Dirichlet Process Gaussian Mixture Model Gibbs Sampler for a 1-dimensional Behavioural Time Series Segmentation

Samuel Harris
harrissp@aston.ac.uk

April 29, 2015

Contents

List of Figures

Abstract

This project looks to implement methods introduced in the mathematics report, such as the *Dirichlet process* (DP) and the *Chinese restaurant process* (CRP) as well as new techniques including *mixture models* and *Gibbs sampling*. A general overview of the mathematics report is given before expressing the motivation for this project. This is to perform behavioral time series segmentation on some experimental movement data collected using a smartphone, with applications in modeling some typical Parkinsons disease (PD) symptoms. The next two chapters are literature reviews of both the preliminaries required for this work and mixture models. Firstly, *probabilistic graphical models* (PGMs) are explored in general both symbolically and graphically before describing a relevant example. Using the knowledge of stochastic processes from the report allows us to discuss *Markov chains*, *Markov chain Monte Carlo* (MCMC) and Monte Carlo integration. These are required in order to fully understand *Gibbs sampling*, a key topic in this project. The general Gibbs sampler is explained and applied to a PGM, ready for its application in mixture models. Both finite and infinite mixture models are discussed in general with some specific examples. Namely the *Gaussian mixture model* (GMM) and the *Dirichlet process mixture model* (DPMM), each with a PGM for visualisation. These are combined to form a *Dirichlet process Gaussian mixture model* (DPGMM), the primary topic of this project. These models have an associated Gibbs sampler which is explained and displayed symbolically. We are then able to put these methods into practice, but first we require both synthetic data, generated in Matlab, and experimental data, which is, in this case, accelerometer data collected from a smartphone. How these data sets were created and what they consist of is described with the aid of graphs. Each data set is then fed into the DPGMM Gibbs sampler, written in Matlab and the resulting plots are shown and explained, having tuned some parameters. These plots show time against acceleration displaying both the original data and the DPGMM Gibbs result. The objective is to reduce noise in the data but retain the abrupt jumps (or spikes) which are observed due to the nature of the experimental tests, instructions to which are in an appendix along with the Matlab code containing relevant comments and references to equations from the literature review. In the final chapter a summary of this project is given and we evaluate the performance of the Gibbs sampler algorithm. Alternative methods are mentioned which may have performed differently. Also some scope for future work is provided which would involve higher dimensional problems along with changing the orientation of the smartphone.
Keywords: *Gibbs sampling, mixture models, Dirichlet, Gaussian*.

Chapter 1

Introduction

1.1 Context

This mathematics project follows on from the mathematics report, which consisted of an exposition of a particular stochastic process, the *Dirichlet process* (DP). The report described some fundamental probabilistic and statistical concepts such as the foundations of probability, Bayesian statistics and a particular class of probability distributions, called the *exponential family*. The report discussed crucial properties of the Dirichlet distribution such as the conjugate posterior update property, the mean and variance, and the aggregation property. The construction of a stochastic process from a finite collection of random variables was then explained. The report also discussed how a stochastic process can be *exchangeable* and how this relates to *de Finetti's theorem*. Using this knowledge, the report then proved that a unique, exchangeable stochastic process with the properties of the DP exists. We then discussed some generative methods to draw samples from a DP. Using the DP model and these sampling methods, we can perform *DP clustering* or *DP mixture modelling*. This allows us to model some observed data with a mixture of distributions, where the number of component distributions in the mixture is not fixed but can grow to automatically accommodate observed data of increasing size. The *Dirichlet process mixture model* (DPMM), which is the primary subject of this project, is such a *Bayesian nonparametric method* and is very useful in practice. Such models have effectively infinite-dimensional parameter space. Advances in these methods have significantly developed over the last few years, for example, the *Gibbs sampling* inference method, which is introduced and implemented for the DPMM in this report.

1.2 Motivation

Today's smartphones contain a rich set of sensor devices including accelerometers, gyroscopes and GPS receivers. Because individuals carry smartphones with them continuously, it is possible to envisage using this sensor data to infer aspects of daily behaviour, which may also have medical implications for certain conditions. For example, Parkinson's disease (PD) is a progressive neurodegenerative disorder which affects approximately one in every 500 people in the UK, most of whom are aged over 50. The disease affects particular neurons in the brain stem which are implicated in movement. The accurate and objective quantification of PD movement symptoms may play a crucial role in developing specifically targeted treatments. However, there is currently no reliable technology for quantifying PD symptoms, which raises the possibility of using smartphones to accurately monitor symptoms of PD. This would also be advantageous because it is low cost and easily accessible to patients.

In this project we will look to perform behavioral time series *segmentation* on some experimental movement data collected using a smartphone. That is, we want to identify behavioural changes using the techniques described in this report. Using DP mixture models and an appropriate sampling algorithm,

will enable us to effectively ‘smooth’ out the noise in the measured sensor data between abrupt jumps, which we will assume occur at changes in behaviour.

Chapter 2

Preliminaries

2.1 Probabilistic Graphical Models (PGMs)

We can model and visualise (multivariate) *joint probability distributions* by means of diagrams called *probabilistic graphical models* (PGMs). This formalism helps us to manipulate the models to learn parameters and make inferences about unobserved variables from the data. Inspecting the graph allows us to determine important properties of the distribution such as the conditional independence structure [3]. Random variables are represented by nodes with links which encode the probabilistic relationships and (in)dependence between these variables. There are two types of graphs which represent joint distributions, namely *directed* where distributions can be factorised into *conditional distributions*, where the direction of arrows represent dependence between the random variables and *undirected* where dependence is factorised into so called 'potentials'. For our purpose we will only use *directed PGMs* due to the conditional probabilities we will encounter, these are also known as *Bayesian networks*. Using the product rule of probability we can write the *joint distribution* of N RVs $p(x_1, ..., x_N)$ in the form

$$p(x_1, ..., x_N) = p(x_N|x_1, ..., x_{N-1})...p(x_2|x_1)p(x_1), \tag{2.1}$$

which is a product of factors resulting in a graph with N nodes. For node x_1 to be a *parent* of node x_2, there must be a link going from node x_1 to node x_2, and we can say that node x_2 is a *child* of node x_1. In general the joint distribution is represented by a *factorisation* defined as

$$p(\mathrm{x}) = \prod_{n=1}^{N} p(x_n|pa_n), \tag{2.2}$$

where pa_n is the parent set of x_n and $\mathrm{x} = (x_1, ..., x_N)$. Graphical models have *Markov* properties due to the conditioning we have discussed. A set of parents and child nodes for a particular node is known as the *Markov blanket* for that node which is the set of nodes needed to treat that node as independent of all the others in the graph. Also, we define *co-parents*, a node which shares a common child with another [11]. We next show two examples for clarification.

2.1.1 Examples of PGMs

Figure 2.1 is a trivial, *fully connected* graphical model over the variables x, y and z, each with an associated node and arrows representing conditioning.

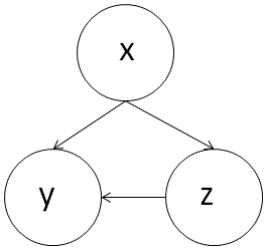

Figure 2.1: Fully connected PGM

This represents the joint distribution

$$p(x, y, z) = p(y|x, z)p(z|x)p(x). \tag{2.3}$$

By contrast, Figure 2.2 represents the joint distribution over the variables x, v, w, y and z

$$p(x, v, w, y, z) = p(x)p(v)p(y|x, v)p(w|x)p(z|w), \tag{2.4}$$

this can be derived by inspection of the graph. For example, in this case y depends on x and v. We can use these principles to form more meaningful PGMs.

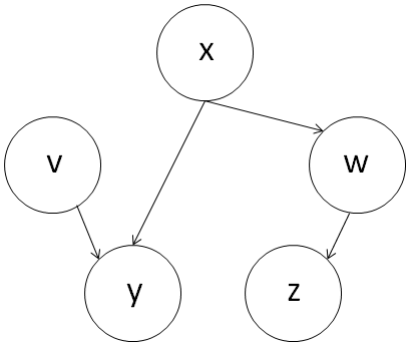

Figure 2.2: An example of a 5-node probabilistic, directed graphical model.

We will use directed PGMs to describe some of the models and techniques in this report.

2.1.2 A Graphical Representation of de Finetti's Theorem

In the initial mathematics report, de Finetti's theorem was introduced. The theorem states that if the distribution of some RVs is *exchangeable*, then there exists another random object upon which each RV depends. Figure 2.3 shows a graphical representation of this situation where $x_1, x_2, ..., x_N$ are N RVs all depending upon the same variable θ. Shaded nodes indicate a variable which is conditional on another variable. Using a *plate* we can simplify the repeated aspects of this graphical representation by putting all replicated variables and links into a single box; use of a *plate* for de Finetti's directed graph would take the form of Figure 2.4 [11].

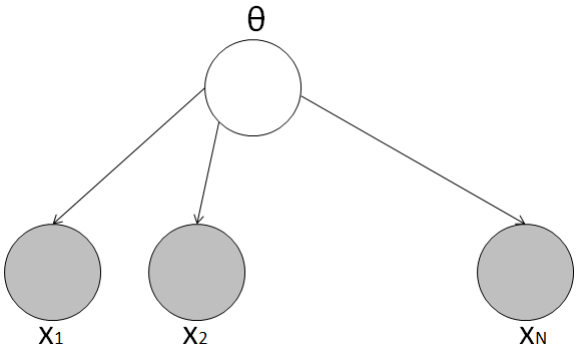

Figure 2.3: de Finetti's theorem represented as a PGM.

