KAO, YIMIN. Advances in Nonparametric Bayesian Methods for Clustering and Classification. (Under the direction of Brian J. Reich.)

Nonparametric Bayesian methods have proven to be extremely useful due to their flexibility and applicability to a wide range of problems. In this thesis, several nonparametric Bayesian techniques are presented for classification and clustering. The first chapter is motivated by a computer security problem, and a nonparametric Bayesian model is proposed by implementing the Dirichlet process mixture (DPM) prior for classifying programs as benign or malicious and simultaneously clustering malicious programs. The novelty of the model is using this clustering algorithm to improve the classification accuracy. In an analysis of malicious and benign programs obtained from Los Alamos National Lab, the DPM model gives better classification performance than the elastic net logistic (ENL) model, and is competitive with the support vector machine (SVM). More importantly, the DPM model identifies clusters of programs during the classification procedure which is useful for reverse engineering.

In the second Chapter, we propose a new method for the fundamental task of testing for dependence between two groups of variables. The response densities under the null hypothesis of independence and the alternative hypothesis of dependence are specified by nonparametric Bayesian models. Under the null hypothesis, the joint distribution is modeled by the product of two independent DPM priors; under the alternative, the full joint density is modeled by a DPM prior. The test is then based on the posterior probability of favoring the alternative hypothesis. The proposed test not only has good performance for testing linear dependence among other popular nonparametric tests, but is also preferred to other methods in testing many of the nonlinear dependencies we explored.

Finally, in the third Chapter an efficient density estimation method is proposed for high-dimensional problems. Classical density estimation methods have difficulty in high dimensions due to the curse of dimensionality. Our proposed model first factorizes the original dimensions to several lower-dimensional groups, and uses the DPM prior to model the lower-dimensional densities. Our results show that when such a low-dimensional structure exists, the proposed model is more efficient than competing methods.
Advances in Nonparametric Bayesian Methods for Clustering and Classification

by

Yimin Kao

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Statistics

Raleigh, North Carolina

2014

APPROVED BY:

-----------------------------
Howard D. Bondell

-----------------------------
Sujit Ghosh

-----------------------------
Yichao Wu

-----------------------------
Grady Miller

-----------------------------
Brian J. Reich
Chair of Advisory Committee
Kao, Yimin was born in Taipei, Taiwan and grew up in Taipei. He earned his Bachelor degree in Statistics in 2007 from National Chench University and Master degree in Statistics in 2011 from North Carolina State University. After completing his master’s study, he continued to pursue his Ph.D. degree in North Carolina State University. After graduation, he will continue his research in statistics in either industrial or academic areas.
ACKNOWLEDGEMENTS

First, I want to express my great appreciation to my advisor Dr. Brian Reich. His profound knowledge of statistics benefits my journey of pursing professional knowledge. More importantly, his passion and patience both influence my attitude towards research. I feel extremely thankful to work with my advisor.

I also want to take this opportunity to thank Dr. Curtis Storlie for co-advising the first chapter of my thesis and Dr. Howard Bondell for co-advising the second and third chapters of my thesis. Also, I want to thank Dr. Sujit Ghosh and Dr. Yichao Wu for joining my committee and giving me many suggestions on my work.

Finally, I want to thank to my family and friends who always give me support and understanding.

“A dream doesn’t become reality through magic; it takes sweat, determination and hard work.” (Colin Powell)
# TABLE OF CONTENTS

**LIST OF TABLES** ................................................................. vii

**LIST OF FIGURES** ............................................................... viii

**Chapter 1 Introduction** .................................................. 1

**Chapter 2 Malware detection using nonparametric Bayesian clustering and classification techniques** .............................................. 3

- 2.1 Introduction ........................................................................ 3
- 2.2 Statistical model for malware detection ................................. 6
  - 2.2.1 Description of dynamic trace data ................................... 6
  - 2.2.2 Prior for the transition probability matrix distribution ........ 7
  - 2.2.3 Cluster indexes and marginalizing over the transition probability matrix ................................................................. 9
- 2.3 Classification and clustering ................................................. 10
  - 2.3.1 The full model ............................................................. 10
  - 2.3.2 Bayesian classification ................................................ 11
  - 2.3.3 Bayesian criterion for clustering ..................................... 12
- 2.4 Simulation Study .............................................................. 13
  - 2.4.1 Data generation ........................................................... 14
  - 2.4.2 Simulation results ........................................................ 15
- 2.5 Analysis of real malicious and benign programs ...................... 17
  - 2.5.1 Data description .......................................................... 17
  - 2.5.2 Classification results .................................................... 18
  - 2.5.3 Detection under different lengths of traces ....................... 18
  - 2.5.4 Clustering results ........................................................ 19
- 2.6 Conclusion ......................................................................... 20

**Chapter 3 A nonparametric Bayesian test of independence** ............ 24

- 3.1 Introduction ........................................................................ 24
- 3.2 Statistical model .................................................................. 26
  - 3.2.1 The independent DPM prior .......................................... 26
  - 3.2.2 The joint DPM prior ..................................................... 27
  - 3.2.3 Bayesian test of independence ........................................ 27
- 3.3 Computing details .............................................................. 28
  - 3.3.1 Reparameterization and hyperparameters ......................... 28
  - 3.3.2 Pseudo code for the DPM test of independence algorithm .... 29
  - 3.3.3 Steps of the RJMCMC algorithm .................................... 29
- 3.4 Simulation Study .............................................................. 31
B.1.2 Full conditionals for the joint DPM prior ........................ 78
B.2 Test of independence by E-statistics ................................. 79
B.3 Heller-Heller-Gorfine test of association based on Euclidean distance metric ......................................................... 80
B.4 Distribution-free tests of association based on data derived partitions 81
B.5 Maximal information coefficient for measuring dependence of two variables ................................................................. 82
Appendix C Supplementary materials for efficient Multidimensional Bayesian density estimation using factored densities .................. 83
C.1 Full conditionals for the DPM model under single factorized component 83
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Seven scenarios in the simulation study.</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>FDR, Power, and AUC subscripted with standard errors for the ENL, SVM and DPM models under seven scenarios. The Wilcoxon signed-rank test for the pairwise differences of the ENL, SVM and DPM models under Power, and AUC are all significant.</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Overall TPC and FPC subscripted with standard errors for scenario 2-7. (∗ represents the Wilcoxon signed-rank test for the pairwise differences of spectral clustering method and DPM models are shown to be significant, p-value &lt; 0.05. No ∗ represents non-significant.)</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>FDR, Power, and AUC subscripted with standard errors for the ENL, SVM, and the DPM models with { (K_0, K_1) = (150, 200) }.</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Clustering results for 5 testing malicious programs (ID:4, 87, 252, 342, 972), each lists the top 5 training samples ID with highest pairwise clustering probabilities in the parenthesis.</td>
<td>21</td>
</tr>
<tr>
<td>3.1</td>
<td>Hyperparameters ((a, b)) under different sample sizes (n).</td>
<td>29</td>
</tr>
<tr>
<td>3.2</td>
<td>Power of each test (columns) for each simulation settings and sample size (n) (rows). A ∗ indicates that the power is significantly different than the power of DPM test.</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Numbers of rejections (of the (N = 4371) tests), and Cohen’s (\kappa) statistics for each pair of methods.</td>
<td>36</td>
</tr>
<tr>
<td>4.1</td>
<td>Posterior probabilities of pairwise associations.</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>10-fold cross validation of deviances for the independent DPM (I), joint DPM (J), and the proposed RJ method.</td>
<td>52</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.1</td>
<td>( P(\xi_s^* = 1 \mid Z_{\xi_s^*}, Z) ) of the ENL (black lines) and the DPM models (red lines) under different lengths of traces for 4 malware samples.</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>Overall clustering heat map for ordered training and testing malware with ((K_0, K_1) = (150, 200)).</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>True log density (background color) and one simulated data set (points) for each simulation design.</td>
<td>33</td>
</tr>
<tr>
<td>3.2</td>
<td>Six pairs of genes where there are disagreements among the tests. The red lines are the linear fitted lines.</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>KLD under high-dimensional data.</td>
<td>47</td>
</tr>
<tr>
<td>4.2</td>
<td>Factorization probability (that is, the posterior probability that each pair of variables are in the same cluster) plots for four simulated data sets.</td>
<td>48</td>
</tr>
<tr>
<td>4.3</td>
<td>Tracks of 79 tropical storm tracks after year 2005.</td>
<td>49</td>
</tr>
<tr>
<td>4.4</td>
<td>Pairwise scatter plots for the tropical storm track data.</td>
<td>50</td>
</tr>
<tr>
<td>4.5</td>
<td>Factorization probability plot of the 344 tropical storm tracks.</td>
<td>51</td>
</tr>
<tr>
<td>4.6</td>
<td>Estimated density function of initial (left) and final (right) spatial locations.</td>
<td>52</td>
</tr>
<tr>
<td>4.7</td>
<td>Estimated conditional density function of the final location given the initial spatial location (denoted as a solid triangle).</td>
<td>54</td>
</tr>
<tr>
<td>4.8</td>
<td>2-dimensional data with ( b = 0, 0.3, 0.5, 0.8 ). Panel (a) plots the KLD for the 100 simulated data sets, and Panel (b) plots the posterior probabilities of the true factorization under the RJ model for the 100 data sets.</td>
<td>55</td>
</tr>
<tr>
<td>4.9</td>
<td>3-dimensional data with ( b = 0, 0.3, 0.5, 0.8 ). Panel (a) plots the KLD for the 100 simulated data sets, and Panel (b) plots the posterior probabilities of the true factorization under the RJ model for the 100 data sets.</td>
<td>55</td>
</tr>
<tr>
<td>4.10</td>
<td>4-dimensional data with ( b = 0, 0.3, 0.5, 0.8 ). Panel (a) plots the KLD for the 100 simulated data sets, and Panel (b) plots the posterior probabilities of the true factorization under the RJ model for the 100 data sets.</td>
<td>56</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Nonparametric Bayesian models have been widely used in many applications due to their flexibility and applicability. In this thesis, several nonparametric Bayesian techniques are presented for classification and clustering. The new method is motivated by a computer security problem which requires statistical methods to quickly and accurately flag malicious programs. We propose a nonparametric Bayesian approach by implementing the Dirichlet process mixture (DPM) model for classifying programs as benign or malicious, and simultaneously clustering malicious programs (Chapter 2). The novelty of the model is using this clustering algorithm to improve the classification accuracy. The simulation study shows that the DPM model outperforms the elastic net logistic (ENL) regression and the support vector machine (SVM) in classification performance under most of the scenarios we considered, and also outperforms the spectral clustering method for grouping similar malware. In an analysis of malicious and benign programs obtained from the Los Alamos National Lab, the DPM model gives better classification performance than the ENL model, and is competitive with the SVM method. More importantly, the DPM model identifies clusters of programs during the classification procedure which is useful for reverse engineering.

Next, we propose a new method for the fundamental task of testing for dependence between two groups of variables (Chapter 3). The response densities under the null hypothesis of independence and the alternative hypothesis of dependence are specified by nonparametric Bayesian models. Under the null hypothesis, the joint distribution is modeled as the product of two independent DPM priors; under the alternative, the full joint density is modeled by a DPM prior. The test is then based on the posterior probability
of favoring the alternative hypothesis. The proposed test not only has good performance for testing linear dependence, but is also preferred to other methods in testing many of the nonlinear dependencies we explored. In the analysis of the gene expression data, we compare different methods for testing pairwise dependence between genes. The proposed test identified some dependent structures that are not detected by other methods.

