Department of Engineering Mathematics

Dirichlet Process mixture models and their application in cancer bioinformatics

Shaun Dowling

Project Report submitted in support of the degree of Master of Engineering

Abstract

The genomic era has left biologists with more data than they know how to handle. Bioinformatics is a field dedicated to working with this data, which can come in a number of different forms. One thing that all this data makes possible is a that categorisation of subtypes of major diseases, such as cancer, from samples taken from the tumour. One approach to such classification is through the use of statistical models.

In this report a number of statistical models (more precisely Dirichlet process Gaussian mixture models) are developed to try and perform this task. This process involves outlining a general statistical model, followed by specifying the distributions that underlie the model, then developing a Markov chain Monte Carlo sampler, and finally some methodology to make sense of the results.

The success of these models is quantified by testing with rich set of randomly generated data as well as with several different gene expression datasets. The synthetic data was used to explore the impact of changing hyperparameters in the model or features in the data before being tested on the real data.

Contents

Chapter 1

Introduction

1.1 Objective

The genomic era has resulted in an explosion of biological data thanks to huge advances in the fields of molecular biology and genomics. Now, in the post-genomic era, researchers in these areas have been overwhelmed by experimental data. The field of bioinformatics refers to the creation, maintenance, and analysis of biological information such as nucleotide and amino acid sequences. Bioinformatics revolves around trying to answer important biological questions by making use of analytical machinery from statistics and computer science, but always being driven by biological understanding. These techniques are used to answer some very interesting and important questions such as

- what are the functional roles of different genes and in what cellular processes do they participate;
- how are genes regulated, how do genes and gene products interact, and what are these interaction networks;
- how does gene expression level differ in various cell types and states, how is gene expression changed by various diseases or compound treatments. [1]

As methods for attaining data develop, such as DNA sequencing and measuring gene expression data, there will be an ongoing need for ever more sophisticated methods in order to understand all the data that is produced.

The aim of this report will be related to the third question, with a particular focus on breast cancer subtypes. A key component of all cancer treatment is to correctly diagnose the patient's cancer subtype so that the optimal treatment can be provided. Since the most effective treatment can vary dramatically between cancer subtypes, giving the correct diagnosis is a very important step in the road to recovery. A major problem in classification is that it is not always possible to know what distinguishes these different subgroups on a genetic level, or sometimes even how many subgroups there are. In this report, machine learning techniques are employed to attempt to find breast cancer subtypes in a set of gene expression data.

A Dirichlet process Gaussian mixture model is a hierarchical Bayesian nonparametric statistical model that can classify data when the number of clusters is unknown. Unlike other clustering algorithms that require the number of clusters to be set before each run, by making use of the Dirichlet process, the resulting mixture model is capable of discovering the cluster structure in a number of different datasets, with no adjustment to the model.

The final goal is to investigate *data integration* (the combination data from a number of sources) by making use of a hierarchical Dirichlet process mixture model [2, 3, 4], which builds upon the Dirichlet mixture. However, during the development of this model it became apparent that the level of constraint placed on the covariance matrix of the mixture components plays a very important role in its success and , consequently models with various covariance structures will be explored in detail. Each model will then be applied to both synthetic and real data to draw comparisons, before considering the more advanced hierarchical model that builds on this foundation.

This project was written in Python as an experiment to explore the effectiveness of current open-source software solutions for scientists. In addition, a framework was developed to enable very easy and flexible use of the cloud-based *Platform as a Service* (PaaS) provider, Amazon Web Services (AWS). This service was utilised for any large simulations. This involved the dynamic initialisation of virtual servers and then distributing work on anyway up to 50 servers concurrently. Developing such a system allows for a potential unlimited decrease in runtime for parrallelisable simulations.

1.2 Background

1.2.1 Data

Here is an overview of how the microarray data is gathered and processed, and the technologies that make this possible. A more in depth description can be found in [5].

Microarrays

Hybridisation is the process of establishing a bond between two of more complementary strands of nucleic acids. The range of technologies involved in hybridising DNA or RNA samples to a large number of oligonucleotide strings arrayed on a solid surface (normally either glass or silicon) is collectively known as oligonucleotide microarray technology, or more commonly as microarrays.

Oligonucleotide strings are short single-stranded DNA or RNA that are used throughout genetic testing, research, and forensics and are usually found as small RNA molecules that function in the regulation of gene expression [6]. A key feature of oligonucleotide strings is that they readily bind to their respective complementary oligonucleotide strings. This binding property makes them effective probes for detecting DNA or RNA.

Oligonucleotide microarrays contain a large number of such oligonucleotide strings arrayed in a known order, where each oligonucleotide string is referred to as a cell of the array. The array is hybridised to a flourescently labelled DNA or RNA sample. Automated laser confocal microscopy is then used to determine if hybridisation has occurred in each cell of the array, indicating that that DNA is being expressed within the sample. More than 10^6 different oligonucleotide strings can be arrayed on a 1.28 cm $\times$ 1.28 cm glass surface in regular square cells as small as 5 µm.

Gene expression

Microarrays completely changed the way hybridisations are inspected by allowing a very large number of hybridisations to be carried out in parallel, a job that would otherwise take years of effort. This makes microarrays very effective when investigating patterns of gene expressions in different tissues. This type of profiling is the most common use of microarray technology.

Arrays in which each probe represents a gene are hybridised to fluorescently labelled DNA prepared from different tissue samples. The results from any two samples are compared; if hybridisation is stronger in one sample for a specific gene then that gene is said to be *differentially expressed* in that sample. The knowledge of which tissue a specific gene is expressed in helps to suggest the function of the gene. Likewise, by looking at a sample from a person with a disease when compared against a control healthy sample, it is possible to see what genes that disease is causing to be over or under expressed. This second technique becomes particularly useful when looking at cancer patients because different cancer types can often be clearly differentiated by their expression profiles.

Nowadays there is a vast amount of this kind of data readily available online from sources such as: Gene Expression Omnibus (GEO), the Cancer Genome Atlas, and Metabric. Due to the unnormalised nature of array data, it is not possible to use data from multiple sources in a single run of the simulation which is the reason for considering the Heirarchical Dirichlet process as a means of data integration.

There are two common ways for the gene expression matrix to be studied:

- comparing expression profiles of genes;
- comparing expression profiles across samples

The dimensionality involved in these problems varies dramatically. When comparing expression profiles of genes, the dimensionality is the number of samples taken which is normally in the hundreds. On the other hand, the dimensionality across samples is the number of genes which is of the order of 10,000. It is the later, more complex, problem that will be considered in this report, which leads to a number of extra challenges due to the size of the data vectors involved.

1.2.2 Clustering methods

A machine learning algorithm is one that learns from experience with respect to some class of tasks and performance measure [7, 1]. Machine learning techniques are useful in bioinformatics because they can infer or explain complex relationships within very high dimensional data by constructing classifiers or hypotheses. Often these hypotheses are then given to a domain expert who can suggest some experiments to validate or refute the hypothesis. This feedback loop is a very important characteristic of machine learning in bioinformatics [1].

There are two general types of machine learning: supervised learning where some data labels have been provided *a priori* (in the case of cancer data this would refer to already knowing some of the patients cancer subtypes); and unsupervised learning where no labeled data is available. In this report the task is to classify, characterise, and cluster the input data, and the approach taken is the more complex unsupervised learning. There are a number of clustering algorithms available to tackle such problems, from more heuristic methods such as hierarchical clustering to statistical models like Gaussian mixtures. However, most of these techniques require either prior knowledge of the number of clusters, or for multiple experiments to be run to enable the use of model averaging or model selection techniques. In addition, it is often not trivial to calculate the posterior over the number of clusters in the data which is of interest in this kind of problem since a level of certainty over the number of subtypes would be useful.

The goal of clustering is to group together objects (in this case patient samples) that have similar properties. This can be thought of as reducing the dimensionality of the system down to a single categorical dimension. Over time a number of clustering algorithms have been developed and several of these have been used effectively when considering gene expression data. Some of the most widely used techniques are *hierarchical* [8], *k-means* [9, 10], and *model-based* [11, 12, 13, 14, 15] clustering methods and an overview of each technique will now be provided.

k-means clustering

The k-means algorithm is simple and fast. Given a pre-specified number of clusters, k, it partitions the data set into exactly k disjoint subsets. The nature of k-means is to iteratively optimise an objective function, stoping once the output of the objective function has converged to a solution. Although fast, k-means does have some issues when classifying gene expression data [16]. Firstly, the number of clusters k, despite most likely being unknown, must be set in advance. This means, in order to find the true number of clusters the algorithm must be run a number of times varying k, then comparing the results. For larger data sets, such as gene expression data, this may not be practical. Secondly, and more significantly, the k-means algorithm forces every data point, perhaps unnaturally, to be part of a cluster. Gene expression data is typically contains a huge amount of noise which the k-means algorithm, by forcing data points in this way, is sensitive to.

Hierarchical clustering

Unlike k-means, hierarchical clustering generates a series of nested clusters which can be graphically represented by a tree, called a *dendogram.* The branches of a dendogram record both the formation of clusters and the similarity between clusters. Hierarchical clustering can be further divided into *agglomerative* (bottom-up) and *divisive* (top-down) approaches based on how the dendogram is formed. Divisive starts with one cluster containing all the data objects which is iteratively split until only clusters containing individual objects remain. In contrast, agglomerative starts with all data points in individual clusters and iteratively combines the closest points until only one cluster remains. After joining two clusters, the distances between all other clusters and the new, joined cluster are recalculated. There are different methods for deciding on which distance should considered when deciding which clusters to join next. The complete linkage, average linkage, and single linkage based on the maximum, average, and minimum distance between the members of two clusters respectively [17].

Model-based clustering

Model-based clustering provides a statistical framework to model the cluster structure of data. It is an example of *generative method* and the various models in this report are described as such. A generative method is one which attempts to model the underlying probability distribution that created the data, often with the aid of hidden variables (otherwise known as latent variables) such as class labels. These models include the Gaussian mixture model, Naive Bayes, and Hidden Markov models. Discriminative methods, in contrast to generative models, simply try to classify data without attempting to understand, or model, the underlying distribution that lead to its production. Examples of discriminative methods are things such as support vector machines (SVMs), traditional neural networks, and logistic regression.

In this report the data is assumed to be generated from a finite *mixture* of Gaussian distributions and is an example of a *mixture distribution* [18, 19]. A mixture in this sense is a linear combination of several simple distribution, enabling the modelling much more complicated probability distributions (an example of which can be seen in Figure 1.1). By using a mixture model, almost any continuous density can be approximated to arbitrary accuracy [20] by correctly adjusting the parameters of each component (in this case a number of Gaussians).

When using a Gaussian mixture in the context of data classification, the data is assumed to be distributed such that every point is drawn from one of a number of M dimensional Gaussian distributions, where M is the number of attributes in the data. When defined this way, each Gaussian represents a different underlying class. The probability density over the mixture is calculated by adding the probability from each distribution where each is weighted to ensure it is a valid probability density. That is, to ensure that the probability density integrated over the entire probability space sums to one.

In this report a Dirichlet process is used so that the number of clusters can remain unknown. This is a particularly useful trait when clustering data since clustering is often one of the first step in data mining and knowledge discovery [16] where a clustering algorithm minimal prior knowledge about the data before being applied. An additional advantage of the Dirichlet process mixture is that, unlike hierarchical and k-means clustering, it is possible to extend the model to make use of data fusion. This enables multiple data sources to be used to infer a cluster structure.

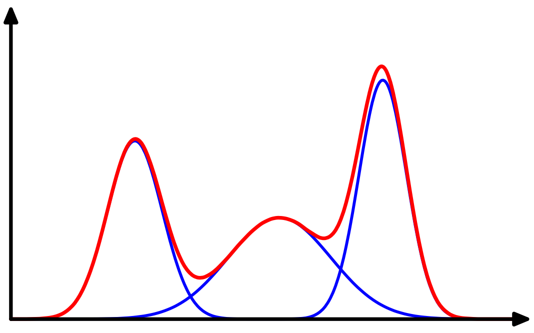


Figure 1.1: A Gaussian mixture model made of three separate Gaussian distributions [20]. This is an example of a distribution that would not be modelled effectively with a single Gaussian but can be accurately modelled by using a mixture.

1.2.3 Using Python

All programming in the report was done using freely available open source software that anyone can download and participate in the development of. Python appeared in 1991 and was created by a Dutch computer programmer Guido van Rossum. Since then, Python has become one of the most widely used languages used today. One of Python's strongest traits is its versatility spawning from its use of object-oriented, imperative, functional, procedural and reflective programming paradigms. This means that today it is possible to build desktop applications, websites, servers, and perform scientific research using a single language. One exciting feature of Python is that it is completely open source and is also General Public License (GPL) compatible enabling the developers or businesses to customise and keep the changes private, which has helped with Python adoption and maturity.

A number of scientific packages have been produced that are all similarly open source. In this report, Numpy, Scipy, MatPlotLib, Scikits-sparse, and Cython were used to replicate MATLAB functionality using nothing but freely available open-source software so as to experiment with, and demonstrate, the quality of the software currently available for free.

1.2.4 Amazon Web Services

Any computationally intensive simulations (some of which took several days to complete, that were required in the development and analysis of these models were run on Amazon Web Services (AWS). This was done to experiment with, and demonstrate the kind of computational power and flexibility that is now easily accessible by everyone.

It is becoming easier and easier to make use of very large computational resources with no upfront cost. There are a number of competing companies in this space, such as Google and Microsoft, but Amazon Web Services is one of the biggest players. With AWS it is possible to rent variable sized Elastic Cloud Compute (EC2) *instance* (or virtual server) for as little as $0.06 an hour. This means that, combined with open-source Python, it is possible for companies and researchers to cheaply and effectively make use of large-scale, flexible and reliable computational power. The source code for this project was run on AWS EC2 instances ranging in size from "Tiny", the free instance, to "Large" for when additional memory was required, as was the case when working with 6,000 dimensional matrices.

1.3 Aims

1. Develop a Python implementation of a Dirichlet process Gaussian mixture model and analyse its effectiveness in classifying different types of synthetic data.

2. Test the resulting model on real cancer gene expression data.
3. Investigate the impact of using different covariance structures on the model.
4. Develop a framework to compare the resulting clusters that are produced from the different models.
5. Make use of an effective Markov Chain Monte Carlo sampling scheme in order to approximate the posterior over the cluster assignments.
6. Develop a scheme for analysing and comparing the resulting Markov chains.
7. Explore the possibility of using Heirachical Dirichlet process for data integration.
8. Compare Variational Bayes methods to Markov Chain Monte Carlo sampling as options for inference.

Chapter 2

Dirichlet Process

In this section a quick summary of the Dirichlet distribution will be given since it forms the foundation for the definition of a Dirichlet Process. The Dirichlet process will be described and constructed in various ways to give some intuition as to what it is and how it behaves. Finally, the theory behind the Dirichlet process mixture model will be presented to highlight how it is useful for unsupervised learning.

2.1 Dirichlet distribution

The Dirichlet distribution was popularised as the conjugate prior of a multinomial distribution and is the multivariate generalisation of the Beta distribution. The Dirichlet distribution can be considered as a probability distribution over probability vectors. The Dirichlet distribution has probability density

$$p(\theta|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{w=1}^{W}\alpha_w\right)}{\prod_{w=1}^{W}\Gamma(\alpha_w)}\prod_{w=1}^{W}\theta_w^{\alpha_w-1}$$

where θ is a vector in the W-dimensional probability simplex, i.e. $\sum_w \theta_w = 1$. The entries of the α vector are the parameters of the Dirichlet distribution [21].

Figure 2.1 gives examples of the probability density of the Dirichlet distribution over the 3-dimensional probability simplex. A draw, or sample, from this distribution can be interpreted as a probability distribution over three outcomes. In the case when $\boldsymbol{\alpha} = (1, 1, 1)$, the probability of selecting any probability distribution is equally likely. If this value is increased but kept uniform then the probability mass becomes more concentrated in the centre. This means that more uniform outcomes are more likely i.e. closer to $\boldsymbol{\alpha} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. If the values for $\boldsymbol{\alpha}$ drop below 1 the probability mass is pushed further to the outside of the simplex resulting in more extreme outcome distributions i.e. more like $\boldsymbol{\alpha} = (1, 0, 0)$. If $\boldsymbol{\alpha}$ is imbalanced, the probability is more concentrated along the dimension that has the highest value.