As explained, this graph can be abbreviated in the following way:

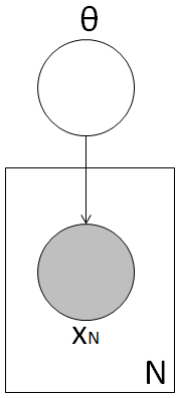

Figure 2.4: de Finetti's theorem represented with an abbreviated PGM using *plates* (the box enclosing the x_N).

2.2 Markov Chains

In essence, a *Markov chain* is simply a sequence of discrete RVs which are serially dependent. It generates a *discrete time stochastic process* which has the *Markov property*, which is as follows. Given a stochastic process in discrete time $(x_n; n = 0, 1, ...)$, where x_n is a sequence of RVs, the distribution of each x_n is solely determined by the value of the preceding variable x_{n-1}. So, given x_{n-1} the distribution of x_n is independent of the 'history' of the process i.e. $(x_{n-2}, x_{n-3}, ...)$. In other words, the process has no 'memory'. We can define this using conditional probabilities:

$$p(x_n|x_0, x_1, ..., x_{n-1}) = p(x_n|x_{n-1}). \tag{2.5}$$

For example, *random walks*, defined as $x_n = x_{n-1} + y_n$ where y_n is another random variable, has the Markov property. If the x_n's are discrete RVs over the sample space Ω and let $x_n = i$ denote that the process is in state i at time n, then we can define the conditional probability of this process:

$$p(x_n = j|x_{n-1} = i) \equiv p_{ij}^n, \tag{2.6}$$

where p_{ij}^n is the probability that the process with state $i \in \Omega$ will make a transition to state $j \in \Omega$ in the step from $(n-1)$ to (n). Of course we must have that $p_{ij}^n \geq 0$ and $\sum p_{ij}^n = 1$ for $i = 0, 1, ...$ If these *transition probabilities* p_{ij}^n are the same for all n, then the Markov chain is *homogeneous* and we can

drop the superscript n. For the distribution of x_n to converge to a *stationary distribution*, then the chain must be *irreducible*, *aperiodic* and *positive recurrent*, roughly speaking, it must have no states which are unreachable from any other, it must not fall into finite, cyclic sequences of states, and all states must eventually be visited [8].

2.3 Markov Chain Monte Carlo (MCMC)

MCMC [9] is a sampling method for generating samples from a set of joint RVs. MCMC methods have an extremely wide range of applications in *Bayesian inference* and statistical modeling, some of which we will use and discuss in this project. Such techniques will allow us to perform inference on mixture models. The essential idea of MCMC is to construct a *transition distribution* q which leads to a Markov chain whose stationary distribution is the joint distribution over the joint RVs x in the problem that we wish to model. This will be used to generate a sequence of samples whose distribution converges on the desired distribution, using the following scheme:

$$x_{N+1} \sim q(x_{N+1}|x_N). \tag{2.7}$$

Figure 2.5 depicts this method graphically.

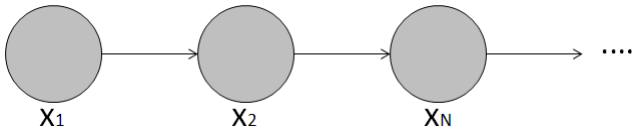

Figure 2.5: MCMC graphical model

2.4 Monte Carlo Integration

Given MCMC samples from the joint distribution over the RVs x, we often want to summarise this distribution using various moments. We can compute the expectation g:

$$\int g(x)\mathrm{d}P(x), \tag{2.8}$$

where P is the stationary distribution of the MCMC chain. We will form estimates of these expectations by averaging over MCMC samples [8]. *Monte Carlo integration* approximates the value

$$E[g(x)] \approx \frac{1}{N}\sum_{n=1}^{N} g(x_n), \tag{2.9}$$

which is a sample expectation. Here $x_n, n = 1, ..., N$ are the samples drawn from the MCMC process described above. In other words, we take samples from the converged MCMC sampler and form empirical moment estimates.

2.5 Gibbs Sampling

Gibbs sampling methods are a class of convenient MCMC algorithms used in computing with DPs [10]. This is an inference method to sample from the required joint distribution from the RVs x. In general, this iterative algorithm is as follows

1. Initialize the variables $x_1, x_2, ..., x_N$,

2. For $r = 1, ..., R$:
   Sample $x_1^{(r+1)} \sim p(x_1|x_2^{(r)}, x_3^{(r)}, ..., x_N^{(r)})$.
   Sample $x_2^{(r+1)} \sim p(x_2|x_1^{(r+1)}, x_3^{(r)}, ..., x_N^{(r)})$.
   .
   .
   .
   Sample $x_N^{(r+1)} \sim p(x_N|x_1^{(r+1)}, x_2^{(r+1)}, ..., x_{N-1}^{(r+1)})$.

where r is the iteration number and R is the maximum number of iterations. Another way of describing this is that on each iteration, we replace the variable x_i by a sample from $p(x_i|\mathrm{x}_{/i})$, where $\mathrm{x}_{/i}$ is the sequence $x_1, x_2, ..., x_N$ with x_i omitted. Each sample is conditional only the previous set of variables, demonstrating the *Markov* property. It can be shown that the resulting Markov chain has a stationary distribution which is the desired joint distribution over the RVs x [3]. There are numerous variations of this method as described next.

2.6 Gibbs Sampling in PGMs

We can apply the Gibbs sampling algorithm in the context of PGMs. For this iterative method we need to know the conditionals $p(x|.....)$, where we sample each of the variables conditioned on all the other variables, in turn. So in the case of Figure 2.2 the following sequence of samples would be obtained on each iteration:

1. $p(x|v, w, y, z)$
2. $p(w|x, v, y, z)$
3. $p(z|x, v, w, y)$
4. $p(v|x, w, y, z)$
5. $p(y|x, v, w, z)$

We update all the variables with their latest sample. However, in many cases we will already know the value of some of the variables (these would come from obtained data, for example) so in this case we do not re-sample these variables, and simply fix them at the observed value. The rest of the (unobserved) RVs are sampled conditional on these fixed values.

Chapter 3

Mixture Models

3.1 Finite Mixtures

In a parametric context, a *finite mixture model* (FMM) with independent and identically distributed (i.i.d.) observations has probability density function:

$$p(x;\theta,\pi) = \sum_{k=1}^{K} \pi_k f(x;\theta_k), \tag{3.1}$$

where θ_k are the component-specific parameters and π_k are known as *mixture coefficients* which satisfy $\sum_{k=1}^{K} \pi_k = 1$, and $0 \leq \pi_k \leq 1$. These coefficients can be interpreted as prior probabilities that an observation is from a particular mixture component [13]. We will denote a data set as $X_1, ..., X_N$ and any one observation, X_i, from this set, has the distribution above.

FMMs have many *clustering* applications, where the data is from a mixture of *probability distributions* which we can interpret as different clusters. We wish to partition the data in to a fixed number, K, of clusters each with their own distribution. We can think of the weights π_k as prior probabilities of residing to a particular cluster, and we can write the probability that a data point will be assigned to a particular cluster as $p(Z_i = k) = \pi_k$, where the *latent indicator variable* $Z_i = k$ if observation X_i is assigned to cluster k. Once we have determined an FMM, the *posterior* probability of a data point belonging to a particular cluster is:

$$p(Z_i = k | X_i = x) = \frac{\pi_k f(x;\theta_k)}{\sum_{j=1}^{K} \pi_j f(x;\theta_j)} \tag{3.2}$$

Given some class assignments Z_i drawn from the *categorical distribution Cat* (π), a fully Bayesian, conjugate FMM can be written as:

$$\begin{aligned} \pi &\sim \text{Dir}\,(\pi; c) \\ Z_i &\sim \text{Cat}\,(Z_i|\pi) \\ \theta_k &\sim H \\ X_i &\sim f(X_i|\theta_k) \end{aligned} \tag{3.3}$$

where $\text{Dir}(\pi; c)$ is a *Dirichlet distribution* with concentration parameter vector c on the sample space $\pi = (\pi_1, ..., \pi_K)$. H is a prior over the component parameters (usually chosen as a conjugate to f). As $K \to \infty$ this FMM converges to a *Dirichlet Process Mixture Model* (DPMM), as we describe next.

3.2 Infinite Mixtures

An *infinite mixture* is simply an FMM where the number of clusters $K \to \infty$. In the finite case we predetermine the number of clusters, K, each with an associated parameter θ. For infinite mixtures, we will be unsure about the 'true' number of clusters but we will learn this from the data by using an inference algorithm. We explore specific infinite mixture models below.

3.3 Gaussian Mixture Model (GMM)

The univariate Gaussian (or normal) distribution is PDF

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right), \tag{3.4}$$

where μ and σ^2 are the mean and variance respectively. For a Gaussian mixture model (GMM), the mixture distribution is

$$p(X = x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2). \tag{3.5}$$

We can express the conditional distribution in terms of the latent indicator RVs, $Z_1, ..., Z_N$, where $Z_i = k$ means that the i^{th} observation is assigned to the k^{th} cluster [3]:

$$p(X = x | Z_i = k) = \mathcal{N}(x | \mu_k, \sigma_k^2). \tag{3.6}$$

Similarly, the joint probability over all the data $\{X_1, \ldots, X_N\}$ is

$$p(X_1 = x_1, ..., X_N = x_N) = \prod_{i=1}^{N} \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i | \mu_k, \sigma_k^2) \right). \tag{3.7}$$

We will make extensive use of these probabilities within the Gibbs sampler for making inferences.