An efficient density estimation method is proposed in Chapter 4 for high-dimensional problems. Classical density estimation methods have difficulty in high-dimensional problems due to the curse of dimensionality. Our proposed model first factorizes the original dimensions to several lower-dimensional groups, and uses the DPM prior to model the lower-dimensional densities. Uncertainty about the correct factorization is accounted for in the Bayesian approach. The transition between different factorizations is a trans-dimensional problem which requires implementing the reversible jump MCMC algorithm. The simulation study shows that the proposed method outperforms the independent DPM, joint DPM, and kernel density estimation methods when the true density is the product of several lower-dimensional components. However, when the true density has complex high-dimensional joint structures, the proposed method is limited. We apply our method to estimate the spatiotemporal density of tropical storms in the Atlantic from 1981-2009. Cross validation shows that the proposed method has better performance than the independent DPM and the joint DPM models.
Chapter 2

Malware detection using nonparametric Bayesian clustering and classification techniques

2.1 Introduction

Malware is a term used to describe a variety of forms of hostile, intrusive, or annoying software or program code. It was recently estimated that 30% of computers operating in the US are infected with malware (PandaLabs, 2012). More than 286 million unique variants of malware were detected in 2010 alone (Symantec, 2011), and it is widely believed that the release rate of malicious programs is now far exceeding that of legitimate program applications (Symantec, 2008). The cost incurred by US companies due to malware was estimated in 2011 to be $338 Billion/year (Consumer Reports 2011). A large majority of new malware is created through simple modifications to existing malicious programs or by using code obfuscation techniques such as a packer (Royal et al., 2006). A packer compresses a program much the same way a compressor like Pkzip does, and then attaches its own decryption/loading stub which unpacks the program before resuming execution normally at the programs original entry point.

Malware is growing at such a rate that commercial antivirus vendors (AV) are not able to adequately keep up with new variants. Even though most new malware is very similar to known malware, it will often not be detected by signature-based antivirus programs (Christodorescu and Jha, 2003; Perdisci et al., 2006) until the malware signature
eventually works its way into the database, which can take weeks or even longer. There are two methods for antivirus scanners to implement their technology. The first is via a static signature scanning method, which uses a sequence of known bytes in a program and tests for the existence of this sequence. The second is using heuristic detection technologies which are intended to protect against zero-day (i.e., new) malware and malware not in the signature database.

Because of the signature-based susceptibility to new malware, classification procedures based on statistical and machine learning techniques have been employed with varied success to make a decision about the integrity of an unknown program (Reddy et al., 2006; Reddy and Pujari, 2006; Stolfo et al., 2005; Dai et al., 2009). These methods have generally revolved around n-gram analysis of the static binary or dynamic trace of the malicious program. Recently, some promising results have come from a Markov chain representation of the program trace (Anderson et al., 2011; Storlie et al., 2013). In these works, dynamic traces from many samples of malware and benign programs were used to train a classifier. A dynamic trace is a record of the sequence of instructions executed by the program as it is actually running. Although some success has been achieved using disassembled static binary files, this cannot always be done, particularly if the program uses an unknown packer, and therefore this approach has similar shortcomings to the signature based methods.

The classification of malware is a very important problem by itself, but it is only the beginning of malware analysis. Once malware is discovered on a host in an enterprise network, for example, the time consuming process of reverse engineering (RE) (Chikofsky and Cross, 1990) commences. A highly trained cyber professional can spend a day to weeks uncovering the purpose of the malware, its sender, and the extent of the attack. Not only does this tedious process require substantial time and labor, but it also results in a slower response. An organization cannot adequately respond to a malware infection until they understand what it does.

An approach to automatically cluster malware programs accurately would be a large benefit to the RE process. For example, when a new malware instance is detected, it can be clustered into a group, where perhaps some of the group members have already been reverse engineered by an analyst. The analyst can then use these previous efforts to more quickly understand the nature, functionality, origin, etc., of the newly suspected malicious program. However, clustering malware is a difficult problem because of the high
dimensionality required to adequately represent a computer program. Previous work that addresses the malware clustering problem relies on spectral clustering techniques (Anderson et al., 2012) and other heuristic approaches (Storlie et al., 2013).

The proposed model in this chapter builds upon the work of Anderson et al. (2011) and Storlie et al. (2013) by explicitly modeling the dynamic trace (DT) data as a Markov chain. Unlike the previous works, a rigorous probabilistic model is established for the generation of a program trace. Specifically, the nonparametric Bayesian model (Ghosh and Ramamoorthi, 2003) accomplishes both clustering and classification via discriminant analysis. This general approach has been applied successfully in density estimation problems (MacEachern, 1994; Escobar, 1994; West et al., 1994; Escobar and West, 1995; Muller and Quintana, 2004; Dunson, 2010), and also for classification and clustering purposes (Jackson et al., 2007; Shahbaba and Neal, 2009; Davy and Tourneret, 2010; Zhu et al., 2012). The contribution of this chapter is to tailor this general approach to malware detection, and to illustrate that this approach has benefits compared to current approaches to this problem. The DTs are assumed to be Markovian, which has been demonstrated to be a reasonable approximation in previous work (Storlie et al., 2013), and thus the proposed model focuses on the transition probability matrix. To perform the classification of the programs, the proposed method fits separate models for the distributions of transition matrices for malware and benign programs. To ensure that the model is flexible enough to capture the complexities of DTs, the Dirichlet process mixture (DPM) (Antoniak, 1974) prior is applied to span the entire space of row-stochastic matrices. After fitting the models using the training data, it classifies programs based on the posterior probability of being a malware in a way that controls the Bayesian false discovery rate. A natural by-product of the DPM model is that it places programs into clusters. Therefore, in addition to classification, the proposed model simultaneously obtains Monte Carlo estimates of the probability that a new program clusters with each malware in the training set, which can be used to aid reverse engineering.

The remainder of this chapter proceeds as follows. In Chapter 2.2, the DPM model is introduced for the distribution of transition matrices. Chapter 2.3 discusses how to use the model for classification and clustering. A simulation study is then given in Chapter 2.4 to compare the DPM model with recently-proposed methods for classification and clustering. Chapter 2.5 provides an analysis of real sets of malicious and benign programs provided by Los Alamos National Laboratory (LANL). Chapter 2.6 concludes the
2.2 Statistical model for malware detection

2.2.1 Description of dynamic trace data

A dynamic trace (DT) is a sequence of processor instructions called during execution. Let \( N \) be the total number of programs in the sample indexed by \( s, s = 1, ..., N \). The dynamic trace of the \( s^{th} \) program is denoted as \( \{Y_{s1}, ..., Y_{sN_s}\} \), where \( Y_{st} \in \{1, ..., M\} \) is the \( t^{th} \) instruction called during execution, \( N_s \) is the length of the dynamic trace, and \( M \) is the number of classes of processor instructions. As Storlie et al. (2013), we treat similar instructions as identical, by grouping similar instructions together into one of \( M \) classes. Given the first-order Markov structure, it is sufficient to specify the distribution of the first instruction \( Y_{s1} \) and the distribution of the \( M \times M \) transition matrix \( Z_s \), where the \((r,c)\) component, \( r,c \in \{1, ..., M\} \), of the \( Z_s \) is the count

\[
Z_{src} = \sum_{t=2}^{N_s} I(Y_{st-1} = r, Y_{st} = c).
\]

Both the initial instruction distribution and the transition matrix distribution are allowed to vary by programs. The initial instruction of a program \( s \) has distribution \( P(Y_{s1} = j) = q_{sj} \), with \( \sum_{j=1}^{M} q_{sj} = 1 \). The subsequent instructions, \( t = 2, ..., N_s \), are modeled as \( P(Y_{st} = c \mid Y_{st-1} = r) = P_{src} \), with \( \sum_{c=1}^{M} P_{src} = 1 \), for all \( r \). \( P_s \) is the \( M \times M \) matrix with \((r,c)\) component \( P_{src} \) which is the row-stochastic transition probability matrix for \( Z_s \).

Because \( N_s \) is typically very large, the initial instruction \( Y_{s1} \) provides little information about the characteristics of the DT. Therefore, \( Y_{s1} \) can be ignored and the sufficient statistic for the DT is then the transition matrix \( Z_s \). The likelihood function for one single program \( s \) has a simple form

\[
f(Z_s \mid P_s) = \prod_{r=1}^{M} \prod_{c=1}^{M} P_{src}^{Z_{src}}. \tag{2.1}
\]
The objective of this study is to focus entirely on the transition matrix $Z_s$, and to build a flexible nonparametric Bayesian model for the distribution of $P_s$ to capture variability across programs.

The remainder of this chapter only describes the model for the malicious programs to simplify notation. The model for benign programs is identical. Chapter 2.3 describes the full model including both benign and malicious programs. Given this full model, a new program can be classified as either benign or malicious, and also identify clusters of similar programs.

### 2.2.2 Prior for the transition probability matrix distribution

The transition probability matrix $P_s$ for one single program $s$ is assumed to have $P_s \overset{iid}{\sim} F$ for all $s$, where $F$ represents the distribution of transition probability matrices across malicious programs. In practice, there is only limited information about the form of the distribution before the data are observed. Therefore, rather than specify a particular parametric model, the proposed method places a prior on $F$ to ensure enough flexibility to capture variability among the malicious programs.

A straightforward nonparametric Bayesian model is to assume $F$ has a Dirichlet process prior (Ferguson, 1973)

$$
P_s \overset{iid}{\sim} F
$$

$$
F \sim DP(\alpha, F_b),
$$

where $\alpha > 0$ is the concentration parameter and $F_b$ is the base distribution of the Dirichlet process prior; large $\alpha$ forces the prior of $F$ closer to $F_b$, and vice versa. In our application, $F_b$ is the Matrix Dirichlet (MD) distribution, $F_b = MD(\gamma \tilde{P})$, which is centered on a constant $M \times M$ matrix $\tilde{P}$ with a concentration scalar parameter $\gamma > 0$. This MD distribution implies that if an $M \times M$ random matrix $X$ is from $MD(\gamma \tilde{P})$, then each row of $X$ independently follows the $M$-dimensional Dirichlet distribution with concentration parameters equal to the corresponding row of $\gamma \tilde{P}$

$$
X_r \overset{iid}{\sim} \text{Dirichlet}(\gamma \tilde{P}_r), \quad r = 1, ..., M,
$$
where \( X_r, \tilde{P}_r \) are the \( r \)th row of \( X \) and \( \tilde{P} \) respectively. In other words, \( \tilde{P} \) controls the general shape of the base distribution \( F_b \), and \( \gamma \) controls the variance of \( F_b \) with large \( \gamma \) corresponding to smaller variance.

The Dirichlet process prior in (2.2) can be written as a mixture density (Ferguson, 1983)

\[
f(P_s \mid w, \lambda) = \sum_{k=1}^{\infty} w_k \cdot \delta(P_s \mid \lambda_k),
\]

(2.3)

where \( \lambda = \{\lambda_k : k = 1, ..., \infty\} \), \( w = \{w_k : k = 1, ..., \infty\} \), and the mixture probabilities \( w_k > 0 \) satisfy \( \sum_{k=1}^{\infty} w_k = 1 \). \( \delta \) is the Dirac Delta density with point mass at \( P_s = \lambda_k \), and \( \lambda_k \sim F_b \). Equation (2.3) can be thought of as mixing infinitely-many point mass distributions, and \( \lambda_k \) controls the shape of the \( k \)th point mass distribution.