When considered in a Dirichlet process each element of $\boldsymbol{\alpha}$ are always equal and greater that one, such that $\alpha_1 = \alpha_2 = \ldots = \alpha_w = \alpha$ where $\alpha \geq 1$. Hence, the associated Dirichlet distribution is always symmetric and either evenly distributed or concentrated in the centre of the probability simplex. The relevance of Dirichlet distributions in the Dirichlet process will be described in the next section.

2.2 Dirichlet process

The Dirichlet process (DP) can be considered as an infinite dimensional Dirichlet distribution. DPs were first considered in [22] but a more modern, and digestible, description can be found in [23] from which this overview was adapted.

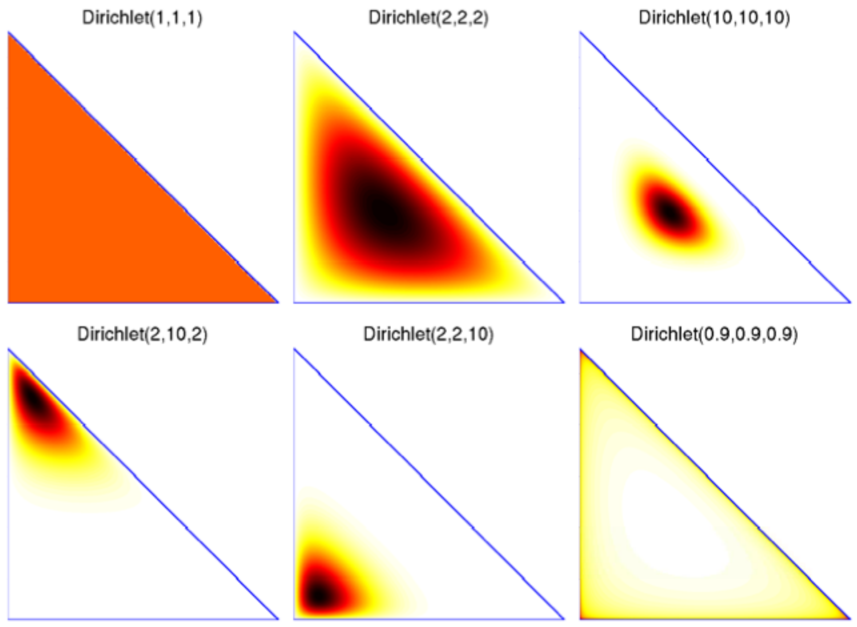


Figure 2.1: An example of a 3-D Dirichlet distribution with a variety of α parameters. (Taken from a lecture on Non-parametric Bayesian Methods by Zoubin Ghahramani)

A DP is a stochastic process over probability distributions where from which samples can be interpreted as as distributions over some parameter space Θ. In order for a random distribution G to be distributed according to a DP, its marginal distribution must be Dirichlet distributed [22]. Let G_0 be a distribution over a parameter space Θ and the Dirichlet parameter α be some positive real number. For any partition $A_1, \ldots, A_r$ of Θ, the vector $(G(A_1), \ldots, G(A_r))$ is random since the distribution G itself is random. G is Dirichlet process distributed, with some base distribution G_0 and concentration parameter α, if and only if

$$(G(A_1), \ldots, G(A_r)) \sim \mathcal{D}(\alpha G_0(A_1), \ldots, \alpha G_0(A_r)) \tag{2.1}$$

for every partition $A_1, \ldots, A_r$ of Θ.

Defined in this way it is very hard to imagine how such a process can be useful, and this definition is almost never used when constructing Dirichlet process mixtures. However, since the DP was introduced there have been a number of different constructions that allow samples to be drawn from a DP and that make nature of the process much easier to understand and apply. These constructions will be described in more detail in Section 2.3.

A draw (or sample), G, from a DP is denoted as $G \sim \mathcal{DP}(\alpha, G_0)$. The definition of the base distribution, G_0, and the concentration parameter, α, play important roles when defining a DP. The base distribution can be thought of as the mean of the DP. That is, for any set $A \subset \Theta$, $\mathbb{E}[G(A)] = G_0(A)$. In other words, the expected distribution over all parameters is the base distribution.

The concentration parameter, α, can be understood as quantifying the precision of G such that for any set $A \subset \Theta$, $\text{var}[G(A)] = G_0(A)(1 - G_0(A))/(\alpha + 1)$. This implies that as α is increased, the samples from the DP become more densely distributed around the mean distribution G_0. Hence, as $\alpha \to \infty$, $G(A) \to G_0(A)$, although interestingly this is not equivalent to saying that $G \to G_0$ [23].

2.3 Dirichlet process constructions

Several constructions have been developed since the formulation of the Dirichlet process in [22] and there has been a lot of work put into proving such a process exists in a general form [24]. In the process of proving its existence, several different schemes have been created which required powerful mathematical machinery to develop, but are actually very simple to reproduce and understand. To give a sense of what exactly a Dirichlet process is, a brief overview of the main three constructions will be given.

2.3.1 Blackwell-MacQueen Urn Scheme

In [25] the first construction for a Dirichlet process was developed and results in a predictive distribution $\theta_1, \theta_2, \ldots$ over the $\theta \in \Theta$. The metaphor used in the Blackwell-MacQueen Urn scheme is a number of coloured balls in an urn. Each value of the parameter space Θ is represented by a unique colour. Initially the urn is empty so a colour, θ_1, is chosen at random, and a ball of that colour is added to the urn. On all subsequent steps, a completely new coloured ball (sampling the colour $\theta_{n+1} \sim G_0$) is added with probability $\frac{\alpha}{\alpha+n}$. Otherwise, the colour of the new ball, θ_{n+1}, corresponds to the colour, θ_i, of a ball chosen from the urn with probability $\frac{n_i}{\alpha+n}$, where n_i is the number of balls of colour θ_i in the urn.

The Blackwell-MacQueen Urn scheme was the first proof of the existence of the DP but it also gives a very intuitive representation of the outcome from a DP, in this case an urn filled with coloured balls, as being a discrete distribution over the parameter space, Θ, where the likelihood of drawing any colour, or parameter set, is dependent on the number of balls of that colour in the urn.

2.3.2 Chinese restaurant

Perhaps one of the most popular metaphors for the Dirichlet distribution is the Chinese restaurant process (CRP) [26, 27]. Not only does this construction allow for easy explanation of the process, the CRP construction is also particularly effective and intuitive when used in the context of clustering.

Consider a Chinese restaurant with an infinite number of circular tables, each of which can seat an infinite number of customers. Each time another customer enters the restaurant they choose a table at which to sit. The first customer, c_1, simply sits at a random table θ_i. The choice of all subsequent customers, $\{c_2, c_3, \ldots, c_n\}$, when there are currently M occupied tables, is stochastic process based on the probabilities:

$$\begin{aligned} p(\theta_i | c_1, \ldots, c_{n-1}) &= \frac{|\theta_i|}{n-1+\alpha} \\ p(\theta_{M+1} | c_1, \ldots, c_{i-1}) &= \frac{\alpha}{n-1+\alpha} \end{aligned}$$

where $|\theta_i|$ is the number of customers currently sitting at table θ_i and θ_{M+1} signifies sitting at a new table.

In this construction each table represents one cell of a partition of all customers. In order for $G \sim \mathcal{DP}(G_0, \alpha)$ each partition must represent some draw from the base distribution G_0. Hence, a new set of parameters θ_i is sampled from G_0 every time a customer sits at a new table. After this process, G is the discrete distribution

$$p(\theta_i | c_1, \ldots, c_N) = \frac{|\theta_i|}{N}, \text{ for } i \in \{1, \ldots, M\}$$

2.3.3 Stick-breaking representation

Finally, the stick-breaking process, which was first derived in [24], can be written generally as

$$\beta_k \sim \text{Beta}(1, \alpha) \qquad \theta_k \sim H$$

$$\pi_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) \qquad G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}$$

where δ_{θ_k} is a unit point mass at θ_k.

This process can be thought of as taking a stick of length 1 and breaking it into smaller pieces. At the first iteration a fraction, β_1, of the stick is broken off leaving a stick of length $1 - \beta_1$ remaining. This results in a first piece of size $\pi_1 = \beta_1$. Now a fraction, β_2, of the remaining stick is broken off which results in a piece that is of length $\pi_2 = \beta_2(1 - \beta_1)$. Next a fraction, β_3, of the remaining stick is broken resulting in a piece that is of length $\pi_3 = \beta_3(1 - \beta_2)(1 - \beta_1)$. At this point the recursion apparent and it is clear that, although there is a potentially infinite number of pieces, the size of the pieces will eventually become negligibly small.

The strength of the stick-breaking construction is that it is not only the simplest, but also the most general process for developing a DP. It is also this process that will be used in the statistical models developed later in this report. However, an approximation to a true stick-breaking process will be made by truncating the weights at $i = K$ meaning $\beta_K = 1$. This means that there is no longer any stick left to break since $\pi_{K+1} = \beta_{K+1}(1 - \beta_K) \ldots (1 - \beta_1) = 0$. If K is large enough then this results in a very close approximation to an "infinite" stick breaking process and, in the case of mixture models, avoids constant addition and removal of cluster components.

2.4 Application

The Dirichlet process is popular across a range of applications of Bayesian analysis in statistics and machine learning. Some of the most popular are: Bayesian model validation, density estimation, and clustering via mixture models [23]. In this report, the flexible nature of the Dirichlet process is utilised in an unsupervised classification model. More precisely, the Dirichlet process will be used as a prior over the distribution of weights of the mixture components in a Gaussian mixture model.

The use of the Dirichlet process in a mixture model is also referred to as an *infinite mixture model* because, even if data exhibits a finite number of components, adding any additional data can exhibit previously unseen components [28]. In other words, the number of clusters increases as the size of the data set increases. This approach is also known as *non-parametric* because there is not a fixed number of parameters and that the number of parameters can grow with the data.

In the models developed in this report, the mixture components are always Gaussian. This means that the parameter space, Θ, described above is the space containing all possible combinations of means and covariances for a Gaussian distribution or, in general, a multivariate Gaussian. When related to the various constructions this means that each ball colour, table, and piece of stick represent some Gaussian distribution and the fraction of balls of that colour, fraction of people at that table, and length of stick represent the prior probability of a new data point being assigned to a cluster with those parameters. In this report a number of different constraints are added to the covariance matrices that can be drawn from the base distribution G_0. In effect this is shrinking the underlying parameter space Θ.

Chapter 3

Statistical Model

In this chapter the reason why covariance structure is important will be explained and the reasoning behind their importance argued. The accompanying equations will be derived for three different covariance structures are developed and some intuition as to the benefits and shortcomings of each will be given before an experimental analysis is performed in Chapter 6.

In this chapter and throughout the report a notational convenience is employed when discussing conditional probability distribution. Rather than writing $p(x_i|\theta_i) \sim \mathcal{N}(\theta_i)$ instead only the contents of the probability is written as $x_i|\theta_i \sim \mathcal{N}(\theta_i)$. This is a standard notation adopted across a variety of papers simply due to the number of these distributions that need to be written.

3.1 General Model

The purpose of this model is to categorise patient's gene expression data such that patients assigned to the same category (or cluster) likely share a subtype of cancer, or that a cancer subtype has similar characteristics with the others in the same cluster. In this model it is assumed that the data points are Normally distributed and a Dirichlet process prior is placed over the mixture components. Other models [29, 13, 2] were used as inspiration as to the form that this model should take and as a foundation for the creation of new models.

Let the data, denoted by $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, represent N microarray samples each of M genes, where N is the total number of patients and $\mathbf{x}_i$ represent the M-dimensional vector for one patient's gene expression levels (as described in Section 1.2.1). Let $\mathbf{z} = (z_1, \ldots, z_N)$ denote the *latent variable* for the cluster assignment of each data point such that $z_i = j$ indicates that subject i was assigned to the j-th cluster for $i = 1, \ldots, N$ and $1 \leq j \leq K$m where K denotes the total number of clusters in the mixture and is unknown.

It is assumed that the gene expression data is generated according to a mixture of several unknown Gaussian distributions with parameters θ_j, $1 \leq j \leq K$. Hence,

$$\mathbf{x}_i|\theta_1, \ldots, \theta_K = \sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

where $\theta_j = \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \pi_j\}$ is the set of parameters for component j where: $\boldsymbol{\mu}_j$ is the mean of the multivariate Gaussian; $\boldsymbol{\Sigma}_j$ is the covariance matrix; and π_j is cluster j's weight. The combined vector $\boldsymbol{\pi}$ is referred to as the *mixing proportions*. The mixing proportions are necessary for the mixture to be a true probability density and all elements must be positive and sum to one.

This is a Dirichlet process mixture model and so a Dirichlet Process prior is placed over the mixture components, meaning:

$$G \sim \mathcal{DP}(\alpha, G_0)$$
$$\theta_j \sim G$$

where G is a distribution of point masses representing parameters drawn from the base distribution G_0. Any point on the support of this distribution that can be thought of as defining a cluster component. The mixing proportions are equal to the probability associated with drawing any one of the parameter sets, θ_j, from the discrete distribution, G, over a subset of the parameter space Θ (with more detail given in Section 2.2).

This corresponds to sampling θ_j and π_j in accordance with one of the different Dirichlet process representations as described in Section 2.3, each of which involves drawing new θ_js from the base distribution G_0. In this model the *stick-breaking construction*, as described in [30], will be used due to its convenient application with Blocked Gibbs sampling. Although a true stick-breaking process does form a potentially infinite number of cluster components, the additional weights become so small that truncating at a sufficiently large index, K, results in a very close approximation to the true distribution.

As described in Section 2.3.3 this requires sampling $\beta_i \sim \text{Beta}(1, \alpha)$ for $i = \{1, 2, \ldots, K\}$ then mixture weights, π_i as:

$$\pi_i = \beta_i \prod_{l=1}^{i-1} (1 - \beta_l)$$

where $\beta_K = 1$ determines at what point the truncations occurs.

Hence, the Dirichlet process can be written as:

$$\theta_j \sim G_0$$
$$\boldsymbol{\pi}|\alpha \sim \text{Stick-Breaking}(\alpha)$$

With these parameters defined, the prior probability of a data points being assigned to a component is simply a multinomial over $\boldsymbol{\pi}$.

$$z_i|\boldsymbol{\pi} \sim \text{Mult}(\pi_1, \ldots, \pi_K)$$

In summary the generative model can be defined as:

$$\begin{array}{ll} \mathbf{x}_i|z_i = j, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) & z_i|\boldsymbol{\pi} \sim \text{Mult}(\pi_1, \ldots, \pi_K) \\ \boldsymbol{\theta}_j = (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \sim G_0 & \boldsymbol{\pi}|\alpha \sim \text{Stick-Breaking}(\alpha) \end{array}$$

The model can be tuned through the choice of the joint base distribution, G_0, which includes setting hyperparameters and/or hyperpriors, and the Dirichlet, or concentration, parameter. The concentration parameter controls how often new clusters are created which impacts the expected number of clusters without placing any strict constraints on it. More precisely, the probability that θ_j is a new, unseen, draw from G_0 is proportional to α.

3.2 Covariance structure

The model component with the biggest impact on the outcome is the definition of the base distribution G_0. This can be thought of as the prior over the parameter values and determines the types of mixtures possible. In effect this determines the size of the parameter space that can be explored and the probabilities associated with that exploration. In this section, an overview of the process of choosing potential base distributions will be presented.

3.2.1 Overview

In the original attempt to define the model, a Gaussian-Wishart prior was used as it was in [31, 32, 29]. This model was implemented in Python, making use of the derivations and MATLAB source code from [33].

The Gaussian-Wishart prior is the conjugate pair of a multivariate Gaussian distribution when both the mean and variance are unknown. After failed experimentation when using this model at the kind of high dimensions required, and conversation with Yeh Whye Teh and Mark Girrolami, it appeared that the issue was with the number of parameters being approximated. This number arose from the full covariance matrix that was not suitable at the dimensions being considered. Using this base distribution, the number of free parameters in the model is KM from the mean vectors and KM^2 from the covariance matrices, that is $\mathcal{O}(KM^2)$ in total. This becomes unfeasible when $M \approx 10,000$ as in gene expression data. As a result a different joint prior was necessary and instead a Gaussian-Gamma conjugate prior was used. This approach is also taken by [2, 34, 13] which all focus on clustering high dimensional gene expression data. However, in this report the methods were derived from scratch.

3.2.2 Covariance structures

Since it became apparent that covariance structure played such a significant role in the success of the model at high dimensions, a number of different options were explored. A good background to various covariance structures is provided in [15], which describes (in ascending order of parameter complexity) *equal volume spherical*, *unequal volume spherical*, *unequal volume ellipsoid*, and *unconstrained* models, all of which were implemented for finite mixture models in MCLUST [35]. The Gaussian-Wishart model is an example of an *unconstrained* covariance structure, which includes the largest number of parameters possible and, as indicated by the name, puts no constraint on the covariances that can be drawn from $G \sim \mathcal{DP}$.