3.3.1 Graphical Model

A widely used representation of a FMM is indeed a *mixture of Gaussians* represented graphically in Figure 3.1:

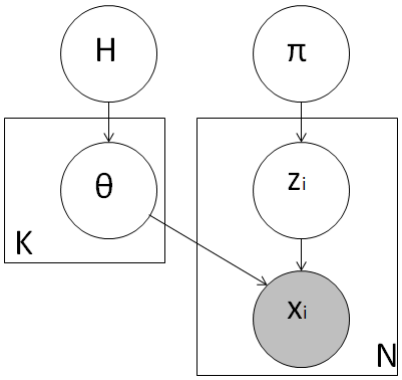

Figure 3.1: Finite mixture model (FMM) graphical model.

3.3.2 Gibbs Sampler for GMM

Equations 3.8 and 3.9 below are the two conditional probabilities we require to apply the Gibbs sampler with K components, mean parameters $\mu_{1,...,K}$ and variances $\sigma^2_{1,...,K}$ to given data. For $i = 1, ..., N$, the latent indicator variables have distribution

$$p(Z_i = k|\mu) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \sigma_k^2)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \sigma_j^2)}), \tag{3.8}$$

and for $k = 1, ..., K$, the component parameters have PDF

$$p(\mu_k|x) = \mathcal{N}\left(\mu_k; \frac{\sigma_k^2 \mu_0 + \sigma_0^2 \sum_{i:z_i=k} x_i}{\sigma_k^2 + \sigma_0^2 N_k}, \frac{\sigma_k^2 \sigma_0^2}{\sigma_k^2 + \sigma_0^2 N_k}\right), \tag{3.9}$$

where σ_0^2 is the prior variance of μ, and where $N_k = |\{i : z_i = k\}|$, that is, the number of observed data points assigned to component k. We now can perform Gibbs sampling, using the following iteration:

1. Sample the indicators z from $p(z|\mu)$,
2. Sample the component means μ from $p(\mu|z)$.

We repeat these two steps until convergence on the stationary distribution $p(x, z, \mu)$.

3.4 Dirichlet Process Mixture Model (DPMM)

The DPMM is a clustering model which is an infinite, Bayesian mixture model [12], hence this mixture model has a countably infinite number of components. When more data is observed, the number of components can grow with this increase in data. Given some exchangeable data $\{X_1, ..., X_N\}$, the model is defined by [19]:

$$\begin{aligned} G &\sim \text{DP}(cH) \\ \theta_i &\sim G \\ X_i &\sim f(X_i|\theta_i), \end{aligned} \tag{3.10}$$

where X_i is the i-th observation, and θ_i is the i.i.d. parameter drawn from the DP G, which is discrete [6]. The parameters θ_i, the observations X_i and G are all independent of each other. Draws from the DP are discrete PDFs over the sample space Ω with (scalar) concentration parameter c and base measure H over Ω [14]. However, since G is discrete, there is a non-zero probability of repeated samples θ_i sharing a value, as described in the mathematics report. Hence some of the X_i's share identical component parameters θ_i, and we will find that there are $K \leq N$ representative component parameter values. This is the origin of the clustering effect in DPMMs. In this hierarchical Bayesian model, the DP is the *nonparametric* prior [1] for this mixture model with infinitely many components. Below, we will demonstrate one way to use Gibbs sampling to perform inferences for the DPMM.

3.4.1 Graphical Model

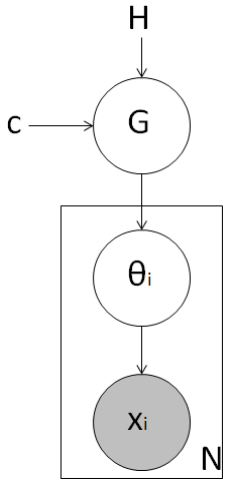

Figure 3.2: Graphical model for the Dirichlet process mixture model (DPMM).

3.5 Chinese Restaurant Process Mixture (CRPM)

In the associated report, we discussed the *Chinese restaurant process* (CRP). We revisit the concept here. In order to sample from the DPMM, we will need to sample a *partition* of the set of samples $\{1, \ldots, N\}$ from this process. As a reminder, given a list of samples with the K representatives $\theta_1, \ldots, \theta_K$ and a set of indicator variables $Z_1, \ldots, Z_n$, the probability of sampling from an existing representative value for θ_n is:

$$p\left(Z_{n+1} = k\right) = \frac{n_k}{c+n}, \tag{3.11}$$

for $k = 1, \ldots, K$, where n_k is the number of representatives for which $Z_i = k$, for $1 \leq i \leq n$. The probability of sampling a new representative is

$$p\left(Z_{n+1} = K+1\right) = \frac{c}{c+n}, \tag{3.12}$$

where c is a concentration parameter. This allows us to generate a sequence of indicators, which amounts to sequentially sampling a partition over $\{1, \ldots, N\}$ using the CRP. This sampling procedure is crucial to using the Gibbs sampler for DPMM inference.

3.6 Gibbs Sampler for the DPGMM

Given the *posterior predictive distribution* of X_i:

$$p(X_i = x_i | \mu_0, \sigma_0^2) = \mathcal{N}(x_i; \mu_0, \sigma^2 + \sigma_0^2), \tag{3.13}$$

we can derive an infinite Gaussian mixture model sampler. As before we require two conditional probabilities:

$$p(Z_i = k | \mu) = \frac{N_{k,-n}\mathcal{N}(x_i; \mu_k, \sigma^2)}{c\mathcal{N}(x_i; \mu_0, \sigma^2 + \sigma_0^2) + \sum_{j=1}^{K} N_{j,-n}\mathcal{N}(x_i; \mu_j, \sigma^2)} \tag{3.14}$$

for $k \in 1, ..., K$ and

$$p(Z_i = K + 1|\mu) = \frac{c\mathcal{N}(x_i; \mu_0, \sigma^2 + \sigma_0^2)}{c\mathcal{N}(x_i; \mu_0, \sigma^2 + \sigma_0^2) + \sum_{j=1}^{K} N_{j,-n}\mathcal{N}(x_i; \mu_j, \sigma^2)} \tag{3.15}$$

This gives us all the ingredients with which to apply the DPGMM Gibbs sampler:

1. Sample the indicators z from $p(z|\mu)$. We will sample new component means μ from the posterior if new components need to be generated,
2. Sample the component means μ from $p(\mu|z)$.

Equations 3.14 and 3.15 exploit the CRP distribution. If sampling z_i a new indicator is generated, we sample a new μ_{K+1} using the PDF

$$p\left(\mu_{K+1}\right) = \mathcal{N}\left(\mu_{K+1}; \frac{\sigma^2\mu_0 + \sigma_0^2 x_i}{\sigma^2 + \sigma_0^2}, \frac{\sigma^2\sigma_0^2}{\sigma^2 + \sigma_0^2}\right). \tag{3.16}$$

We then update K to $K + 1$ and continue to sample z_{i+1}.

Chapter 4

Experimental Results

All of the algorithms in this chapter are implemented using Matrix Laboratory (MatLab).

4.1 Synthetic Experiments

4.1.1 Generating Synthetic 1D Data

Synthetic data will be required in order to demonstrate that the algorithm is working correctly before applying to real data. Here we require an iterative function which samples a partition from a CRP to generate cluster indicators. The original code can be found at [5]. We input a concentration parameter, c, and the number of objects, N, we want in the partition. The output of this code is a set of N indicators z and a number of clusters K, which we can then use to sample associated component parameters μ and observations x, or x_{syn} in this case. Figure 4.1 shows a plot of the generated observations.

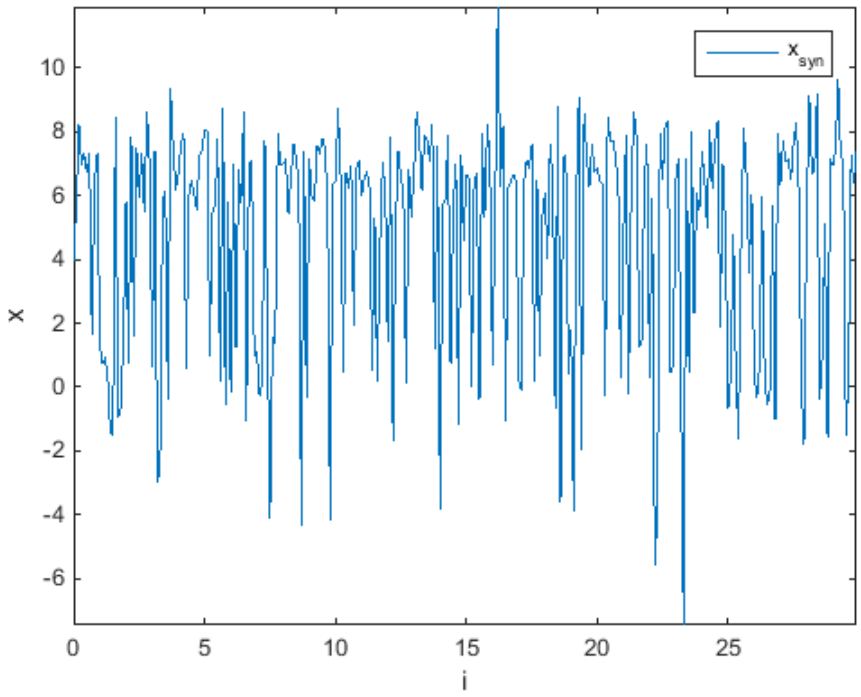

Figure 4.1: Synthetic DPGMM cluster data for $c = 0.8$, $N = 300$.