While the Dirichlet process prior in (2.3) has attractive features, it is not appropriate for the analysis of transition probability matrices because it gives a discrete distribution. In other words, it implies that the transition probability matrices of two executable programs could be identical, which is unrealistic. Therefore, the proposed model considers a Dirichlet process mixture (DPM) prior (Antoniak, 1974; Neal, 2000) to alleviate this concern. The DPM prior has the form

\[
P_s \mid \theta_s, \sigma \sim \text{MD}(\sigma \theta_s)
\]

\[
\theta_s \sim G
\]

\[
G \sim \text{DP}(\alpha, F_b),
\]

(2.4)

where the scalar parameter \( \sigma > 0 \) controls the variance of the distribution of \( P_s \) with large \( \sigma \) corresponding to smaller variance, the \( M \times M \) matrix \( \theta_s \) is the shape parameter of the distribution of \( P_s \), \( \alpha \) and \( F_b \) are defined as in (2.2). Compared with (2.2), the distribution of \( P_s \) is now continuous as the MD distribution is continuous, and therefore the probability of two transition probability matrices are identical is zero. This DPM model can also be written as a mixture model density

\[
f(P_s \mid w, \lambda, \sigma) = \sum_{k=1}^{\infty} w_k \cdot f_{MD}(P_s \mid \sigma \lambda_k),
\]

(2.5)

where \( w = \{w_k : k = 1, ..., K\} \), and \( f_{MD} \) is the probability density function of MD distribution.
Equation (2.5) is the stick-breaking representation of a DPM model (Ferguson, 1983). The weights \( w_k \) are modeled in terms of latent \( v_k \sim \text{Beta}(1, \alpha) \). The first weight is \( w_1 = v_1 \). The remaining elements are modeled as \( w_k = v_k \prod_{j=1}^{k-1} (1 - v_j) \), where \( \prod_{j=1}^{k-1} (1 - v_j) = 1 - \sum_{j=1}^{k-1} w_j \) is the remaining probability after accounting first \( k - 1 \) mixture weights. By construction, little is lost by truncating the sum at a large number of terms, say \( K \), where the last term \( v_K \) is fixed to be 1, and thus \( \sum_{k=1}^{K} w_k = 1 \). Accuracy of the approximation is monitored by inspecting the posterior of the final term \( w_K \) which should be near zero if the approximation is sufficient.

### 2.2.3 Cluster indexes and marginalizing over the transition probability matrix

The mixing structure in (2.5) can also be viewed as a clustering model by introducing an auxiliary variable \( g_s \in \{1, \ldots, K\} \) to represent the cluster index of \( s^{th} \) malware observation, where \( g_s = k \) indicates \( s^{th} \) malware is in cluster \( k \). The DPM model assumes that the \( P_s \) of all the malware in the same cluster are from the same MD distribution, and the prior probability of a malware being in cluster \( k \) is \( w_k \). This representation facilitates Bayesian clustering in the hierarchical structure

\[
f(Z_s \mid P_s) = \prod_{r=1}^{M} \prod_{c=1}^{M} p_{src}^{Z_{src}}\tag{2.6}
\]

\[
P_s \mid \lambda, \sigma, g_s = k \, \text{iid} \sim \text{MD}(\sigma \lambda_k)\tag{2.7}
\]

\[
\lambda_k \mid \gamma \, \text{iid} \sim \text{MD}(\gamma \bar{P})\tag{2.8}
\]

\[
P(g_s = k \mid w) = w_k.\tag{2.9}
\]

The structure above includes the transition probability matrix \( P_s \), however, \( P_s \) itself is not of interest and including \( P_s \) in the model will increase the computing time. Therefore, by marginalizing over \( P_s \), the likelihood function of \( Z_s \) can be derived as a Matrix Multivariate Pólya (MMP) distribution (derived in Appendix A.1)

\[
Z_s \mid \lambda, \sigma, g_s = k \sim \text{MMP}(\lambda_k, \sigma)\tag{2.10}
\]
Therefore, (2.6) and (2.7) are replaced by (2.10), which is the hierarchical model used in the study. Similarly, (2.10) can be written as the mixture model density

\[ f(Z_s \mid \lambda, \sigma, w) = \sum_{k=1}^{K} w_k \cdot f_{MMP}(Z_s \mid \lambda_k, \sigma), \]

(2.11)

where \( f_{MMP} \) is the probability density function of MMP distribution.

## 2.3 Classification and clustering

This chapter describes how the DPM model classifies and clusters programs. First, the full model is specified which includes both benign and malicious programs. After the full model is defined, the Bayesian classification and clustering algorithms are described.

### 2.3.1 The full model

The full model is constructed by introducing another auxiliary variable \( \xi_s \), where

\[ \xi_s = \begin{cases} 
1 & \text{if program } s \text{ is malicious} \\
0 & \text{if program } s \text{ is benign}
\end{cases} \]

(2.12)

and \( P(\xi_s = 1) = \psi \) is the prior probability of being a malicious program. The full model of a single program \( s \) can be written as

\[ f(Z_s \mid \xi_s = i, \Theta_i) = \sum_{k=1}^{K_i} w_{ik} \cdot f_{MMP}(Z_s \mid \lambda_{ik}, \sigma_i), \quad i = 0, 1 \]

(2.13)

\[ \lambda_{ik} \overset{iid}{\sim} \text{MD}(\gamma_i \tilde{P}_i) \]

\[ \xi_s \sim \text{Bern}(\psi), \]

where \( \Theta_0 = \{\lambda_0, \gamma_0, \sigma_0, w_0, K_0\} \) and \( \Theta_1 = \{\lambda_1, \gamma_1, \sigma_1, w_1, K_1\} \) are the collections of parameters under benign and malicious models respectively, which means the proposed method allows different models for benign and malicious programs respectively, which means the proposed method allows different models for benign and malicious programs, i.e., different mixture model densities and different base distributions in the DPM priors. The truncated number of clusters \( K_i \) could be different depending on the sample size of each class of programs.

The priors, \( \sigma_i \) (controls the variance within clusters, \( i \in \{0, 1\} \)) and \( \gamma_i \) (controls the
variance between clusters) are assigned to have non-informative gamma distributions. The hyperparameter of the stick breaking representation in (2.5) is \( \alpha_i \sim \text{Gamma}(a_i, b_i) \), where \( a_i \) and \( b_i \) are picked such that the prior on the number of clusters (i.e., number of unique values of \( g_s \)) is fairly uninformative. More details of the model settings are in Appendix A.2.

### 2.3.2 Bayesian classification

Let \( Z \) be the collection of all observed \( Z_s \) where \( \xi_s \) is known. The model is then trained by \( Z \), and the posterior probability of a new program \( s^* \) being a malware is

\[
P(\xi_{s^*} = 1 \mid Z_{s^*}, Z) = \frac{f(Z_{s^*} \mid \xi_{s^*} = 1, Z) \cdot P(\xi_{s^*} = 1)}{f(Z_{s^*} \mid \xi_{s^*} = 1, Z) \cdot P(\xi_{s^*} = 1) + f(Z_{s^*} \mid \xi_{s^*} = 0, Z) \cdot P(\xi_{s^*} = 0)}.
\]

Under the Bayes rule, the new program \( s^* \) is classified as malicious if this probability exceeds a predefined threshold.

Let \( U_{s^*} \) denote the approximation of the posterior probability of being a malware calculated by post processing the MCMC samples (described in Appendix A.3)

\[
U_{s^*} = \frac{\sum_{l=1}^{n_M} f(Z_{s^*} \mid \Theta_1^{(l)}, Z)}{\sum_{l=1}^{n_M} f(Z_{s^*} \mid \Theta_1^{(l)}, Z) + \sum_{l=1}^{n_M} f(Z_{s^*} \mid \Theta_0^{(l)}, Z)},
\]

where \( \Theta_i^{(l)} \) is the collection of parameters in the \( l^{th} \) MCMC step under class \( i \) (\( i = 1 \) for malicious, \( i = 0 \) for benign program), and \( n_M \) is the number of MCMC iterations. The classification is to label program \( s \) as malicious if \( U_{s^*} > T \), where \( T \) is a tuning threshold. One way to determine the threshold \( T \) is by using the Bayesian false discovery rate (BFDR) control procedure (Efron and Tibshirani, 2002; Newton et al., 2004; Storey et al., 2004; Muller et al., 2006) under the observed samples, which is described below:

1. Suppose there are \( n_Z \) observed programs. Define BFDR under threshold \( t \) as

\[
\text{BFDR}_t = \frac{\sum_{s=1}^{n_Z} r_s (1 - U_s)}{\sum_{s=1}^{n_Z} r_s}, \text{ where } r_s = I(U_s > t).
\]

In practice, to avoid \( \sum_{s=1}^{n_Z} r_s \) being too small (only very few programs are classified as malicious), \( t \) is restricted to satisfy \( t : \sum_{s=1}^{n_Z} r_s > R_0 \), where \( R_0 \) is a reasonable positive
number. In this chapter, \( R_0 \) is set to be 5.

2. Let \( U_h^{(1)} < U_h^{(2)} < \ldots < U_h^{(n_Z)} \) be the ordered values of \( \{ U_s : s = 1, \ldots, n_Z \} \). For controlling BFDR under \( \alpha \) level, the optimal threshold is

\[
T = U_h^{(c)}, \quad \text{where} \ c = \min\{ j : \text{BFDR}_{U_h^{(j)}} \leq \alpha, \sum_{s=1}^{n_Z} r_s > R_0 \}
\]

3. Then for the new program \( s^* \), if \( U_{s^*} > T \), we classify it as malicious.

The above procedure controls the BFDR, however, by assuming independence between observations, Muller et al. (2004) and Muller et al. (2007) showed that BFDR control implies frequentist FDR control.

### 2.3.3 Bayesian criterion for clustering

Besides classifying the new program \( s^* \) as either benign or malicious, the proposed model can also cluster the new program with existing programs that share some common features. Recall in Chapter 2.2.3, we introduce an auxiliary variable \( g_s = k \), which implies that the \( s^{th} \) executable program is in cluster \( k \) (where clusters are defined separately for benign and malicious classes). However, the cluster label itself is not of interest, as the labels vary over MCMC iterations. The main interest is in whether two programs are in the same cluster. In other words, for a new program \( s^* \) that is classified as a malware, the main objective is to calculate the posterior probability that \( s^* \) is clustered with an existing malware \( s' \). This pairwise cluster probability can be approximated by using the MCMC output

\[
c_{s^*s'} = P(g_{s^*} = g_{s'}, \mathbf{Z}_{s^*}, \mathbf{Z}) = \frac{1}{n_M} \sum_{l=1}^{n_M} I(g_{s^*} = g_{s'} \mid \Theta_1^{(l)}, \mathbf{Z}_{s^*}, \mathbf{Z}).
\]

Note that the calculation of \( c_{s^*s'} \) is free of the values of the cluster labels \( g_{s^*} \) and \( g_{s'} \).
2.4 Simulation Study

In the simulation study, samples of transition matrices are generated from seven different scenarios. The number of instructions \(M = 3\) is considered and the length of dynamic trace equal to 10,000 (plus 1 fixed initial instruction) for the entire simulation study. For each scenario, 100 data sets are generated, with each data set consisting of \(n_Z = 200\) training samples and \(n_v = 200\) testing samples. Within each training and testing sample, exactly half of the samples are assigned to be malware and the other half are benign (except for the data generated from logistic regression, the true malicious indicator \(\xi_s\) is sampled randomly with probability \(P(\xi_s = 1 \mid P_s, \beta)\)). For classification, the proposed DPM model is compared with the elastic net logistic (ENL) regression model (Zou and Hastie, 2005) (described in Appendix A.4), and the support vector machine (SVM) (Cortes and Vapink, 1995) (described in Appendix A.5), which was applied in Anderson et al. (2011). Both ENL and SVM use the elements of the empirical transition matrix from each program as its features for classification.