The approach taken in this report was to first develop the unequal volume spherical model before developing the simpler equal volume spherical and more complex unequal volume ellipsoid models to allow for comparisons (where complexity refers to parameter complexity i.e. the number of parameters). The covariances will be explained in ascending order of this complexity.

Equal volume spherical covariance

The simplest covariance structure possible is *equal volume spherical covariance*. In this case, $\mathbf{\Sigma} = \sigma \mathbf{I}$ for some $\sigma > 0$ shared across all components. In essence, this means that all cluster components are assumed to be identically variant in every dimension, and to each other. This results in $\mathcal{O}(KM)$ parameters in the model or, to be exact, $KM + 1$ with K lots of M for the mean vectors of the cluster components and an additional σ value shared across the cluster components.

The issue with such a simplified model is that it severely reduces the forms of data that truly match the model. If the data itself is identically variant in all dimensions and across all clusters then these constraints are not an issue as the model can accurately represent the data. In fact, this model should be more effective with a smaller number of samples due to the smaller number of parameters that need approximating. However, if the data is dissimilarly variant in even one direction for a single cluster then the model may have issues classifying the data well.

Unequal volume spherical covariance

The slightly more complex option is to allow each cluster to have a unique covariance matrix but maintain that it must be equal in all dimensions. This is called *unequal volume spherical covariance*, and the covariance matrix for a cluster component j can be written as $\mathbf{\Sigma}_j = \sigma_j \mathbf{I}$. In this case the number of parameters is $KM + K$ with K lots of M for the mean vectors of the cluster components and K σs describing the potentially unique spherical covariance matrices for each cluster. Despite this added complexity, however, this is still asymptotically $\mathcal{O}(KM)$ parameters.

This is capable of accurately capturing both tight and loose cluster structures within the same dataset. This is important when considering cancer subtypes because it may be that one subtype is naturally a lot more variant than another.

This covariance structure may still run into difficulty if the variance in the expression levels of some genes are significantly more that the rest. When there are $M \approx 10,000$ genes being considered at least some of them must vary differently across patients. For example, there undoubtedly be some gene expression levels that are completely unaffected by the cancer and as such will results have most invariant expression readings. In contrast, a cancer subtype may cause a spike in the expression levels of a gene that effects certain patients more than others. This would result in a highly variant expression readings within a single cluster. Both these cases would result in potential misclassification of the data and, most likely, an over approximation to the number of clusters would be made by breaking one cluster into several along the highly variant dimension when trying to fit a spherical model. If every cluster in the data is equally variant in all dimensions but potentially different across clusters, it would be better to use this model than a more complex one because it would be capable of making a better estimate of the true distribution with a smaller number of samples.

Unequal volume ellipsoid

The last covariance structure to be considered is when the variance in every dimension is independent of all other dimensions, but can be unique along the diagonal of the covariance matrix. This is known as *unequal volume ellipsoid* covariance since it allows for ellipsoid shaped distributions. The covariance for a cluster, j, would be written

$$\mathbf{\Sigma}_{\boldsymbol{j}} = \begin{pmatrix} \sigma_{j1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{j1}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_{jm}^2 \end{pmatrix}$$

where $\{\sigma_{j1}^2, \dots, \sigma_{jm}^2\}$ are free to be unique.

In this case each of the K cluster components introduces M parameters for the mean vectors and M parameters for the covariance matrices. Hence the total number of parameters is $2KM$ which, although being the biggest in terms of constant factors, is still $\mathcal{O}(KM)$ asymptotically. The case described previously that would intuitively cause problems for the previous two covariance structures would be able to be modelled accurately by the ellipsoid structure.

Unconstrained

Using a Gaussian-Wishart prior results in *unconstrained* covariance matrices meaning that any covariance matrix is possible. In this case, the model considers the potential covariance between every pair of dimensions. By removing all constraints, the model increases from having $\mathcal{O}(KM)$ (in the case of equal volume spherical, unequal volume spherical, and unequal volume ellipsoid) to having $\mathcal{O}(KM^2)$, This asymptotic difference results in a huge increase in free parameters when $M \approx 10,000$ and all constrained models are much more feasible when looking at data sets where the ratio of sample size to dimensionality is small (as is almost always the case with gene expression data). In general, trying to accurately approximate $\mathcal{O}(1,000,000)$ parameters with only $\mathcal{O}(100)$ data points is going to be very difficult!

3.2.3 Resulting base distributions

In all models a conjugate prior is used so as to make the Gibbs sampling stage possible analytically. In the case of a Gaussian mixture, imposing this constraint on the choice of prior causes the prior distribution of the mean of a component to be dependant on the covariance. This has been shown to have a potentially negative effect effect on density estimation [29]. The upside, however, is that the posterior is now of the same form as the prior which makes sampling from the posterior much simpler (as long as a simple prior is used, as is the case here).

Since conjugacy is required, the most obvious choice for a prior is the Gaussian-Wishart since it is the conjugate prior of a general multivariate Gaussian distribution. However, for the reasons given above, this prior results in an unsuitable covariance structure and results in an unfeasible number of parameters. Hence, all models will have be some form of Gaussian-Gamma distribution, which is the conjugate prior of a univariate Gaussian distribution with unknown mean and variance. With each different covariance structure it is possible to make use of the applied constraints to simplify the posterior so that it is in a form that allows the conjugacy of the Gaussian-Gamma and univariate Gaussian to be leveraged.

3.3 Model Derivations

3.3.1 Gaussian-Gamma: Unequal Volume Spherical Covariance

In this case, each multivariate Gaussian distribution can be considered, probabilistically, as a combination of univariate Gaussians with different means thanks to the independence between dimensions.

The generative model would be as follows:

$$\begin{aligned}\mathbf{x}_i|z_i = j, \boldsymbol{\theta} &\sim \mathcal{N}(\boldsymbol{\mu}_j, \sigma_j^2\mathbf{I})\\ \boldsymbol{\mu}_j|\boldsymbol{\mu}_0, \lambda, \sigma_j^2 &\sim \mathcal{N}(\boldsymbol{\mu}_0, \frac{\sigma_j^2}{\lambda}\mathbf{I})\\ \sigma_j^2|a, b &\sim \text{Inv-Gamma}(a, b)\end{aligned}$$

where $\theta_j = \{\boldsymbol{\mu}_j, \sigma_j\}$.

$\boldsymbol{\mu}_0, \lambda, \alpha$ and β are hyperparameters of the model that can be adjusted based on prior knowledge of the system or the data. $\boldsymbol{\mu}_0$ is the mean of the cluster centres. λ is how precise (inverse covariate) the clusters are compared to the cluster centres. α and β are the shape and scale of the Beta distribution and together effect where and how densely the probability mass of σ_j is distributed.

There are several probabilities that are necessary in order to perform Gibbs sampling considered with this model some other probabilities that are useful for analysis. Details of the sampling method used will be outlined in more detail in Section 4.3. The full list of probabilities can be found in Appendix A.1.1.

Joint posterior distribution

In its full form, the joint posterior over the parameters for a cluster j is

$$\begin{aligned}p(\theta_j|\mathbf{X}, \mathbf{z}) &\propto p(\mathbf{X}_j|\theta_j, \mathbf{z})p(\theta_j)\\ &= \left(2\pi\sigma_j^2\right)^{-\frac{N_jM}{2}} \left(\frac{\lambda}{2\pi}\right)^{\frac{M}{2}} \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma_j^2)^{-\frac{M}{2}-\alpha-1} \exp\left\{ -\frac{\sum_{x_i\in j} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2 + \lambda||\boldsymbol{\mu}_j - \boldsymbol{\mu}_0||^2 + 2\beta}{2\sigma_j^2}\right\}\end{aligned} \tag{3.1}$$

where the sum over $x_i \in j$ means summing over every x_i for which $z_i = j$.

In order to make sampling from (3.1) simpler, each parameter will be considered in turn. The total set of all parameters in this model is:

$$\boldsymbol{\theta} = \{\mu_{11}, \mu_{12}, \dots, \mu_{1M}, \mu_{21}, \dots, \mu_{KM}, \sigma_1, \sigma_2, \dots, \sigma_j\} \tag{3.2}$$

It is possible to consider each of the parameters in (3.2) in turn and wait for the sampling to converge to the true posterior. Sampling from a Normal-Gamma distribution is analogous to first sampling from a Gamma distribution before sampling from a Normal distribution conditioned on the result so sampling the variance will be considered first.

Posterior over the variance

The variance is assumed equal in every dimension so given a mean, the posterior over the variance is only dependent on the sum of all data point's divergence from the mean, summed along each dimension. Hence, all the data points assigned to a cluster can be combined into one univariate Gaussian resulting in a posterior in the form of a Gamma distribution. By making this simplification, N_j elements in the multivariate are projected into N_jM data points in a univariate.

$$\begin{aligned} p(\sigma_j^2 | \mathbf{X}, \boldsymbol{\mu}_j, \mathbf{z}) &\propto p(\mathbf{X}_j | \boldsymbol{\mu}_j, \sigma_j^2, \mathbf{z}) p(\sigma_j^2 | \alpha, \beta) \\ &= \frac{\beta^\alpha}{\Gamma(\alpha + (2\pi)^{\frac{N_jM}{2}})} (\sigma_j^2)^{-(\alpha + \frac{N_jM}{2}) - 1} \exp\left\{ - \frac{\sum_{x_i \in j} \sum_{m=1}^M ||x_{im} - \mu_{jm}||^2 + \beta}{\sigma_j^2} \right\} \end{aligned}$$

which is a Gamma distribution with effectively N_jM data points assigned to it (one for each dimension of each data point). From [36, 20], the posterior over the variance when holding the mean fixed is

$$\sigma | \alpha, \beta, x_1, \dots, x_n \sim \text{Inverse-Gamma}\Big(\alpha + \frac{n}{2}, \beta + \frac{1}{2} \sum (x_i - \mu)\Big)$$

With the flattened distribution taken into account, the posterior of the variance for a cluster is

$$\sigma_j | \alpha, \beta, \mathbf{x}, \mathbf{z} \sim \text{Inverse-Gamma}\Big(\alpha + \frac{N_jM}{2}, \beta + \frac{1}{2} \sum_{x_i \in j} \sum_{m=1}^{M} x_{im} - \mu_{jm}\Big) \tag{3.3}$$

where the N_jM represents the effective number of data points in the projected univariate.

The distribution 3.3 is slightly awkward to sample from since it requires the storage of every data point in the component so that it can be calculated because μ_j, and hence the deviation from μ_j, will change every time a new set of parameters is drawn. However, in practise this shortcoming didn't cause too much of an increase in runtime. This cost in runtime seems especially small when compared to the cost that results from working with a full 6,000 dimensional covariance matrix as is be the case with an unconstrained model.

Posterior over the mean

Considering the posterior over the mean for cluster component k gives:

$$\begin{aligned} p(\boldsymbol{\mu}_j | \mathbf{X}, \sigma_j^2, \mathbf{z}) &\propto p(\mathbf{X}_j | \boldsymbol{\mu}_j, \sigma_j^2, \mathbf{z}) p(\boldsymbol{\mu}_j | \boldsymbol{\mu}_0, \sigma_j^2, \lambda) \\ &= \left(\frac{1}{2\pi(\lambda^{N_j+1}\sigma_j^2)} \right)^{\frac{M}{2}(N_j+1)} \exp\left\{ - \frac{\sum_{x_i \in j} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2 + \lambda ||\boldsymbol{\mu}_j - \boldsymbol{\mu}_0||^2}{2\sigma_j^2} \right\} \end{aligned} \tag{3.4}$$

However, by making use of the spherical covariance constraint, and implied independence between dimensions, (3.3.1) can be broken up into univariate Gaussian distributions along each dimension conditioned on the value in the m-th dimension of all the data points belong to that cluster . Hence,

$$p(\boldsymbol{\mu}_j|\mathbf{X}, \sigma_j^2, \mathbf{z}) = \prod_m p(\mu_{jm}|\mathbf{X}, \sigma_j^2, \mathbf{z})$$
$$p(\mu_{jm}|\mathbf{X}_m, \sigma_j^2, \mathbf{z}) \propto p(\mathbf{X}_{jm}|\mu_{jm}, \sigma_j^2, \mathbf{z}) p(\mu_{jm}|\mu_{0_m}, \sigma_j^2, \lambda)$$

which is a product of two normal distributions meaning that the resulting posterior is also a Normal distribution due to conjugacy. Sampling (3.3.1) after sampling the variance from the Gamma distribution in (3.3) and holding the new σ fixed is equivalent to sampling from a univariate Normal distribution with known variance and unknown mean.

From [36], in the univariate case where $x_i|\mu \sim \mathcal{N}(\mu_j, \sigma_j^2)$ i.i.d and $\bar{\mu} \sim \mathcal{N}(\mu_0, \sigma_0^2)$,

$$\mu|x_1, x_2, \dots, x_n \sim \mathcal{N}\left(\frac{\sigma_0^2}{\frac{\sigma^2}{n} + \sigma_0^2}\bar{x} + \frac{\frac{\sigma^2}{n}}{\frac{\sigma^2}{n} + \sigma_0^2}\mu_0, \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)^{-1}\right)$$

where $\sigma_0^2 = \frac{\sigma_j^2}{\lambda}$ is the variance of the cluster centres.

Therefore, a new sample for the mean for each cluster component, j, is made from the distribution

$$\mu_m|x_{1m}, x_{2m}, \dots, x_{nm} \sim \mathcal{N}(\hat{\mu}_m, \hat{\sigma}^2)$$

with

$$\hat{\mu}_m = \frac{n}{\lambda + n}\bar{x}_m + \frac{\lambda}{\lambda + n}\mu_{0_m}$$

where $\bar{x}_m = \frac{\sum_{i=1}^N x_{im}}{n}$ is the sample mean of the m-th element of the data points assigned to cluster j and

$$\hat{\sigma}^2 = \left(\frac{\lambda}{\sigma_j^2} + \frac{n}{\sigma_j^2}\right)^{-1}$$

3.3.2 Gaussian-Gamma: Equal Volume Spherical Covariance

The generative model can be written as:

$$\mathbf{x}_i|z_i = j, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_j, \sigma^2\mathbf{I})$$
$$\boldsymbol{\mu}_j|\boldsymbol{\mu}_0, \lambda, \sigma^2 \sim \mathcal{N}(\boldsymbol{\mu}_0, \frac{\sigma^2}{\lambda}\mathbf{I})$$
$$\sigma^2|a, b \sim \text{Inv-Gamma}(a, b)$$

which is almost identical to the unequal case but with only one variance parameter, σ, which leads to very similar likelihood and prior distributions that can be found in Appendix A.1.2.

Now the full set of parameters is

$$\boldsymbol{\theta} = \{\mu_{11}, \mu_{12}, \dots, \mu_{1M}, \mu_{21}, \dots, \mu_{KM}, \sigma\}$$

Posterior over the variance

Equal volume spherical covariance implies the variance is assumed equal for every cluster along every dimension. This means that each dimension of every data point can be combined into a single univariate Gaussian. The resulting posterior is in the form of a Gamma distribution matching the prior as required, such that

$$p(\sigma^2|\mathbf{X}, \boldsymbol{\mu}_j, \mathbf{z}) \propto p(\mathbf{X}_j|\boldsymbol{\mu}_j, \sigma^2, \mathbf{z}) p(\sigma^2|\alpha, \beta)$$

When considered with the addition of this flattened distribution, the posterior of the variance for all clusters is

$$\sigma^2|\alpha, \beta, \mathbf{x}, \mathbf{z} \sim \text{Inverse-Gamma}\Big(\alpha + \frac{NM}{2}, \beta + \frac{1}{2}\sum_{j=1}^{K}\sum_{x_i \in j}\sum_{m=1}^{M} x_{im} - \mu_{jm}\Big)$$

which as before requires the storage and iteration over every data point so that it can be calculated. The difference with this model compared to the unequal volume spherical covariance is that the variance must be sampled just once for all cluster components.

Posterior over the mean

Making use of the spherical covariance, and hence independence between dimensions, the posterior over the mean can be broken up into univariate distributions along each dimension conditioned upon the value along the m-th dimension of all the data points in that cluster.

$$p(\mu_{jm}|\mathbf{X}_m, \sigma^2, \mathbf{z}) \propto p(\mathbf{X}_{jm}|\mu_{jm}, \sigma^2, \mathbf{z})p(\mu_{jm}|\mu_{0_m}, \sigma^2, \lambda) \tag{3.5}$$

Sampling from 3.5 is almost identical to the previous case. μ_j is sampled such that

$$\mu_m|x_{1m}, x_{2m}, \dots, x_{nm} \sim \mathcal{N}(\hat{\mu}_m, \hat{\sigma}^2_m)$$

where,

$$\hat{\mu}_m = \frac{n}{\lambda + n}\bar{x}_m + \frac{\lambda}{\lambda + n}\mu_{0_m} \tag{3.6}$$

$$\hat{\sigma}^2 = \Big(\frac{\lambda}{\sigma^2} + \frac{n}{\sigma^2}\Big)^{-1} \tag{3.7}$$

3.3.3 Gaussian-Gamma: Unequal Volume Elliptical Covariance

The equations related to the likelihoods are no longer so easy to write analytically. In all equations, D_j, will represent the diagonal matrix,

$$\mathbf{D}_j = \begin{pmatrix} \sigma^2_{j1} & 0 & \cdots & 0 \\ 0 & \sigma^2_{j1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma^2_{jm} \end{pmatrix}$$

leading to the generative model,

$$\mathbf{x}_i|z_i, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_j, \mathbf{D}_j)$$
$$\boldsymbol{\mu}_j|\boldsymbol{\mu}_0, \lambda, \sigma^2_j \sim \mathcal{N}(\boldsymbol{\mu}_0, \frac{1}{\lambda}\mathbf{D}_j)$$
$$\sigma^2_{jm}|a, b \sim \text{Inv-Gamma}(a, b)$$

where $\boldsymbol{\mu}_0, \lambda, \alpha$ and β have the same meaning as before. Again, full equations for this covariance structure can be found in Appendix A.1.3. and only a summary will be given here.

Joint posterior distribution

Calculating the posterior over the parameters becomes more complex in this case but only really involves additional calculations since independence along each dimension is maintained.

The total set of all parameters to be sampled in this model is:

$$\boldsymbol{\theta} = \{\mu_{11}, \mu_{12}, \ldots, \mu_{1M}, \mu_{21}, \ldots, \mu_{KM}, \sigma_{11}, \sigma_{12}, \ldots, \sigma_{1M}, \sigma_{21}, \ldots, \sigma_{KM}\} \tag{3.8}$$

Again it is possible to consider sampling the parameters in (3.8) as sampling from a set of univariate Normal-Gamma distributions but this time a larger number of samples is required per iteration.

Posterior over the variance

The variance of each cluster is not assumed equal in every dimension, but it is assumed to be independent of all other dimensions. Hence, the sampling step can be broken up into $m = \{1, \ldots, M\}$ draws from a Gamma distribution per cluster component, conditioned on the value along the m-th dimension of each data points in that cluster.

Using the same theory as before, this boils down to sampling each σ_{jm} in D_j such that

$$\sigma_{jm}|\alpha, \beta, \mathbf{x}, \mathbf{z} \sim \text{Inverse-Gamma}\Big(\alpha + \frac{N_j}{2}, \beta + \frac{1}{2}\sum_{x_i \in j} x_{im} - \mu_{jm}\Big)$$

Posterior over the mean

The posterior over the mean remains unchanged except that a different variance must be considered along each dimension. That is,

$$\mu_m|x_{1m}, x_{2m}, \ldots, x_{nm} \sim \mathcal{N}(\hat{\mu}_m, \hat{\sigma}_m^2)$$

with

$$\hat{\mu}_m = \frac{n}{\lambda + n}\bar{x}_m + \frac{\lambda}{\lambda + n}\mu_{0_m}$$
$$\hat{\sigma}_m^2 = \Big(\frac{\lambda}{\sigma_{jm}^2} + \frac{n}{\sigma_{jm}^2}\Big)^{-1}$$

Chapter 4

Markov Chain Monte Carlo

For most probabilistic models of practical interest, exact posterior inference is intractable and therefore some sort of approximation is required. The two most popular methods are *Markov Chain Monte Carlo* (MCMC) sampling and *Variational Bayes* methods. In this report MCMC was used and this chapter will provide a short overview of MCMC explaining why it is applicable in this context. Further detail will then be provided with regards to how it is utilised in this model. Finally, an outline of how the outcome of the Markov chain is analysed will be given before reviewing results in Chapter 6.

4.1 Overview

With the simplest and most common distributions it is possible to analytically transform the distribution in terms of an inverse cumulant function. This allows a uniformly sampled number, $u \in \{0, 1\}$ on the y-axis can be converted to a position on the support of the distribution. Each point on the support sampled in this way will be made in accordance with the probability distribution. However, in complex cases producing such a transformation in closed form is almost always impossible meaning that other techniques are required.

The idea behind MCMC sampling is to draw samples based on a Markov Chain where the next draw is conditional on the *state* the sampler is currently in. Here the state of the sampler means the most recently sampled point from the parameter space. For a given chain, the marginal probability for a particular variable can be expressed in terms of the marginal probabilities of all previous samples in the chain such that

$$p\Big(\mathbf{z}^{(m+1)}\Big) = \sum_{n=1}^{m} p\Big(\mathbf{z}^{(n+1)}|\mathbf{z}^{(n)}\Big)p\Big(\mathbf{z}^{(n+1)}\Big) \tag{4.1}$$

where $\mathbf{z}^{(m+1)}$ represents the value of $\mathbf{z}$ at iteration $m + 1$.

In this report, a *blocked Gibbs* sampling scheme was developed (with help from Dimitris Vavoulis), and will be described in the next section.

4.2 Gibbs sampling

Gibbs sampling is a simple yet widely applicable class of Markov Chain Monte Carlo algorithms. It is a special case of the more general Metropolis-Hastings algorithm [20] although both were developed individually.

4.2.1 Blocked Gibbs Sampling

The models developed in Section 3.3 will all make use of a blocked Gibbs sampler. One major advantage of blocked Gibbs over collapsed Gibbs is the ability to parallelise the updates for the cluster assignments because, within each block, the updates are independent.

4.3 Applied to the model

This section will outline the Gibbs sampler used to sample from the posterior of the model. To recap, the underlying, generative statistical model that applies to all models (described in Section 3.1) is

$$\begin{aligned} \mathbf{x}_i | z_i = j, \boldsymbol{\theta} &\sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), \quad i = 1, \dots, N \\ z_i | \boldsymbol{\pi} &\sim \text{Mult}(\pi_1, \dots, \pi_K) \\ (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) &\sim G_0 \\ \boldsymbol{\pi} | \alpha &\sim \text{Stick-Breaking}(\alpha) \end{aligned}$$

The idea of the blocked sampler is to iteratively perform updates to all of one type of parameter in blocks, including the latent cluster assignments. Enough iterations should be performed so that there is enough time for the chain to converge to the posterior distribution and for enough samples from this distribution to be taken. Since this is a MCMC method the assignments at the next time step are only dependent on the assignments at the current time step.

In each iteration, the following updates will be performed when there are K clusters:

- Sample new parameter, $\theta_j^{(m+1)} = \left\{\mu_j^{(m+1)}, \Sigma_j^{(m+1)}\right\}$, for each cluster according to

$$p\Big(\theta_j^{(m+1)} \Big| \mathbf{X}, \mathbf{z}^{(m)}\Big) \propto p\Big(\mathbf{X}_j \Big| \theta_j^{(m+1)}, \mathbf{z}^{(m)}\Big) p\Big(\theta_j^{(m+1)}\Big) \tag{4.2}$$

  where $\mathbf{X}_j$ denotes the points belonging to cluster k at iteration m i.e. those x_i for which $z_i = j$.

- Update each data points cluster assigned based on the updated parameters

$$p\Big(z_i^{(m+1)} = k \Big| \pi_j^{(m)}, \boldsymbol{\theta}^{(m+1)}, \mathbf{x}_i\Big) \propto p\Big(\mathbf{x}_i \Big| \theta_j^{(m+1)}\Big) \pi_j^{(m)} \tag{4.3}$$

  which boils down to a Multinomial

$$p\Big(z_i = k^{(m+1)} \Big| \pi_j^{(m)}, \boldsymbol{\theta}^{(m+1)}, \mathbf{x}_i\Big) \sim \text{Mult}\Big(p\Big(\mathbf{x}_i \Big| \theta_1^{(m+1)}\Big)\pi_1^{(m)}, p\Big(\mathbf{x}_i \Big| \theta_2^{(m+1)}\Big)\pi_2^{(m)}, \dots, p\Big(\mathbf{x}_i \Big| \theta_K^{(m+1)}\Big)\pi_K^{(m)}\Big) \tag{4.4}$$

- Recalculate the cluster weights based on an adjusted stick-breaking process such that:

$$p\Big(\beta_j^{(m+1)} \Big| \mathbf{z}^{(m+1)}\Big) = \text{Beta}\Big(\alpha + N_j^{(m+1)}, \alpha + N - \sum_{l=1}^{K} N_j^{(m+1)}\Big) \tag{4.5}$$

$$p\Big(\pi_j^{(m+1)} \Big| \beta_j^{(m+1)}\Big) = \beta_j^{(m+1)} \prod_{l=1}^{K-1} \Big(1 - \beta_l^{(m+1)}\Big) \tag{4.6}$$

In (4.2) to (4.6) the independence of the updates is particularly useful when updating the latent variable $\mathbf{z}$ because these updates can be performed without considering the latent variables for every other data point. This allows the parallelisation of the large number of updates resulting in a significant increase in speed. It can be shown that, in the limit, this algorithm generates clusterings from the posterior distribution of clusters as required [30].

Chapter 5

Posterior Inference

Due to the stochastic nature of Gibbs sampling, it is necessary to define a method to determine what final cluster assignments were implied by the Markov chain. This is a major short-coming of MCMC sampling methods when compared to other deterministic approaches such as Expectation-Maximisation (EM) and Variational Bayes in which the outcomes require no extra analysis. The reason there is less complexity with both of these methods is because they both involve maximising the likelihood of the data which means there is only one local maxima given any starting position and the final clustering can be taken as the optimum result from the algorithm. With MCMC, the assignments at the final iteration are no more valid than any other. Hence, deciphering the results can be somewhat challenging and there is no "best" way to do it. After some research a scheme for inferring the outcome from the Markov chain, and also how effectively that outcome classifies the data, was developed and will be explained below.

5.0.1 Cluster assignments

Performing S samples of the Markov Chain results a set of potentially different cluster assignments $(\mathbf{z}^1, \mathbf{z}^2, \ldots, , \mathbf{z}^S)$. An obvious way to determine the most accurate cluster is to find the set of assignments, $\mathbf{z}_{MAP}$, that maximises the probability of the maximum a posteriori (MAP) assignments. More precisely, for each set of assignments the dataset would be broken up into its respective clusters; the most likely parameters are assigned to each cluster conditioned on the data; the joint probability of all these parameters would be calculated; and then the most probable set of assignments taken.

Although this seems like a completely reasonable approach to finding the best assignment, in practise the MAP clustering may only be slightly more probably than the next best alternative, yet represent very different cluster assignments. This indicates that the resulting clustering is not particularly reliable or stable. Another issue is that using such a method discards the majority of the Markov Chain, which is a waste of valuable information.

An alternative method suggested in [13] consists of three parts: calculating the pairwise probability of each pair of data points being in the same cluster given the results of the entire Markov Chain; subtracting this matrix from a matrix of ones to calculate a distance matrix; and then use the distances in some sort of hierarchical clustering algorithm to determine the clusters. [14] highlights some issues with this approach in that it is somewhat strange to apply a sophisticated statistical model to some data and then determine the final output based on an ad-hoc heuristic based approach like hierarchical clustering. Instead, in [14], a method they refer to as *least-squares model-based clustering* is proposed which was utilised effectively in [32]. A key aim when developing this method was to produce stable results and consider as much information as possible from the entire Markov Chain. It is this adjusted method that will be used in this report.

With each of these methods, as well as any other inference methods for MCMC sampling, there is always a *burn-in period* such that it is possible to assume that the resulting cluster assignments $\mathbf{z}^B, \mathbf{z}^{B+1}, \ldots$ after a sufficiently large number of iterations, B, accurately approximate draws from the posterior distribution. Any results before B are discarded as noise occurring before convergence to the true posterior distribution.

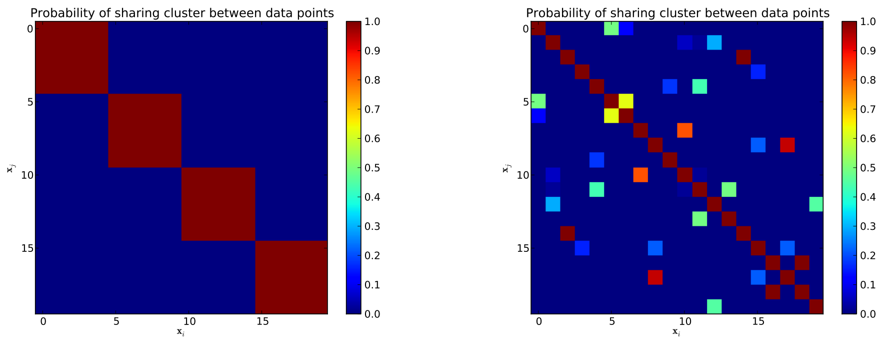
When considered in the context of breast cancer data, patient's samples that get assigned to the same cluster a large proportion of the time after the burn-in period, are likely to have been drawn from the same underlying cluster component. This suggests that it is likely the two patients have the same cancer subtype. Thanks to the flexibility of the Dirichlet process prior an additional benefit is that the posterior distribution over the number of clusters can also be easily estimated.

Pairwise probabilities

Let the number of iterations considered to be in burn-in period be denoted B and the total number of samples, including those used as burn-in be denoted S. This step involves considering the cluster assignments $(\mathbf{z}^B, \mathbf{z}^{B+1}, \mathbf{z}^{B+2}, \ldots, , \mathbf{z}^S)$ at each iteration of Gibbs sampler. A probability is then assigned to each pair of samples indicating how empirically likely it was for them to appear in the same cluster. More formally, for samples i and j

$$P_{ij} = \frac{\text{count}(\#\text{ of samples after burn in for which } z_i = z_j)}{S - B} \tag{5.1}$$

which results in an $N \times N$ matrix. This can be represented as a heat map showing how likely any two data points were clustered together. Figure 5.1 shows a the results from 5.1 showing a perfectly clustered and randomly clustered set of results.



(a) Normal case (b) Random case

Figure 5.1: Different probability plots. 5.1a shows when there are clearly defined clusters. 5.1b shows when the data is randomly generated.

Least-squares model-based clustering

Now that the probabilities associated with each pair of data points being in the same cluster have been calculated. The next step is to find a way to find the most representative clustering from this set. Using the approach laid out in [14], the goal is to find the least-squares cluster assignments, $\mathbf{z}_{\text{LS}} = \mathbf{z}^{(m)}$ for $m \in B, \ldots, S$, that minimises the distance between its association matrix, $\delta\big(\mathbf{z}^{(m)}, \big)$ and the pairwise probability matrix, $\mathbf{P}$, calculated previously (and demonstrated in Figure 5.1). Hence, $\mathbf{z}_{\text{LS}}$ can be calculated as

$$\mathbf{z}_{\text{LS}} = \underset{\mathbf{z} \in \{\mathbf{z}^B, \ldots, \mathbf{z}^{\mathbf{S}}\}}{\arg\min} \sum_{i=1}^{M} \sum_{j=1}^{M} (\delta_{i,j}(\mathbf{z}) - P_{i,j})^2 \tag{5.2}$$

The advantage of taking this approach is that it uses information from all stages of the Markov chain and is an attractive option because it selects the "average". This meaning that its accuracy should increase with the number of iterations performed.

5.0.2 Number of clusters

The number of clusters is of great interest in this model. As mentioned before, one of the great strength of the Dirichlet process prior is that it allows the number of clusters to be left unset *a priori*. During a run of the model there are a constant number, K, of potential cluster components. When using blocked Gibbs sampling, only a subset of those clusters will have any data points allocated to them.

The number of clusters at any point in time is the number of active cluster components. An important characteristic of Dirichlet process mixture models is that there is a continuously varying number of active clusters, each of which is initialised with a fraction of the data points. This results in a very noisy output but by considering the number of cluster components at every iteration of the model, a posterior probability over the number of clusters can be calculated. In its simplest form the probability of any number of clusters can be approximated by taking the number of times that a certain number of clusters appears over all cluster assignments $(\mathbf{z}^B, \ldots, , \mathbf{z}^S)$. To avoid confusing the output due to the inherent noise over the number of clusters, only clusters that contain more than one point will be counted at each time step. This can make a significant different to the posterior over the number of closers. Figure 5.2 shows just how significant the change can be. Since, a cluster with one element is effectively not a clustering, this is a fair adjustment to make.