4.1.2 DPGMM Gibbs Sampler Results

It is useful to plot other variables within the DPGMM Gibbs sampler to better describe this algorithm, specifically K and μ, which are used to store the Gibbs samples of μ per iteration. Figure 4.2 shows how the number of clusters, K increases and converges with more iterations, also, the vector μ will increase in size as K increases. In this case, the sampler converged to $K = 6$. Each time we run the algorithm we will get a different value for K as it is a RV, which is a characteristic of the DPMM. However, with a large number iterations K will eventually stabilize as the Markov chain converges. Figure 4.3 shows the component means μ against iteration number, which shows the effect of increasing K against iteration number r. As K becomes large, the plots of μ become unclear as there will be too many lines to see the individual component parameters. Finally, Figure 4.4 is the result of the Gibbs sampler with $c = 0.8$, $N = 300$, $R = 3000$, $\mu_0 = 2$ and $\sigma_0 = 5$. This plot shows the Monte Carlo estimated values of the component means $\bar{\mu}_k = \frac{1}{R}\sum_{r=1}^{R} \mu_r$ generated using the Gibbs sampler, indexed by the mode value $\bar{z}_i$ over all Gibbs iterations, $r = 1, \ldots, R$. These are most probable (*maximum a posteriori*, MAP) summaries of the posterior values of the observed data, produced by the DPGMM Gibbs sampler.

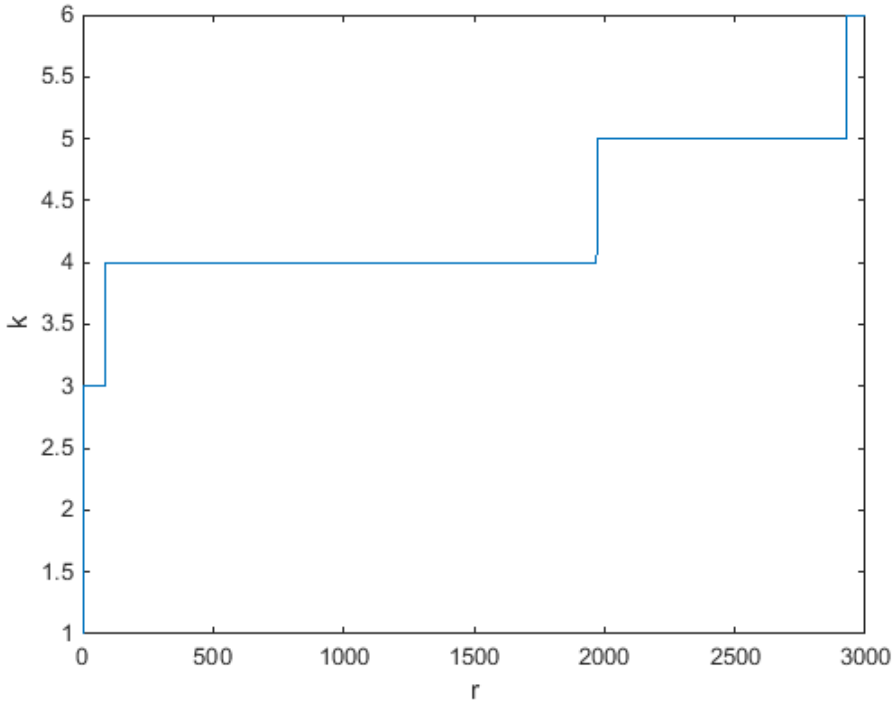

Figure 4.2: Plot of k against iteration r of the DPGMM Gibbs sampler, applied to synthetic data.

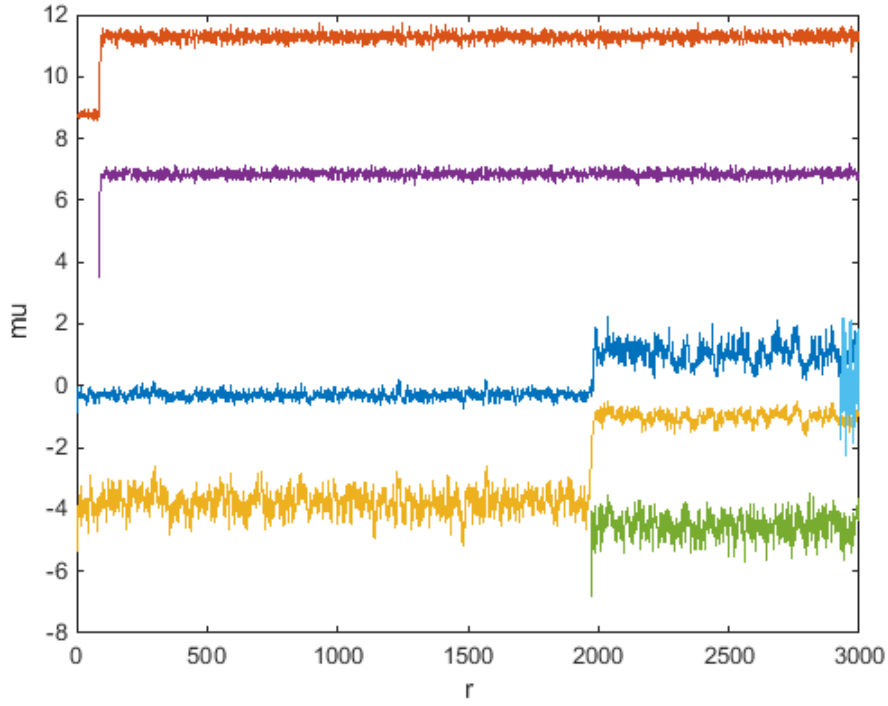

Figure 4.3: Plot of μ against iteration number r of the DPGMM Gibbs sampler, applied to synthetic data x_{syn}.

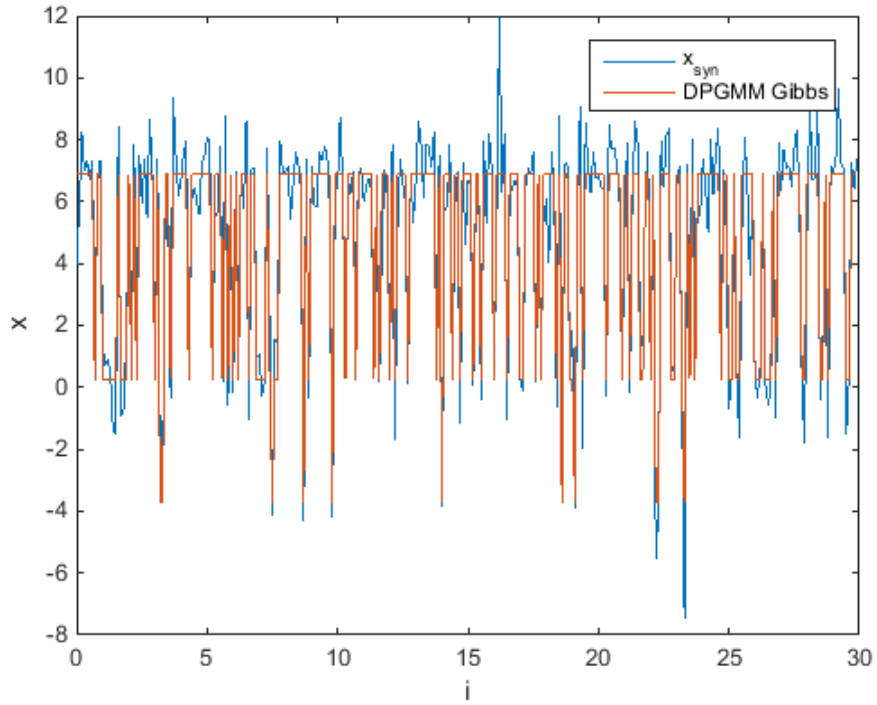

Figure 4.4: Plot of the MAP estimates of the DPGMM sampler, of the observed data against observation index i, estimated from all iterations of the DPGMM Gibbs sampler, applied to synthetic data.

This last figure shows that the Gibbs sampler is able to recover a simplified approximation to the synthetic data, as expected. The next step is to apply the Gibbs sampler on some real data. It is worth noting that the the Dirichlet prior concentration parameter, c, was not tuned here but this variable will need to be tuned for experimental tests in the next section.

4.2 Behavioural Time Series Segmentation

4.2.1 Experimental Accelerometer Data

Some labelled test data was collected by Reham Badawy, a PhD Researcher at Aston University, from accelerometer sensors in a smartphone. In all there are 20 tests each with a different experimental protocol and purpose. Each data set is of size $N = 300$ observations. Observations consist of time, t (seconds) for each input data point which is uniformly increasing in increments of 0.1s, with a total time of 30s. Each sensor observation contains x, y and z axes values. The tests simulate certain activities that might enable us to monitor typical PD symptoms, such as gait impairment, balance problems and rigidity. The activities include the user walking (Test 5), dropping/picking up their phone (Test 12) and falling (Test 15). It is these three tests that will be examined here in terms of the effect of applying the DPGMM. The purpose of each test is as follows: Test 5 identifies the accelerometry pattern of walking while the phone is in the user's pocket. Test 12 identifies the accelerometry pattern of the phone being dropped, followed by the phone being picked up, and the phone being dropped and being picked up quickly. Finally, Test 15 identifies the accelerometry pattern for when the phone is in the user's cardigan pocket when falling, getting up before falling and getting up straight away. As an example, Figure 4.6 shows the sensor data obtained during Test 12.