Denote \(d_s\) as the classification result of program \(s\) under each method (\(d_s = 1\) for malicious, \(d_s = 0\) for benign) and let \(\xi_s\) be the true class. Methods are compared in terms of the following three criteria under the testing samples

- False discovery rate (FDR) = \(\frac{\sum_{s=1}^{n_v} d_s(1-\xi_s)}{\sum_{s=1}^{n_v} d_s}\).

- Sensitivity (Power) = \(\frac{\sum_{s=1}^{n_v} d_s \xi_s}{\sum_{s=1}^{n_v} \xi_s}\).

- Area under the ROC curve (AUC).

FDR represents the expected proportion of incorrectly specified programs as malware under the model. The FDR is controlled with \(\alpha = 0.1\) level in the DPM model using the procedure introduced in Chapter 2.3.2, and for the ENL model and the SVM, FDR are directly controlled by the testing samples also with \(\alpha = 0.1\). Sensitivity is the proportion of true malicious programs that are identified as malicious. AUC represents the probability that the model will rank a randomly chosen malware higher than a randomly chosen benign program on the basis of probability of maliciousness (assuming the malware ranks higher than benign program).

In the scenarios where data are generated with clusters, the DPM model is compared with the spectral clustering algorithm presented in Anderson et al. (2012) (described in
Appendix A.6). The performances are compared under testing samples using

\[
\text{True pairwise cluster (TPC)} = \frac{\sum_{i=1}^{n_h} \sum_{j=1}^{n_v} I(g_{s_i} = g_{s_j}) D_{ij}}{\sum_{i=1}^{n_h} \sum_{j=1}^{n_v} I(g_{s_i} = g_{s_j})},
\]

\[
\text{False pairwise cluster (FPC)} = \frac{\sum_{i=1}^{n_h} \sum_{j=1}^{n_v} I(g_{s_i} \neq g_{s_j}) D_{ij}}{\sum_{i=1}^{n_h} \sum_{j=1}^{n_v} I(g_{s_i} \neq g_{s_j})},
\]

where under the DPM model, \( D_{ij} = c_{s_i,s_j} \) is the posterior pairwise cluster probability in (2.17), and for the spectral clustering method, \( D_{ij} = I(g_{s_i} = g_{s_j} \mid Z_h, Z_v) \) which is the pairwise clustering decision. The reason that \( D_{ij} \) are defined differently is because the DPM model presents a soft clustering result, but the spectral clustering method gives a hard clustering result. The hard clustering of the DPM model by setting \( D_{ij} = I(c_{s_i,s_j} > 0.5) \) was also computed and found to be similar to the soft clustering DPM model.

2.4.1 Data generation

The data sets are generated under seven different scenarios. The classification results of the DPM, the ENL, and the SVM methods are compared for all scenarios, and the clustering results of the DPM and the spectral clustering methods are compared under scenario 2 to scenario 7 (scenario 1 has no clustering).

Data is generated from logistic regression in the first scenario by

\[
\text{logit} \left[ P(\xi_s = 1 \mid P_s, \beta) \right] = \sum_{r=1}^{M} \sum_{c=1}^{M} P_{src} \beta_{rc},
\]

where the true \( P_s \) is generated from \( \text{MD}(1) \), each \( \beta_{rc} \) is generated from \( \text{Uniform}(-10, 10) \), and \( \xi_s \) are sampled from \( \text{Bernoulli}(p_s) \) with probability \( p_s = P(\xi_s = 1 \mid P_s, \beta) \).

Scenarios 2 to 5 are generated from the first-order Markovian, and each with different levels of clustering controlled by (a) the true number of clusters \( K^* \) (assuming the true number of cluster under benign and malicious programs are the same), (b) within-cluster variation (\( \sigma_0 \) and \( \sigma_1 \), where large values give strong clustering), and (c) between-cluster variation (\( \gamma_0 \) and \( \gamma_1 \), where small values give strong clustering). The transition matrix \( Z_s \) is obtained from (2.13) with the true \( \tilde{P}_0 = \tilde{P}_1 = \tilde{P} \sim \text{MD}(1) \), the cluster probability
Table 2.1: Seven scenarios in the simulation study.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Model</th>
<th>$K^*$</th>
<th>$\sigma_0$</th>
<th>$\sigma_1$</th>
<th>$\gamma_0$</th>
<th>$\gamma_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Logistic</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1st-order Markov</td>
<td>2</td>
<td>10</td>
<td>40</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>1st-order Markov</td>
<td>5</td>
<td>10</td>
<td>40</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>1st-order Markov</td>
<td>5</td>
<td>30</td>
<td>30</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1st-order Markov</td>
<td>5</td>
<td>10</td>
<td>40</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>2nd-order Markov</td>
<td>5</td>
<td>10</td>
<td>40</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>2nd-order Markov</td>
<td>5</td>
<td>30</td>
<td>30</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

vectors are $\mathbf{w}_i = (\frac{1}{K^*}, ..., \frac{1}{K^*})$, and both benign and malicious programs have the same true number of clusters. Examination of the sensitivity of the model assumption is also given by comparing with the programs generated from the second-order Markovian. Scenarios 6 and 7 are generated from the second-order Markov model, which assumes the current instruction being called in time $t$ not only depends on the instruction at time $t - 1$ but also on the instruction at $t - 2$. The transition probability for program $s$ now is a three-dimensional array, where $P_{slrc} = P(Y_{st} = c \mid Y_{st-1} = r, Y_{st-2} = l)$. The data sets are again generated from (2.13) similar to the first-ordered Markovian data sets but $Z_s, P_s,$ and $\lambda_i (i = 0, 1)$ are now three-dimensional arrays, and the matrix distributions MD and MMP are now array distributions, i.e., the vectors across the last dimension of the array for any given values of the first two dimensions are assumed to be Dirichlet. Table 3.2 gives the summary of these seven scenarios.

### 2.4.2 Simulation results

The classification results of the simulation study are presented in Table 2.2. The results show that when the data sets are generated from the logistic regression model, the ENL model performs better than the DPM model by 1% and also better than the SVM by 0.5% in AUC criterion. The ENL model also has higher power than other two methods. Therefore, the ENL model generally performed better than other two methods which is not surprising since the data was originally generated from the logistic regression. On the other hand, when the data sets are generated from the first-order or second-order Markov models, the DPM model outperforms the ENL and the SVM methods significantly in terms of both power and AUC criterions.

By comparing the classification performances under different clustering levels, we
found that when the true number of clusters $K^*$ increases from 2 to 5 (Scenario 2 and 3), the DPM model has much higher AUC than the other methods. Also, when the between cluster variation increases or within cluster variation decreases (Scenario 3, 4, and 5), the DPM model again has higher power and higher AUC than the other methods. Scenario 4 has the best classification results among all seven scenarios as this setting has the largest between-cluster variation and smallest within-cluster variation. For the last two scenarios where the data are generated from the second-order Markov model, the DPM of the DPM model is difficult to control around the nominal rate. As the proposed model is constructed under the first-order Markov assumption, this suggests the FDR for our first-order model may be slightly sensitive to model misspecification.

Clustering results for the simulation study are in Table 2.3. The DPM model implements a clustering technique along with the classification process, and the spectral clustering does not. Therefore, for simplicity, the clustering results of both algorithms are compared under the assumption that the programs are well-classified. When the data sets are generated from the first-order Markov model, the DPM clustering always performs better than the spectral clustering method with higher TPC and lower FPC. For the last two scenarios where the data are generated from the second-order model, both the spectral clustering and the DPM methods have lower clustering ability, but the DPM model still has lower FPC than the spectral clustering method.
Table 2.3: Overall TPC and FPC subscripted with standard errors for scenario 2-7. (* represents the Wilcoxon signed-rank test for the pairwise differences of spectral clustering method and DPM models are shown to be significant, p-value < 0.05. No * represents non-significant.)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>TPC</th>
<th>FPC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spectral clustering</td>
<td>DPM</td>
</tr>
<tr>
<td>2</td>
<td>0.613* (0.003)</td>
<td>0.690* (0.011)</td>
</tr>
<tr>
<td>3</td>
<td>0.333* (0.003)</td>
<td>0.574* (0.008)</td>
</tr>
<tr>
<td>4</td>
<td>0.464* (0.005)</td>
<td>0.894* (0.007)</td>
</tr>
<tr>
<td>5</td>
<td>0.426* (0.004)</td>
<td>0.814* (0.008)</td>
</tr>
<tr>
<td>6</td>
<td>0.434* (0.006)</td>
<td>0.535* (0.009)</td>
</tr>
<tr>
<td>7</td>
<td>0.323* (0.006)</td>
<td>0.320 (0.003)</td>
</tr>
</tbody>
</table>

2.5 Analysis of real malicious and benign programs

2.5.1 Data description

Here a modified version of the Ether Malware Analysis framework (Dinaburg et al., 2008) was used to perform the dynamic trace data collection. Collecting dynamic traces can be slow, however, the current implementation is sufficient for a sandbox environment (i.e., the program is passed along to the user that requested it, while being run on a separate machine devoted to analysis) (Goldberg et al., 1996). For example, many institutions implement an email/http inspection system to filter for spam and malware. Inserting the proposed methodology into this process allows for a more robust approach to analyzing new threats as they are received. The data set used here contains dynamic traces from a five minute run of the program for 1613 malicious and 634 benign programs, respectively, for a total of 2247 observations with the number of instruction classes $M = 8$. This data set is a randomly selected subset of that used in Storlie et al. (2013). There, the malware sample was obtained via a random sample of programs from the website http://www.offensivecomputing.net/, which is a repository that collects malware instances in conjunction with several institutions.

Malware samples are acquired by the Offensive Computing Website through user contributions, capture via mwcollectors and other honeypots (i.e., traps set up on a network for the purpose of collecting malware or other information about possible network attacks), discovery on compromised systems, and sharing with various institutions. Ad-
mittedly, this is not a truly random sample from the population of all malicious programs that a given network may see, but it is one of the largest publicly available malware collections on the Internet (Quist, 2012). To obtain a sample of benign programs, Storlie et al. (2013) gathered a collection of many programs that were running on Los Alamos National Laboratory (LANL) systems during 2012. If these programs passed through a suite of 25 AV engines as "clean" then they were treated as benign.

2.5.2 Classification results

The analysis implements a repeated random sub-sampling validation (RRSV) (which is also known as Monte Carlo cross-validation (Picard and Cook, 1984; Xu and Liang, 2001)) to evaluate the classification for the ENL, the SVM, and the DPM models. A 20-fold RRSV method is used for evaluating the classification performance. Each of the RRSV data set is obtained by randomly and equally split the whole samples into training and testing sets. The evaluation will be based on the performances over these 20 RRSV data sets. The FDR, power, and AUC for each model are presented in Table 2.4. The results for the DPM model are presented with truncation values \((K_0, K_1) = (150, 200)\) in (2.13). Other values of \((K_0, K_1)\) are also implemented but found that a smaller number of clusters will hurt the classification performances, and a larger number of clusters give similar results. Also, by checking the posterior median of the last mixture probability \(w_{iK} (i = 0, 1)\), the values are close to 0.03.