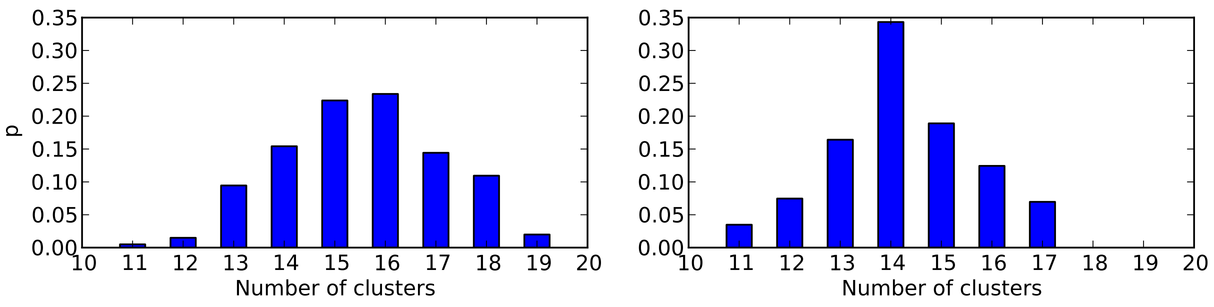


Figure 5.2: The posterior over the number of cluster when including components with one element (left) and excluding these from the count (right). By excluding the single point clusters there is an unsurprising shift to the a smaller number of clusters on average. More interestingly, there is a result that becomes significantly more likely than the rest indicating a stronger predictive outcome.

5.0.3 Clustering accuracy

In the case of synthetic data, or real data with known class labels, the effectiveness of each model must be measured in some way so that different models can be compared against each other to find the best. A common metric is the Rand index which originated in [37] and was used in [2] among others. However, the adjusted Rand index that was developed in [38] will be used in this report. The adjusted Rand index has been used effectively in [13, 15, 34, 39], and was also the recommended index of choice from an extensive study performed in [40].

The Rand index will be briefly described and then built upon to define the adjusted index, highlighting the improvements therein. In both cases the discussion revolves around a set of n data points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ where $U = \{u_1, \ldots, u_R\}$ and $V = \{v_1, \ldots, v_C\}$ represent two different partitions of those data points such that $\cup_{i=1}^{R} u_i = X = \cup_{j=1}^{C} v_j$ and $u_i \cap u_{i'} = \emptyset = v_j \cap v_{j'}, \forall i, i' \in \{1, \ldots, R\}$ and $\forall j, j' \in \{1, \ldots, C\}$, where $i \neq i'$ and $j \neq j'$. When applied to this model, U represents the known underlying classes of the data and V represents the cluster assignments output from the model after applying least-squares based clustering.

Rand index

The idea behind the Rand index is to compare the number pairs of points that were correctly clustered to the number of pairs of points that were incorrectly clustered, with the assumption that more pairs of points clustered together correctly means a better overall clustering.

There are four possible outcomes after an experiment. The count of each possible outcome will be assigned to a different variable. Let a be the number of data points in the class class in U and the same cluster in V. Let b be the number of pairs of data points that are in the same class in U but different cluster in V. Let c be the number of pairs of data points that are in a different class in U but the same cluster in V. Let d be the number of data points that are in different classes in U and clusters in V.

Defined in this way, a and d can be considered correct and b and c as incorrect. The Rand index is simply the ratio of the correct counts to the total. That is, $RI = \frac{a+d}{a+b+c+d}$. Hence, a Rand index is closer to one when there are more correct assignments (with a maximum of $RI = 1$ when the clustering match completely) and gets closer to zero as the number of disagreements increases.

Adjusted Rand index

An issue with the Rand index raised in [38] is that the expected value of two random partitions does not take a constant values (which would ideally be zero). The adjusted Rand index developed in [38] assumes the generalised hypergeometric distribution as the model of randomness. Put more simply, it assumes that the U and V partitions are picked at random such that the number of data points in the classes and clusters are fixed. Let n_{ij} be the number of objects that are in both class u_i and cluster v_i and $n_{i.}$ and $n_{.j}$ be the number of data points in class i and cluster j respectively. A tabular representation of these assignments can be seen in Table 5.1.

Class \Cluster	v_1	v_1	$\dots$	v_C	Sums
u_1	n_{11}	n_{12}	$\dots$	n_{1C}	$n_{1.}$
u_1	n_{21}	n_{22}	$\dots$	n_{2C}	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
u_1	n_{R1}	n_{R2}	$\dots$	n_{RC}	$n_{R.}$
Sums	$n_{.1}$	$n_{.2}$	$\dots$	$n_{.3}$	$n_{..} = n$

Table 5.1: Tabular representation of the adjusted Rand index calculations

The general form of an index with a constant expected value is $\frac{\text{index}-\text{expected index}}{\text{maximum index}-\text{expected index}}$, which is bounded above by 1, and takes the value 0 when the index equals its expected value.

Under the generalised hypergeometric model used in [38], it was shown that

$$\mathbb{E}\left[\sum_{i,j}\binom{n_{ij}}{2}\right] = \left[\sum_i\binom{n_{i.}}{2}\sum_j\binom{n_{.j}}{2}\right]\Big/\binom{n}{2}$$

Since $a + d$ can is equivalent to $\sum_{i,j}\binom{n_{ij}}{2}$, the adjusted Rand index to be written out as

$$\text{AR} = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i\binom{n_{i.}}{2}\sum_j\binom{n_{.j}}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{n_{i.}}{2} + \sum_j\binom{n_{.j}}{2}\right] - \left[\sum_i\binom{n_{i.}}{2}\sum_j\binom{n_{.j}}{2}\right]/\binom{n}{2}}$$

The advantage of this index is that the expected value for independent clusterings is zero resulting in a better quantification of how well two clusterings match because any result above 0 can be taken as success in some way. The same is not true of the Rand index.

Chapter 6

Results

In this chapter the results from the various models when run on various different data will be presented. After initial tests with real data, the behaviour of the model, and the hyperparameters, were very hard to understand with so few datasets all of which were very noisy and hard to characterise. This meant that optimising the method of setting the hyperparameters in the model and fairly assessing and comparing each model was very difficult. As a result, the production and analysis of synthetic data allowed, not only for the models to be compared based on outcome, but also allowed for greater intuition and understanding of a Dirichlet process and the various covariance structures used. Since the data sets involved with this kind of classification are often rather small, these synthetic data sets are being used to provide a richer set of data to validate on, saving the true data purely as test data.

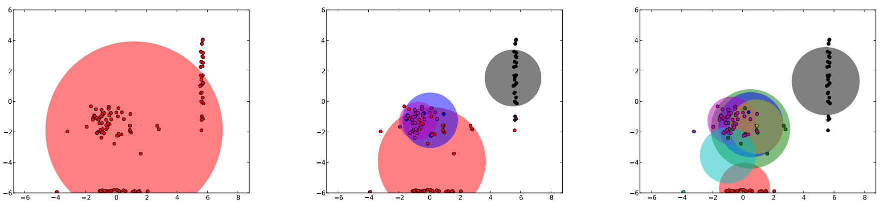
This section starts by giving some simple examples of the Dirichlet process mixture in action in a number of different situations as well as highlighting some key characteristics of the model. The framework used for generating a range of synthetic data will then be explained followed by a set of experiments used to decide on hyperparameters for the models. Then the performance of each model will be compared using wider range of datasets, making use of these hyperparameters. Some extra analysis goes into the posterior over the number of clusters from each model, before running experiments on much higher dimensional data. Finally, a number of real gene expression data sets are introduces and the performance of each model given.

6.1 The model in practise

Before jumping straight into analysis of the model, a sense for exactly how this model works this section will show some simple examples of the process.

6.1.1 The different covariance structures

To demonstrate a typical run of the algorithm, 500 iterations of the model were performed (with an iteration described in Chapter 4) initialised with all points assigned to a single cluster to help with the visualisation. Figures 6.1, 6.2, and 6.3 shows the model at the start and the state of the model at iterations 10, 50, 100, 200, and 500. In this case some non-spherical synthesised data is used with the spherical, equal volume spherical (evspherical), and elliptical covariance structures respectively. In these tests, the additional flexibility of the elliptical covariance is very apparent. However, this may not be indicative of how well it classifies higher dimensional data which will result in the larger number of parameters known to impact on its performance. With the spherical covariance, the varying sized clusters highlight how the model is free to assign different covariance matrices to each of the cluster components. With the equal volume case, this is not so and the additional constraint is clearly visible. Another point of interest is that the equal volume covariance seems to have a harder time breaking up the original big cluster when compared to the other two.


(a) Iteration 0 (b) Iteration 10 (c) Iteration 50

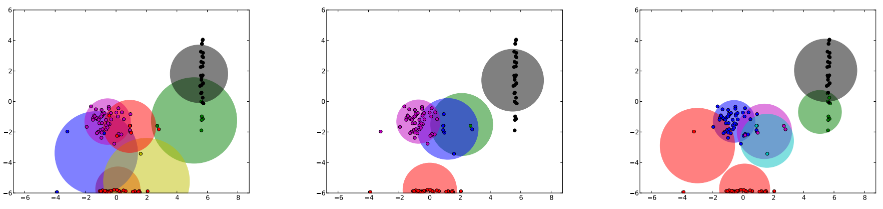
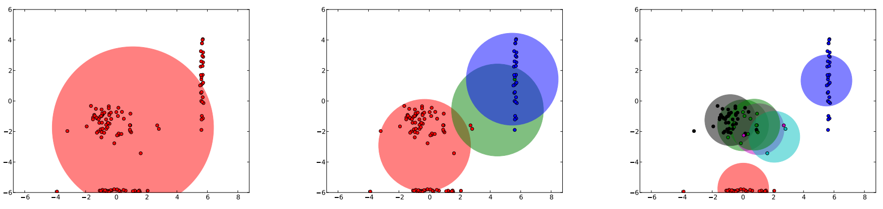

(d) Iteration 100 (e) Iteration 200 (f) Iteration 500

Figure 6.1: Spherical covariance. Here it is possible to see how the spherical covariance structure struggles to correctly classify data that is not spherically shaped. Although in iteration 200 it would actually classify the data well, iterations 100 and 500 show how this is not a stable classification and hence will not result in a very confident result.


(a) Iteration 0 (b) Iteration 10 (c) Iteration 50

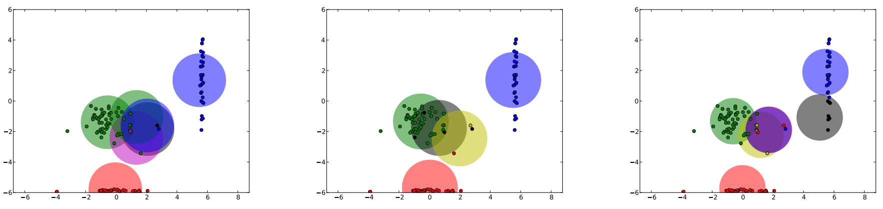
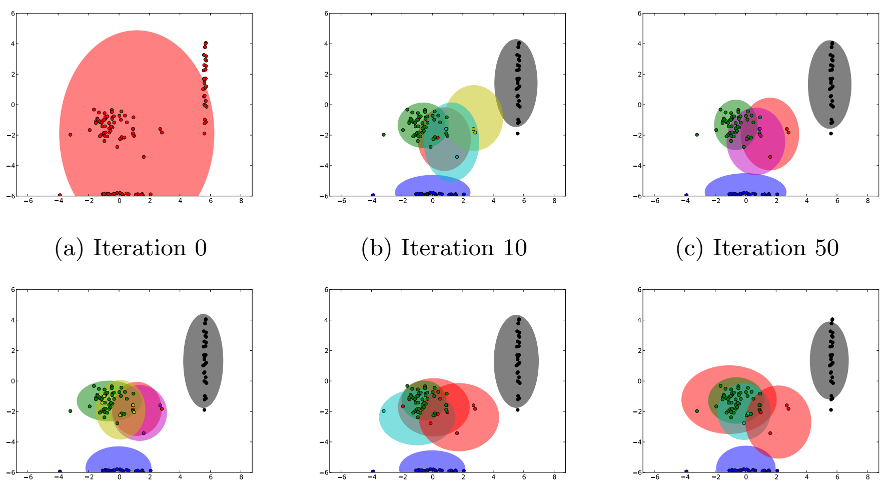

(d) Iteration 100 (e) Iteration 200 (f) Iteration 500

Figure 6.2: Equal volume spherical covariance. These plots just visualise the additional constraint applied to the equal volume spherical covariance and show how although it does manage to pick up on some key parts of the data, it does not *look* like the right kind of mixture model.

The dynamic nature of the simulations shown in Figures 6.1, 6.2, and 6.3 are very characteristic of the Dirichlet process mixture model. It can be seen clearly that the number of clusters is constantly changing and the position and variances of the clusters is being updated constantly.

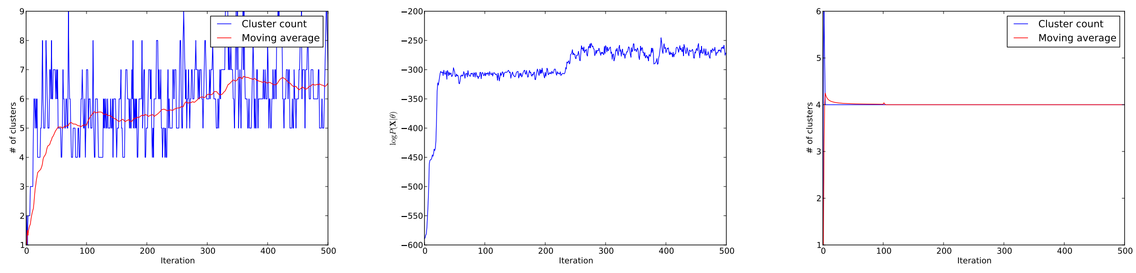
Figure 6.4 shows the cluster count of the model as well as the likelihood of the data over time for the same run. The cluster count is very noisy and shows how cluster components are constantly added and removed. The moving average line shows how the number of clusters can maintain a relatively constant value. Together with the plot of the likelihood of the data, it can be seen how the discovery of a new, stable cluster component results in a sudden jump in the likelihood of the data.



(a) Iteration 0 (b) Iteration 10 (c) Iteration 50

(d) Iteration 100 (e) Iteration 200 (f) Iteration 500

Figure 6.3: Elliptical covariance. This figures show how stably the elliptical covariance can classify clusters that are not spherical in their shape. The blue and black clusters in this plot would come out as very confident clustering in the post markov chain analysis.



(a) 2-d cluster counts (b) Likelihood of the 2-d data (c) 100-d cluster counts

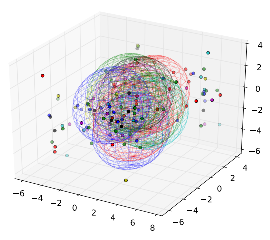
Figure 6.4: Likelihood of the data

The nature of the Dirichlet process is in contrast to other algorithms such as k-means or the EM-algorithm which converge to a single solution in a deterministic manner. Figure 6.4 shows just how noisy the number of clusters is but how this constantly changing model allows for big jumps in the likelihood of the data, and hence much better classifications than would otherwise be possible. However, when the Dirichlet process is run at low dimensions the number of clusters is significantly more volatile than when at higher dimensions. Figure 6.4c shows the cluster counts when considering 100 dimensional data rather than 2 dimensional data. In this case the number of clusters very quickly converges to a likely solution and there is no change over the subsequent 500 iterations. This highlights a significant problem when considering much higher dimensional data because the chance of forming a new cluster becomes much smaller, and hence fewer new clusters are formed. One way that is common when using a Dirichlet process mixture model is to instead start with the points randomly assigned across all the clusters. By doing this the flexibility of the model is given a greater chance to find the different clusters of points and the exact tuning of the hyperparameters becomes less important.

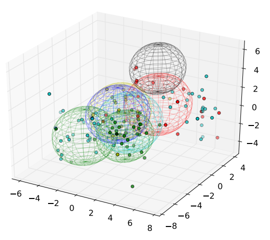
6.1.2 Higher dimensions

When considering much higher dimensional data, the model can converge very quickly to a solution if the cluster centres are sufficiently spread. In this case, the Dirichlet process prior only plays a part in how these very early assignments come about but allows for a very speedy classification to be made. However, it can mean that the Dirichlet process gets stuck for long periods of time on sub optimal clustering. The reason for this behaviour is that is it very unlikely for a new draw from the base distribution, G_0, to describe a cluster component that falls close enough to some data points to allow it to take that point away from some other cluster that already has a large number of data points. Part of the reason for this problem at higher dimensions is thanks to the very *rich-get-richer* nature of the Dirichlet process which stack the probability in favour of the larger cluster, where this is compounded by the exceptional low probabilities associated with high dimensional space. However, it is exactly the *rich-get-richer* trait that makes it so applicable as a clustering algorithms. The number of iterations required at $M \approx 1000$ dimensions for a new cluster component to be assigned any points a lot more than when $M \approx 100$. Although no in-depth analysis was done in this report, the exact nature of this change when considering higher dimensions could be interesting.

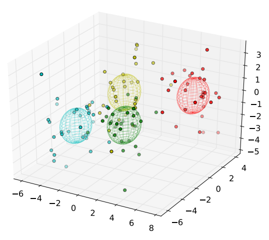
Hence, rather than running the simulation for tens of thousands of iterations a much faster and more stable solution will be used. The same model-base least squares technique that was used during a single simulation will be used to take the "average" clustering that resulted from a large number of such simulations. In each simulation the points will be randomly assigned to each cluster component and then will be run for a only 20 iterations. The analysis from this point can be considered the same as if each one of these best clustering was a single iteration of the model. That is: the probability of any two points being in the cluster will be calculated; converted to a distance matrix; then the cluster assignment with an association matrix that is closest to this will be taken as the best clustering.



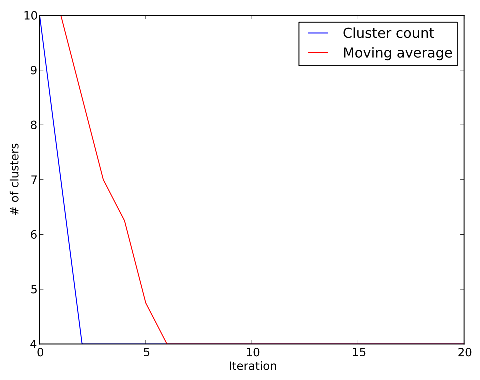
(a) Iteration 0



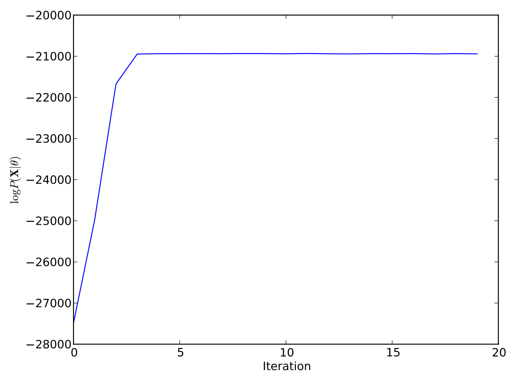
(b) Iteration 1



(c) Iteration 5



(d) Cluster counts



(e) Log-likelihood of the data

Figure 6.5: 100-d with few iterations. Each plot shows the first three dimensions of the data. This show just how quickly the higher dimensional data can converge to the correct result. Each cluster starts as a very random, and unlikely, selection of points and over the first few iterations this big sets are broken down into much smaller cluster components. By starting from random a lot more of the parameters space relevant to the data is considered much more quickly.

Although these plots allow for nice visualisation of the data at low dimensions when considering $M \approx 1000$ they become a lot less insightful, especially when the majority of the data is noise. Instead, the success of the classification is reduced down to a much more manageable index like the Rand index so that comparisons between models and across data can be made.

6.2 Synthetic data sets

Testing on real data early in the development of the model was not instructive as to the relative strengths and weaknesses of different models or what impact the hyperparameters played in the success of the model. It became apparent that covariance structure plays a particularly important role when considering the very high dimensional data associated with gene expression samples. Another important feature of a model is how well it handles the noise that is ubiquitous in gene expression readings.

In order to produce informative results about the strength and weaknesses of each model, significant effort was put into producing data that can satisfy a range of properties. The creation of the data is dependent on a number of features:

- Data covariance type - is the covariance of each cluster spherical or ellipsoidal. Is the covariance of each cluster the same (equal volume) or does it vary across clusters (unequal volume).
- Cluster variance, s - what is the average diagonal entry of the covariance matrix. With spherical covariance this is equivalent to the variance along each dimension.
- Cluster centre variance, s_0 - What is the variance of the centres of the clusters when they are created.
- Percentage noise components - How many of the dimensions are simply random noise that give no information as to the cluster assignments.
- Dimensionality, M - dimensionality of the data
- Number of clusters, K - how many different cluster centres are there.

6.3 Setting the hyperparameters

After experimentation, it became clear the setting of the hyperparameters *can* have a massive impact on how successfully the model can classify the data. In order to get a better sense of exactly how these parameters influence the success of the model a large number of simulations were performed varying a single parameter each time. These simulations were set up in such a way as to allow easy comparison between the effect of varying different hyperparameters in the model.

Initially synthesised data sets with the following traits were used:

- Dimension M - 2, 5, 15, 50
- Cluster centre variance σ_0 - 40
- Cluster variance σ_j - 40
- Covariance type: Equal volume spherical (all clusters shared the same spherical covariance matrix)
- Number of noise components: 0

The data was randomly generated for each simulation but using the same random number seed so that the data was identical for all simulations of the same dimension. Data was recorded for two different seeds. Ideally more seeds (and hence more data sets) would be used so that the outcome from the model is more representative of how well the hyperparameters perform and less to do with chance. However, at this stage having two should suffice to get a sense for how the hyperparameters effect the outcome of the model.

Similarly, because of the number of simulations that need to be run a relatively short burn-in period of 200 iterations and a total simulation length of 400 will be used.

The hyperparameters of interest are:

- α_0 - the concentration parameter for the Dirichlet process
- α - the shape of the variance prior
- β - the scale of the variance prior
- λ - the relative precision of the clusters compared to the means

In order to get a sense of how the setting of these parameters effected the outcome, a single parameter was adjusted while the rest were help constant. Initial arbitrary choices for the parameters letting $\alpha_0 = 1$, $\alpha = 2$, and $\lambda = 1$ leaving β to be calculated as the overall standard deviation of the data as in [34].

The first step to finding the right parameters for the model was to see if these initial guesses were suitable. Hence, each parameter was tested over a range of values for a number of different dimensional data for each model covariance structure. Twenty different values were tested for each parameter across the three models over four different dimension, each run with two different seeds. This results in 480 simulations per parameter and should give plenty of indication as to the behaviour of each.

In order to quickly and efficiently gather this much data, Amazon Web Services (AWS) was utilised and significant effort was put into creating a dynamic and scalable server management system. This involved flexible addition and removal of servers so that a large number of simulation could be run in parallel. At times anywhere up to 50 servers were being used in order to quickly run the simulations required for thorough analysis of each model and parameter.

Figure 6.6 shows how adjusting λ effects the outcome of the model. It appears that in general the elliptical covariance structure does not perform well.

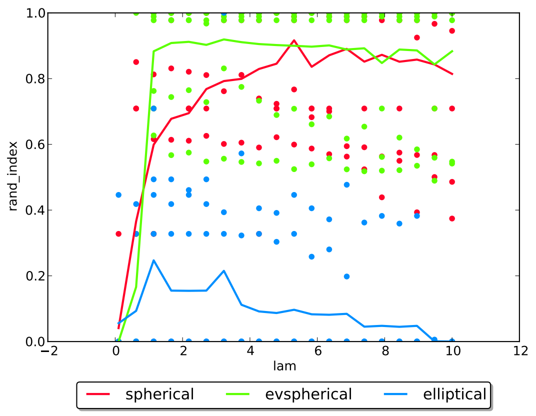
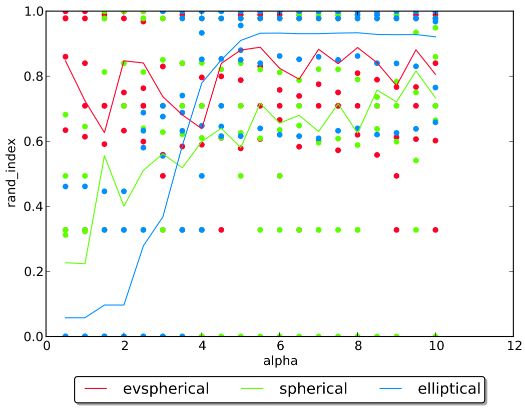


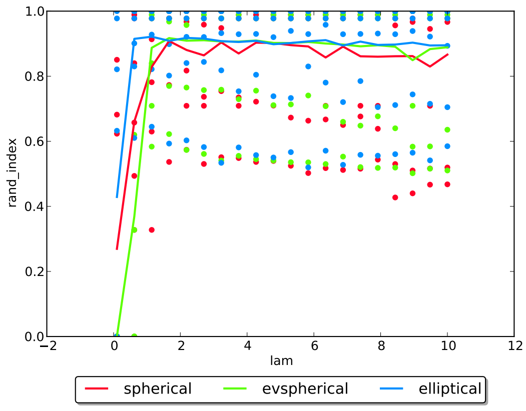
Figure 6.6: The results of adjusted the λ hyperparameter keeping all others fixed (or in the case of β calculating it as the variance of the data). The elliptical covariance appears to do poorly in all cases.

Figure 6.7a shows how successful each model was at clustering the data when α was set to a variety of values. It appears that although low alpha values cause bad clustering, there is no penalty for an overlarge alpha value.

In order to account for the simulation was re-run with a higher α value. Figure 6.7b shows how much better the model performs for all λ with $\alpha = 10$. Figure 6.7b also highlights how low values for λ have a serious negative effect on the accuracy of the resulting clusterings. However, it appears that the outcome is not that dependant on the exact value of λ as long as it isn't too small. Hence, $\lambda = 5$ was used for the remainder of the simulations.



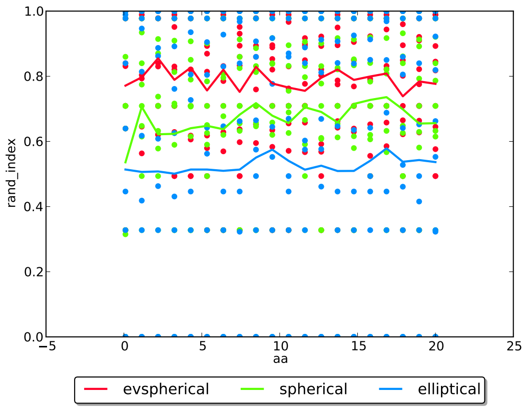
(a) Adjusting α



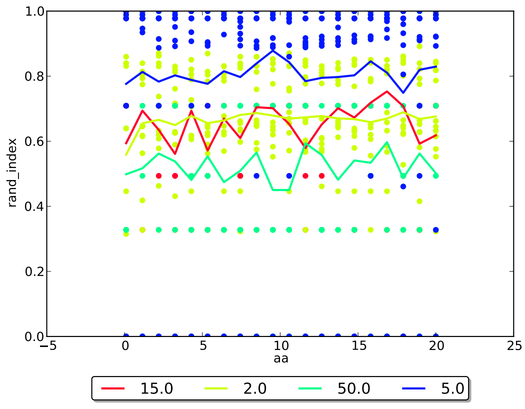
(b) Adjusting λ

Figure 6.7: Figure 6.7a show the results of adjusted the α hyperparameter keeping all others fixed (or in the case of β calculating it as the variance of the data). There is a very clear increase in the success of the clustering as α increases, particularly for the elliptical covariance model. Figure 6.7b shows the results of adjusted the λ hyperparameter keeping all others fixed (or in the case of β calculating it as the variance of the data) but with $\alpha = 10$. The results are clearly much stronger than in the previous case and in fact indicate that the outcome isn't particularly dependent on λ.

Another hyper parameter is the concentration parameter α_0. Similar tests were run with a variety of values and Figure 6.8 shows how the value of the α_0 parameter plays a minimal role in how the model performs.



(a) Varying α_0 coloured by base distribution G_0



(b) Varying α_0 coloured by dimensionality M

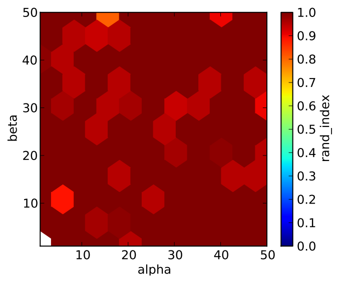
Figure 6.8: The results of adjusting the α_0 hyperparameter keeping all others fixed. The results indicate that the exact value of α_0 has only a very small effect on the success of the model. There appears to be no reliable maximum/minimum when considering either base distribution or dimensionality.

These experiments provide no conclusive evidence for particularly strong or weak values for λ and α_0 except to provide some sort of lower bound for λ. As such, from this point onwards it will be assumed that $\lambda = 5$ and $\alpha_0 = 1$ are reasonable values for these hyperparameters. This selection of α_0 is the same as in [34] although this, somewhat arbitrary, choice is questioned in [14] for being an under approximation of the expected number of clusters.

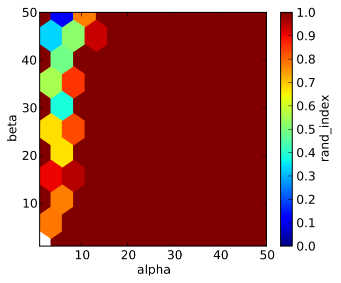
Tests up until this point have not been conclusive about how α and β should be selected. In order to get a better idea, a full range of combinations of these two parameters will be run on just $M = 15$ synthesised data but averaged across five different seed values and also considering the three different base distributions.

Figure 6.9 shows how the effect of changing α and β varies greatly between each model. In the case of the equal volume spherical covariance, α and β appear to play no part although the model never reaches a point of consistently classifying perfectly. With the spherical covariance however, there is a very clear issue when α is too small but otherwise the model seems unaffected. The biggest impact comes when considering the unequal volume elliptical model. The pattern across this parameter space is the most complex but also shows an apparent pattern. There are clearly issues when the α value is too small the there also seem to be problems when $\frac{\beta}{\alpha} \approx 2$. There appears to be strong results when $\frac{\beta}{\alpha} \approx \frac{1}{2}$. The scale of the values does not seem as important.

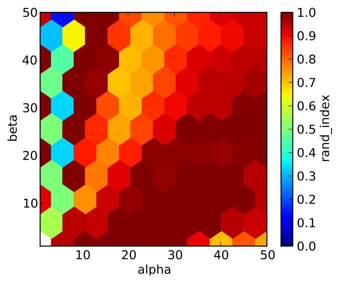
Again, as in [34] the overall standard deviation of the data will be used as the α value and β will be half of that. Having performed a number of tests to ensure this is valid, it seems reasonable.



(a) EVSpherical covariance



(b) Spherical covariance



(c) Elliptical covariance

Figure 6.9: The results of various values of α and β for the different covariance structures. This results show how the EVSpherical covariance does not seem particularly effected by the α and β parameters but it never reaches as consistently correct outcome as the spherical which is effected at low α values. Elliptical seems to be the most effected by the correct tuning of these parameters and there appears to be consistently strong performance when $\alpha = 2\beta$.

6.4 Testing across a range of datasets

At this point choices have been made as to how to set the full set of hyperparameters α, β, λ, and α_0 for each model. The next question is how well these models fair when clustering different types of data. In particular, since the final goal is to use these models on gene expression data, high-dimensional and noisy data sets will also be considered. Before testing on much higher dimensional synthetic data, some experiments with different cluster and cluster centre variances will be performed as well as with varying number of clusters.

Since the reason for this experimentation is to get a feeling for what situations these models can handle, four different classes of data will be classified. These settings that led to the production of these sets can be seen below:

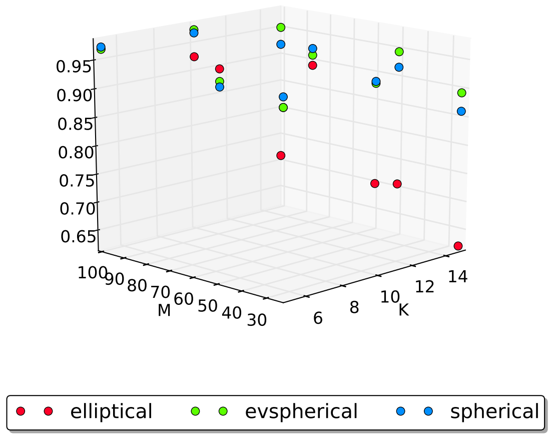
Name	% noise dimensions	s_0	s
Noiseless clear clusters	0	50	5
Noiseless less clear clusters	0	10	5
Noisy clear clusters	70	50	5
Noisy less clear clusters	70	10	5

Table 6.1: Descriptions of the different synthesised datasets being used.

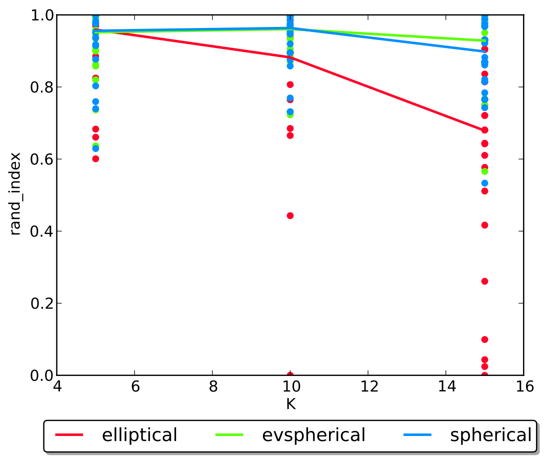
For noise dimensions, all data generated along that axis is drawn a random Normal distribution centred on the origin with a random variance.

Each of the datasets Table 6.1 were tested containing 5, 10, and 15 clusters with $M = 25, 50, 100$ and using spherical, equal volume diagonal and unequal volume diagonal covariances. Every model was tested against this data with 20 potential cluster components, $\alpha_0 = 1$, $\lambda = 5$, and with α and β calculated as described above.

One result that was to see how differently the models perform as the dimensionality and various parameters are changed. Figure 6.10 shows shows how the elliptical model struggles when there are more clusters. Figure 6.10 also shows how all models actually perform better as M increases.



(a) Average Rand indices over various M and K values



(b) How the average Rand index for each model varies with the number of clusters K

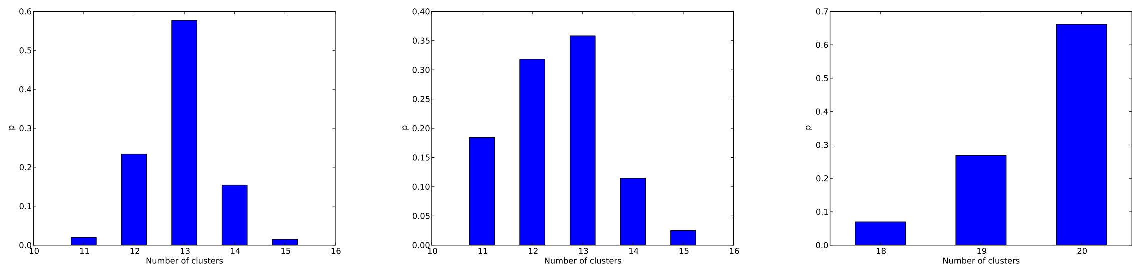
Figure 6.10

6.5 Number of clusters

In this section the behaviour of the different covariance structures on the inferred number of clusters will examined. The same range of datasets in Table 6.1 will be used. The simplest is clearly clustered noiseless spherically variant data, the most complex being noisy, less clearly clustered, elliptically variant data. Since the number of clusters had a big impact on the success of the models a medium value of $K = 10$ will be used. To compare the effects of dimensionality simulations were run for $M \in \{25, 100\}$.

A number of the simulations resulted in a posterior which was effectively a point mass at a single value indicating apparent certainty of the number of clusters. However, with some datasets this apparently certain value was different for each model. An example of this occurring was with noiseless clearly separated clusters with diagonal covariance. In this case, the elliptical model correctly predicted 10 clusters with 100% confidence, the spherical model predicted 9 with 100% confidence, and the equal volume spherical predicted 11 with 100% confidence.

On most of the data the models all performed well, achieving near perfect inference. In the more noisy and complicated data however, there were clearly some issues. One example is with the noisy, unclearly separated, spherical covariance data in which there was a large spread over cluster counts. Figure 6.11 shows has the posterior over the cluster changes between covariance structure. These results how just how much of an impact the covariance structure has in accurately representing the data. This plot demonstrates a sever shortcoming of the elliptical covariance and its tendency to create additional clusters that closely match the data but don't actually exist.



(a) Spherical covariance (b) EVSpherical covariance (c) Elliptical covariance

Figure 6.11: The posterior over cluster counts for the different models for noisy, unclearly separated, spherical covariate data

6.6 High dimensional data

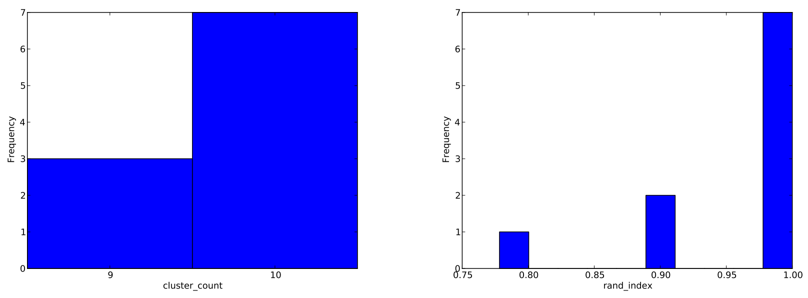
Since the nature of the model is to converge very quickly to a solution at high dimensions, in order to get a better sense of the true results, multiple simulations are performed concurrently for the same data but with various initial assignments. The best outcome from each of the Markov chains are then combined and considered as iterations in the same assignment averaging process used in the model. Although this is not ideal (since it involves the kind of model-averaging techniques that were to be avoided), by using such a scheme the model becomes much more effective and consistent in higher dimensions.

As a final test on synthesised data before seeing how the models fare on true data, the same general data sets were used as in the last tests. The difference now is that the number of dimensions will be increased dramatically to $M = 5000$. The results from these tests should give a better indication as to how well the models will be able to cluster true data and allow for any final adjustments before the last tests.

These tests involve trying ever combination of base distribution over 16 different randomly generated sets of data. Each test was run with 10 different starting assignments and the model-based square divergence was used to select the best clustering and there were only two non-perfect results. In both cases it was the the elliptical covariance that struggles on spherically distributed clusters.

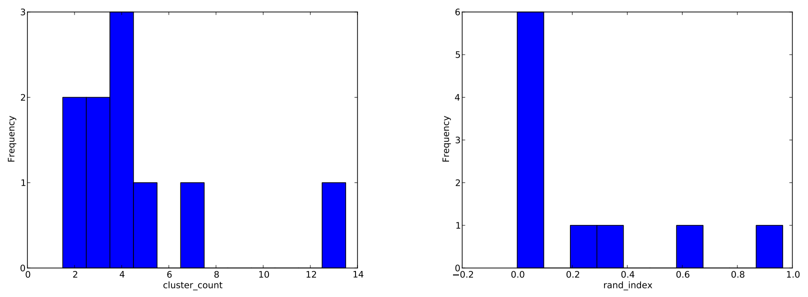
In the first case, where the data was noiseless and in spherical, but less separated clusters, the elliptical under approximated the number of clusters by 1 on several of the runs. Figure 6.12 shows the distribution over resulting Rand indices for these assignments. It is worth mentioning that these Rand indexes could not be calculated during this simulation since the labels required to calculate them are unknown. This is an example of where the averaging technique has not chosen the best clustering. Figure 6.12b shows how a number of the resulting cluster assignments were in fact perfect but the averaging chose differently. Since there were only 10 runs here, it could be expected with a higher number of simulations the result would have been better. In a number of other tests that resulted in a perfect scores, there were a few non-perfect runs of the simulation. However, by taking the “average” result it was possible to perfectly infer the data classes resulting in near perfect results.

The second, and much more severe failure was when trying to model the noisy, spherically variant clusters which were less spread. In this case the elliptical model seemed to have a lot of trouble correctly predicting the number of clusters. Figure 6.13 shows the distribution over the number of clusters in the resulting from each run, and the calculated Rand indices associated with those clusterings. It is clear that this covariance structure is constantly under approximating the number of clusters in the data and in doing so results in regularly poor Rand indices. However, it can be seen that one of the simulations did in fact very nearly perfectly classify the data but was unfortunately drowned out when the model-based least squares technique was applied.



(a) Likelihood of the 2-d data (b) Likelihood of the 2-d data

Figure 6.12: Likelihood of the data



(a) 2-d cluster counts (b) Likelihood of the 2-d data

Figure 6.13: Likelihood of the data

6.7 Real data

Four different real data sets will be considered and the results will determine how successful this model is at classifying gene expression data. Initially a low dimension ecoli data set will be run that has 5 dimensions but contains very asymmetric cluster distributions some of which are categorical. Secondly, a known larger leukaemia data set will be used that has 7 000 dimensions will be tested an the results compared between models. Finally, an unlabelled breast cancer data set will be run and the models will be used to try to classify the data and draw some useful conclusions.

After seeing the success of the least squares technique in the previous section, this will be applied to all models. However, with lower dimensional data the run will involve a larger number of iterations since it takes longer for the Markov chain to coverage when M is smaller. In all cases 100 different simulations will be performed, all with different randomly allocations of the data points. With the lower dimensional E.coli and yeast data sets 100 iterations will be performed. In the case of the higher dimensional colon cancer, leukaemia, and breast cancer data sets 20 iterations be simulation will be run. In both cases half of the number of iterations will be considered as burn-in.

The advantage of performing tests in this way is that in theory it is very parallelisable. When these simulations were performed they can be broken up into blocks of work each the size of a run of a single simulation. Since these simulations run in a matter of minutes, it is possible to run as many simulations as required in this time by introducing more machines to run them on.

All data, except the breast cancer data, can be found at `http://www.inf.ed.ac.uk/teaching/courses/dme/html/datasets0405.html`.

6.7.1 E.coli dataset

The E.coli data set has been used in [41, 1, 42] with the objective of predicting cellular localisation sites of E.coli proteins. In this data set there are 8 different cellular sites which represent the classes of the data. The attributes are signal sequence recognition methods, the presence of charge on N-terminus of predicted lipoproteins and 3 different scoring functions on the amino acid contents whether predicted as a outer membrane or inner membrane, cleavable or uncleavable sequence signal.

The reason this data set is of particular interest here is that some attributes in the data are categorical resulting in in symmetric cluster shapes. Also, if one of these categorical attributes is a strong indicator of the class of the data point, the variance will potentially be zero along that axis. Hopefully, this data will highlight the strength of an elliptical covariance in being able to handle this kind of unusual data.

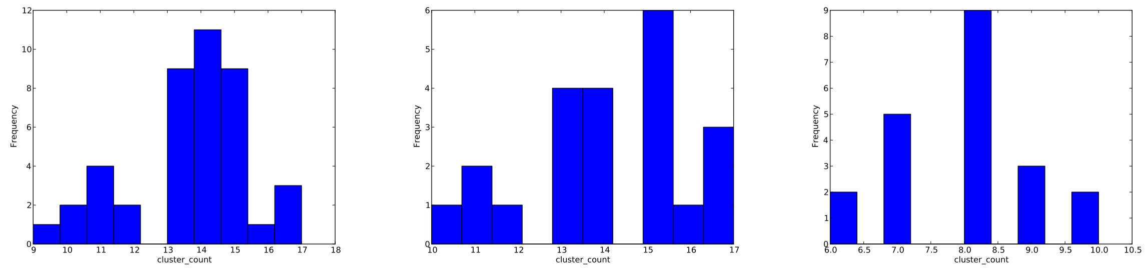
Since this data is relatively low dimensional, the outcome will be based on a single, long simulation with a large number of iterations since this has been shown to explore the parameter space more efficiently than at higher dimensions. 1000 iterations were used with a burn-in of 500 iterations. The simulation was run 10 times so are the consistency of the results could be compared. Since the data is of such a low scale, it was scaled up by a factor of 10 to make the numbers easier to handle.

This data set turned out to be relatively successful for all models and the results can be seen in Table 6.2.

Model	% Rand index
Spherical	0.529
Equal volume spherical	0.556
Elliptical	0.728

Table 6.2: Outcome from running on E.Coli data set.

The nature of the adjusted Rand index means that any number above 0 indicates some successful clustering so the model clearly was able to make some sense of the data. It appears that since the data contains categorical data, the elliptical covariance structure was able to better classify when compared to the other two. Figure 6.14 shows the resulting posteriors over the number of clusters.



(a) Spherical covariance (b) EVSpherical covariance (c) Elliptical covariance

Figure 6.14: The posterior over the number of clusters for the three different covariance structures. The actual number of clusters is 8 which means that all models are over approximating the number of clusters. However, the elliptical covariance is only slightly over approximating whereas the other two are severely over approximating.

6.7.2 Lung cancer

This dataset was used in [43] and resulted in clear clusters. It contains 71 samples of 918 genes with five different classes.

Classification on this data set varied drastically the spherical-based covariance and the elliptical covariance. However, all resulted in near zero adjusted Rand index scores. The elliptical covariance ended up assigning data points very randomly to clusters, with no resulting pattern, whereas the spherical covariances simply joined all data points into a single cluster. This dataset really highlighted some issues with the model and would be a great data set to explore further to try and find what parameters, if any, could result in a more successful result. The fact that all points are being combined into a single cluster indicates that the hyperparameters effecting the distribution over variances may not be set properly.

6.7.3 Breast cancer dataset

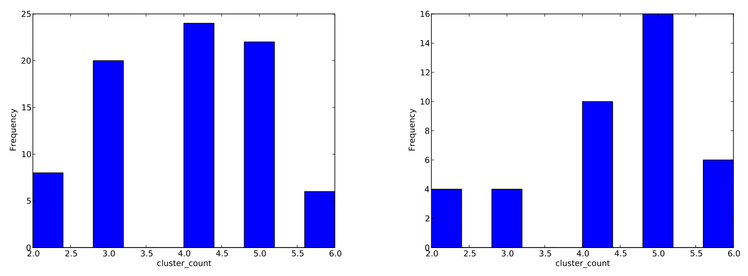
This dataset contains samples of 8278 gene expression levels from 135 patients. This dataset was provided by Colin Campbell and will be used as the final classification task. In this case the labels are unknown which means the Rand index cannot be calculated.

The outcomes can be seen in Table 6.3 shows the estimated number of clusters from each model.

Model	% Cluster count
Spherical	2
Equal volume spherical	4
Elliptical	20

Table 6.3: Outcome from running on E.Coli data set.

In the case of the elliptical covariance, the 20 represents the total number of cluster components that the model had. In other words it hit the limit of how many it could possibly use. This means that this result should be discarded because it has clearly not run correctly. The posterior over the cluster counts for the other two covariances can be seen in Figure 6.15. There is no very clear outcome but there is more of a distinctive most likely outcome in the equal volume spherical case. It is hard to determine a level of confidence for the classifications made by the model but looking at the posterior over the number of clusters can give some indication as to how stable the solution was.



(a) Spherical covariance (b) EVSpherical covariance

Figure 6.15: The posterior over the number of clusters for breast cancer data.

Chapter 7

Conclusions and future work

This report set out to determine if a Dirichlet process mixture model, extending it to the Hierarchical Dirichlet process mixture model, can be used effectively when classifying high dimensional gene expression data. All development work was done using purely open-source languages and software to investigate how effectively this software can be used. In addition, modern cloud-based capabilities were utilised to perform computationally expensive simulations, again to experiment with the kind of computational power that is now readily available for everyone.

Dirichlet process mixture models undoubtedly provide a very elegant and flexible method to potentially sidestep the need for model selection and averaging techniques, and hopefully this report has given some intuition as to what a Dirichlet process is and what impacts its success when used in a mixture model. Early on in this investigation it became apparent that the original Gaussian-Wishart prior used as a base distribution was not suitable at the dimensionality required. This led to analysing what impact, and what limits, different covariance structure resulting from imposing different constraints on the mixture model. The difference in the classification of the final results highlights just how much difference these changes can make.

Although performing well with synthesised data, all the model had issues when classifying real data. Data from microarrays inherently provide a number of challenges when performing clustering of any sort. There is an always a large amount of noise due to the process of gathering the data, as well as the systems themselves that are begin sampled. It is this noise, together with the challenges of working with such high dimensional probability distributions, that led to a great deal of misclassification. One shortcoming of the Dirchlet process as it was applied in these models is that is appears to lose the dynamic addition and removal of cluster components that make it effective at lower dimension. To get around this, multiple runs of the simulation were performed but doing this defeats one of the main advantages in using the Dirichlet process (that it can avoid the need for multiple runs) and hence warrants further investigation to try and find any was of avoiding this technique.

A large portion of the report was spent trying to understand, through the use of synthesised data, what caused the severe changes in performance, as a result there is still a thorough analysis of more real data sets to be performed. Ideally, the model would be tested more thoroughly on a wider variety of real data since this seemed to cause a great deal of problems even after coming from what was actually quite intermeshed synthetic data. To do this some sort of framework better than the brute force approach of testing all possible parameter values across all parameters should be devised.

Although initially the aim was to find a was to utilise data integration with gene expression data it became apparent that the creating a non-hierarchical Dirichlet mixture model to even classify data at one level is exceptionally challenging. Clearly, developing the model so as to enable data integration across a variety of datasets is still an ongoing goal.

Finally, the level of impact that the hyperparameters can have it set incorrectly (particularly with the elliptical model) indicates the need for more precise settings. Ideally some sort of hyperpriors (priors over the hyperparameters) would be used instead of fixed hyperparameter values so that the model can better adjust to various forms of input. As an alternative, a better technique for preprocessing the data in order to infer the right parameters could also be explored. This could be in the form of several smaller runs on subsets of the data runs before a final full run is performed.

Appendices

Appendix A

Model

A.1 Probabilities

A.1.1 Spherical covariance

$$p(\mathbf{x}_i|\boldsymbol{\mu}_j, \sigma_j^2, z_i = j) = \Big(2\pi\sigma_j^2\Big)^{-\frac{M}{2}} \exp\Bigg\{ - \frac{1}{2\sigma_j^2}||\mathbf{x}_i - \boldsymbol{\mu}_j||^2 \Bigg\}$$

$$p(\sigma_j^2|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma_j^2)^{-\alpha-1} \exp\Bigg\{ - \frac{\beta}{\sigma_j^2} \Bigg\}$$

$$\begin{aligned} p(\theta_j|\mathbf{X}, \mathbf{z}) =& \Big(2\pi\sigma_j^2\Big)^{-\frac{N_j M}{2}} \Big(\frac{\lambda}{2\pi}\Big)^{\frac{M}{2}} \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma_j^2)^{-\frac{M}{2}-\alpha-1} \\ & \exp\Bigg\{ - \frac{\sum_{x_i \in j} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2 + \lambda||\boldsymbol{\mu}_j - \boldsymbol{\mu}_0||^2 + 2\beta}{2\sigma_j^2} \Bigg\} \end{aligned}$$

$$\begin{aligned} p(\sigma_j^2|\mathbf{X}, \boldsymbol{\mu}_j, \mathbf{z}) =& \frac{\beta^\alpha}{\Gamma(\alpha + (2\pi)^{\frac{N_j M}{2}})}(\sigma_j^2)^{-(\alpha + \frac{N_j M}{2})-1} \\ & \exp\Bigg\{ - \frac{\sum_{x_i \in j}\sum_{m=1}^M ||x_{im} - \mu_{jm}||^2 + \beta}{\sigma_j^2} \Bigg\} \\ p(\boldsymbol{\mu}_j|\mathbf{X}, \sigma_j^2, \mathbf{z}) =& \left(\frac{1}{2\pi(\lambda^{N_j+1}\sigma_j^2)}\right)^{\frac{M}{2}(N_j+1)} \\ & \exp\Bigg\{ - \frac{\sum_{x_i \in j} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2 + \lambda||\boldsymbol{\mu}_j - \boldsymbol{\mu}_0||^2}{2\sigma_j^2} \Bigg\} \\ p(\mu_{jm}|\mathbf{X}_m, \sigma_j^2, \mathbf{z}) =& \Big(2\pi\sigma_j^2\Big)^{-\frac{N_j}{2}} \Big(\frac{\lambda}{2\pi\sigma_j^2}\Big)^{\frac{1}{2}} \\ & \exp\Bigg\{ - \frac{1}{2\sigma_j^2}\sum_{x_i \in j}(x_{im} - \mu_{jm})^2 - \frac{\lambda}{2\sigma_j^2}(\mu_{jm} - \mu_{0_m})^2 \Bigg\} \end{aligned}$$

A.1.2 Equal volume spherical

$$\log p(\mathbf{x}_i|\boldsymbol{\mu}_j, \sigma^2, z_i = j) = -\frac{M}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}||\mathbf{x}_i - \boldsymbol{\mu}_j||^2$$

$$\begin{aligned} p(\boldsymbol{\mu}_j|\boldsymbol{\mu}_0, \lambda, \sigma^2) =& \Big(\frac{\lambda}{2\pi\sigma^2}\Big)^{\frac{M}{2}} \exp\Bigg\{ - \frac{\lambda}{2\sigma^2}\sum_{m=1}^M(\mu_{jm} - \mu_{0_m})^2 \Bigg\} \\ p(\sigma^2|\alpha, \beta) =& \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-\alpha-1} \exp\Bigg\{ - \frac{\beta}{\sigma^2} \Bigg\} \\ p(\boldsymbol{\mu}_j, \sigma^2) =& \Big(\frac{\lambda}{2\pi}\Big)^{\frac{M}{2}} \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-\frac{M}{2}-\alpha-1} \\ & \exp\Bigg\{ - \frac{\lambda\sum_{m=1}^M(\mu_{jm} - \mu_{0_m})^2 + 2\beta}{2\sigma^2} \Bigg\} \end{aligned}$$

$$\begin{aligned} p(\theta_j|\mathbf{X}, \mathbf{z}) \propto & \; p(\mathbf{X}_j|\theta_j, \mathbf{z})p(\theta_j) \\ =& \Big(2\pi\sigma^2\Big)^{-\frac{N_j M}{2}} \Big(\frac{\lambda}{2\pi}\Big)^{\frac{M}{2}} \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-\frac{M}{2}-\alpha-1} \\ & \exp\Bigg\{ - \frac{\sum_{x_i \in j} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2 + \lambda||\boldsymbol{\mu}_j - \boldsymbol{\mu}_0||^2 + 2\beta}{2\sigma^2} \Bigg\} \end{aligned}$$

$$\begin{aligned} p(\sigma^2|\mathbf{X}, \boldsymbol{\mu}_j, \mathbf{z}) \propto & \; p(\mathbf{X}_j|\boldsymbol{\mu}_j, \sigma^2, \mathbf{z})p(\sigma^2|\alpha, \beta) \\ =& \frac{\beta^\alpha}{\Gamma(\alpha + (2\pi)^{\frac{NM}{2}})}(\sigma^2)^{-(\alpha + \frac{NM}{2})-1} \\ & \exp\Bigg\{ - \frac{\sum_{j=1}^K\sum_{x_i \in j}\sum_{m=1}^M ||x_{im} - \mu_{jm}||^2 + \beta}{\sigma^2} \Bigg\} \end{aligned}$$

$$\begin{aligned} p(\boldsymbol{\mu}_j|\mathbf{X}, \sigma^2, \mathbf{z}) \propto & \; p(\mathbf{X}_j|\boldsymbol{\mu}_j, \sigma^2, \mathbf{z})p(\boldsymbol{\mu}_j|\boldsymbol{\mu}_0, \sigma^2, \lambda) \\ =& \left(\frac{1}{2\pi(\lambda^{N_j+1}\sigma^2)}\right)^{\frac{M}{2}(N_j+1)} \\ & \exp\Bigg\{ - \frac{\sum_{x_i \in j} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2 + \lambda||\boldsymbol{\mu}_j - \boldsymbol{\mu}_0||^2}{2\sigma^2} \Bigg\} \end{aligned}$$

$$\begin{aligned} p(\mu_{jm}|\mathbf{X}_m, \sigma^2, \mathbf{z}) \propto & \; p(\mathbf{X}_{jm}|\mu_{jm}, \sigma^2, \mathbf{z})p(\mu_{jm}|\mu_{0m}, \sigma^2, \lambda) \\ =& \Big(2\pi\sigma^2\Big)^{-\frac{N_j}{2}} \Big(\frac{\lambda}{2\pi\sigma^2}\Big)^{\frac{1}{2}} \\ & \exp\Bigg\{ - \frac{1}{2\sigma^2}\sum_{x_i \in j}(x_{im} - \mu_{jm})^2 - \frac{\lambda}{2\sigma^2}(\mu_{jm} - \mu_{0_m})^2 \Bigg\} \end{aligned}$$

A.1.3 Gaussian-Gamma: Unequal Volume Elliptical Covariance

$$\begin{aligned} p(\mathbf{x}_i|\boldsymbol{\mu}_j, \sigma_j^2, z_i = j) =& (2\pi)^{-\frac{M}{2}}\big|\mathbf{D}_j\big|^{-\frac{1}{2}} \\ & \exp\Bigg\{ - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^\top\mathbf{D}_j^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_j) \Bigg\} \\ =& (2\pi)^{-\frac{M}{2}}\Big(\prod_{m=1}^M \sigma_{jm}^2\Big)^{-\frac{1}{2}} \\ & \exp\Bigg\{ - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^\top\mathbf{D}_j^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_j) \Bigg\} \end{aligned}$$

$$\begin{aligned} \log p(\mathbf{x}_i|\boldsymbol{\mu}_j, \sigma_j^2, z_i = j) =& -\frac{M}{2}\log(2\pi) - \frac{1}{2}\sum_{m=1}^M \log(\sigma_{jm}^2) \\ & - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^\top\mathbf{D}_j^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_j) \end{aligned}$$

$$\begin{aligned} p(\boldsymbol{\mu}_j|\boldsymbol{\mu}_0, \lambda, \sigma_j^2) =& (2\pi)^{-\frac{M}{2}}\Big(\prod_{m=1}^M \frac{\sigma_{jm}^2}{\lambda}\Big)^{-\frac{1}{2}} \\ & \exp\Bigg\{ - \frac{1}{2}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_0)^\top\Big(\frac{\mathbf{D}_j}{\lambda}\Big)^{-1}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_0) \Bigg\} \end{aligned}$$

$$p(\sigma_{jm}^2|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma_{jm}^2)^{-\alpha-1} \exp\Bigg\{ - \frac{\beta}{\sigma_{jm}^2} \Bigg\}$$

Bibliography

[1] A. C. Tan and D. Gilbert, "An empirical comparison of supervised machine learning techniques in bioinformatics," in *Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003 - Volume 19*, APBC '03, (Darlinghurst, Australia, Australia), p. 219222, Australian Computer Society, Inc., 2003.

[2] L. Wang and X. Wang, "A non-parametric bayesian clustering for gene expression data," in *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 556–559, Aug.

[3] D. M. B. T. L. Griffiths and M. I. J. J. B. Tenenbaum, "Hierarchical topic models and the nested chinese restaurant process," in *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, vol. 16, p. 17, 2004.

[4] S. Rogers, A. Klami, J. Sinkkonen, M. Girolami, and S. Kaski, "Infinite factorization of multiple non-parametric views," *Mach Learn*, vol. 79, pp. 201–226, May 2010.

[5] D. P. Sudbery, *Human Molecular Genetics.* Benjamin Cummings, 3 ed., June 2009.

[6] B. A. Pierce, *Genetics: A Conceptual Approach.* W. H. Freeman, Dec. 2010.

[7] T. M. Mitchell, "Machine learning. 1997," vol. 45, 1997.

[8] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," vol. 95, no. 25, p. 1486314868, 1998.

[9] J. A. Hartigan, *Clustering algorithms.* John Wiley & Sons, Inc., 1975.

[10] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church, "Systematic determination of genetic network architecture," *Nat Genet*, vol. 22, pp. 281–285, July 1999.

[11] C. Fraley and A. E. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," vol. 41, no. 8, p. 578588, 1998.

[12] D. Ghosh and A. M. Chinnaiyan, "Mixture modelling of gene expression data from microarray experiments," *Bioinformatics*, vol. 18, pp. 275–286, Feb. 2002.

[13] M. Medvedovic and S. Sivaganesan, "Bayesian infinite mixture model based clustering of gene expression profiles," *Bioinformatics*, vol. 18, pp. 1194–1206, Sept. 2002.

[14] D. B. Dahl, "Model-based clustering for expression data via a dirichlet process mixture model," p. 201215, 2006.

[15] K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery, and W. L. Ruzzo, "Model-based clustering and data transformations for gene expression data," *Bioinformatics*, vol. 17, pp. 977–987, Oct. 2001.

[16] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: a survey," vol. 16, no. 11, pp. 1370–1386, 2004.

[17] A. Brazma and J. Vilo, "Gene expression data analysis," vol. 480, no. 1, p. 1724, 2000.

[18] G. McLachlan and D. Peel, "Finite mixture models. 2000," 1997.

[19] G. J. McLachlan and K. E. Basford, "Mixture models. inference and applications to clustering," vol. 1, 1988.

[20] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, Feb. 2007.

[21] R. E. Madsen, D. Kauchak, and C. Elkan, "Modeling word burstiness using the dirichlet distribution," in *Proceedings of the 22nd international conference on Machine learning*, ICML '05, (New York, NY, USA), p. 545552, ACM, 2005.

[22] T. S. Ferguson, “A bayesian analysis of some nonparametric problems,” *The Annals of Statistics*, vol. 1, pp. 209–230, Mar. 1973. ArticleType: research-article / Full publication date: Mar., 1973 / Copyright 1973 Institute of Mathematical Statistics.

[23] Y. W. Teh, “Dirichlet process,” p. 280287, 2010.

[24] J. Sethuraman, “A constructive definition of dirichlet priors,” tech. rep., DTIC Document, 1991.

[25] D. Blackwell and J. B. MacQueen, “Ferguson distributions via plya urn schemes,” p. 353355, 1973.

[26] D. Aldous, “Exchangeability and related topics,” p. 1198, 1985.

[27] J. Pitman, “Combinatorial stochastic processes,” tech. rep., Technical Report 621, Dept. Statistics, UC Berkeley, 2002. Lecture notes for St. Flour course, 2002.

[28] C. E. Rasmussen, “The infinite gaussian mixture model,” vol. 12, no. 5.2, p. 2, 2000.

[29] D. Grr and C. Edward Rasmussen, “Dirichlet process gaussian mixture models: Choice of the base distribution,” vol. 25, no. 4, p. 653664, 2010.

[30] H. Ishwaran and L. F. James, “Gibbs sampling methods for stick-breaking priors,” vol. 96, no. 453, pp. 161–173, 2001.

[31] J. Kivinen, E. Sudderth, and M. Jordan, “Learning multiscale representations of natural scenes using dirichlet processes,” in *IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007*, pp. 1–8, Oct.

[32] S. Kim, M. G. Tadesse, and M. Vannucci, “Variable selection in clustering via dirichlet process mixture models,” vol. 93, no. 4, p. 877893, 2006.

[33] Yee Whye Teh, “Gaussian-derivation,” Aug. 2007.

[34] Z. S. Qin, “Clustering microarray gene expression data using weighted chinese restaurant process,” *Bioinformatics*, vol. 22, pp. 1988–1997, Aug. 2006.

[35] C. Fraley and A. E. Raftery, “MCLUST: software for model-based cluster analysis,” vol. 16, no. 2, p. 297306, 1999.

[36] M. I. Jordan, “The conjugate prior for the normal distribution,” Feb. 2010.

[37] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” vol. 66, no. 336, pp. 846–850, 1971.

[38] L. Hubert and P. Arabie, “Comparing partitions,” *Journal of Classification*, vol. 2, pp. 193–218, Dec. 1985.

[39] K. Y. Yeung and W. L. Ruzzo, “An empirical study on principal component analysis for clustering gene expression data,” tech. rep., Technical report, Department of Computer Science and Engineering, University of Washington, 2000.

[40] G. W. Milligan and M. C. Cooper, “A study of the comparability of external criteria for hierarchical cluster analysis,” vol. 21, no. 4, pp. 441–458, 1986.

[41] C. A. Ratanamahatana and D. Gunopulos, “Scaling up the naive bayesian classifier: Using decision trees for feature selection,” 2002.

[42] H. Zhang and C. X. Ling, “An improved learning algorithm for augmented naive bayes,” in *Advances in Knowledge Discovery and Data Mining* (D. Cheung, G. J. Williams, and Q. Li, eds.), no. 2035 in Lecture Notes in Computer Science, pp. 581–586, Springer Berlin Heidelberg, Jan. 2001.

[43] M. E. Garber, O. G. Troyanskaya, K. Schluens, S. Petersen, Z. Thaesler, M. Pacyna-Gengelbach, M. van de Rijn, G. D. Rosen, C. M. Perou, and R. I. Whyte, “Diversity of gene expression in adenocarcinoma of the lung,” vol. 98, no. 24, p. 1378413789, 2001.