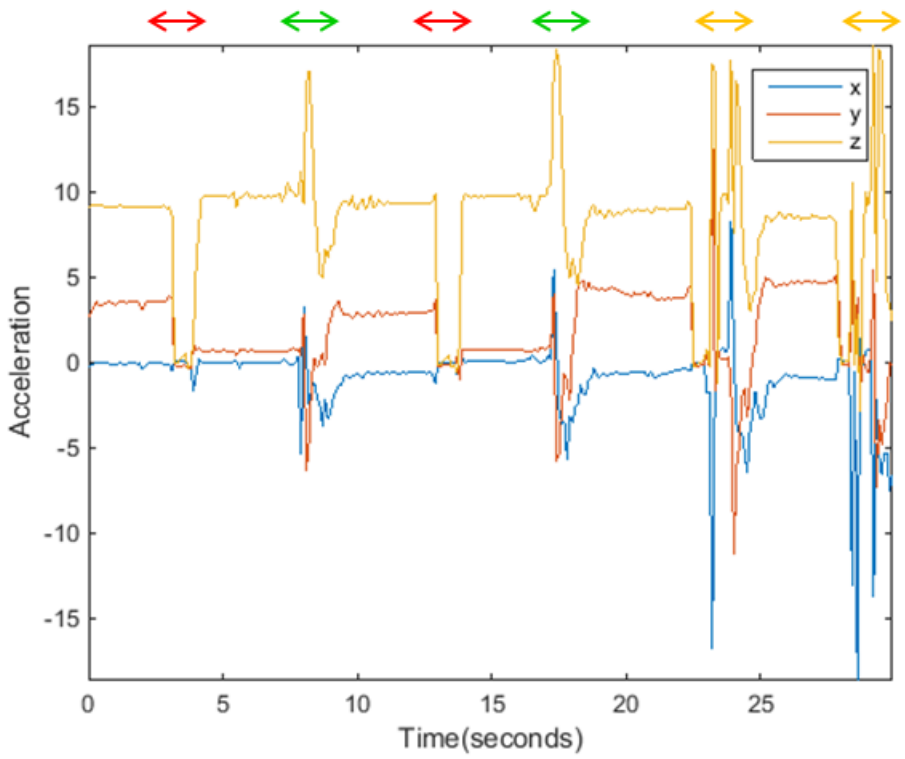

Figure 4.5: Plot of experimental accelerometer smartphone data for Test 12.

The red arrows at the top of the figure represent a drop, the green arrows represent a pick-up and

the orange arrows represent a drop/pick-up instantaneously. Relatively flat acceleration occurs when the phone is stationary.

We will now apply the DPGMM Gibbs sampler to these experimental data. The concentration parameter c, will be varied to provide a meaningful behavioural segmentation. Essentially, this variable tunes the number of clusters for each kind of test. Here, we will plot time, t, for each input data point against summaries of the posterior values of the observed data. However, because the Gibbs iterations only converge for large iteration numbers close to R, we will use summaries obtained for the last 10% of iterations. Since the number of iterations of the Gibbs sampler is $R = 3000$, the last 10% corresponds to the 300 time increments (i.e. 0 to 30 seconds in increments of 0.1s) we have for the data. For each time increment we can select the MAP value over these last 10% of iterations. We use the value of K at convergence. The original data is also included in the plot to assess how the MAP Gibbs sampler results compare to the original data and if the clustering is a meaningful representation of this experimental data. The primary objective here is to achieve a 'smoother' result than the experimental data, while retaining the abrupt changes in behaviour.

4.2.2 DPGMM for Test 5: Walking

For this test $c = 0.001$ was used. Figure 4.6 shows the MAP Gibbs result for the x-axis accelerometer data. It can be seen that there is less fluctuation in the DPGMM Gibbs output compared to the original data, confirming that the algorithm is indeed behaving as it should by 'smoothing' the data in between abrupt changes. In this case, the final $K = 7$. Figure 4.7 shows the same result the y-axis data, where $K = 6$. For the z-axis data (Figure 4.8), $K = 5$. All three graphs are quite similar.

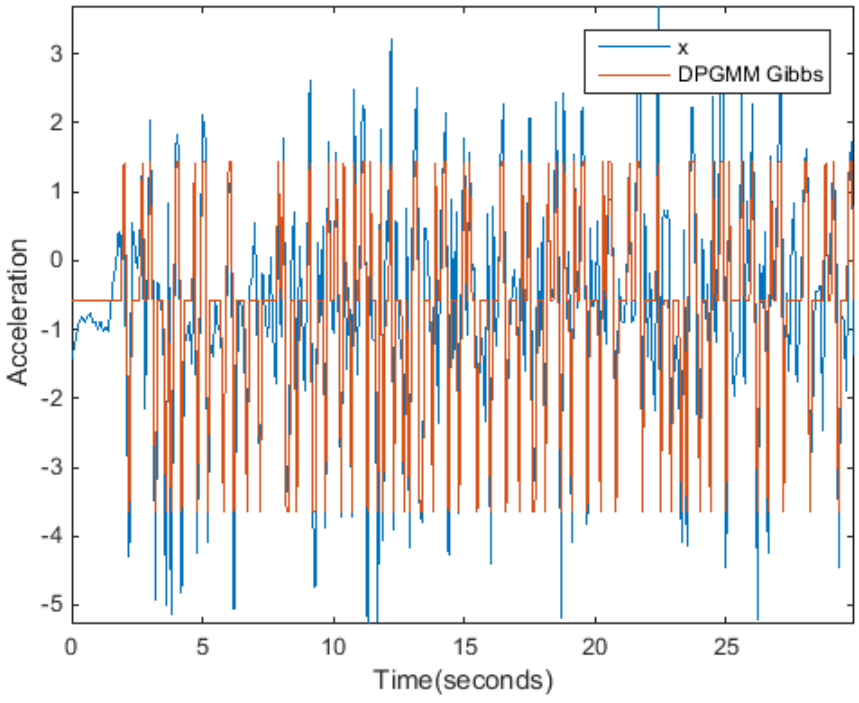

Figure 4.6: Time t against the MAP DPGMM Gibbs results, x-axis accelerometry data, for experimental Test 5.

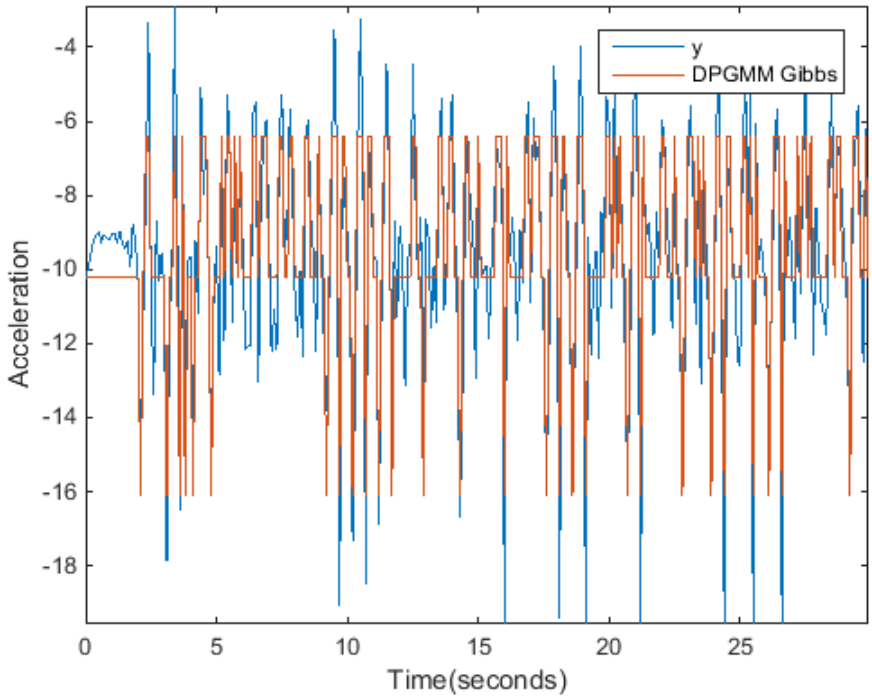

Figure 4.7: Time t against the MAP DPGMM Gibbs results, y-axis accelerometry data, for experimental Test 5.

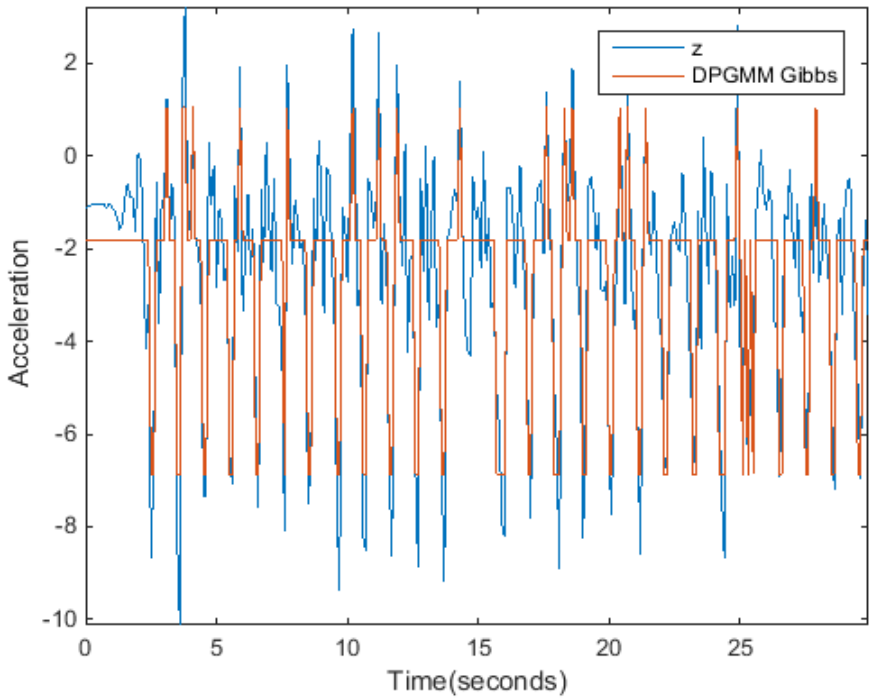

Figure 4.8: Time t against the MAP DPGMM Gibbs results, z-axis accelerometry data, for experimental Test 5.

We can see that the DPGMM Gibbs algorithm has 'standardised' the heights of the 'spikes' in the plots for all three axes, and small fluctuations in the observed data have been replaced with flat lines. Thus, it provides a useful segmentation of the data into periods of consistent and changing behaviour.

4.2.3 DPGMM for Test 12: Dropping Phone

The first two 'spikes' in this data occur when the phone was dropped and picked up again before repeating but in quick succession, represented by spikes which are closer together. Here, $c = 0.0001$ was used. Figure 4.9 shows the final MAP Gibbs sampler result for the x-axis data, where $K = 3$. In this case, it is only the very large spikes which the algorithm reduces, smaller spikes have not been reduced although flat lines have replaced small fluctuations. Figure 4.10 shows the result for the y-axis data, where $K = 10$. Again, taller spikes have been reduced by the algorithm to leave a smoother result. Figure 4.11 shows the final result for the z-axis data, where $K = 5$. This has taller spikes than the other two axes, due to the nature of this test.

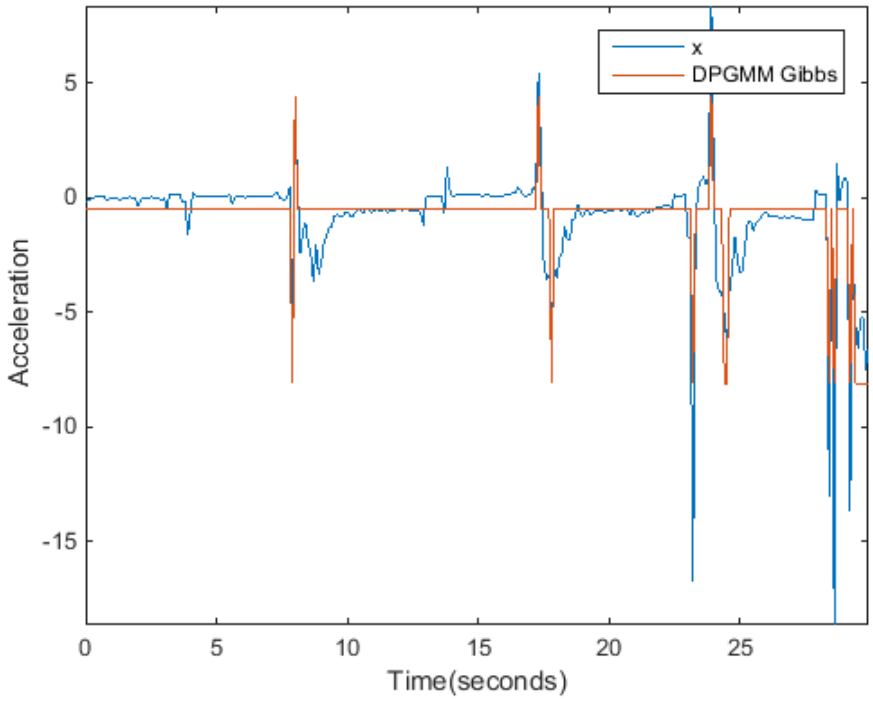

Figure 4.9: Time t against the MAP DPGMM Gibbs results, x-axis accelerometry data, for experimental Test 12.

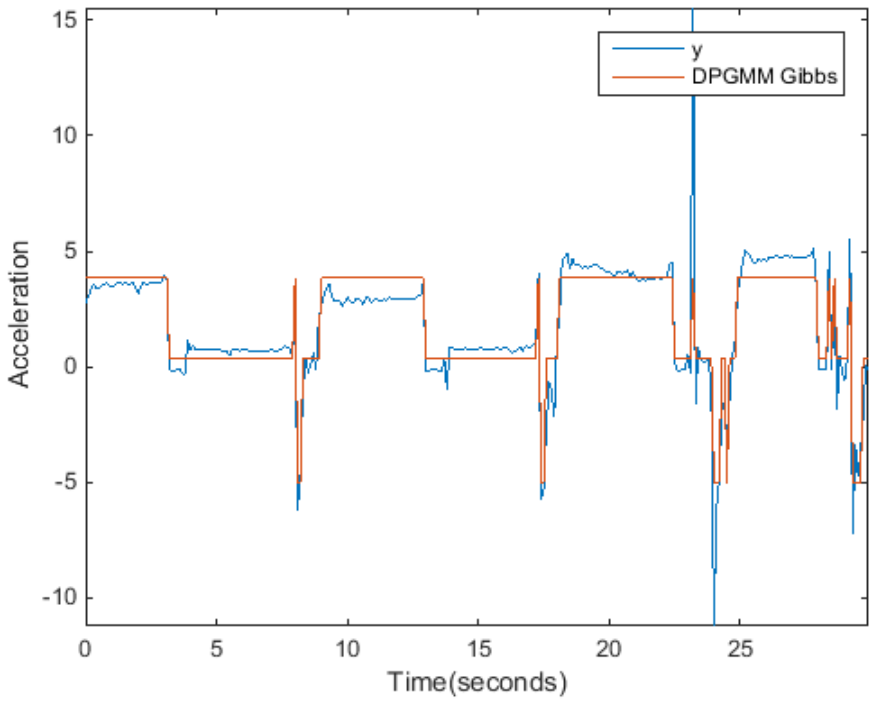

Figure 4.10: Time t against the MAP DPGMM Gibbs results, y-axis accelerometry data, for experimental Test 12.

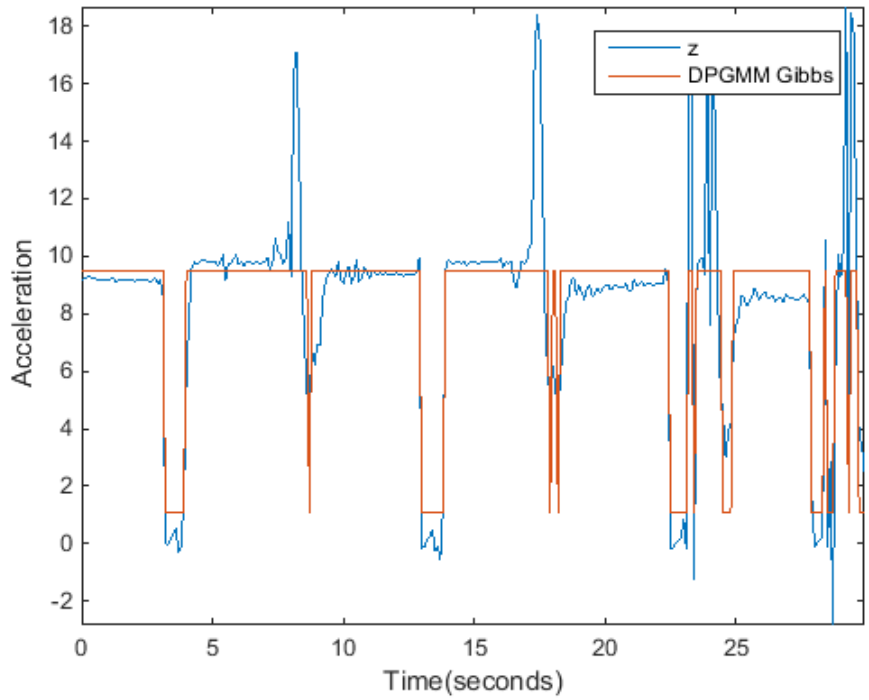

Figure 4.11: Time t against the MAP DPGMM Gibbs results, z-axis accelerometry data, for experimental Test 12.

From the final results it can be clearly seen that the small fluctuations are replaced, by the *Gibbs* algorithm, with flat lines, reducing the noise, and that the abrupt jumps are still retained.

4.2.4 DPGMM for Test 15: Falling

For this test $c = 0.001$ was used and this gave visibly good results. Figure 4.12 shows the final MAP result for the x-axis data, where $K = 8$. For x-values between approximately 5 and -9, the MAP results corresponds very closely to the original data. Some of the spikes are 'smoothed' out by the algorithm. Figure 4.13 shows the same result for the y-axis data, where $K = 5$. This is similar to the corresponding x-axis. Figure 4.14 shows the result for the z-axis data, where $K = 7$. This is a good demonstration of the smoothing effect algorithm for small fluctuations.

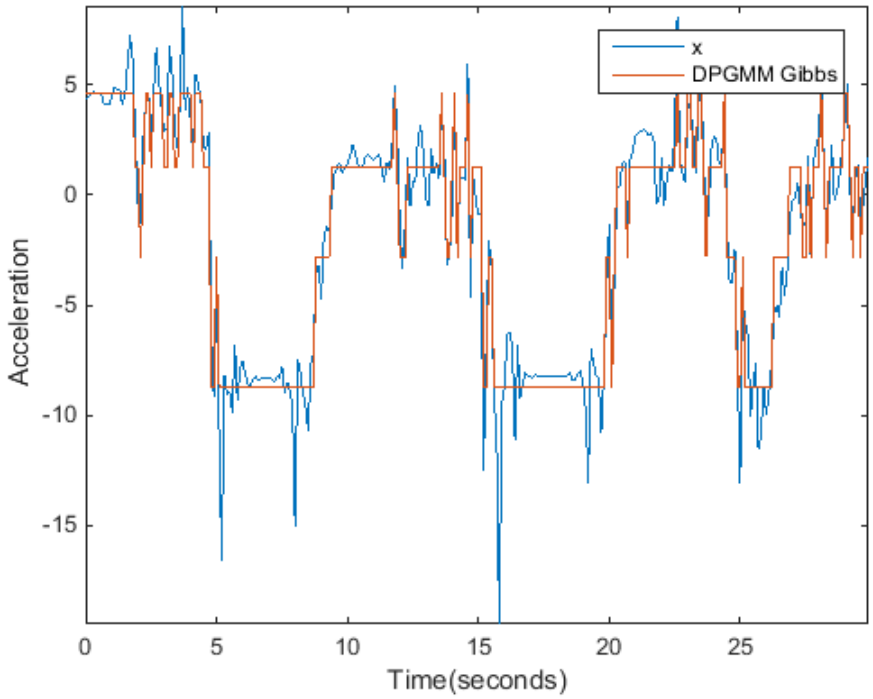

Figure 4.12: Time t against the MAP DPGMM Gibbs results, x-axis accelerometry data, for experimental Test 15.

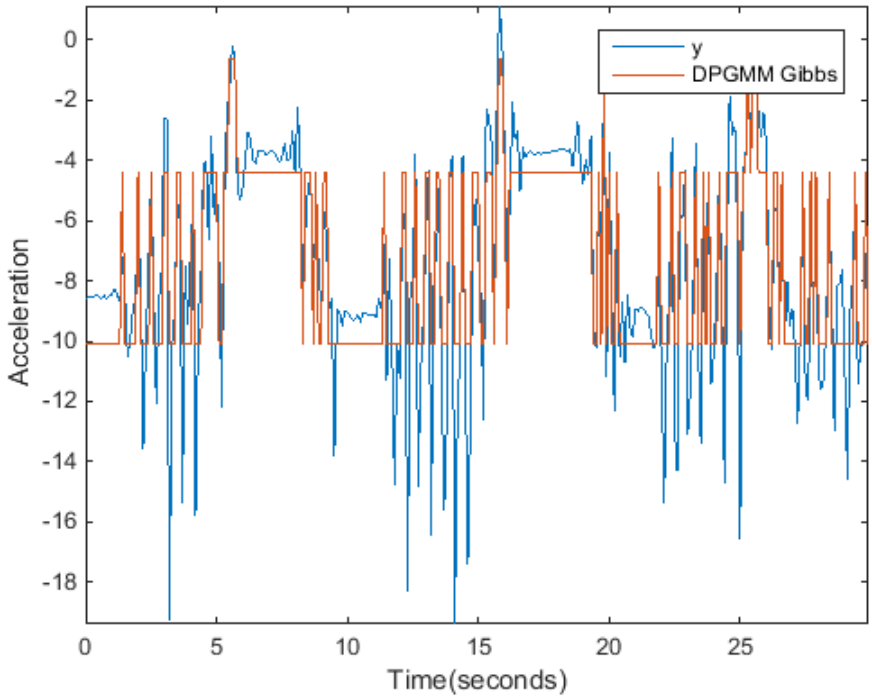

Figure 4.13: Time t against the MAP DPGMM Gibbs results, y-axis accelerometry data, for experimental Test 15.

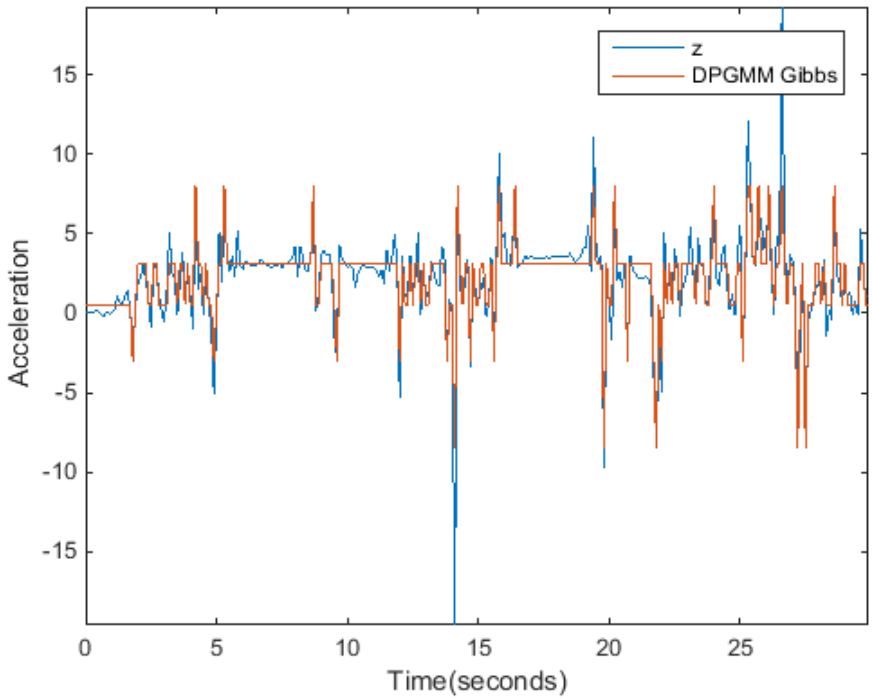

Figure 4.14: Time t against the MAP DPGMM Gibbs results, z-axis accelerometry data, for experimental Test 15.

Chapter 5

Ending

5.1 Summary

The aim of this project was to expand upon ideas from the associated report and put them into practice, specifically the DPMM. The Gibbs sampling inference method was introduced and an exposition of this and relevant concepts was provided. Firstly, some preliminaries were required which included PGMs and Markov chains before Gibbs sampling could then be discussed from a foundational level. Next, using the knowledge of stochastic processes, from the report, mixture models were discussed in detail. This included the CRP sampler based on the CRP distribution discussed in the report. After explaining the GMM and DPMM, the Gibbs sampler for inference in the DPGMM was introduced. This inference algorithm was tested on some synthetic data generated using the CRP GMM. Running the DPGMM Gibbs sampler showed good results giving confidence that the algorithm could be applied to some experimental data. These synthetic data tests suggested that the concentration parameter, c, may need to be tuned for each experimental data set. Experimental data from various behavioural tests was described and plotted. DPGMM Gibbs inference algorithm was then applied to the experimental test data for three different experimental tests, for each of the three accelerometer axes x, y and z. The algorithm was able to smooth out noise between abrupt jumps yet retain the jumps fairly well. The maximum number of clusters, K, was relatively consistent across axes for each test.

5.2 Conclusions

Overall, the DPGMM Gibbs sampler algorithm produced much smoother approximations to the input data and segmented the experimental data to a certain extent. Hence the algorithm worked as expected. However there were certain limitations to this work. This algorithm was slow to converge for the number of iterations required to get a meaningful result, and iteration required high computational cost. This was demonstrated by timing the results using Matlab, which showed approximately 0.1-0.2 seconds per iteration, which, for a total of $R = 3000$ iterations to reach convergence required 150-300 seconds per experimental data set. One difficulty with the Gibbs sampler is knowing when to stop the iterations: i.e. to know when the algorithm has converged on the stationary distribution. Techniques exist to test for convergence but there are no definitive answers to this problem [7]. Further experiments could assess convergence after each iteration and store the history, which poses another difficulty for this algorithm. The prior Dirichlet concentration parameter, c, can be used for tuning the model. A different c was used for Test 12, as for the y data K was very high ($K = 100$) compared to the other axes ($K = 6$). Reducing c for this test gave a more reliable result and more experiments could have been performed in regards to this, i.e. more extensive exploration of the tuning. This may have given more accurate results. Furthermore, the hyperparameters μ_0 and σ_0 were simply chosen by hand, in classical Bayesian fashion,

but certain inference methods could have been used to obtain these, such as *empirical Bayes* [15]. An alternative to the rather simple Gibbs sampler for inference is the *Metropolis Hastings* algorithm [2], which will also provide a full joint posterior distribution of the model. Also, since in practice, in this behavioural segmentation problem, we only really care about the MAP result, the recently developed *maximum-a-posterior Dirichlet process mixture* (MAP-DPM) algorithm, which is a fast method for fitting a DPGMM could be used, however it only results in a point, modal estimate of the joint likelihood of the model. Numerous other approximation methods, including *variational methods* [4] exist which could also be used for inference.

5.3 Scope for Future Work

Analysis of a selection of experimental tests was performed, this selection representing a variety of activities (walking, falling and dropping phone) with the phone in different places (hand and pockets), on all of which the algorithm was able to obtain useful results. However, this project only concerned behavioral changes, but changes in *orientation* of the smartphone could have also been taken into consideration. An example of an orientation test would be for the user to rotate and tilt the phone in their hand, while walking at the same time. It would be interesting to see how the algorithm would perform on such orientation changing experiments, although doubtless, changes to the algorithm may have to be made to accommodate for these. Using hierarchical clustering, for example, by using a *hierarchical DP* [16], it may be possible to use a DPGMM to establish subgroups (clusters) of PD patients classified according to particular symptoms. This would, however, require significant changes to this particular Gibbs sampler. Similar work in this field has consisted of using a DPMM to identify disease subgroups based on patient symptoms [18]. Other research has used finite mixture modeling to identify disease phenotypes using a Bayesian approach [17]. Such work could lead to more extensive contributions in this area.

5.4 Acknowledgements

I would like to thank both Reham Badawy and Yordan Raykov, PhD Researchers at Aston University, for their advice.

Appendix A

MatLab Code for the DPGMM Gibbs Sampler

The *'s in the following code indicate what has to be manually changed depending on what test and axis is being used.

```
clear all;
randn('seed',1);

R = 3000; % Number of Gibbs iterations
tenit = R/10; % To be used later for the last 10% of the Gibbs iterations

% When using synthetic data, add the next 5 lines:
%-------------------------------------------------------------------------------------------
mu0 = 2;
sigma0 = 5;
[Y, mugen] = synthetic(mu0, sigma0); % Function discussed and referenced in section 4.1.1
Y=Y';
Y=Y(1,:);
%-------------------------------------------------------------------------------------------

N = 300; % (Fixed) number of rows in all of the data sets
load('interp_data.mat'); % Data collected by Reham Badawy
 expTest = interp_data_test5_1; % Manually select which experimental test to use *
% expTest = interp_data_test12;
% expTest = interp_data_test15;

% When using the experimental test data, add the next 2 lines:
%-------------------------------------------------------------------------------------------
Y = expTest(:,2); % Manually choose column index 2, 3 or 4 for x, y or z axis respectively *
Y = Y';
%-------------------------------------------------------------------------------------------

% Associated time indexes for each data point
t = (expTest(:,1) - expTest(1,1)); % Equivalent to t = 0.1:0.1:30

% Gaussian distribution
fn = @(y,mu,sigma) 1/sqrt(2*pi*sigma)*exp(-0.5*(y-repmat(mu,1,size(y,2))).^2/sigma);

% Now ready to perform Gibbs sampling

% Prior variance of mu
sigma0 = 1.0;
```

```
% Prior Dirichlet concentration of indicator probability p
c = 1e-3; % 1e-4 for Test 12

% Likelihood variance of mu
sigma = 1.0;

% Initialization of the variables discussed
Kr = ones(R,1);
xr = ones(R,N);
mur = nan(R,N);
mur(1,1) = 0;

for r = 1:(R-1)

    tic; % To time the iterations
    x = xr(r,:);
    K = Kr(r);
    mu = mur(r,1:K);

    for n = 1:N

        % Calculate conditional probability
        pxy = zeros(K+1,1);
        for k = 1:K
            i = (x == k);
            Nkni = sum(i);
            if (x(n) == k)
                Nkni = Nkni - 1;
            end

            % Likelihood PDF w/ variance sigma: numerator in equation 3.14
            pxy(k) = Nkni*fn(Y(n),mu(k),sigma);
        end
        % Compound PDF w/ variance sigma0+sigma: numerator in equation 3.15
        pxy(K+1) = c*fn(Y(n),0,sigma0+sigma);

        pxy = pxy/sum(pxy); % Normalise

        % Sample indicator x from pxy
        x(n) = randsample(K+1,1,true,pxy);

 % As shown in section 3.6 it needs to be determined whether a new component needs to be created
 % See equation 3.16
        if (x(n) > K)
            K = K + 1;
            muhat = sigma0*Y(n)/(sigma+sigma0);
            sigmahat = sigma*sigma0/(sigma+sigma0);
            mu(K) = sqrt(sigmahat)*randn+muhat;
        end
    end

  % Sample means mu_x from conditional p(mu_(x_n)|y_n,x_n), see equation 3.16
    for k = 1:K
        i = (x == k);
        muhat = sigma0*sum(Y(:,i),2)/(sigma+sigma0*sum(i));
        sigmahat = sigma*sigma0/(sigma+sigma0*sum(i));
        mu(k) = sqrt(sigmahat)*randn+muhat;
    end
  % Storing outputs for future reference
```

```
    xr(r+1,:) = x;
    Kr(r+1) = K;
    mur(r+1,1:K) = mu;

  % Extract the last 10% of iterations of the Gibbs sampler for xr and mur
     xrp = xr(R - tenit:R,:);
     murp = mur(R - tenit:R,:);

  % Calculate the mode of xr and mean of mu for these last 10% of iterations
      xmode = mode(xrp);
      mumean = nanmean(murp); % 'nanmean' ignores the many 'NaN' values
     toc;
end

% Plot the raw data and the Gibbs approximation on the same plot
figure;
% When plotting the synthetic data:
%--------------------------------
  plot(t, Y', '-')
%--------------------------------

% When plotting the experimental test data:
%-------------------------------------
 plot(t,expTest(:,2),'-'); % *
%-------------------------------------
    hold on

  plot(t, mumean(xmode),'-'); % DPGMM Gibbs sampler result on same plot
    hold off

% When plotting the synthetic data:
%-------------------------------------
    ylabel('x');
    xlabel ('i');
    legend('x_{syn}','DPGMM Gibbs');

    figure; % (demo of synthetic data)
    plot(t, Y', '-')
    ylabel('x');
    xlabel ('i');
    legend('x_{syn}');

%-------------------------------------
% When plotting the experimental test data:
%-------------------------------------
    ylabel('Acceleration');
    xlabel ('Time(seconds)');
    legend('x','DPGMM Gibbs'); % *
%-------------------------------------
    axis tight;

figure;
  plot(Kr); % To show how k increases - see section 4.1.2
    xlabel('r');
    ylabel('k');

figure;
  plot(mur); % Cluster assignments - see section 4.1.2
    xlabel('r');
    ylabel('mu');
```

Appendix B

Protocols for Experimental Tests

This experimental test data was collected by Reham Badawy, a PhD Researcher at Aston University, in the following manner.

B.1 Test 5: Walking

This is also known as the 'gait' test.

1. Press start button.
2. Put phone in pocket.
3. Wait for buzzer to vibrate.
4. Walk 20 yards, turn and then walk another 20 yards.
5. Wait for buzzer to vibrate to signal end of test.

B.2 Test 12: Dropping Phone

1. Press start button while phone holding phone in hand.
2. Wait for buzzer to vibrate (phone still in hand).
3. Remain stationary (in order to be able to distinguish below step in accelerometer data).
4. Drop phone.
5. Leave phone on floor for a few seconds (to distinguish accelerometry pattern when phone is being dropped and when phone is being picked up in below step).
6. Pick up phone and hold in hand.
7. Remain stationary for a few seconds.
8. Drop phone.
9. Leave phone on floor for a few seconds.
10. Pick up phone and hold phone in hand.
11. Remain stationary for a few seconds.

12. Drop and pick up phone straight away.
13. Remain stationary while holding phone in hand (to identify whether it possible to distinguish accelerometry pattern when phone is being dropped and when phone is being picked up without pauses in between).
14. Drop and pick up phone straight away.
15. Remain stationary while holding phone in hand until buzzer vibrates to signal end of test.

B.3 Test 15: Falling

1. Press start button and place phone in cardigan pocket.
2. Wait for buzzer to vibrate (phone still in hand).
3. Walk for a few seconds and then 'fall'.
4. Remain stationary on floor for a few seconds (in order to be able to distinguish below step in accelerometry data).
5. Get up slowly (phone still in hand).
6. Remain stationary for a few seconds.
7. Walk for a few seconds and then 'fall'.
8. Remain stationary on floor for a few seconds.
9. Get up slowly (phone still in hand).
10. Remain stationary for a few seconds.
11. Walk for a few seconds, 'fall' and get up straight away, and then continue walking.
12. Walk until buzzer vibrates to signal end of test.

Bibliography

[1] C.E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The annals of statistics*, 2(6):1152–1174, 1974.

[2] L.J. Billera and P. Diaconis. A geometric interpretation of the Metropolis-Hastings algorithm. *Statistical science*, 16(4):335–339, 2001.

[3] C.M. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

[4] D.M. Blei and M.I. Jordan. Variational methods for the Dirichlet process. In *21st international conference on machine learning*. ACM Press, 2004.

[5] F. Caron. http://www.stats.ox.ac.uk/˜caron/code/abs2014/html/BNP_clustering.html. *(Accessed 04/03/2015)*, 2014.

[6] T.S. Ferguson. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, 1(2):209–230, 1973.

[7] C.J. Geyer. Practical Markov chain Monte Carlo. *Statistical science*, 7(4):473–511, 1992.

[8] W.R Gilks, S. Richardson, D.J. Spiegelhalter, and G.O. Roberts. *Markov chain Monte Carlo in practice*, volume 39. Chapman & Hall, 1996.

[9] W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrica*, 57(1):97–109, 1970.

[10] H. Ishwaran and L.F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American statistical association*, 96(453):161–168, 2001.

[11] M.I. Jordan. Graphical models. *Statistical science*, 19(1):140–155, 2004.

[12] B. Kulis and M.I. Jordan. Revisiting k-means: new algorithms via Bayesian nonparametrics. In *29th international conference on machine learning*, pages 1–4, 2012.

[13] V. Melnykov and R. Maitra. Finite mixture models and model-based clustering. *Statistics surveys*, 4:80–116, 2010.

[14] R.M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[15] Herbert Robbins. An Empirical Bayes approach to statistics. In *Proceedings of the third Berkeley symposium on mathematical statistics and probability*, volume 1, pages 157–163. University of California Press, 1956.

[16] Kyung Ah Sohn and Eric P. Xing. A hierarchical Dirichlet process mixture model for haplotype reconstruction from multi-population data. *Annals of applied statistics*, 3(2):791–821, 2009.

[17] N. White, H. Johnson, P. Silburn, G. Mellick, N. Dissanayaka, and K. Mengersen. Probabilistic subgroup identification using Bayesian finite mixture modelling: a case study in Parkinson's disease phenotype identification. *Statistical methods in medical research*, 21(6):563–583, 2012.

[18] N. White, H. Johnson, P. Silburn, and K. Mengersen. Dirichlet process mixture models for unsupervised clustering of symptoms in Parkinson's disease. *Journal of applied statistics*, 39:2363–2377, 2012.

[19] O. Zobay. Mean field inference for the Dirichlet process mixture model. *Electronic journal of statistics*, 3:507–545, 2009.