By comparing the DPM and the ENL methods, the DPM has 4.4% higher power than the ENL model and 3.3% higher in AUC. The differences are all statistically significant. The DPM model performs slightly better than the SVM method under the auc criterion, but the differences are not significant, which indicates that these two methods are competitive. However, the advantage of using the DPM model is that the classification probability is calculated and can be obtained for each sample. More importantly, a clustering algorithm is the by-product of the DPM model.

2.5.3 Detection under different lengths of traces

Another important perspective in comparing malware classifiers is determining how long of an instruction trace is needed before identifying the program as malicious or benign. In order to address this problem, four testing malware samples are picked from one
RRSV data set. The transition probability matrix $P_s$ for each sample is estimated by the corresponding transition matrix $Z_s$. Then from this estimates $\hat{P_s}$, new transition matrices $Z_s'$ are generated with different lengths of traces. For each sample, we generate lengths of traces from 100 to 35,000. 50 samples are generated under different lengths. Two soft classifiers: the ENL model and the DPM model are compared. Figure 2.1 shows a 95% confidence interval of $P(\xi^{s*}_s = 1 \mid Z_{s^*}, Z)$ for the ENL model and a 95% credible interval of $P(\xi^{s*}_s = 1 \mid Z_{s^*}, Z)$ for the DPM model. In sample A, both methods correctly classify the sample from shorter lengths (i.e. two lines are overlapped). In sample B, C, and D, both models correctly classify the sample eventually, however the DPM model requires shorter lengths than the ENL model to achieve narrower 95% interval. In general, the DPM model requires shorter traces to achieve higher probability for accurately classifying a program and gives a narrower 95% intervals than the ENL model. Also, as the length of the trace increases, the probabilities of being malicious go to one for the DPM model for these four examples, which is showing that the flexible DPM model can make better use of the trace data in some cases.

### 2.5.4 Clustering results

To illustrate how clustering can be used in practice, we first look at the overall clustering between training and testing malware for one RRSV data set. Figure 2.2 is the heat map of the training malware along with the testing malware after ordering by their clustering probabilities calculated from the DPM model with $(K_0, K_1) = (150, 200)$. Darker color represents higher pairwise cluster probabilities. There are very distinct clusters of malicious programs. The blocks along the diagonal represent clusters of programs with pairwise cluster probabilities all near one. Also, the black rectangles off the diagonal show
some similarities between different clusters.

We also choose five malicious testing samples, and present the top five training samples that might be clustered with these testing sets under. From the results in Table 2.5, one advantage of the DPM algorithm is that we could rank the clustering programs and also obtain the pairwise clustering probability instead of the hard clustering method as spectral clustering.

To test the performance of our clustering algorithm in a case where the true clustering is known, we use the case study of the Bagle virus (bag, Accessed 9 December 2013) with the original real data set. The Bagle virus is known to spread via its own smtp engine and by using filesharing networks such as KaZaA. The main purpose of the bagle virus was to create a large botnet for the purpose of spam dissemination. Four programs in the Bagle family are examined: Bagle.n, Bagle.o, Bagle.q and Bagle.r. For each sample, say Bagle.n, the model is trained with the other three programs: Bagle.o, Bagle.q and Bagle.r added to the training programs. We then compute the clustering probability that Bagle.n is clustered with one of Bagle.o, Bagle.q and Bagle.r. The DPM method was able to successful cluster programs Bagle.o, Bagle.q and Bagle.r with at least one member of this family with probability one. However, Bagle.n was only clustered with another member of the family with probability 0.02. Compared to the other variants, Bagle.n is unique as it contains polymorphic code to infect other executables found on a windows system. Our results were able to clearly find this difference. For comparison, we applied the spectral clustering method of Anderson et al. (2012), and found that none of these four programs were clustered with their family, further highlighting the utility of the proposed method.

2.6 Conclusion

The DPM model implements a nonparametric Bayesian method in classifying and clustering programs. The clustering algorithm improves the ability of classification by identifying a program that has common structure with the observed samples. In the simulation study, the proposed model outperforms the ENL and the SVM methods in classification when the data sets are generated with clusters. The proposed model also has higher clustering accuracy than the spectral clustering method in the simulation study. In the analysis of the real data sets provided by LANL, the DPM model provides
Table 2.5: Clustering results for 5 testing malicious programs (ID:4, 87, 252, 342, 972), each lists the top 5 training samples ID with highest pairwise clustering probabilities in the parenthesis.

<table>
<thead>
<tr>
<th>Malware ID</th>
<th>4</th>
<th>87</th>
<th>252</th>
<th>342</th>
<th>972</th>
</tr>
</thead>
<tbody>
<tr>
<td>1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>753&lt;sub&gt;(1,000)&lt;/sub&gt;</td>
<td>97&lt;sub&gt;(0.827)&lt;/sub&gt;</td>
<td>371&lt;sub&gt;(0.422)&lt;/sub&gt;</td>
<td>1022&lt;sub&gt;(0.937)&lt;/sub&gt;</td>
<td>913&lt;sub&gt;(0.666)&lt;/sub&gt;</td>
</tr>
<tr>
<td>2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>885&lt;sub&gt;(1,000)&lt;/sub&gt;</td>
<td>1172&lt;sub&gt;(0.163)&lt;/sub&gt;</td>
<td>662&lt;sub&gt;(0.199)&lt;/sub&gt;</td>
<td>74&lt;sub&gt;(0.936)&lt;/sub&gt;</td>
<td>74&lt;sub&gt;(0.662)&lt;/sub&gt;</td>
</tr>
<tr>
<td>3&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>1192&lt;sub&gt;(1,000)&lt;/sub&gt;</td>
<td>914&lt;sub&gt;(0.163)&lt;/sub&gt;</td>
<td>651&lt;sub&gt;(0.199)&lt;/sub&gt;</td>
<td>925&lt;sub&gt;(0.936)&lt;/sub&gt;</td>
<td>1022&lt;sub&gt;0.662&lt;/sub&gt;</td>
</tr>
<tr>
<td>4&lt;sup&gt;th&lt;/sup&gt;</td>
<td>727&lt;sub&gt;(1,000)&lt;/sub&gt;</td>
<td>1512&lt;sub&gt;(0.163)&lt;/sub&gt;</td>
<td>646&lt;sub&gt;(0.199)&lt;/sub&gt;</td>
<td>913&lt;sub&gt;(0.787)&lt;/sub&gt;</td>
<td>925&lt;sub&gt;(0.661)&lt;/sub&gt;</td>
</tr>
<tr>
<td>5&lt;sup&gt;th&lt;/sup&gt;</td>
<td>1086&lt;sub&gt;(1,000)&lt;/sub&gt;</td>
<td>557&lt;sub&gt;(0.163)&lt;/sub&gt;</td>
<td>672&lt;sub&gt;(0.199)&lt;/sub&gt;</td>
<td>12&lt;sub&gt;(0.531)&lt;/sub&gt;</td>
<td>12&lt;sub&gt;(0.550)&lt;/sub&gt;</td>
</tr>
</tbody>
</table>

significantly better classification performance than the ENL method and is competitive with the SVM model. Beyond classification, the proposed model also provides clustering among the samples which can be useful for reverse engineering.
Figure 2.1: $P(\xi_{s^*} = 1 \mid Z_{s^*}, Z)$ of the ENL (black lines) and the DPM models (red lines) under different lengths of traces for 4 malware samples.
Figure 2.2: Overall clustering heat map for ordered training and testing malware with $(K_0, K_1) = (150, 200)$
Chapter 3

A nonparametric Bayesian test of independence

3.1 Introduction

A fundamental statistical task is to determine whether two groups of variables are dependent. For example, in genomic analysis, we might want to test whether two groups of genes are associated to identify dependence between genetic pathways. In the brain imaging research, we may want to discover whether sets of voxels from different parts of the brain are related to explore functional connectivity. In general, high-dimensional data analysis can be simplified by identifying sets of independent variables.

Testing of dependence is often reduced to testing for linear dependence. Pearson correlation coefficient is a classical and widely used method for quantifying the strength of linear dependence between two univariate variables. Spearman’s rank correlation coefficient (Spearman, 1904) is a ranked-based version of Pearson correlation coefficient which quantifies monotone correlation. Tests based on correlation are powerful for testing specific types of association, but lose power for other general types.

For testing more general associations, the $\chi^2$ test of independence and Hoeffding’s test of independence (Hoeffding, 1948) are two classic and widely used nonparametric methods. These tests are based on partitioning data into a contingency table. The main drawback for $\chi^2$ test is that the result is sensitive to the way the data are partitioned. Several approximations of the test statistics of the Hoeffding’s test are studied: Blum et al. (1961) introduce an approximation by the concordances and discordances of a $2 \times 2$
contingency table formed by the data, and Wilding and Mudholkar (2008) propose an approximation by using the two Weibull extensions. A relation between the Hoeffding’s test and the $\chi^2$ test statistics was noted by Thas and Ottoy (2004), and they also suggested extending the idea of Blum et al. (1961) to a $k \times k$ contingency tables, for $k > 2$. More recent methods related to the Hoeffding’s test have been proposed by Heller et al. (2013) and Kaufman et al. (2013). Both of these tests are proved to be consistent under general types of associations. Other methods for testing for independence include the distance correlation test of Szekely et al. (2007) and the maximal information coefficient of Reshef et al. (2011). Both the tests of Heller et al. (2013) and Szekely et al. (2007) can be extended to higher dimensions for testing joint independence of two or more random vectors. Several Bayesian methods are introduced for testing of independence. The simplest test of linear dependence between two univariate random variables can be achieved by fitting a linear model and inspecting the posterior distribution of the correlation coefficient. Other methods were proposed for testing of independence based on a contingency table (Nandram and Choi, 2006, 2007; Nandram et al., 2013).

In this chapter, we propose a nonparametric Bayesian test of independence between two groups of variables. We test the null hypothesis of independence and the alternative hypothesis of dependence. We specify nonparametric Bayesian models for the response density under both hypotheses. Under the null hypothesis, the joint distribution is taken to be the product of two independent densities, both with nonparametric priors; under the alternative, the full joint density has a nonparametric prior. The test is based on the posterior probability of the alternative hypothesis. By specifying nonparametric Bayesian models under each hypothesis, we obtain an extremely flexible test which can capture both linear and complex nonlinear relationships between groups of variables.

The remainder of the chapter proceeds as follows. In Chapter 3.2, we introduce the statistical algorithm. The details of the reversible jump MCMC algorithm use to compute the posterior probability of the alternative hypothesis are provided in Chapter 3.3. In Chapter 3.4, we present a simulation study to compare the power of the proposed test with other tests of linear and nonlinear relationships. The method is illustrated using a genetic data analysis in Chapter 3.5. Chapter 3.6 concludes.
3.2 Statistical model

Let $X_1 \in \mathbb{R}^{D_1}$ and $X_2 \in \mathbb{R}^{D_2}$ be random vectors in $D_1$ and $D_2$ dimensions, respectively, and denote $X = (X_1, X_2)$. The objective is to test whether $X_1$ and $X_2$ are independent. The hypotheses are

$$H_0 : \text{ } X_1 \text{ and } X_2 \text{ are independent and } f(X) = f_1(X_1)f_2(X_2)$$
$$H_1 : \text{ } X_1 \text{ and } X_2 \text{ are dependent and } f(X) \text{ cannot be factorized}$$

In other words, when they are independent, the joint density can be factorized as the product of two lower-dimensional densities.

Under both hypotheses, the densities are modeled using Dirichlet process mixture (DPM) priors as introduced in Chapter 2. Under $H_0$, $f_1(X_1)$ and $f_2(X_2)$ follow independent DPM priors; under $H_1$ when $X_1$ and $X_2$ are not independent, the joint distribution is assumed to follow a DPM prior. The following subchapters describe the independent and joint DPM priors.

3.2.1 The independent DPM prior

When $X_1$ and $X_2$ are independent, $f_j(X_j), \ j = 1, 2$, are assumed to follow the DPM prior independently. The DPM prior can be written as the infinite mixture

$$f_j(X_j) = \sum_{l=1}^{\infty} w_{lj} \phi_j(X_j \mid \mu_{lj}, \Sigma_j), \quad (3.1)$$

where $w_{lj}$ is the mixture weight, $\phi_j$ is assigned to be the $D_j$-dimensional multivariate normal distribution (MVN) in this analysis, $\mu_{lj}$ is the mean vector of the $l$th mixture component, and $\Sigma_j$ is the covariance matrix.

The mixture weights $w_{lj}$ are modeled by the stick-breaking construction with concentration parameter $d_j$. The weights $w_{lj}$ are modeled in terms of latent $v_{lj} \sim \text{Beta}(1, d_j)$. The first weight is $w_{1j} = v_{1j}$. The remaining elements are modeled as $w_{lj} = v_{lj} \prod_{i=1}^{l-1} (1 - v_{ij})$, where $\prod_{i=1}^{l-1} (1 - v_{ij}) = 1 - \sum_{i=1}^{l-1} w_{ij}$ is the remaining probability after accounting first $l-1$ mixture weights. The number of mixture components is truncated by a sufficiently large number $K$ (i.e. $l = 1, ..., K$), where the last term $v_K$ is fixed to be 1 to ensure that $\sum_{l=1}^{K} w_{lj} = 1$. 
The mean vectors $\mu_{lj}$ have priors $\mu_{lj} \sim \text{MVN}(0, \Omega_j)$. The covariance matrices $\Sigma_j$ and $\Omega_j$ are parameterized as $\Sigma_j = rS_j$, and $\Omega_j = (1 - r)S_j$. Under this model, $S_j$ is the covariance matrix for $X_j$ marginally over the mixture means $\mu_{lj}$, and $r$ is the proportion of the total variance attributed to the variance within each mixture component. The marginal covariance $S_j$ is assigned to have inverse Wishart prior distribution, and to facilitate computing, the prior of $r$ is a discrete uniform distribution with support $r \in \{0, 0.01, \ldots, 1\}$. The concentration parameter $d_j$ has prior distribution $\text{Gamma}(a, b)$.

### 3.2.2 The joint DPM prior

When $X_1$ and $X_2$ are not independent, $f(X)$ is assumed to follow the joint DPM prior

$$f(X) = \sum_{l=1}^{\infty} w_l \phi(X \mid \mu_l, \Sigma),$$

(3.2)

where $w_l$ is the mixture weight, $\phi$ is the $(D_1 + D_2)$-dimensional MVN distribution, $\mu_l$ is the mean vector of the $l^{th}$ mixture component, and $\Sigma$ is the covariance matrix. The number of mixtures is truncated by the same number $K$ as in the independent model. The mixture weights $w_l$ are again modeled by the stick-breaking algorithm with concentration parameter $d$. The mean vectors $\mu_l$ have priors $\mu_l \sim \phi(0, \Omega)$. The covariance matrices $\Sigma$ and $\Omega$ are modeled as $\Sigma = r \cdot \text{diag}(S)$ and $\Omega = (1 - r)S$, where $S$ is the covariance matrix for $X$, and $\text{diag}(S)$ is the diagonal form of $S$. In other words, under the joint DPM prior, we assign non-diagonal structure for the $\Omega$, and diagonal structure for the $\Sigma$. The priors for $S$, $r$, and $d$ are the same as in the independent DPM prior.

### 3.2.3 Bayesian test of independence

The Bayesian hypothesis test of independence is based on the Bayes factor (BF)

$$BF = \frac{P(H_1 \mid X)/P(H_0 \mid X)}{P(H_1)/P(H_0)} = \frac{P(X \mid H_1)}{P(X \mid H_0)}.$$  

(3.3)

The null is rejected if $BF > T$, where $T$ is a threshold parameter. The threshold parameter $T$ can be chosen based on rules of thumb about the weight of evidence favoring $H_1$. For example, Kass and Raftery (1995) suggest that $BF = 10$ is a strong evidence for $H_1$. Alternatively, in the simulation study in Chapter 3.4, we select $T$ to control the Type I
error rate. In the analysis of genetic data in Chapter 3.5, multiple tests are performing simultaneously, therefore we select $T$ to control the Bayesian false discovery rate.

### 3.3 Computing details

Computing the Bayes factor requires computing the posterior probability of each hypothesis. This is accomplished using a reversible jump MCMC (RJMCNNC) algorithm as described below.

#### 3.3.1 Reparameterization and hyperparameters

The updating algorithm of the DPM prior is facilitated by introducing the clustering model as in Chapter 2. The mixture form in (3.1) can be written as

$$f_j(X_j \mid g_j = l) = \phi_j(X_j \mid \mu_{lj}, \Sigma_j),$$

which draws an auxiliary cluster label $g_j \in \{1, ..., K\}$ with $P(g_j = l) = w_{lj}$. Similarly, the model in (3.2) is equivalent to

$$f(X \mid g = l) = \phi(X \mid \mu_l, \Sigma),$$

with cluster label $g$ and $P(g = l) = w_l$. Under the clustering model, the full conditionals of all the parameters are conjugate.

In addition, we introduce model indicator parameter $M$, where

$$M \in \begin{cases} 
I & \text{if } X_1 \text{ and } X_2 \text{ are independent (H}_0 \text{ is true)} \\
J & \text{if } X_1 \text{ and } X_2 \text{ are not independent (H}_1 \text{ is true)}
\end{cases}.$$ 

Under each MCMC step, we propose a new indicator $M'$ in the Markov chain, and decide whether to accept the new status $M'$. The probability $P(H_1 \mid X)$ is then approximated by $\sum_{i=1}^{N} I(M^{(i)} = J)/N$, where $N$ is the number of MCMC samples and $M^{(i)}$ is the model status for the $i^{th}$ MCMC sample.

Throughout this chapter, we let the number of mixture components truncated at $K = 20$ and the hyperparameters in the stick-breaking procedure $(a, b)$ are fixed under different sample sizes $n$ as presented in Table 3.1.
Table 3.1: Hyperparameters \((a, b)\) under different sample sizes \(n\).

<table>
<thead>
<tr>
<th>(n)</th>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>200</td>
<td>1.0</td>
<td>4.0</td>
</tr>
<tr>
<td>300</td>
<td>1.0</td>
<td>4.5</td>
</tr>
<tr>
<td>500</td>
<td>0.8</td>
<td>4.6</td>
</tr>
</tbody>
</table>

3.3.2 Pseudo code for the DPM test of independence algorithm

Let \(\Theta_M\) denote the DPM parameters \((\Theta_M = \{\mu_1, ..., \mu_K, r, S_1, S_2, w_1, ..., w_K, d_1, d_2\}\) if \(M = I\), and \(\Theta_M = \{\mu_1, ..., \mu_K, r, S, w_1, ..., w_K, d\}\) if \(M = J\). The algorithm of the DPM test of independence is described as follows:

**Step 0:** Select initial values for \(M\) and \(\Theta_M\).

**Step 1:** Update \(\Theta_M\) given \(M\) using the Gibbs sampling.

**Step 2:** Update \(M\) given the parameters \(\Theta_M\).

**Step 2.1:** Generate proposed model status \(M'\) with \(P(M' = I) = P(M' = J) = 0.5\).

**Step 2.2:** If \(M = M'\), then so back to **Step 1**.

**Step 2.3:** If \(M = I\) and \(M' = J\), then propose \(\Theta_{M'}\) required for the joint DPM prior \((H_1)\).

**Step 2.4:** If \(M = J\) and \(M' = I\), then propose \(\Theta_{M'}\) required for the independent DPM prior \((H_0)\).

**Step 2.5:** Accept \(M'\) with probability \(\min\{1, \alpha(M, M')\}\).

**Step 3:** Back to **Step 1**.

The full conditionals requires for Step 1 are all standard and are given in Appendix B.1.1 for \(M = I\), and Appendix B.1.2 for \(M = J\). Details on the RJMCMC steps are provided below in Chapter 3.3.3.

3.3.3 Steps of the RJMCMC algorithm

The parameter spaces under the independent and the joint DPM priors are different, so moving between these two parameter spaces becomes a trans-dimensional problem. Reversible jump MCMC (RJMCMC) was first introduced by Green (1995), which can be thought of as a generalized Metropolis-Hastings algorithm for the trans-dimensional
updates.

Under the current model status \( M \), the propose model status \( M' \) is randomly assigned to be either \( I \) or \( J \) with acceptance probability \( \min\{1, \alpha(M, M')\} \), where

\[
\alpha(M, M') = \frac{l_{M'} \cdot \pi_{M'} \cdot q_{M' \mid M'} \cdot p_{M' \rightarrow M}}{l_M \cdot \pi_M \cdot q_{M' \mid M} \cdot p_{M \rightarrow M'}},
\]

(3.4)

where \( l_M \) and \( \pi_M \) are the likelihood function and the prior distribution under model \( M \), \( q_{M' \mid M} \) is the candidate distribution of the parameters when proposing for model \( M' \) under model \( M \), \( p_{M \rightarrow M'} \) is the probability of proposing \( M' \) conditional on the current status \( M \), and \( |J| \) is the Jacobian. As \( M' \) is randomly picked from \( \{I, J\} \), \( p_{M \rightarrow M'} \) and \( p_{M' \rightarrow M} \) are equal in the algorithm. Note that when \( M = M' \), it becomes the usual fixed-dimensional MCMC algorithm as \( \alpha(M, M') = 1 \); when \( M \neq M' \), the candidate distribution of the parameters \( q \) is then for balancing the parameter spaces between the independent and joint models.

Recall that \( \Theta_M \) and \( \Theta_M' \) denote the DPM parameters under models \( M \) and \( M' \), respectively, and the truncated number \( K \) under both models are assigned to be identical. We first examine the case when \( X_1 \) and \( X_2 \) are univariate random variables \( (D_1 = D_2 = 1) \) with the current model status \( M = I \), and the proposed model is \( M' = J \). Denote the covariance matrix under the joint model as \( S = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \). We assign the \( 2 \times K \) mean vector \( \mu \) to be the same in both the independent and joint DPM models. Also, we assign the variances \( S_{11}^2 \) and \( S_{22}^2 \), and \( r \) to be the same across different model statuses. Therefore, this move only requires proposing the parameters under the joint DPM prior in (3.2): the cluster label \( g_{M'}' \), \( \rho_{M'}' \), the concentration parameter \( d_{M'}' \), and the mixture weights \( w_{M'}' \). The concentration parameter \( d_I' \) is proposed by \( d_I' \sim \text{Gamma}(\bar{d}_I, 1) \), where \( \bar{d}_I \) is the mean of \( d_I \), and then the mixture probabilities \( w_I' \) is proposed from the stick-breaking procedure with concentration parameter \( d_I' \). The cluster label \( g_I' \) is proposed from the full conditional distribution given in Appendix B.1. The details of the mapping for each parameter is described in the end of this section.

Conversely, if the current model status is \( M = J \) and the proposed model status is \( M' = I \), the parameters of the independent model described in (3.1) are proposed as follow: The concentration parameter \( d_I' \sim \text{Gamma}(d_I, 1) \), and the mixture weights \( w_I' \) are again proposed by the stick-breaking procedure with concentration parameter \( d_I' \). The cluster label \( g_I' \) is again proposed by the full conditional distribution given in Appendix
B.1.

For dimension matching under the RJMCMC algorithm, the bijection map is described below for the case where $M = I$ and $M' = J$. The reverse move uses the same map. Let

\[ \theta_M = \{ \mu, S_{11}, S_{22}, r, w_I, d_I, g_I \} \]
\[ u = \{ \rho'_J, w'_J, d'_J, g'_J \} \]
\[ \theta_{M'} = \{ \mu, S_{11}, S_{22}, r, w_J, d_J, g_J, \rho_J \} \]
\[ u' = \{ w'_I, d'_I, g'_I \} \].

Then we assign $\Theta_M = \{ \theta_M, u \}$, $\Theta_{M'} = \{ \theta_{M'}, u' \}$. The bijection function $h$ has the form

\[ h(\Theta_M) = h(\theta_M, u) = \Theta_{M'} = \{ \theta_{M'}, u' \}, \]

which is a one-to-one bijection map with: $w_I \rightarrow w'_I$, $d_I \rightarrow d'_I$, $g_I \rightarrow g'_I$, $\rho'_J \rightarrow \rho_J$, $w'_J \rightarrow w_J$, $d'_J \rightarrow d_J$, and $g'_J \rightarrow g_J$. Hence, the Jacobian $|J| = \left| \frac{\partial(\theta_{M'}, u')}{\partial(\theta_M, u)} \right| = 1$.

When $D_1 + D_2 > 2$, the transition of the covariance matrices between the independent and joint models becomes more complicated as the off-diagonal elements are harder to propose than in the bivariate case. One way to alleviate this concern is to assume the covariance matrix $S$ under the joint model is a block-diagonal matrix $S = \begin{pmatrix} S_1 & 0 \\ 0 & S_2 \end{pmatrix}$, where $S_i$ is a $D_i \times D_i$ covariance matrix of $X_i$ for $i = 1, 2$. However, in the simulation study and the real data analysis of this chapter, we will focus on the case where $X_1$ and $X_2$ are univariate random variables.

### 3.4 Simulation Study

The simulation study focuses on testing for dependence between two univariate variables. The objective is to compare the power of each method under linear and nonlinear dependence. In the following subchapters, we introduce the data generation procedure, the competing methods, and the simulation results.
3.4.1 Data generation

The seven different types of data sets are described as follows. Scenarios 5 and 6 are designed from Kaufman et al. (2013).

1. Independent normal (Null): \( X_j \sim N(0, 1) \), for \( j=1,2 \).

2. Bivariate normal (BVN): \( (X_1, X_2) \sim \text{BVN} \left[ 0, \left( \begin{array}{c} 1 \\ \rho \\ 1 \end{array} \right) \right] \), where \( \rho = 0.2 \).

3. Horseshoe (HS): \( X_1 \sim N(0, 1), X_2 | X_1 \sim N(\rho X_1^2, 1) \), where \( \rho = 0.2 \).

4. Cone: \( X_1 \sim U(0, 1), X_2 | X_1 \sim N \left[ 0, (\rho X_1^2 + 0.1)^2 \right] \), where \( \rho = 0.1 \).

5. W: \( X_1 \sim \frac{1}{n} \sum_{i=1}^{n} U(a_i, a_i + \frac{1}{3}), X_2 | X_1 \sim U \left[ 3(X_1^2 - \frac{1}{2})^2, 3(1 + X_1^2 - \frac{1}{2}) \right] \), where \( a_1 = -1, n \) is the number of samples, and \( a_i = a_{i-1} + \frac{2}{n}, \) for \( i > 1 \).

6. Circle: \( (X_1, X_2) \sim \frac{1}{n} \sum_{i=1}^{n} \text{BVN} \left[ \theta_i, \left( \begin{array}{c} \frac{1}{2} \\ \frac{1}{2} \end{array} \right) \right] \), where \( \theta_i = [\sin(a_i\pi), \cos(a_i\pi)] \), and \( a_i \) is defined as in W.

Each scenario is generated with the algorithms introduced above with sample size \( n = 100, 200, \) and \( 500 \). Then for each dimension, we standardize the data to have mean zero and variance one. We plot the data when \( n = 200 \) in Figure 3.1 along with the true density. The responses are dependent for designs 2-6. Design 3-6 are all examples of the challenging dependent but uncorrelated and thus the usual test of correlation will miss this dependence.

3.4.2 Methods for testing of independence

We compare six methods in the simulation study. Each method is controlled to have type I error rate approximately equal to 0.05.

1. Linear regression (LR): The model \( X_2 = \beta_0 + \beta_1 X_1 + \epsilon, \ \epsilon \sim N(0, 1) \) is fitted by least squares and the linear association is determined by the test of \( \beta_1 = 0 \).

2. E-statistics (ES) (Szekely et al., 2007): The testing procedure is by calculating the distance covariance between \( X_1 \) and \( X_2 \). Details are given in Appendix B.2.

3. Heller-Heller-Gorfine method (HHG) (Heller et al., 2013): The test statistic is based on the sum of all likelihood ratio tests of \( 2 \times 2 \) contingency tables formed by the pairwise distances within each of \( X_1 \) and \( X_2 \). Details are given in Appendix B.3.
4. Data Derived Partitions method (DDP) (Kaufman et al., 2013) with $3 \times 3$ contingency tables: The DDP method is similar to the HHG method, but only designed for univariate random variables. The test statistic is based on the sum of all likelihood ratio tests of $3 \times 3$ contingency tables formed by the observed values. More details are given in Appendix B.4.

5. Maximal Information Coefficient method (MIC) (Reshef et al., 2011): It is a rank-order test statistic which is calculated from the largest achievable mutual information under different grid sizes. The details are given in Appendix B.5.
6. The DPM test of independence (DPM): The proposed test is described in Chapter 3.2. The $X$ is first marginally transformed to the normal scores, which transforms the data to the expected values of the order statistics of a same-size standard normal distribution. The normal score transformation is to make the proposed method becomes a distribution-free testing procedure. Therefore, the threshold for the BF in Chapter 3.2 can be determined by the permutations of the transformed data. The threshold $T$ for the Bayes factor is computed from 300 permutations of the sample.

### 3.4.3 Simulation results

The results are presented in Table 3.2 with sample sizes $n = 100, 200$ and $500$. The first three rows of the table are the type I error rate for each method under different sample sizes, which is controlled for all methods (Type I error rate is between 0.03 to 0.09). The following rows give the power of each method under different scenarios and sample sizes. It is clear that as the sample size $n$ increases, the powers increase for all the methods except the LR method under the HS, Cone, and Circle scenarios because of the nonlinear associations of these scenarios.

When the data are generated from bivariate normal distribution, the LR method has the highest power. This is expected because the LR method is theoretically the most powerful test under this scenario. The ES and DPM tests are the second best among other comparing tests.

The DPM test outperforms all other methods when data are generated from the HS and the W shapes. Under the Cone shape data, the HHG and the DPM tests both perform well. For the Circle design, the HHG, DDP, and DPM tests all have power greater than 0.9 starting from small sample sizes, and the ES and MIC have lower power.

In summary, the LR method is able to capture linear association but loses power in the nonlinear cases. The ES method is able to capture linear and nonlinear associations, but lose power in some of the nonlinear cases. The HHG and DDP methods both have high power in testing of nonlinear associations, but lose power in the linear association, especially the HHG method. The MIC method is a relatively conservative test compared to all other methods, and this problem is discussed by Heller et al. (2012). The proposed method not only shows the ability to capture the linear association, but is also powerful for detecting nonlinear associations in the simulation study.
Table 3.2: Power of each test (columns) for each simulation settings and sample size \( n \) (rows). A * indicates that the power is significantly different than the power of DPM test.

<table>
<thead>
<tr>
<th>Type</th>
<th>( n )</th>
<th>LR</th>
<th>ES</th>
<th>HHG</th>
<th>DDP</th>
<th>MIC</th>
<th>DPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>100</td>
<td>0.06</td>
<td>0.04</td>
<td>0.03</td>
<td>0.02</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.06</td>
<td>0.05</td>
<td>0.03</td>
<td>0.02</td>
<td>0.07</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.09</td>
<td>0.08</td>
<td>0.05</td>
<td>0.09</td>
<td>0.02</td>
<td>0.09</td>
</tr>
<tr>
<td>BVN</td>
<td>100</td>
<td>0.57*</td>
<td>0.49</td>
<td>0.24*</td>
<td>0.38*</td>
<td>0.13*</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.84*</td>
<td>0.78</td>
<td>0.37*</td>
<td>0.61*</td>
<td>0.25*</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.99</td>
<td>0.99</td>
<td>0.83*</td>
<td>0.99</td>
<td>0.40*</td>
<td>0.99</td>
</tr>
<tr>
<td>HS</td>
<td>100</td>
<td>0.11*</td>
<td>0.22</td>
<td>0.42</td>
<td>0.39</td>
<td>0.12*</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.05*</td>
<td>0.48</td>
<td>0.53*</td>
<td>0.60*</td>
<td>0.25*</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.10*</td>
<td>0.96</td>
<td>0.99</td>
<td>1.00</td>
<td>0.42*</td>
<td>1.00</td>
</tr>
<tr>
<td>Cone</td>
<td>100</td>
<td>0.03*</td>
<td>0.25</td>
<td>0.54*</td>
<td>0.33</td>
<td>0.17*</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.08*</td>
<td>0.56</td>
<td>0.87</td>
<td>0.73*</td>
<td>0.37*</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.09*</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.80*</td>
<td>1.00</td>
</tr>
<tr>
<td>W</td>
<td>100</td>
<td>0.54*</td>
<td>0.42</td>
<td>0.55*</td>
<td>0.75*</td>
<td>0.34*</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.84*</td>
<td>0.83*</td>
<td>0.92*</td>
<td>1.00</td>
<td>0.70*</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.98*</td>
<td>1.00</td>
</tr>
<tr>
<td>Circle</td>
<td>100</td>
<td>0.00*</td>
<td>0.00*</td>
<td>0.96</td>
<td>0.99</td>
<td>0.20*</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.00*</td>
<td>0.22*</td>
<td>1.00</td>
<td>1.00</td>
<td>0.37*</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.00*</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.95*</td>
<td>1.00</td>
</tr>
</tbody>
</table>

3.5 Real data analysis

We compare the six methods in the simulation study on the gene expression data set from Hughes et al. (2000). Studies of associations between genes can be found in de la Fuente et al. (2004) and Bhardwaj and Lu (2005). The number of observations is \( n = 300 \) for each gene, and we select 94 genes on chromosome 1 after removing samples with missing values. The objective is to test the pairwise associations within these 94 genes. A total of \( \binom{94}{2} = 4371 \) hypotheses tests of independence are performed. Because of the large number of tests, we control false discovery rate (FDR) at the 0.05 level rather than Type I error. The Bayesian FDR (BFDR) control procedure is applied (Efron and Tibshirani, 2002; Newton et al., 2004; Storey et al., 2004; Muller et al., 2006) for the DPM test, and the Benjamini–Hochberg procedure (Benjamini and Hochberg, 1995) is applied for the other methods.
The Cohen’s $\kappa$ statistic (Cohen, 1960) is used to measure agreement between tests. The $\kappa$ statistic is

$$\kappa = \frac{P_a - P_e}{1 - P_e},$$

where $P_a$ is the proportion of agreements between the two methods among the $N = 4371$ tests, and $P_e$ is the theoretical proportion of agreements under independence. Larger values of $\kappa$ represent more agreement between the tests. The number of rejections among $N = 4371$ tests and the $\kappa$ statistics of pairwise methods are presented in Table 3.3.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Number of rejections</th>
<th>LR</th>
<th>ES</th>
<th>HHG</th>
<th>DDP</th>
<th>MIC</th>
<th>DPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>2404</td>
<td>1.00</td>
<td>0.472</td>
<td>0.301</td>
<td>0.404</td>
<td>0.082</td>
<td>0.452</td>
</tr>
<tr>
<td>ES</td>
<td>3352</td>
<td>–</td>
<td>1.000</td>
<td>0.686</td>
<td>0.830</td>
<td>0.036</td>
<td>0.779</td>
</tr>
<tr>
<td>HHG</td>
<td>3442</td>
<td>–</td>
<td>–</td>
<td>1.000</td>
<td>0.751</td>
<td>0.032</td>
<td>0.720</td>
</tr>
<tr>
<td>DDP</td>
<td>3350</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.000</td>
<td>0.036</td>
<td>0.814</td>
</tr>
<tr>
<td>MIC</td>
<td>249</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.000</td>
<td>0.042</td>
</tr>
<tr>
<td>DPM</td>
<td>3231</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 3.3: Numbers of rejections (of the $N = 4371$ tests), and Cohen’s $\kappa$ statistics for each pair of methods.

The $\kappa$ statistics show that the ES, HHG, DDP, and the DPM tests have similar testing powers in this gene expression data sets, and the number of rejections among these tests are similar (3231 to 3442). The LR test only captures the linear associations between genes, and the MIC has the lowest power as in the simulation study.

In Figure 3.2, we plot six pairs of genes where there are disagreements among the tests. In the upper two plots (gene 94 versus gene 8, and gene 88 versus gene 15), the associations between these pairs of genes are detected by the DPM test, but not the other tests. The figure shows that between gene 94 and gene 8, there is a horseshoe pattern of dependence, and a nonlinear relationship between gene 88 and gene 15. In the middle two plots (gene 17 versus gene 1, and gene 89 versus gene 24), the ES, HHG, DDP, and DPM tests all flag associations between genes, but not the LR and the MIC tests. The figure shows that gene 17 and gene 1 have a cone-shape association, and genes 89 and 24 have a clustering relationship. The bottom two plots (gene 92 versus gene 2 and gene
30 versus gene 6) are the cases where only the LR, ES and DPM tests flag associations between genes. These three tests are powerful in testing the linear associations, and the figure shows linear relationships between genes in these two pairs.

Figure 3.2: Six pairs of genes where there are disagreements among the tests. The red lines are the linear fitted lines.

3.6 Conclusion

We propose a nonparametric Bayesian test of dependence by calculating the Bayes factor using the Dirichlet process mixture model and the reversible jump MCMC algorithm. We compare our method with the linear model, distance correlation method, HHG, DDP, and MIC in the simulation study and also in the gene expression data sets. The simulation results show that the proposed test is competitive in testing both linear
and nonlinear relationships.

In the gene expression data analysis, we performed 4371 multiple testing on the gene expression data in comparing pairwise genes. The proposed test shows similar performance with the distance correlation, DDP, and HHG methods, and detects some cases that other methods do not detect. It also shows that the proposed method is powerful on both linear and nonlinear relationships in the pairwise gene comparisons.
Chapter 4

Efficient multidimensional Bayesian density estimation using factored densities

4.1 Introduction

In this chapter, we propose a new method to estimate an unknown probability density function. While there are many classical methods that are efficient in low dimensions, the objective of this chapter is to estimate the density when the dimension is moderate or large. Some existing low-dimensional density estimation methods are kernel density estimation (Rosenblatt, 1956; Sheather and Jones, 1991; Jones et al., 1991; Seaman and Powell, 1996) or locally adaptive methods such as local likelihood method (Loader, 1996; Hjort and Jones, 1996) and adaptive kernel density estimator (Terrell and Scott, 1992; Botev et al., 2010). Other classical nonparametric density estimation methods are discussed in Izenman (1991). However, in high-dimensional problems, these methods have major difficulties due to the curse of dimensionality and bandwidth selection.

High-dimensional density estimation has many applications, includes exploring the structure of gene expression data, pattern recognition in pixel images, and storm tracking analysis. Several methods for high-dimensional density estimation implement dimension reduction procedures within the algorithm (Liu et al., 2007; Vadivel et al., 2011; Buchman et al., 2011; Moghaddam, 2002). Other methods for high-dimensional density estimation are such as mixture models (Moghaddam and Pentland, 1997), log-spline model (Stone,
In the nonparametric Bayesian setting, Dirichlet process priors (Ferguson, 1973; Antoniak, 1974) are a common approach. MacEachern (1994) and Escobar and West (1995) apply the Dirichlet process mixtures (DPM) prior in univariate density estimation problems, and a more flexible model which allows the number of mixtures to be unknown is by using the reversible jump MCMC algorithm under univariate case (Richardson and Green, 1997). Muller et al. (1996) extend the problem to multivariate case to estimate the joint density of \((y, x)\) using multivariate Dirichlet mixture models for solving regression problem of \(y \mid x\). Rodriguez et al. (2007) applied the nonparametric Bayesian density estimation on estimating multiple functional curves, and Canale and Dunson (2011) consider jointly model the density of mixed-scale data. Other applications of nonparametric Bayesian density estimation are studied in Muller and Quintana (2004) and Dunson (2010). Convergence rates of the DPM model under univariate density estimation are introduced in Ghosal and Van der Vaart (2001) and Ghosal and van der Vaart (2007). A more recent study presents a rate-adaptive density estimation method with multivariate Dirichlet mixtures (Shen et al., 2013), which guarantee the convergence rate is minimax optimal if the true density belongs to the smoothness class under certain conditions of the priors. However, when the dimension increases, the result shows that the convergence rates become slower due to the curse of dimensionality.

In this chapter, we propose a nonparametric Bayesian density estimation method for high-dimensional problems. The novelty of the proposed model is to factorize the high-dimensional joint distribution into the product of several independent lower-dimensional components, and then apply Dirichlet process mixtures for estimating these lower-dimensional densities. Since estimating several low-dimensional densities is easier than estimating a single high-dimensional density, we are able to alleviate the curse of dimensionality in cases where this low-dimensional structure exists.

The chapter proceeds as follows. In Chapter 4.2, we provide the statistical model, and computational details are given in Chapter 4.3. Two different simulation studies are given in Chapter 4.4 and 4.5 for lower and higher dimensional scenarios, respectively. An analysis of spatial density estimation for tropical storm tracks is conducted in Chapter 4.6, and the conclusion is in Chapter 4.7.
4.2 Density estimation under factorized components

Let $X \in \mathbb{R}^{D}$ denote a $D$-dimensional random vector with joint density $f(X)$. We assume $X$ can be factorized into an unknown number $C \leq D$ independent components. Let $m_j \in \{1,\ldots,C\}$ denote the component to which variable $j = 1,\ldots,D$ is assigned, and $Y_i = \{X_j \mid m_j = i\}$ as the collection of variables assigned to component $i$. The joint density then factors into the product of lower-dimensional densities

$$f(X \mid m) = \prod_{i=1}^{C} f_i(Y_i),$$

where $Y_i \in \mathbb{R}^{q_i}$, $i = 1,\ldots,C$, $\sum_{i=1}^{C} q_i = D$, and $m = (m_1,\ldots,m_D)$. After this factorization, the problem of estimating high-dimensional density $f$ is replaced by estimating several low-dimensional densities $f_i$, which is often a simpler problem.

The prior of $m$ is assigned to have equal probabilities among all possible factorizations. The density for each component is modeled by the DPM prior with clustering parameter $g_i$

$$f_i(Y_i \mid g_i = l) = \phi_i(Y_i \mid \mu_{li}, \Sigma_i),$$

(4.1)

where $P(g_i = l) = w_{li}$, $w_{li}$ is the mixture weight, $\phi_i$ is assigned to be the $q_i$-dimensional multivariate normal distribution (MVN), $\mu_{li}$ is the corresponding mean vector, and $\Sigma_i$ is the covariance matrix. The $w_{li}$ are modeled by the stick-breaking construction with concentration parameter $d_i$, as defined in Chapter 2. The mean vector $\mu_{li}$ has prior $\mu_{li} \sim \phi_i(0, \Omega_i)$, and the covariance matrices are modeled as $\Sigma_i = rS_i$ and $\Omega_i = (1-r)S_i$, where $r$ is the percentage of total variance explained within the clusters and $(1-r)$ is the percentage of total variance explained between the clusters.

Unlike the previous chapter, we assume $S$ is a diagonal covariance matrix. This is largely for computational convenience; early attempts with non-diagonal covariance showed poor MCMC convergence. The diagonal covariance model still provides an arbitrarily flexible model for the joint density, but may be inefficient in some cases, such as a density that is well-approximated by a multivariate normal density with strong corrections.

The number of mixture components for each density $f_i$ is truncated by a sufficiently large number $K$ (i.e. $l = 1,\ldots,K$), where $K$ is set to be 20 throughout the analysis. The $j^{th}$ diagonal element of $S_i$, $s_{ij}$, has prior InvGamma($a_1,b_1$). The percentage of total
variance $r$ has a uniform prior from 0 to 1. The mixture weights $w_i$ in this analysis is assumed to be identical for each model of $Y_i$, so that $w_i = w$. The concentration parameter $d$ has prior distribution $\text{Gamma}(a_2, b_2)$. The hyperparameters in the analysis are fixed with $(a_1, b_1) = (0.1, 0.1)$, and $(a_2, b_2) = (1.0, 4.5)$.

4.3 Computing details

4.3.1 Factorization by reversible jump MCMC

The factorization of $X$ is implemented by the reversible jump MCMC (RJMCMC) algorithm introduced by Green (1995), as transitioning between factorizations is a trans-dimensional problem. For each MCMC iteration, a new indicator $m'$ is proposed by a split or merge step, and the RJMCMC algorithm is used to update $m$. After $m$ is determined, the parameters in the DPM model for $f_i$ are then updated based on this factorization. The computing algorithms are provided in the following subchapters. The number of MCMC samples in this chapter is 50,000 with 20,000 burn-in, and the convergence is monitored using the 30,000 post-burn-in samples.

4.3.2 Pseudo code for the density estimation algorithm

Let $\Theta_m = \{\mu, r, S, w, d\}$ denote the DPM parameters under factorization $m$. The algorithm of the proposed density estimation model is described as follows:

Step 0: Select initial values for $m$ and $\Theta_m$.

Step 1: Update $\Theta_m$ given $m$ using Metropolis within Gibbs sampling.

Step 2: Update $m$ given the parameters $\Theta_m$.

Step 2.1: Propose factorization $m'$ as described in Chapter 4.3.3.

Step 2.2: If $m = m'$, then back to Step 1.

Step 2.3: If $m \neq m'$, then propose $\Theta_{m'}$ as described in Chapter 4.3.3.

Step 2.4: Accept $m'$ with probability $\min\{1, \alpha(m, m')\}$, where $\alpha(m, m')$ is defined in (4.2).

Step 3: Back to Step 1.

The full conditionals required for Step 1 are given in Appendix C.1, and the details of Step 2 are given in Chapter 4.3.3.
4.3.3 Details of the reversible jump MCMC step

For proposing a new factorizing indicator \( m' \), we use split and merge moves in the RJMCMC algorithm. The procedure is described below:

1. