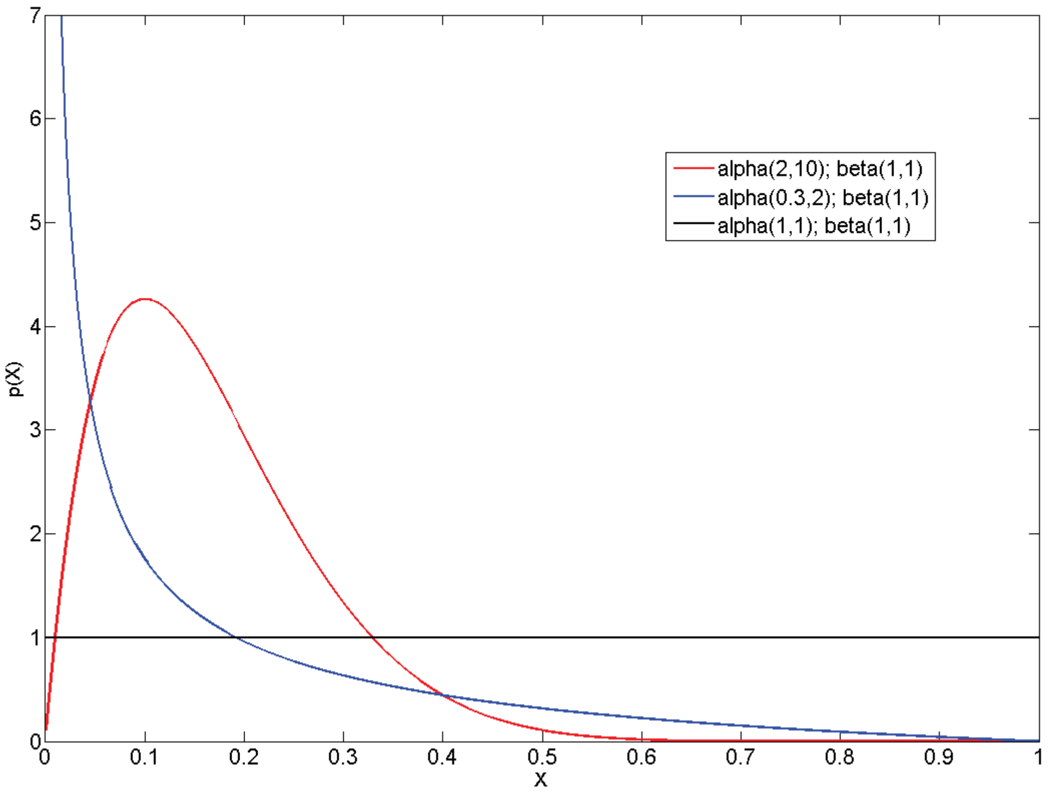


Figure 3.1: Artificial histogram plots when D = 2 describing the properties of the shape parameter.

3.1.2 Scale Parameter

The scale parameter simply controls how the density plot is spread out. In addition, we also notice that the shape of the density is invariant, irrespective of the value of a constant or uniform scale parameter. The mathematical proof of this, is seen in Appendix A.

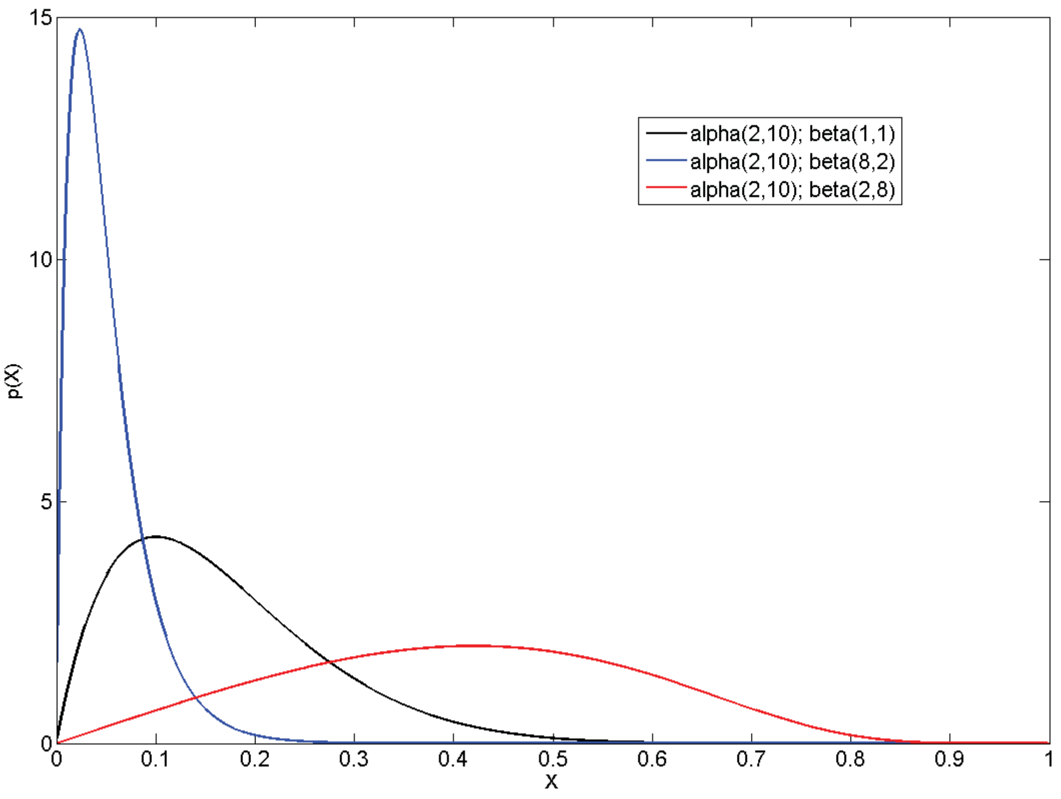


Figure 3.2: Artificial histogram plots when D = 2 describing the properties of the scale parameter.

From figure 3.2, we observe how the varying scale parameter values in the red and blue colored plot affect the spread of the distribution with constant scale parameter.

3.2 Scaled Dirichlet Mixture Model

Formally, to introduce the finite mixtures with the scaled Dirichlet distribution, we assume that $\mathcal{X} = \{\vec{X}_1, \vec{X}_2, ..., \vec{X}_N\}$ our dataset is made up of N vectors and each sample vector $\vec{X}_n = (x_{n1}, ..., ..., x_{nD})$ is D-dimensional. The general idea in mixture modeling is that we assume that our data population is generated from a mixture of sub populations. These sub-populations are usually called clusters. In the case of K-means, these clusters are defined by their cluster centroids. However, in the case of model based clustering we assume our clusters to be defined by the model parameters. So, in the scaled Dirichlet mixture model we intend to discover a mixture of K-components that define our dataset. This mixture model is expressed as:

$$p(\vec{X}_n|\Theta) = \sum_{j=1}^{K} p_j p(\vec{X}_n|\vec{\alpha_j}, \vec{\beta_j}) \tag{3}$$

where the p_j are the mixing weights defined by $\sum_{j=1}^{K} p_j = 1, p_j > 0$. Then the likelihood will be equal to

$$p(\mathcal{X}|\Theta) = \prod_{n=1}^{N} \sum_{j=1}^{K} p_j p(\vec{X}_n|\vec{\alpha_j}, \vec{\beta_j}) \tag{4}$$

We denote the set of parameters by $\Theta = \{\vec{P} = (p_1, ..., ...p_k); \vec{\theta} = ((\vec{\alpha}_1, ..., ...\vec{\alpha}_K), (\vec{\beta}_1, ..., ...\vec{\beta}_K))\}$

3.3 Finite Scaled Dirichlet Mixture Model Estimation

A very significant problem in finite mixture modeling is the estimation of its parameter as identified above. Here, we want to estimate the model parameters of the scaled Dirichlet distribution (SDD). We will make use of the maximum likelihood estimation (MLE) approach because it has become widely popular and acceptable in solving this problem.

The expectation maximization (EM) algorithm is used to compute the maximum likelihood estimates given that we have unobserved latent variables. For the ease of estimating the model parameters, we maximize the log of the likelihood function:

$$\log{(p(\mathcal{X}|\Theta))} = L(\Theta, \mathcal{X}) = \sum_{n=1}^{N} \log{(\sum_{j=1}^{K} p_j \, p(\vec{X}_n|\vec{\alpha_j}, \vec{\beta_j}))} \tag{5}$$

The parameter estimation by maximizing the log-likelihood function is achieved using EM algorithm. Let $Z = (\vec{z}_1, ..., ..., \vec{z}_N)$ denote the hidden assignment or latent variables that are unobserved, where $\vec{z}_n$ is the assignment vector with respect to each jth component for a data sample and z_{nj} is the assignment of a data sample to the jth cluster. In addition, z_{nj} is equal to one if the data sample belongs to cluster j and zero if otherwise.

With the data combined with the latent variables, we can find the Θ_{MLE}. We shall also call (X, Z) our complete data and its log likelihood is as follows:

$$\log\left(p(\mathcal{X}, Z|\Theta)\right) = L(\Theta, \mathcal{X}, Z) = \sum_{n=1}^{N}\sum_{j=1}^{K} z_{nj}\left(\log\ p_j + \log\ p(\vec{X}_n|\vec{\alpha_j}, \vec{\beta_j})\right) \tag{6}$$

where

$$\log\ p(\vec{X}_n|\vec{\alpha_j}, \vec{\beta_j}) = \sum_{n=1}^{N}((\log\ \Gamma(\alpha_+) - \sum_{d=1}^{D}\log\ \Gamma(\alpha_d)) + \sum_{d=1}^{D}[\alpha_d \log\ \beta_d + (\alpha_d - 1)\log\ x_{nd}] \tag{7}$$

$$- \alpha_+ \log\left(\sum_{d=1}^{D}\beta_d x_{nd}\right))$$

$$= \sum_{n=1}^{N}((\log\Gamma(\alpha_+) - \sum_{d=1}^{D}\log\Gamma(\alpha_d)) + \sum_{d=1}^{D}\alpha_d\log\beta_d + \sum_{d=1}^{D}(\alpha_d - 1)\log x_{nd} - \alpha_+\log(\sum_{d=1}^{D}\beta_d x_{nd})) \tag{8}$$

In the E-step of the EM algorithm, the goal is to compute the probability of an object belonging to a cluster j. This, more or less, can be seen as a simple computation of the posterior probability of each data vector assigned to a particular cluster j. The probability of vector $\vec{X}_n$ belonging to cluster j is given by:

$$\hat{z}_{nj} = \frac{p_j p(\vec{X}_n|\vec{\alpha_j}, \vec{\beta_j})}{\sum_{j=1}^{K} p_j p(\vec{X}_n|\vec{\alpha_j}, \vec{\beta_j})} \tag{9}$$

In the M-step, we update the model parameters which result in maximizing or increasing the expectation of the complete log likelihood given by:

$$\log\ p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N}\sum_{j=1}^{K} \hat{z}_{nj}(\log\ p_j + \log\ p(\vec{X}_n|\vec{\alpha_j}, \vec{\beta_j})) \tag{10}$$

We compute the Θ_{MLE} by optimizing the complete log-likelihood.

The maximization of $\log\ p(\mathcal{X}, Z|\Theta)$ under the constraint $\sum_{j=1}^{K} p_j = 1$ gives:

$$p_j = \frac{1}{N}\sum_{n=1}^{N} p(j|\vec{X}_n, \vec{\alpha_j}, \vec{\beta_j}) = \frac{1}{N}\sum_{n=1}^{N} \hat{z}_{nj} \tag{11}$$

To compute the optimal parameters for $(\vec{\alpha_j}, \vec{\beta_j})$ via the MLE framework, we simply take the derivative of the log-likelihood and find the θ_{MLE} when the derivative is equal to zero.

$$\log\ p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N} \hat{z}_{nj} \sum_{j=1}^{K} (\log(p_j) + \log(p(\vec{X}_n|\theta_j)) \tag{12}$$

Calculating the derivative with respect to $\alpha_{jd}, d = 1, ..., D$, we obtain:

$$\frac{\partial}{\partial \alpha_{jd}} \log\ p(\mathcal{X}, Z|\Theta) = G(\alpha) = \sum_{n=1}^{N} \hat{z}_{nj} \frac{\partial}{\partial \alpha_j} \log(p(\vec{X}_n|\vec{\alpha_j}))$$

$$= \sum_{n=1}^{N} \hat{z}_{nj}(\Psi(\alpha_+) - \Psi(\alpha_d) + \log\ \beta_d + \log\ x_{nd} - \log(\sum_{d=1}^{D} \beta_d x_{nd})) \tag{13}$$

Calculating the derivative with respect to $\beta_{jd}, d = 1, ..., D$, we obtain:

$$\frac{\partial}{\partial \beta_{jd}} \log\ p(\mathcal{X}, Z|\Theta) = G(\beta) = \sum_{n=1}^{N} \hat{z}_{nj} \frac{\partial}{\partial \beta_j} \log(p(\vec{X}_n|\vec{\beta_j}))$$

$$= \sum_{n=1}^{N} \hat{z}_{nj}(\frac{\alpha_d}{\beta_d} \frac{\alpha_+ x_{nd}}{\sum_{d=1}^{D} \beta_d x_{nd}}) \tag{14}$$

where $\psi(\alpha_d) = \frac{\Gamma'(\alpha_d)}{\Gamma(\alpha_d)}$ is called the digamma function.

3.3.1 Newton-Raphson method

Considering Eqns.13 and 14, to find the MLE for our model parameters, we can see that a closed form solution does not exist. Therefore, we employ an iterative multivariate optimization technique called Newton-Raphson method to find our model parameters. This Newton-Raphson method will help us to find the roots of the log-likelihood function. In other words, we are simply using this optimization technique to carry out the maximization step of the EM algorithm. The Newton-Raphson method can be expressed as follows:

$$\theta_j^{new} = \theta_j^{old} - H^{-1}G \approx [\alpha_j^{new} = \alpha_j^{old} - H^{-1}G; \beta_j^{new} = \beta_j^{old} - H^{-1}G] \tag{15}$$

Where H is called the Hessian matrix and G is the gradient. Our Hessian matrix is a $2D$ by $2D$ matrix as shown below:

$$\begin{pmatrix} \frac{\partial^2 L}{\partial \alpha_{j1}^2} & \cdots & \frac{\partial^2 L}{\partial \alpha_{j1}\alpha_{jD}} & \frac{\partial^2 L}{\partial \alpha_{j1}\beta_{jD+1}} & \cdots & \frac{\partial^2 L}{\partial \alpha_{j1}\beta_{j2D}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 L}{\partial \alpha_{jD}\alpha_{j1}} & \cdots & \frac{\partial^2 L}{\partial \alpha_{jD}^2} & \frac{\partial^2 L}{\partial \alpha_{jD+1}\beta_{jD+1}} & \cdots & \frac{\partial^2 L}{\partial \alpha_{jD+1}\beta_{j2D}} \\ \frac{\partial^2 L}{\partial \beta_{jD+1}\alpha_{j1}} & \cdots & \frac{\partial^2 L}{\partial \beta_{jD+1}\alpha_{jD}} & \frac{\partial^2 L}{\partial \beta_{jD+1}^2} & \cdots & \frac{\partial^2 L}{\partial \beta_{jD+1}\beta_{j2D}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 L}{\partial \beta_{j2D}\alpha_{j1}} & \cdots & \frac{\partial^2 L}{\partial \beta_{jD+1}\alpha_{jD}} & \frac{\partial^2 L}{\partial \beta_{j2D}\beta_{jD+1}} & \cdots & \frac{\partial^2 L}{\partial \beta_{j2D}^2} \end{pmatrix} \tag{16}$$

To calculate this Hessian matrix, we must compute the second and mixed derivatives of our log-likelihood function. Calculating the second and mixed derivative with respect to $\alpha_{jd}, d = 1, ..., D$, we obtain:

$$\frac{\partial^2}{\partial \alpha_{jd}^2} \log\ p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N} \hat{z}_{nj}\ \psi^{'}(\alpha_+) - \psi^{'}(\alpha_d) \tag{17}$$

$$\frac{\partial^2}{\partial \alpha_{jd_1}\alpha_{jd_2}} \log\ p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N} \hat{z}_{nj}\ \psi^{'}(\alpha_+),\ d_1 \neq d_2,\ d_1, d_2 = 1, ..., D \tag{18}$$

$$H_{(\alpha_{jd},\alpha_{jd})} = \sum_{n=1}^{N} \hat{z}_{nj} \times \begin{pmatrix} \psi^{'}(\alpha_{+}) - \psi^{'}(\alpha_{1}) & \cdots & \psi^{'}(\alpha_{+}) \\ \vdots & \ddots & \vdots \\ \psi^{'}(\alpha_{+}) & \cdots & \psi^{'}(\alpha_{+}) - \psi^{'}(\alpha_{D}) \end{pmatrix} \tag{19}$$

Calculating the second and mixed derivative with respect to α_{jd} and $\beta_{jd}, d = 1, ..., D$ we obtain:

$$\frac{\partial^2}{\partial \alpha_{jd} \beta_{jd}} \log\ p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N} \hat{z}_{nj}\ \frac{1}{\beta_d} - \frac{x_{nd}}{\sum_{d=1}^{D} \beta_d x_{nd}} \tag{20}$$

$$\frac{\partial^2}{\partial \alpha_{jd_1} \beta_{jd_2}} \log\ p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N} \hat{z}_{nj}\ - \frac{x_{nd}}{\sum_{d=1}^{D} \beta_d x_{nd}}, d_1 \neq d_2\ d_1, d_2 = 1, ..., D \tag{21}$$

$$H_{(\alpha_{jd},\beta_{jd})} = \sum_{n=1}^{N} \hat{z}_{nj} \times \begin{pmatrix} \frac{1}{\beta_1} - \frac{x_{nd_1}}{\sum_{d=1}^{D} \beta_d x_{nd}} & \cdots & -\frac{x_{nD}}{\sum_{d=1}^{D} \beta_d x_{nd}} \\ \vdots & \ddots & \vdots \\ -\frac{x_{nd_1}}{\sum_{d=1}^{D} \beta_d x_{nd}} & \cdots & \frac{1}{\beta_D} - \frac{x_{nD}}{\sum_{d=1}^{D} \beta_d x_{nd}} \end{pmatrix} \tag{22}$$

Calculating the second and mixed derivative with respect to β_{jd} and $\alpha_{jd}, d = 1, ..., D$, we obtain:

$$\frac{\partial^2}{\partial \beta_{jd} \alpha_{jd}} \log\ p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N} \hat{z}_{nj}\ \frac{1}{\beta_d} - \frac{x_{nd}}{\sum_{d=1}^{D} \beta_d x_{nd}} \tag{23}$$

$$\frac{\partial^2}{\partial \beta_{jd_1} \alpha_{jd_2}} \log\ p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N} \hat{z}_{nj}\ - \frac{x_{nd}}{\sum_{d=1}^{D} \beta_d x_{nd}}, d_1 \neq d_2\ d_1, d_2 = 1, ..., D \tag{24}$$

$$H_{(\beta_{jd},\alpha_{jd})} = \sum_{n=1}^{N} \hat{z}_{nj} \times \begin{pmatrix} \frac{1}{\beta_1} - \frac{x_{nd_1}}{\sum_{d=1}^{D} \beta_d x_{nd}} & \cdots & -\frac{x_{nd_1}}{\sum_{d=1}^{D} \beta_d x_{nd}} \\ \vdots & \ddots & \vdots \\ -\frac{x_{nD}}{\sum_{d=1}^{D} \beta_d x_{nd}} & \cdots & \frac{1}{\beta_D} - \frac{x_{nD}}{\sum_{d=1}^{D} \beta_d x_{nd}} \end{pmatrix} \tag{25}$$

Calculating the second and mixed derivative with respect to $\beta_{jd}, d = 1, ..., D$, we obtain:

$$\frac{\partial^2}{\partial \beta_{jd}^2} \log\, p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N} \hat{z}_{nj}\, \frac{\alpha_+ x_{nd}^2}{(\sum_{d=1}^{D} \beta_d x_{nd})^2} - \frac{\alpha_d}{\beta_d^2} \tag{26}$$

$$\frac{\partial^2}{\partial \beta_{jd_1} \beta_{jd_2}} \log\, p(\mathcal{X}, Z|\Theta) = \sum_{n=1}^{N} \hat{z}_{nj}\, \frac{\alpha_+ x_{nd_1} x_{nd_2}}{(\sum_{d=1}^{D} \beta_d x_{nd})^2} \tag{27}$$

$$H_{(\beta_{jd},\beta_{jd})} = \sum_{n=1}^{N} \hat{z}_{nj} \times \begin{pmatrix} \frac{\alpha_+ x_{n1}^2}{(\sum_{d=1}^{D} \beta_d x_{nd})^2} - \frac{\alpha_1}{\beta_1^2} & \cdots & \frac{\alpha_+ x_{n1} x_{nD}}{(\sum_{d=1}^{D} \beta_d x_{nd})^2} \\ \vdots & \ddots & \vdots \\ \frac{\alpha_+ x_{nD} x_{n1}}{(\sum_{d=1}^{D} \beta_d x_{nd})^2} & \cdots & \frac{\alpha_+ x_{nD}^2}{(\sum_{d=1}^{D} \beta_d x_{nd})^2} - \frac{\alpha_D}{\beta_D^2} \end{pmatrix} \tag{28}$$

Where ψ' is the trigamma function.

The complete block Hessian matrix H_j has to be transformed to its inverse before it can be used in the Newton-Raphson maximization step $\theta_j^{new} = \theta_j^{old} - H^{-1}G$. The complete Hessian block matrix is given by:

$$H_j = \begin{bmatrix} H_{(\alpha_{jd},\alpha_{jd})} & H_{(\alpha_{jd},\beta_{jd})} \\ H_{(\beta_{jd},\alpha_{jd})} & H_{(\beta_{jd},\beta_{jd})} \end{bmatrix}$$

The inverse of a complete Hessian matrix is difficult to compute. In our case, the Hessian block matrix needs to be positive or semi positive definite before its inverse can be computed. In order to relax this constraint, we make use of its diagonal approximation. This approximation, allows the inverse to be trivially computed.

3.3.2 Initialization and Estimation Algorithm

From [6] we know that the maximum likelihood function of a mixture model is not globally concave. This notion as well as the requirement of initial parameter guesses for the EM algorithm makes the process of initialization very important. For our algorithm to perform optimally, we must initialize properly in order to avoid converging to a local maximum. However, avoiding this local maximum cannot be guaranteed using the EM algorithm. To initialize the p_j parameter, we use the K-means algorithm. And to initialize the model parameters of the scaled Dirichlet mixture $(\vec{\alpha_j}, \vec{\beta_j})$ we make use of the method of moments. The method of moments simply estimates the model parameters based on their moment equations. In the case of the scaled Dirichlet distribution, a closed form solution for its moment equations does not exist in the literature [38]. However, for the purpose of our work, we will initialize using the moment equation of the Dirichlet distribution.

To initialize the $\vec{\beta_j}$ parameter, we assign it a value of 1. Then, it is our desire that during the iterations, the $\vec{\beta_j}$ parameter would be updated and then take its natural value in relation to the observed data.

Initialization Algorithm

(1) Apply K-means algorithm to the data $\mathcal{X}$ to obtain the pre-defined K clusters and its elements.

(2) Calculate the p_j parameter as.

$p_j = \frac{Number\ of\ elements\ in\ cluster\ j}{N}$

(3) Apply the method of moment [20] for each cluster j to obtain the shape parameter vector $\vec{\alpha_j}$.

(4) Initialize the scale parameter vector $\vec{\beta_j}$ with a vector of ones (initialize with equal scaling).

Parameter Estimation Algorithm

(1) Input: the complete data $\mathcal{X}$ and number of clusters K

(2) Apply the Initialization Algorithm

(3) Repeat until convergence criterion is met:

  (a) E Step: Compute the posterior probability of an object assigned to a cluster $\hat{z_{nj}}$ using Eqn.9

  (b) M Step:

    i. Update p_j using Eq.11

    ii. Update β_j and α_j using Eqn.15

(4) If convergence test is passed, terminate and return final parameter estimates and cluster probabilities.

3.4 MML Approach for Model Selection

In the previous section, we noted that we pre-defined the number of clusters before executing the EM algorithm. The role of model selection is to help us infer the number of optimal clusters. First, we assume that our data is fundamentally modeled by a mixture of distributions. The minimum message length (MML) is the approach we implement to solve the problem of model selection.

In reference to information theory, the optimal number of clusters is that which requires minimum information to transmit the data from sender to receiver efficiently [28]. The MML is based on this concept and for a mixture of distributions it is expressed below as:

$$MessLength = -log(\frac{h(\Theta)\, p(\mathcal{X}|\theta)}{\sqrt{|F(\Theta)|}}) + N_p(-\frac{1}{2}log(12) + \frac{1}{2})$$

$$= -log\, h(\Theta) - log\, p(\mathcal{X}|\Theta) + \frac{1}{2} log(|F(\Theta)|) + \frac{N_p}{2}(1 - log(12)) \tag{29}$$

Where $h(\Theta)$ is the prior probability distribution, $\mathcal{X}$ is the data, Θ is the vector of parameters, N_p is the number of free parameters to be estimated and is equal to $K(2D+1)-1$ [38], $p(\mathcal{X}, Z|\theta)$ is the complete data log likelihood, $|F(\Theta)|$ is the determinant of the fisher information matrix which is derived from taking the second derivative of the negative log-likelihood.

Subsequently, we will first develop the Fisher information for a mixture of scaled Dirichlet distributions and then propose a prior distribution about our knowledge of its parameters.

3.4.1 Fisher Information for a Mixture of Scaled Dirichlet Distributions

The Fisher matrix is sometimes called the curvature matrix. This matrix explains the curvature of the likelihood function around its maximum and is the expected value of the negative of the Hessian matrix, which is simply the expected value of the negative of the second derivative of the log-likelihood function [23]. In the case of a mixture model, the authors in [16] proposed that the Fisher information matrix can be computed after the data vectors have been assigned to their respective clusters.

The determinant of the complete-data Fisher information matrix is given as the product of the determinant of the Fisher information of $\theta = (\vec{\alpha_j}, \vec{\beta_j})$ and the determinant of the Fisher information of mixing parameters p_j [23]. This is shown below as follows;

$$|F(\vec{\Theta})| = |F(\vec{P})| \prod_{j=1}^{K} |F(\vec{\alpha_j}, \vec{\beta_j})| \tag{30}$$

$$|F(\vec{\theta})| = \prod_{j=1}^{K} |F(\vec{\alpha_j}, \vec{\beta_j})|$$

The Fisher information of the cluster mixing weights is $F(\vec{P}) = F(p_1, p_2, ..., p_K)$. Its determinant is calculated in [23] as:

$$|F(p_1, p_2, ..., p_K)| = \frac{N^{K-1}}{\prod_{j=1}^{K} p_j} \tag{31}$$

where $p_1 + p_2 + ... + p_K = 1$, for all $j :\ p \geq 0$ and where N is the total number of data observations, and p_j is the mixing weight of each cluster.

$|F(\vec{\alpha_j}, \vec{\beta_j})|$ is the Fisher information of the scaled Dirichlet distribution with parameter $(\vec{\alpha_j}, \vec{\beta_j})$. To find its determinant considering the method proposed by [16], we assume that the jth cluster of the mixture will contain $X_j = (\vec{X}_l, ..., \vec{X}_{l+n_j-1})$ data samples, where $l \leq N$ and n_j is the number of observations in cluster j, with parameter $\vec{\alpha_j}, \vec{\beta_j}$.

We determine $F(\vec{\alpha_j}, \vec{\beta_j})$ by taking the negative of the second derivative of its log-likelihood function:

$$-\,log\, p(\mathcal{X}|\vec{\alpha_j}, \vec{\beta_j}) = -log(\prod_{n=l}^{l+n_j-1} p(\vec{X}|\theta_K)) = -(\sum_{n=l}^{l+n_j-1} log\, p(\vec{X}|\theta_K)) \tag{32}$$

First order derivative is also called the Fisher score function. Calculating this derivative with respect to α_{jd} we obtain:

$$-\frac{\partial log\, p(\mathcal{X}|\vec{\alpha_j}, \vec{\beta_j})}{\partial \alpha_{jd}} = n_j(\Psi(\alpha_+) - \Psi(\alpha_d) + log\beta_d) - \sum_{n=1}^{N}(log\, x_{nd} + log(\sum_{d=1}^{D} log\, \beta_d x_{nd})) \tag{33}$$

Calculating the first order derivative with respect to β_{jd} we obtain:

$$-\frac{\partial log\, p(\mathcal{X}|\vec{\alpha_j}, \vec{\beta_j})}{\partial \beta_{jd}} = n_j(\frac{\alpha_d}{\beta_d}) + \sum_{n=1}^{N}(\frac{\alpha_+ x_{nd}}{\sum_{d=1}^{D} \beta_d x_{nd}}) \tag{34}$$

Calculating the second and mixed derivative with respect to $\alpha_{jd}, d = 1, ..., D$, we obtain:

$$-\frac{\partial^2 log\, p(\mathcal{X}|\vec{\alpha_j}, \vec{\beta_j})}{\partial\alpha_{jd}^2} = -n_j(\psi^{'}(\alpha_+) - \psi^{'}(\alpha_d)) \tag{35}$$

$$-\frac{\partial^2 log\, p(\mathcal{X}|\vec{\alpha_j}, \vec{\beta_j})}{\partial\alpha_{jd_1}\alpha_{jd_2}} = -n_j\, \psi^{'}(\alpha_+),\ d_1 \neq d_2,\ d_1, d_2 = 1, ..., D \tag{36}$$

Calculating the second and mixed derivative with respect to α_{jd} and $\beta_{jd}, d = 1, ..., D$, we obtain:

$$-\frac{\partial^2 log\, p(\mathcal{X}|\vec{\alpha_j}, \vec{\beta_j})}{\partial\alpha_{jd}\beta_{jd}} = -n_j(\frac{1}{\beta_d}) + \sum_{n=1}^{N}(\frac{x_{nd}}{\sum_{d=1}^{D}\beta_d x_{nd}}) \tag{37}$$

$$-\frac{\partial^2 log\, p(\mathcal{X}|\vec{\alpha_j}, \vec{\beta_j})}{\partial\alpha_{jd_1}\beta_{jd_2}} = \sum_{n=1}^{N}(\frac{x_{nd}}{\sum_{d=1}^{D}\beta_d x_{nd}}), d_1 \neq d_2\ d_1, d_2 = 1, ..., D \tag{38}$$

Calculating the second and mixed derivative with respect to β_{jd} and $\alpha_{jd}, d = 1, ..., D$, we obtain:

$$-\frac{\partial^2 log\, p(\mathcal{X}|\vec{\alpha_j}, \vec{\beta_j})}{\partial\beta_{jd}\alpha_{jd}} = -n_j(\frac{1}{\beta_d}) + \sum_{n=1}^{N}(\frac{x_{nd}}{\sum_{d=1}^{D}\beta_d x_{nd}}) \tag{39}$$

$$-\frac{\partial^2 log\, p(\mathcal{X}|\vec{\alpha_j}, \vec{\beta_j})}{\partial\beta_{jd_1}\alpha_{jd_2}} = \sum_{n=1}^{N}(\frac{x_{nd}}{\sum_{d=1}^{D}\beta_d x_{nd}}), d_1 \neq d_2\ d_1, d_2 = 1, ..., D \tag{40}$$

Calculating the second and mixed derivative with respect to $\beta_{jd}, d = 1, ..., D$, we obtain:

$$-\frac{\partial^2 log\, p(\mathcal{X}|\vec{\alpha_j},\vec{\beta_j})}{\partial \beta_{jd}^2} = -\sum_{n=1}^{N} \frac{\alpha_+ x_{nd}^2}{(\sum_{d=1}^{D}\beta_d x_{nd})^2} + n_j(\frac{\alpha_d}{\beta_d^2}) \tag{41}$$

$$-\frac{\partial^2 log\, p(\mathcal{X}|\vec{\alpha_j},\vec{\beta_j})}{\partial \beta_{jd_1}\beta_{jd_2}} = -\sum_{n=1}^{N} \frac{\alpha_+ x_{nd_1} x_{nd_2}}{(\sum_{d=1}^{D}\beta_d x_{nd})^2} \tag{42}$$

$F(\vec{\alpha_j},\vec{\beta_j})$ and can be represented in the following form:

$$\begin{bmatrix} F_{(\alpha_{jd},\alpha_{jd})} & F_{(\alpha_{jd},\beta_{jd})} \\ F_{(\beta_{jd},\alpha_{jd})} & F_{(\beta_{jd},\beta_{jd})} \end{bmatrix} \tag{43}$$

Where each of the sub blocks $F_{(\alpha_{jd},\alpha_{jd})}, F_{(\alpha_{jd},\beta_{jd})}, F_{(\beta_{jd},\alpha_{jd})}, F_{(\beta_{jd},\beta_{jd})}$ is a $(D \times D)$ symmetric matrix.

We compute the determinant $|F(\vec{\alpha}_{jd},\vec{\beta}_{jd})|$ of this block matrix, using the solution provided in [42].

3.4.2 Prior Distribution

The capability of the MML criterion is dependent on the choice of prior distribution $h(\Theta)$ for the parameters of the scaled Dirichlet mixture model. We will have to assign distributions that better describe our prior knowledge of the vectors of mixing parameter and the parameter vectors of the scaled Dirichlet finite mixture model. Since these parameters are independent of each other, we represent $h(\Theta)$ as follows;

$$h(\Theta) = h(\vec{P})h(\alpha)h(\beta) \tag{44}$$

Mixing Weight Prior $h(\vec{P})$

Since we know that the mixing parameter $\vec{P}$ is defined on the simplex $P_1, P_2, ..., P_K : \sum_{j=1}^{K} P_j = 1$. We assume the probability density of $h(\vec{P})$ prior follows a Dirichlet distribution. This is because of its suitability in modeling proportional vectors and this prior is represented as follows;

$$h(P_1, P_2, ..., P_K) = \frac{\Gamma(\sum_{j=1}^{K} \eta_j)}{\prod_{j=1}^{K} \Gamma(\eta_j)} \prod_{j=1}^{K} p_j^{\eta_j - 1} \tag{45}$$

$\vec{\eta} = (\eta_1, ..., \eta_K)$ represents the parameter vector for the Dirichlet distribution. And we choose a uniform prior for this parameter $\vec{\eta}$, ($\eta_1 = 1, ..., \eta_K = 1$).

This allows us to simplify Eq.45 and we obtain:

$$h(P_1, P_2, ..., P_K) = \Gamma(K) = (K-1)! \tag{46}$$

Shape Parameter Prior $h(\alpha)$

For $h(\alpha)$,we consider the $\vec{\alpha}_j : j = 1, ..., K$ are independent and we obtain;

$$h(\alpha) = \prod_{j=1}^{K} h(\alpha_j) \tag{47}$$

In calculating $\vec{\alpha}$, we assume that we don't have prior knowledge or information about the parameter $(\alpha_{jd}), d = 1, ..., D$ and because of this we want this prior to have minimal effect on the posterior [43]. So to achieve this, we assign the prior with a uniform distribution using the principle of ignorance. An assume that $h(\alpha_{jd})$ is uniform over the range of $[0, e^6 \frac{\|\hat{\alpha}_j\|}{\hat{\alpha}_{jd}}]$. This high value is inferred experimentally as seen in [20], so that $[\alpha_{jd} < e^6 \frac{\|\hat{\alpha}_j\|}{\hat{\alpha}_{jd}}]$. Where $\vec{\alpha_j}$ is the estimated parameter vector.

$$h(\alpha_{jd}) = e^6 \frac{\hat{\alpha}_{jd}}{\|\hat{\alpha}_j\|} \tag{48}$$

$$h(\alpha_j) = \prod_{d=1}^{D} e^{6} \frac{\hat{\alpha}_{jd}}{\|\hat{\alpha}_j\|}$$

$$= \frac{e^{-6D}}{\|\hat{\alpha}_j\|^D} \prod_{d=1}^{D} \hat{\alpha}_{jd} \tag{49}$$

We input Eq.48 and Eq.49 into Eq.47 and we get:

$$h(\alpha) = \prod_{j=1}^{K} \left(\frac{e^{-6D}}{\|\hat{\alpha}_j\|^D} \prod_{d=1}^{D} \hat{\alpha}_{jd}\right) \tag{50}$$

$$= e^{-6KD} \prod_{j=1}^{K} \frac{\prod_{d=1}^{D} \hat{\alpha}_{jd}}{\|\hat{\alpha}_j\|^D} \tag{51}$$

Take the logarithm of Eq.51

$$log\, h(\alpha) = -6KD - D\sum_{j=1}^{K} log(\|\hat{\alpha}\|) + \sum_{j=1}^{K}\sum_{d=1}^{D} log(\hat{\alpha}_{jd}) \tag{52}$$

Scale Parameter Prior $h(\beta)$

For the $h(\beta)$, we consider the scale parameter $\vec{\beta_j} : j = 1, ..., K$ to be independent so we obtain;

$$h(\beta) = \prod_{j=1}^{K} h(\beta_j) \tag{53}$$

Since we also don't have prior knowledge about the scale parameter $(\beta_{jd}), d = 1, ..., D$ and we assign a uniform prior for each β_{jd}. For each β_{jd} we assign a uniform distribution and with the use of the principle of ignorance, we assume that β_{jd} falls within the range $[0, e^{6}\frac{\|\hat{\beta}_j\|}{\hat{\beta}_{jd}}]$. This range is assumed to be a sufficiently high value to accommodate the scale parameter, where the estimated parameter vector is $\vec{\beta_j}$ and the norm of the scale parameter vector is $\|\hat{\beta}_j\|$.

$$h(\beta_{jd}) = e^{6} \frac{\hat{\beta}_{jd}}{\|\hat{\beta}_j\|} \tag{54}$$

$$h(\beta_j) = \prod_{d=1}^{D} e^{6} \frac{\hat{\beta}_{jd}}{\|\hat{\beta}_j\|}$$

$$= \frac{e^{-6D}}{\|\hat{\beta}_j\|^D} \prod_{d=1}^{D} \hat{\beta}_{jd} \tag{55}$$

We input Eq.54 and Eq.55 into Eq.53 and we get:

$$h(\beta) = \prod_{j=1}^{K} \left(\frac{e^{-6D}}{\|\hat{\beta}_j\|^D} \prod_{d=1}^{D} \hat{\beta}_{jd}\right) \tag{56}$$

$$= e^{-6KD} \prod_{j=1}^{K} \frac{\prod_{d=1}^{D} \hat{\beta}_{jd}}{\|\hat{\beta}_j\|^D} \tag{57}$$

Take the logarithm of Eq.57

$$log\, h(\beta) = -6KD - D \sum_{j=1}^{K} log(\|\hat{\beta}\|) + \sum_{j=1}^{K} \sum_{d=1}^{D} log(\hat{\beta}_{jd}) \tag{58}$$

Take the logarithm of Eq.46

$$log\, h(\vec{P}) = \sum_{j=1}^{K-1} log(j) \tag{59}$$

We input Eqs .52, 56 and 58 into Eq.44 and take its logarithm and we obtain:

$$\begin{aligned} log\, h(\Theta) = \sum_{j=1}^{K-1} log(j) - 6KD - D \sum_{j=1}^{K} log(\|\hat{\alpha}_j\|) + \sum_{j=1}^{K} \sum_{d=1}^{D} log(\|\hat{\alpha}_{jd}\|) \\ -6KD - D \sum_{j=1}^{K} log(\|\hat{\beta}_j\|) + \sum_{j=1}^{K} \sum_{d=1}^{D} log(\|\hat{\beta}_{jd}\|) \end{aligned} \tag{60}$$

3.4.3 Complete Learning Algorithm

For each candidate value of K:

(1) Run initialization algorithm

(2) Run estimation algorithm of the scaled Dirichlet mixture model as discussed in Section 3.3.2

(3) Calculate the associated criterion of MML(K) using Eq.29

(4) Select the optimal model K^* such that $K^* = argmin_K \ MML(K)$

Chapter 4

Experimental Results

4.1 Overview

In this chapter, we simply aim to test the performance of the scaled Dirichlet finite mixture model in comparison with Dirichlet and Gaussian finite mixture models. This performance is measured in its ability to estimate model parameters and the number of clusters within datasets.

4.2 Synthetic Data

The goal of using synthetic data is to help us objectively evaluate the performance of our learning algorithm with known model parameters and mixture components. To achieve this goal, we will test our algorithm through various synthetic datasets that have different parameter vectors and number of mixture components known a priori. In addition, we will create histogram and $3D$ plots to describe the shape and surface of the datasets used.

It is also important to note that the synthetic data was generated with constant Beta parameters.

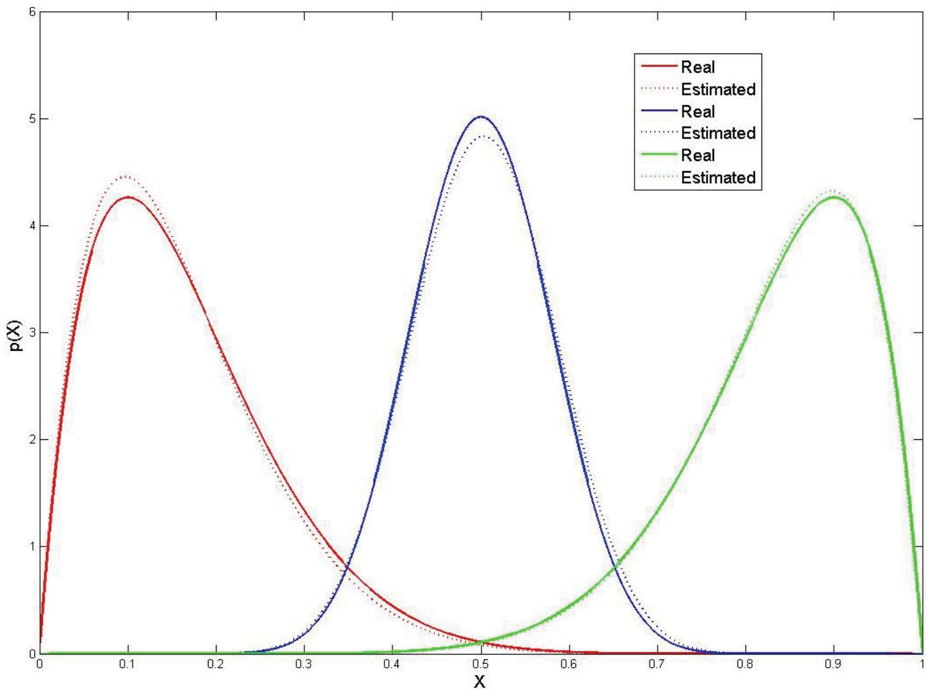
4.2.1 Results

One-Dimensional Data

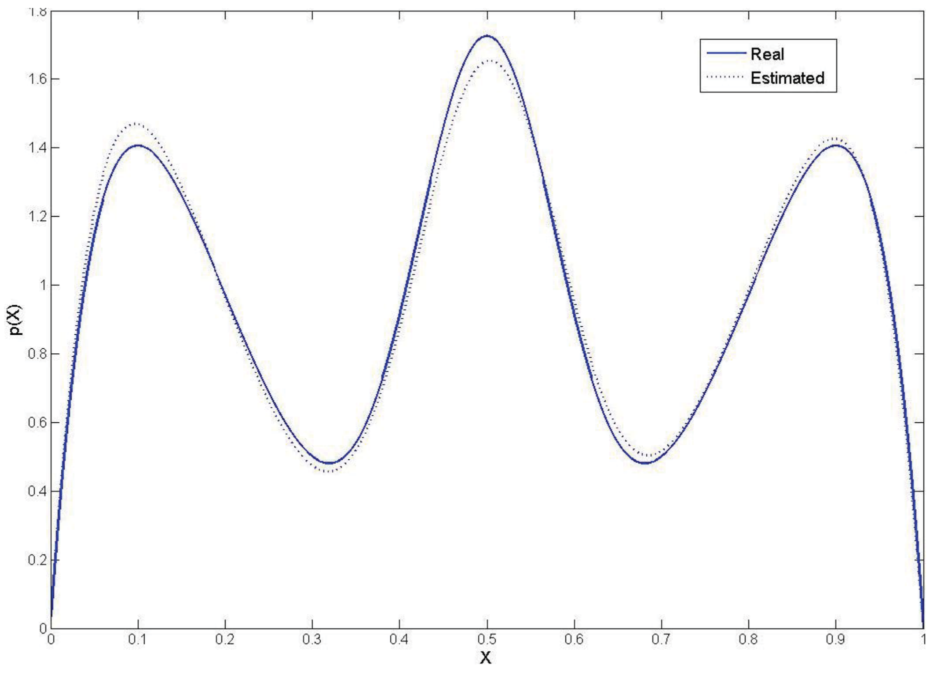
The scaled Dirichlet distribution models D-dimensional vectors (data) and these vectors are represented on a (D-1) dimensional simplex. This is why in this case, our data is called one-dimensional but originally has two dimensions. The two- dimensional equivalent of Dirichlet distribution is called the Beta distribution. And in our case the two dimensional equivalent of the scaled Dirichlet distribution is called the scaled Beta distribution with its pdf given as follows:

$$p(\vec{\mathcal{X}}|\theta) = \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1) \times \Gamma(\alpha_2)} \frac{\beta_1^{\alpha_1} x_{nd_1}^{\alpha_1 - 1} \beta_2^{\alpha_2} (1 - x_{nd_2})^{\alpha_2 - 1}}{(\beta_1 x_{nd_1} + \beta_2 (1 - x_{nd_2}))^{\alpha_1 + \alpha_2}} \tag{61}$$

Given the challenge of generating data with varying scale parameters, we made use of synthetic data generated from a Dirichlet mixture and used our algorithm to learn its shape parameters. This was done by setting the scale parameter to a constant value ($\beta = 1$). Afterwards, we implemented the model selection algorithm to predict the number of mixture components. Figure 4.1 shows the artificial histogram plots. The first histogram in figure 4.1a displays three well separated mixture components while the second plot in figure 4.1b displays the three components overlapping. From the histogram plot in figure 4.1, the dotted line represents the plot of the estimated model while the solid line represents the real model. The values of the real and estimated model parameters are given in table 4.1. According to figure 4.2, we are able to estimate the exact number of clusters. Therefore, we conclude that our algorithm works well with one dimensional synthetic data.



(a)



(b)

Figure 4.1: Artificial histogram plots for one-dimensional data.

Table 4.1: Real and estimated parameters for the generated one-dimension dataset 1 with 3 clusters.

	j	d	n_j	p_j	α_{jd}	$\hat{p_j}$	$\hat{\alpha_{jd}}$	$\hat{\beta_{jd}}$
Data set 1	1	1	1000	0.33	2	0.33	2.03	1
		2			10		10.63	1
	2	1	1000	0.33	20	0.33	18.70	1
		2			20		18.50	1
	3	1	1000	0.34	10	0.34	10.40	1
		2			2		2.07	1

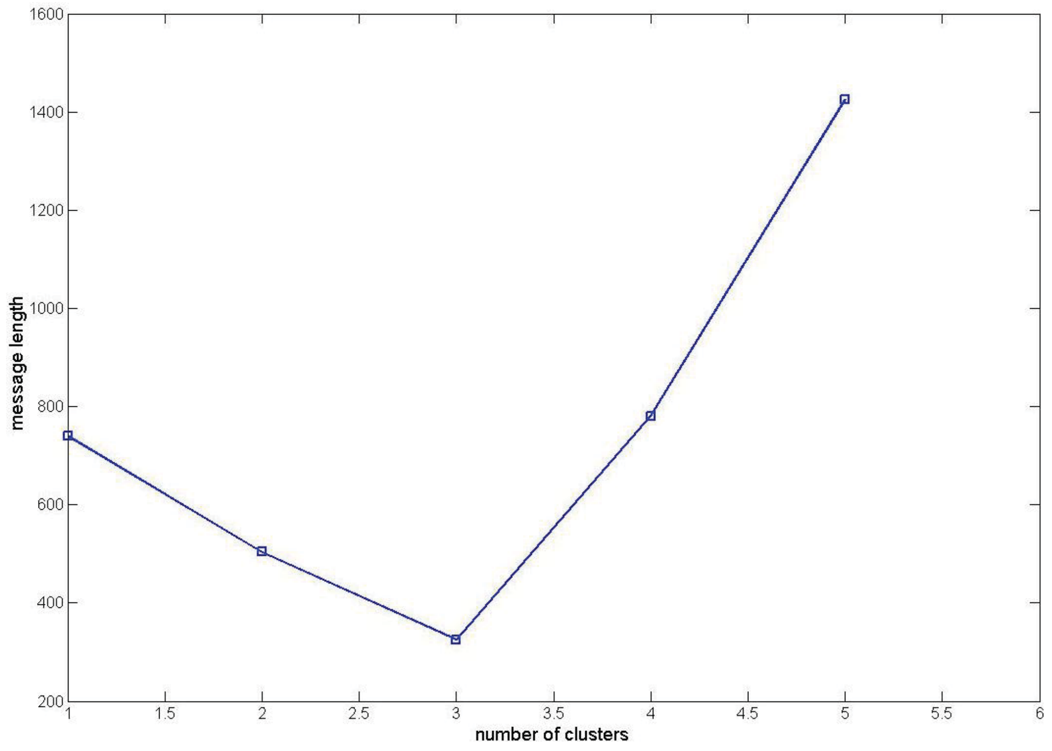


Figure 4.2: Message length plot for the 3-components generated dataset. The X axis represents the number of clusters and the Y axis represents the value of the message length.

Multi-Dimensional Data

In this case, we use $D = 3$-dimensional data. In the first four experiments, we generate synthetic data from two, three, four and five component mixtures respectively. Then, we use our algorithm to carry out parameter estimation and model selection. We display the

3-D plots of the well separated mixtures in figure 4.3. The values of the real and estimated parameters are documented in table 4.2. Results from our model selection algorithm suggest that it works well with synthetic data and predicts the number of clusters accurately.

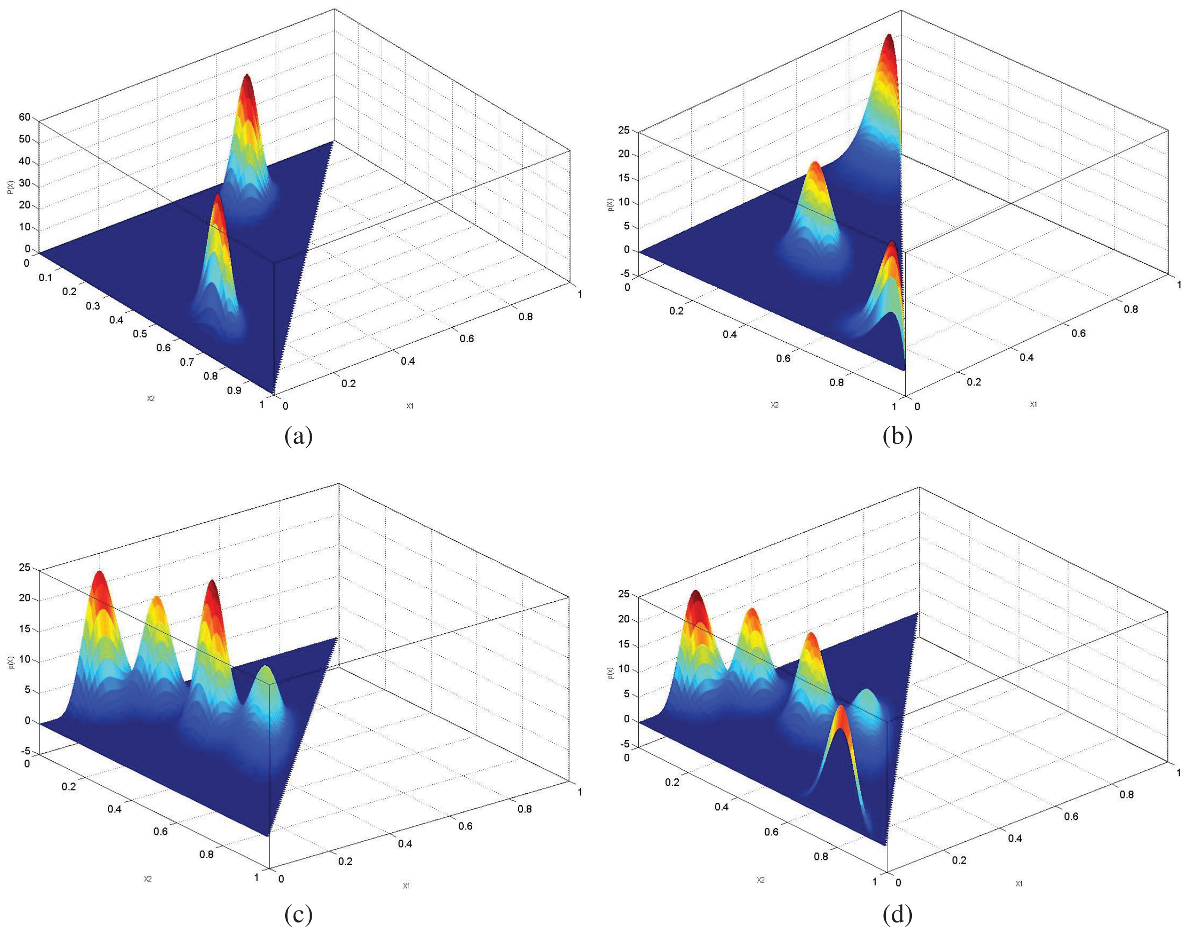
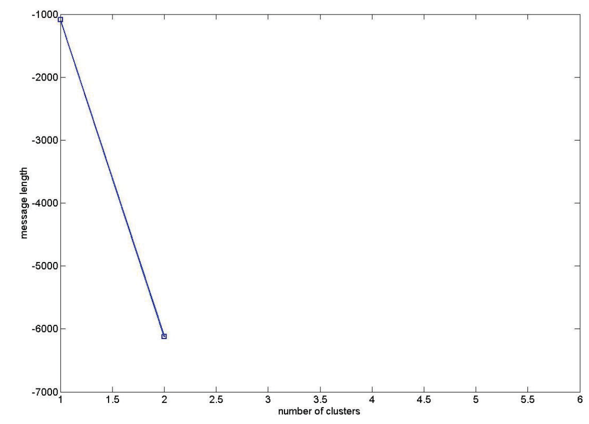


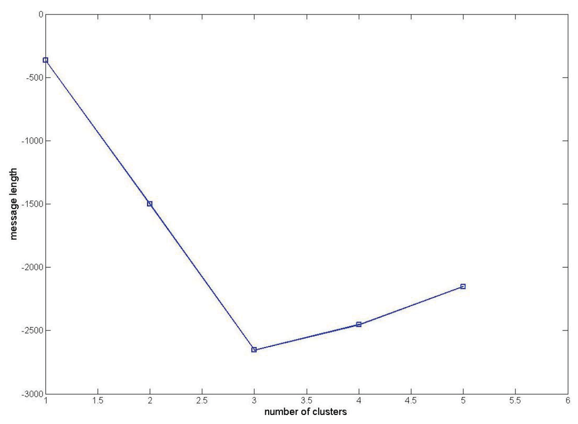
Figure 4.3: 3-D surface plot for generated dataset 2 (4.3a), dataset 3 (4.3b), dataset 4 (4.3c) and dataset 5 (4.3d) with 2, 3, 4 and 5 components, respectively.

Table 4.2: Real and estimated parameters for generated dataset 2, dataset 3, dataset 4, dataset 5 with 2, 3, 4 and 5 components, respectively.

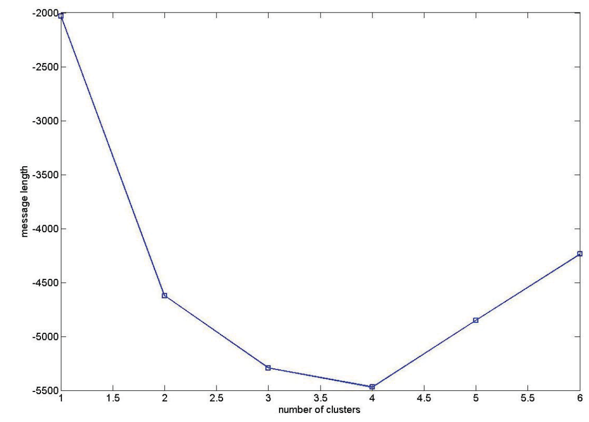
	j	d	n_j	p_j	α_{jd}	$\hat{p_j}$	$\hat{\alpha_{jd}}$	$\hat{\beta_{jd}}$
Data set 2		1			65		64	1
	1	2	1000	0.5	15	0.5	14.52	1
		3			30		29.52	1
		1			15		15.91	1
	2	2	1000	0.5	65	0.5	67.64	1
		3			30		30.96	1
Data set 3		1			2		1.87	1
	1	2	450	0.33	20	0.33	19.07	1
		3			2		1.93	1
		1			23		24.18	1
	2	2	450	0.33	25	0.33	26.20	1
		3			24		24.90	1
		1			20		18.77	1
	3	2	450	0.34	2	0.34	1.87	1
		3			2		1.92	1
Data set 4		1			10		9.55	1
	1	2	500	0.17	2	0.17	1.87	1
		3			40		37.87	1
		1			30		28.95	1
	2	2	1000	0.33	30	0.33	29.07	1
		3			32		30.76	1
		1			15		14.49	1
	3	2	1000	0.33	19	0.33	18.35	1
		3			6		5.71	1
		1			30		29.10	1
	4	2	500	0.17	10	0.17	9.46	1
		3			55		52.57	1
Data set 5		1			10		10.61	1
	1	2	500	0.167	2	0.168	2.09	1
		3			40		41.39	1
		1			30		30.94	1
	2	2	750	0.25	30	0.255	31.36	1
		3			32		33.11	1
		1			15		16.32	1
	3	2	750	0.25	19	0.245	20.75	1
		3			6		6.31	1
		1			30		31.83	1
	4	2	500	0.167	10	0.167	10.69	1
		3			55		57.89	1
		1			2		1.94	1
	5	2	500	0.166	40	0.165	39.56	1
		3		38	10		9.93	1



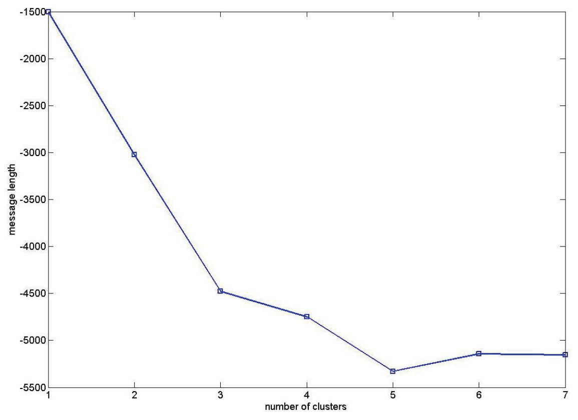
(a)



(b)



(c)



(d)

Figure 4.4: Message length plot for generated dataset 2 (4.4*a*), dataset 3 (4.4*b*), dataset 4 (4.4*c*) and dataset 5 (4.4*d*) with 2, 3, 4 and 5 components, respectively. The X axis represents the number of clusters and the Y axis represents the value of the message length.

4.3 Real Dataset

4.3.1 Iris Dataset

We consider the popular multivariate flower dataset that was first introduced by R. A. Fisher [1] called the Iris dataset. This is a simple benchmark dataset to test clustering algorithms. The 150 Iris flower samples are described with four attributes (Sepal Length, Sepal

[1]Iris flower dataset"https://en.wikipedia.org/wiki/Iris_flower_data_set"

Width, Petal Length and Petal Width). The petal is the colored leaf of the flower, while the sepal is a greenish structure that protects the petal structure. This dataset is composed of 3 different variants, classes or species of the Iris flower (Iris Setosa, Iris Versicolour, and Iris Virginica) [44].

In our experiments, we use our learning algorithm to cluster these samples. But, first we test our model selection algorithm on the dataset to confirm if it is able to determine the exact number of Iris flower species underlying the dataset. According to Figure 4.5, it is clear that our algorithm was able to find the optimal number of clusters.

Results

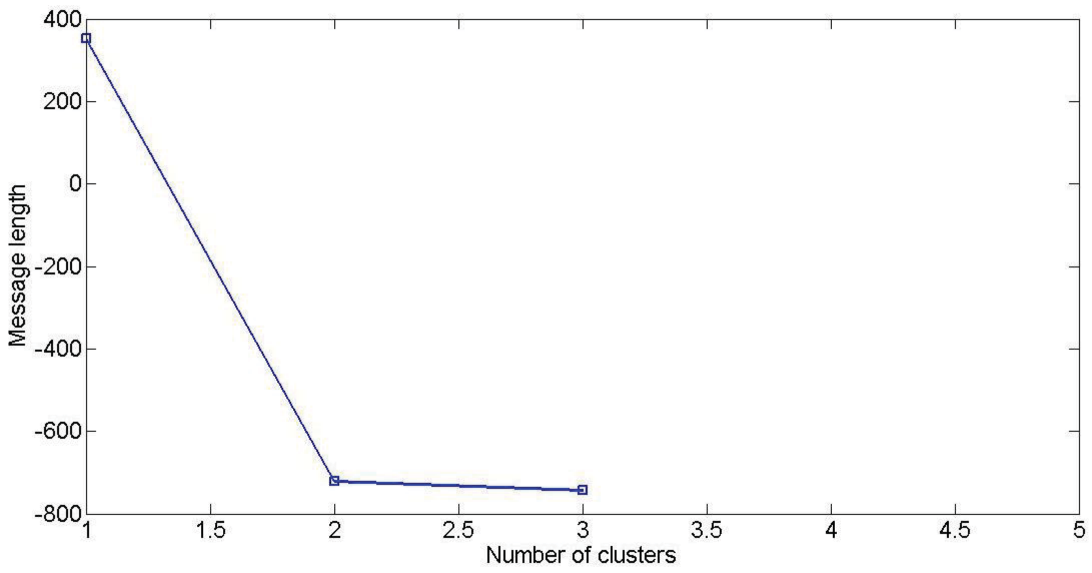


Figure 4.5: Message length plot for the Iris flower dataset. The X axis represents the number of clusters and the Y axis represents the value of the message length.

Table 4.3: Confusion matrix using SDMM, DMM and GMM on the Iris dataset.

		Setosa	Veriscolour	Virginica
SDMM	Setosa	50	0	0
	Veriscolour	0	40	10
	Virginica	0	1	49
DMM	Setosa	50	0	0
	Veriscolour	0	34	16
	Virginica	0	0	50
GMM	Setosa	50	0	0
	Veriscolour	0	35	15
	Virginica	0	12	38

From Table 4.3, we can see that the Setosa flower was accurately classified with no misclassification error in the three tested approaches. While the Versicolour had 10 instances misclassified as Virginica and finally Virginica had 1 instance misclassified as Versicolour, using the SDMM.

We assume that this misclassification between the Versicolour and Virginica is because they have overlapping attribute properties making it difficult to define the cluster parameters that would effectively separate the two clusters. In summary, the overall accuracy of the clustering using scaled Dirichlet mixture model is 93% as compared with 89% and 83% of Dirichlet and Gaussian mixture models, respectively. It is also important to note that we select the matching from our clustering algorithm that gives us the least misclassification rate and because it is rare to have exact classification accuracy at every trial, we repeat the experiment 9 times and take the average result.

4.3.2 Haberman Dataset

We consider another multivariate real dataset. This dataset contains cases from a study that was conducted within the years of 1958 and 1970 at the University of Chicagos billing hospital [45]. The study was focused on the survival of patients after they had undergone surgery for breast cancer.

The dataset contains 306 instances and 4 attributes which includes the class attributes. The 3 attributes describing the 306 instances are: age of the patient at time of operation, patient's year of operation, number of auxiliary nodes detected. Out of the 306 instance, we have 225 instances that belong to class 1 and the other 81 instances belong to class 2. Class 1 represents a situation where the patient survived 5 years or longer after the surgery and Class 2 a situation where the patient died within 5 years of the surgery. According to figure 4.6, we are able to determine the exact number of clusters using our model selection algorithm.

Results

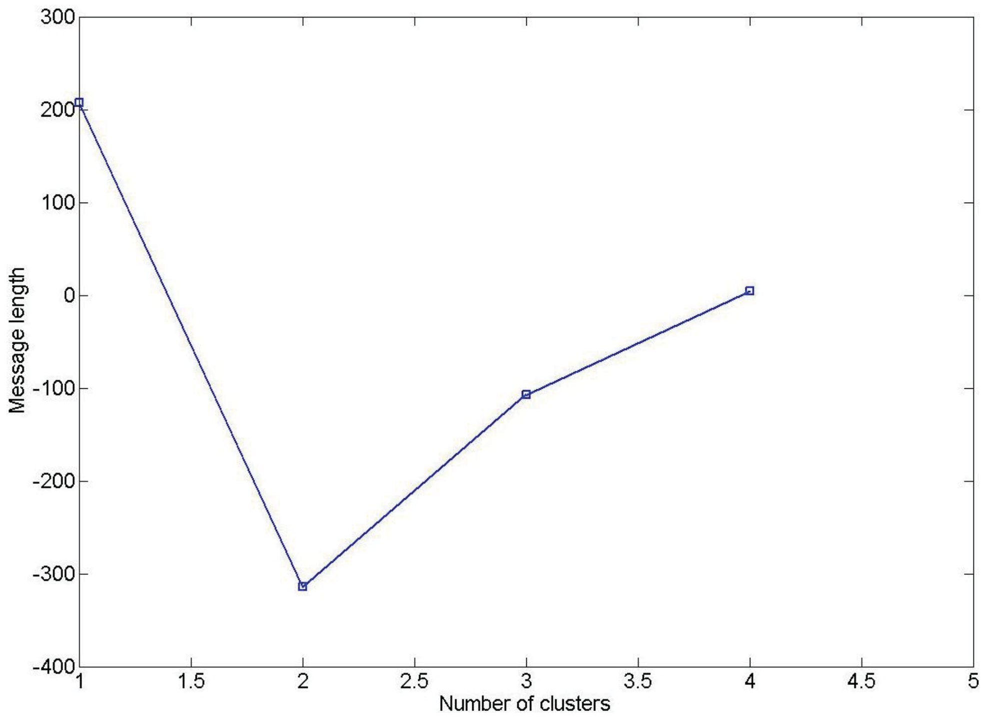


Figure 4.6: Message length plot for the Haberman dataset. The X axis represents the number of clusters and the Y axis represents the value of the message length.

Table 4.4: Confusion matrix using SDMM on Haberman dataset.

	Survived $>$ 5yrs	Died within 5yrs
Survived $>$ 5yrs	205	20
Died within 5yrs	56	25

Table 4.5: Test results for the SDMM, DMM and GMM classification of the Haberman dataset.

Datasets	Learning Algorithm	Results		
		Overall Accuracy	Average Accuracy	Precision
Haberman	SDMM	0.752	0.609	0.556
	DMM	0.415	0.357	0.140
	GMM	0.667	0.667	0.418

From the results in table 4.5, we notice that the scaled Dirichlet mixture model performs better with a higher accuracy than the Dirichlet and Gaussian mixture models. We consider the average accuracy metric because of class imbalance in the dataset. As we notice, our algorithm has a high accuracy in predicting the large class as compared to the small class labels. In other words, this metric helps to even the weight of the accuracy of both classes in situations were overall accuracy is misleading.

4.4 Software Modules Defect Prediction

The challenge of detecting a fault or a defect in a software program is one that has received a lot of research focus. It is also important to point out that it is almost impossible to create a perfect software program without errors. This is because most software development projects involve more than one software developer working on the same project and human error is unavoidable in this scenario.

When a software program is faulty, the most likely event is that the fault is located in some but not all of the software modules. For example, after the Define and Measure phase of a six sigma quality project, the next challenge is to analyze the problem and locate the root cause. This Analysis phase in some cases is synonymous to the process of our algorithm predicting fault prone modules of a software program.

An effective fault prone module prediction algorithm is of high value for businesses focused on developing complex software programs. This is because if the root cause (fault prone modules) is detected early, the software program is improved and optimized which translates to customer satisfaction and considerable higher profits for the company.

The author in, [46] explains the importance of historical datasets in the process of detecting fault prone software modules. This means that the unavailability of these datasets makes the process even more difficult. In addition, it is also very important to select the appropriate metrics that explain the attributes of these software modules. As this, in the long run, would help us classify fault prone software modules from non-fault prone software modules effectively.

In the next section, we provide a brief overview of the most popular software modules complexity metrics in the literature. And for the purpose of our thesis, we consider a statistical approach to tackle this problem. We use the datasets from the PROMISE data repository [2] and we test our algorithm and analyze its performance on some datasets.

4.4.1 Complexity Metrics of Software Modules

According to [47], metrics for measuring software complexity include the code size, McCabes cyclomatic complexity [48] and the Halsteads complexity. The Halsteads and McCabes complexity measures are based on the characteristics of the software modules as explained in [48]. These metrics are useful because they can be computed early during the design and implementation of the software program. A module can be defined as the

[2]The PROMISE Repository of Software Engineering Databases."http://promise.site.uottawa.ca/SERepository"

smallest independent unit of a software that performs a certain function [20].

The McCabes metric includes the following:

(1) Essential complexity

(2) Cyclomatic complexity

(3) Design complexity

(4) Number of lines of code

While the Halsteads complexity metric consists of three groups namely:

(1) Base measures

(2) Derived measures

(3) Line of code (LOC) measures

We perform our experiments on three datasets (JM1, PC1, Datatrieve) from the PROMISE data repository. The JM1 and PC1 contain software modules characterized by 21 attributes from the McCabe and Halstead complexity metrics. Furthermore, we also consider the Datatrieve dataset to be very interesting. This is because it helps us to understand how the characteristics of software modules and its transition across several versions affect the quality of a software code.

In addition, the authors in [49] support the notion that reliability of a software system is directly related to the complexity of a module. In the above, we discover a new dataset that was collected during the development and maintenance of a medical Imaging system (MIS). This MIS data contains 11 software complexity metrics for each module. This dataset is also interesting because it introduces a new feature that measures the number of changes to a module. Authors in [49] explain that after a period of testing, a module's level of being fault prone is based on the number of changes required to remove its fault when discovered. From the dataset we see that between $(0 - 1)$ number of changes describes a non-fault prone module. While from $(2 - 98)$ number of changes describes some level of fault prone within a software module.

4.4.2 Datasets Information

Here we create a table to highlight the basic properties of the datasets used in our experiment. It is important to note that these data were collected from NASA software projects and are currently used as benchmark datasets in this area of research.

Table 4.6: Software modules defect prediction datasets.

	JM1	PC1	Datatrieve	MIS
Language	C	C	BLISS	Pascal, PL/M, FORTRAN
LOC	315k	40k		400K
Modules	10885	1109	130	390
Defects	2106	77	11	276

4.4.3 Performance Criteria

We make use of the confusion matrix to validate the performance of our learning algorithm. This method of measuring performance is suitable because we are dealing with labeled datasets.

Table 4.7: A Typical Confusion Matrix.

		Predicted Outcome	
		No	Yes
Actual Value	No	TN	FP
	Yes	FN	TP

We define the terms as follows:

- True negative (TN): This is the case where we predict that a software module has no defect and it actually has no defect.

- True positive (TP): This is the case where we predict that a software module has a defect and it actually has a defect.
- False positive (FP): Here we predict presence of defect whereas there is no defect.
- False negative (FN): Here we predict no defect whereas there is defect.

Some metrics usually computed are as follows:

- Accuracy: This computes how often our predictive model is accurate.

$$Accuracy = \frac{TP + TN}{(TN + TP + FN + FP)}$$

- Recall /True Positive Rate: This is also called the true positive rate. So, in a situation where it is actually a yes, it computes how often it predicts yes. This is also called detection rate.

$$Recall = \frac{TP}{(FN + TP)}$$

- False Alarm Rate / False positive rate: In a situation when it is actually no, how often does it predict yes

$$False\,positive\,rate = \frac{FP}{(TN + FP)}$$

- Specificity: In a situation when it is actually no, how often does it predict no

$$Specificity = \frac{TN}{(TN + FP)}$$

- Precision: Computes how frequently it is correct when it predicts a yes

$$Precision = \frac{TP}{(FP + TP)}$$

4.4.4 Results

Table 4.8: Confusion matrix for software defect prediction using SDMM on JM1 dataset.

	No Defect	Defect
No Defect	7737	1042
Defect	1779	327

Table 4.9: Confusion matrix for software defect prediction using SDMM on PC1 dataset.

	No Defect	Defect
No Defect	921	111
Defect	68	9

Table 4.10: Confusion matrix for software defect prediction using SDMM on Datatrieve dataset.

	No Defect	Defect
No Defect	113	6
Defect	9	2

Table 4.11: Confusion matrix for software defect prediction using SDMM on MIS dataset.

	Non Fault Prone	Fault Prone
Non Fault Prone	47	67
Fault Prone	56	220

Table 4.12: Test results for the SDMM, DMM and GMM classification of the MIS, JM1, PC1 and Datatrieve Software defect prediction datasets.

Datasets	Learning Algorithm	Results					
		Defects	Overall Accuracy	Average Accuracy	Precision	Recall	False Alarm Rate
MIS	SDMM	0.707	0.685	0.605	0.767	0.797	0.587
	DMM		0.725	0.690	0.826	0.775	0.395
	GMM		0.580	0.687	0.952	0.428	0.053
JM1	SDMM	0.193	0.741	0.518	0.239	0.156	0.119
	DMM		0.707	0.500	0.194	0.164	0.163
	GMM		0.749	0.608	0.359	0.377	0.161
PC1	SDMM	0.069	0.84	0.505	0.076	0.117	0.107
	DMM		0.794	0.492	0.063	0.142	0.158
	GMM		0.752	0.632	0.138	0.493	0.229
Datatrieve	SDMM	0.084	0.885	0.566	0.25	0.182	0.050
	DMM		0.746	0.490	0.077	0.182	0.020
	GMM		0.892	0.529	0.200	0.091	0.033

4.4.5 Assessing Quality of Prediction

It is important to understand the results of this software defect prediction exercise. We see that our results are represented using a confusion matrix. Depending on the application, the results of this confusion matrix can be difficult to interpret.

From the confusion matrices of our results in tables 4.8 - 4.11, we encounter two different types of errors. They are type I and type II errors. Type I error occurs when our learning model predicts a defect in a module when there is actually no defect in that modules. While type II error occurs when our learning model predicts absence of defect in a module when there is actually a defect in that module.

With this understanding, we can say that both types of errors are costly in a software defect correction procedure. Type I error will result in a waste of developers time and effort in testing for errors when there is not. However, type II error is more critical and expensive since the defect goes undetected and if the software product is released to customers, it will result in high quality cost, downtime etc.

From the analysis of our model performance in table 4.8 - 4.11, we notice some cases of higher type II error as compared with type I error and vice versa. However, using the accuracy metric, our approach performs fairly better than the other two models.

4.4.6 Challenges Encountered

The most significant challenge experienced using our learning algorithm was to cluster datasets with class imbalance. This means that the datasets used had more non-defective software modules as compared with software modules with defect. Due to this imbalance, it is obvious that our algorithm could not effectively estimate the parameters that model the cluster of defective modules.

Another challenge is in the application of these prediction techniques in a new software development project. This is because our approach clearly depends on historical data to help developers during software testing. This simply means, that our approach is most suitable for predicting fault prone software modules in new subsequent versions of a software program.

4.5 Cluster Analysis Application in Retail

In this application, we explore the use of our clustering algorithm to find meaningful customer segments within a data population. This sort of application is widely seen in marketing where companies are faced with making decisions regarding budget, amount/ type of goods to supply, personnel, etc. needed to serve a particular customer segment.

We analyze a very popular dataset from the UCI machine learning repository known as the Wholesale Customer dataset. This dataset contains the annual spending in monetary units on diverse product categories of 440 customers. These customers are grouped into two segments based on their spending patterns. The first segment Horeca (Hotel/ Restaurant/ Cafe) Channel and second segment Retail Channel contain 298 and 142 customers, respectively [50].

Useful inference can be gotten from effectively clustering this dataset based on the shopping behavior or pattern of the customers. This inference would help companies plan and make better decisions that are tailored towards a particular customer segment. And in the long run would translate to increased market share and bottom line for such businesses. In addition, it would help improve customer service and satisfaction, improve customer retention as well as allow for effective selection of products for a particular customer segment.

However, our objective is to test and validate the modeling performance of our learning algorithm. And also to find the useful pattern underlying the dataset while maximizing accuracy. But, first it is important to discover the number of clusters using our model selection algorithm. These clusters are the two customer segments described above. Then based on this number of clusters, we will perform classification using our scaled Dirichlet mixture model learning algorithm. According to figure 4.7, we are able to determine the exact number of clusters.

4.5.1 Results

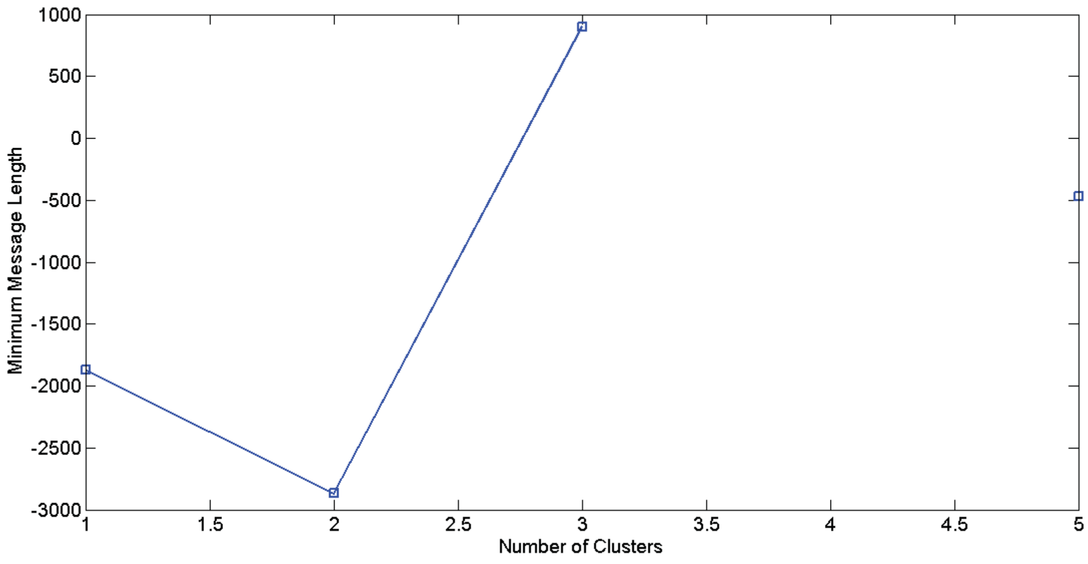


Figure 4.7: Message length plot for the Haberman dataset. The X axis represents the number of clusters and the Y axis represents the value of the message length.

Table 4.13: Confusion matrix using SDMM, DMM and GMM on the Wholesale Customer dataset.

		Horeca	Retail
	Horeca	266	32
SDMM	Retail	48	94
	Horeca	227	71
DMM	Retail	29	113
	Horeca	252	46
GMM	Retail	50	92

Table 4.14: Test results for SDMM, DMM and GMM of the Wholesale Customer dataset.

Datasets	Learning Algorithm	Results		
		Overall Accuracy	Average Accuracy	Precision
Customer	SDMM	0.818	0.777	0.746
	DMM	0.773	0.779	0.614
	GMM	0.782	0.747	0.667

4.5.2 Discussion

From the above results, we can see that our model selection algorithm, using the minimum message length, correctly inferred the K number of clusters as two. From [50], we note that the dataset contains two clusters that are based on channel of distribution as earlier mentioned. In other words, the model selection is very important in discovering similar groupings based on analyzing the multidimensional attribute information of customers. The groupings discovered explored attributes such as annual spending on fresh products, milk, grocery, detergent and paper products, etc. In addition, from the results of the confusion matrix in table 4.14, we see that our algorithm clearly performs better than Dirichlet and Gaussian mixture models. In summary, the method of validating our results is suitable for research purposes. However, in a production environment the business manager would have to drill into customers in the clusters to find meaningful insights that can help in making business decisions.

Chapter 5

Conclusion

In this section, we consider how well we achieved our objectives as outlined in chapter 1. And then give an overview of our work, challenges encountered as well as discuss future works.

To begin with, it is important to note that the task of unsupervised learning, known as clustering is well researched. However, our work was focused in the area of model based clustering. In particular, we proposed the scaled Dirichlet mixture model to further extend the work of modelling multivariate proportional data. The choice of the scaled Dirichlet distribution was motivated by its extra parameters making it more flexible and suitable for data modelling as compared with the Dirichlet distribution. Then we explored the approach of maximum likelihood estimation using the EM algorithm framework to determine the parameters of our mixture model. Given that in real world applications, the task of parameter estimation is not possible without an idea of the number of clusters inherent in our dataset, this led us to implement a model selection technique called the minimum message length to determine the number of clusters.

In addition, we tested and evaluated the modeling strengths of the scaled Dirichlet mixture model on synthetic data by comparing the estimated model parameters with that of the original mixture model parameters, and then went further to carry tests with real datasets.

However, for research purposes we made use of datasets with class information to allow us validate our model using a confusion matrix.

Furthermore, we considered a very popular application in software engineering about predicting defects in software modules. Our clustering algorithm was made to discover two groupings based on some software complexity metrics. This application is very critical in large software projects because it is very costly to carry out tests for all software modules. In addition, we considered another application in retail where we first used our proposed model selection algorithm to determine the number of distribution channels within the customer dataset. Afterwards, we used that information to find groupings of customers particular to a distribution channel based on their spending pattern attributes.

It is important to note that these kinds of applications are very popular and our algorithm works well without knowledge of class information. In other words, we can say that our algorithm produces quality clustering results largely due to its model flexibility.

We experienced a number of challenges from design to experimental stage of this work. One of which is the limitation of our algorithm to handle very high dimensional and sparse datasets. This is because of the difficultly in computing the inverse of the high-dimensional Hessian matrix when estimating model parameters. Tackling this challenge would require more work with Bayesian methods for parameter estimation.

Another challenge is encountered in the use of K-means to initialize our algorithm. It is important to note also that because the EM algorithm is sensitive to its initial values, convergence to the global maxima becomes very difficult. However, further research can be done to explore better methods of initialization.

From the aspect of applying our algorithm to real world problems, we noticed issues surrounding unavailability of datasets especially for clustering algorithms in relevant domains. When considering the scenario of defect prediction as we have explored, we experience issues with class imbalance, software metrics features, etc. The class imbalance issue makes it difficult for the algorithm to find the optimal parameters that define the defect grouping we are interested in. From our software defect prediction experiments, we can see that the small fraction of defects limits our detection ability. In the case of improving the detection performance of our algorithm, we noticed the need for feature engineering. This is because mining of irrelevant features deteriorates the performance of our model.

In a production or real environment, the metrics or attributes to be considered in a software defect prediction exercise are extremely important. This is because defects occur at different phases in a software development cycle which makes early detection a priority for most businesses. It would be necessary to develop metrics suitable for early detection of defected software modules.

Future works will explore other methods of initializing our algorithm. In addition, we will explore more efficient optimization techniques for estimating parameter vectors. Examples include complete Bayesian and variational approaches and another promising future work could be related to online learning.

Appendix A

Proof showing the invariant property of the scale parameter to any constant value

The density function of the scaled Dirichlet distribution is :

$$p(\vec{X}_d|\theta) = \frac{\Gamma(\alpha_+)}{\prod_{d=1}^{D}\Gamma(\alpha_d)} \frac{\prod_{d=1}^{D}\beta_d^{\alpha_d} x_n^{\alpha_d - 1}}{(\sum_{d=1}^{D}\beta_d x_n)^{\alpha_+}} \tag{62}$$

We remove the constant values that do not affect the β parameter

$$\frac{\prod_{d=1}^{D}\beta_d^{\alpha_d}}{(\sum_{d=1}^{D}\beta_d x_n)^{\alpha_+}} \tag{63}$$

We add η which is assumed to be any constant value

$$\frac{\prod_{d=1}^{D}\eta\beta_d^{\alpha_d}}{(\sum_{d=1}^{D}\eta\beta_d x_n)^{\alpha_+}} \tag{64}$$

We simplify out η as follows

$$\frac{\eta^{\sum_{d=1}^{D}\alpha_d}\prod_{d=1}^{D}\eta\beta_d^{\alpha_d}}{\eta^{\alpha_+}(\sum_{d=1}^{D}\eta\beta_d x_n)^{\alpha_+}} \tag{65}$$

$\eta^{\sum_{d=1}^{D}\alpha_d}$ simplifies to η^{α_+}

$$\frac{\eta^{\alpha_+}\prod_{d=1}^{D}\eta\beta_d^{\alpha_d}}{\eta^{\alpha_+}(\sum_{d=1}^{D}\eta\beta_d x_n)^{\alpha_+}} \tag{66}$$

This eventually allows us to eliminate the constant value η^{α_+} which then results to

$$\frac{\prod_{d=1}^{D} \eta\beta_d^{\alpha_d}}{(\sum_{d=1}^{D} \eta\beta_d x_n)^{\alpha_+}} \tag{67}$$

Appendix B

Simple decomposition showing that the Dirichlet density is a special case of the scaled Dirichlet density

The scaled Dirichlet distribution as follows:

$$\frac{\Gamma(\alpha_+)}{\prod_{d=1}^{D}\Gamma(\alpha_d)}\frac{\prod_{d=1}^{D}\beta_d^{\alpha_d}x_n^{\alpha_d-1}}{(\sum_{d=1}^{D}\beta_d x_n)^{\alpha_+}} \tag{68}$$

Below we show the simple decomposition of the Dirichlet from the scaled Dirichlet;

$$\frac{\Gamma(\alpha_+)\prod_{d=1}^{D}x_n^{\alpha_d-1}}{\prod_{d=1}^{D}\Gamma(\alpha_d)} \longrightarrow DirichletPortion \tag{69}$$

$$\frac{\prod_{d=1}^{D}\beta_d^{\alpha_d}}{(\sum_{d=1}^{D}\beta_d x_n)^{\alpha_+}} \longrightarrow ScaledPortion \tag{70}$$

This means that in the case of a Dirichlet density, the scaled portion is equal to 1

$$\frac{\prod_{d=1}^{D}\beta_d^{\alpha_d}}{(\sum_{d=1}^{D}\beta_d x_n)^{\alpha_+}} = 1 \tag{71}$$

Appendix C

Attribute Information for JM1, PC1 Dataset

(1) loc : McCabe's line count of code

(2) v(g) : McCabe ”cyclomatic complexity”

(3) ev(g) : McCabe ”essential complexity”

(4) iv(g) : McCabe ”design complexity”

(5) n : Halstead total operators + operands

(6) v : Halstead ”volume”

(7) l : Halstead ”program length”

(8) d : Halstead ”difficulty”

(9) i : Halstead ”intelligence”

(10) e : Halstead ”effort”

(11) b : Halstead

(12) t : Halstead's time estimator

(13) lOCode : Halstead's line count

(14) lOComment : Halstead's count of lines of comments

(15) lOBlank : Halstead's count of blank lines

(16) lOCodeAndComment

(17) $uniq\_Op$: unique operators

(18) $uniq\_Opnd$: unique operands

(19) $total\_Op$: total operators

(20) $total\_Opnd$: total operands

(21) branchCount : of the flow graph

Appendix D

Attribute Information for DATATRIEVE Dataset

(1) LOC6_0: number of lines of code of module m in version 6.0.

(2) LOC6_1: number of lines of code of module m in version 6.1.

(3) AddedLOC: number of lines of code that were added to module m in version 6.1, i.e., they were not present in module m in version 6.0.

(4) DeletedLOC: number of lines of code that were deleted from module m in version 6.0, i.e., they were no longer present in module m in version 6.1.

(5) DifferentBlocks: number of different blocks module m in between versions 6.0 and 6.1.

(6) ModificationRate: rate of modification of module m, i.e., (AddedLOC + DeletedLOC) / (LOC6.0 + AddedLOC).

(7) ModuleKnowledge: subjective variable that expresses the project team's knowledge on module m (low or high).

(8) ReusedLOC: number of lines of code of module m in version 6.0 reused in module m in version 6.1.

(9) Faulty6_1: its value is 0 for all those modules in which no faults were found; its value is 1 for all other modules.

Appendix E

Attribute Information for MIS Dataset

(1) LOC: number of lines of code of module including comments

(2) CL: number of lines of code of module, excluding comments

(3) TChar: number of characters.

(4) TComm: number of lines of comments

(5) MChar: number of comment characters

(6) DChar: number of code characters.

(7) N = N_1 + N_2 where N_1 and N_2 is the total number of operators and operands respectively

(8) $\hat{N} = \eta_1 log\eta\_1 + \eta\_2 log\eta\_2$ is an estimates program length, where is number of unique operators and operands respectively.

(9) NF = $(log_2\eta_1)! + (log_2\eta_2)!$ is Jensens estimator of program length.

(10) V(G), McCabes cyclomatic number, is one more than the number of decision nodes in the control flow graph

(11) BW is Beladys bandwidth metric

(12) Number of changes

Bibliography

[1] J. D. Wachter, “Is data really the new oil?” 2015. [Online]. Available: http://jorendewachter.com/2015/09/is-data-really-the-new-oil/

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[3] C. Fraley and A. E. Raftery, “Model-based clustering, discriminant analysis, and density estimation,” *Journal of the American statistical Association*, vol. 97, no. 458, pp. 611–631, 2002.

[4] G. McLachlan, S. Ng, and D. Peel, “On clustering by mixture models,” in *Exploratory Data Analysis in Empirical Research*. Springer, 2003, pp. 141–148.

[5] G. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.

[6] N. Bouguila, D. Ziou, and J. Vaillancourt, “Unsupervised learning of a finite mixture model based on the dirichlet distribution and its application,” *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1533–1543, 2004.

[7] M. S. Allili, N. Bouguila, and D. Ziou, “Finite generalized gaussian mixture modeling and applications to image and video foreground segmentation,” in *Computer and Robot Vision, 2007. CRV’07. Fourth Canadian Conference on*. IEEE, 2007, pp. 183–190.

[8] P. Guo and M. R. Lyu, "Software quality prediction using mixture models with em algorithm," in *Quality Software, 2000. Proceedings. First Asia-Pacific Conference on*. IEEE, 2000, pp. 69–78.

[9] K. Sjölander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler, "Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology," *Computer applications in the biosciences: CABIOS*, vol. 12, no. 4, pp. 327–345, 1996.

[10] I. Chatri, N. Bouguila, and D. Ziou, "Classification of text documents and extraction of semantically related words using hierarchical latent dirichlet allocation," Master's thesis, Concordia University, March 2015.

[11] R. D. Gupta and D. S. P. Richards, "The history of the dirichlet and liouville distributions," *International Statistical Review*, vol. 69, no. 3, pp. 433–446, 2001.

[12] J. Aitchison, "A general class of distributions on the simplex," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 136–146, 1985.

[13] S. Medasani and R. Krishnapuram, "A comparison of gaussian and pearson mixture modeling for pattern recognition and computer vision applications," *Pattern recognition letters*, vol. 20, no. 3, pp. 305–313, 1999.

[14] E. A. Cabanlit Jr, R. N. Padua, and K. Alam, "Generalization of the Dirichlet distribution," 2004.

[15] S. Migliorati, G. S. Monti, and A. Ongaro, "E–M algorithm: an application to a mixture model for compositional data," in *Proceedings of the 44th Scientific Meeting of the Italian Statistical Society*, 2008.

[16] M. A. T. Figueiredo and A. K. Jain, “Unsupervised learning of finite mixture models,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 3, pp. 381–396, 2002.

[17] A. Narayanan, “Algorithm as 266: maximum likelihood estimation of the parameters of the dirichlet distribution,” *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 40, no. 2, pp. 365–374, 1991.

[18] G. Ronning, “Maximum likelihood estimation of dirichlet distributions,” *Journal of statistical computation and simulation*, vol. 32, no. 4, pp. 215–221, 1989.

[19] T. Minka, “Estimating a dirichlet distribution,” 2000.

[20] T. Bdiri and N. Bouguila, “Positive vectors clustering using inverted dirichlet finite mixture models,” *Expert Systems with Applications*, vol. 39, no. 2, pp. 1869–1882, 2012.

[21] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[22] M. Giordan and R. Wehrens, “A comparison of computational approaches for maximum likelihood estimation of the dirichlet parameters on high-dimensional data,” *SORT-Statistics and Operations Research Transactions*, vol. 39, no. 1, pp. 109–126, 2015.

[23] D. Z. N. Bouguila, “High-dimensional unsupervised selection and estimation of a finite generalized dirichlet mixture model based on minimum message length,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, 2007.

[24] J. Huang, “Maximum likelihood estimation of dirichlet distribution parameters,” *Distribution; CMU Tech. Rep*, pp. 1–9, 2005.

[25] F. A. Graybill, “Matrices with applications in statistics,” 1983.

[26] M. Dishon and G. H. Weiss, “Small sample comparison of estimation methods for the beta distribution,” *Journal of Statistical Computation and Simulation*, vol. 11, no. 1, pp. 1–11, 1980.

[27] J. Shao, “Linear model selection by cross-validation,” *Journal of the American statistical Association*, vol. 88, no. 422, pp. 486–494, 1993.

[28] C. S. Wallace and D. L. Dowe, “MML clustering of multi-state, poisson, von mises circular and gaussian distributions,” *Statistics and Computing*, vol. 10, no. 1, pp. 73–83, 2000.

[29] N. Bouguila and D. Ziou, “On fitting finite dirichlet mixture using ecm and mml,” in *International Conference on Pattern Recognition and Image Analysis*. Springer, 2005, pp. 172–182.

[30] J. Rissanen, “Modeling by shortest data description,” *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[31] R. K. Blashfield and M. S. Aldenderfer, “The methods and problems of cluster analysis,” in *Handbook of multivariate experimental psychology*. Springer, 1988, pp. 447–473.

[32] M. Meilă, “Comparing clusteringsan information based distance,” *Journal of multivariate analysis*, vol. 98, no. 5, pp. 873–895, 2007.

[33] N. X. Vinh, J. Epps, and J. Bailey, “Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance,” *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2837–2854, 2010.

[34] N. X. Vinh, J. Epps, and Bailey, “Information theoretic measures for clusterings comparison: is a correction for chance necessary?” in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1073–1080.

[35] J. Liu and M. Qian, “Protein function prediction using kernal logistic regresssion with roc curves,” in *International Conference on Information and Management Engineering*. Springer, 2011, pp. 491–502.

[36] N. Bouguila and D. Ziou, “A hybrid sem algorithm for high-dimensional unsupervised learning using a finite generalized dirichlet mixture,” *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2657–2668, 2006.

[37] A. Ongaro, S. Migliorati, and G. S. Monti, “A new distribution on the simplex containing the dirichlet family,” 2008.

[38] G. Monti, G. Mateu-Figueras, V. Pawlowsky-Glahn, and J. Egozcue, “The shifted-scaled dirichlet distribution in the simplex,” in *Proceedings of the 4th International Workshop on Compositional Data Analysis (2011)*, 2011.

[39] V. Pawlowsky-Glahn and A. Buccianti, *Compositional data analysis: Theory and applications*. John Wiley & Sons, 2011.

[40] J. Aitchison, “The statistical analysis of compositional data,” 1986.

[41] K. W. Ng, G.-L. Tian, and M.-L. Tang, *Dirichlet and related distributions: Theory, methods and applications*. John Wiley & Sons, 2011, vol. 888.

[42] P. D. Powell, “Calculating determinants of block matrices,” *arXiv preprint arXiv:1112.4379*, 2011.

[43] A. R. Syversveen, “Noninformative bayesian priors. interpretation and problems with construction and applications,” *Preprint Statistics*, vol. 3, 1998.

[44] R. Fisher, "Uci machine learning repository, irvine, ca: University of california, school of information and computer science, 1963."

[45] H. S. J, "Haberman's survival data set, uci machine learning repository. university of california, school of information and computer science," 1976. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Haberman' s+Survival

[46] C. Ebert, "Software quality management," 2007. [Online]. Available: http://vector.com/portal/medien/vector_consulting/publications/Ebert_QualityManagement.pdf

[47] S. Aleem, L. F. Capretz, and F. Ahmed, "Benchmarking machine learning technologies for software defect detection," *arXiv preprint arXiv:1506.07563*, 2015.

[48] T. J. McCabe, "A complexity measure," *IEEE Transactions on software Engineering*, no. 4, pp. 308–320, 1976.

[49] M. R. Lyu *et al.*, *Handbook of software reliability engineering*. IEEE computer society press CA, 1996, vol. 222.

[50] Bache and Lichman, "The wholesale data set on the uci machine learning repository," 2013. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Wholesale+customers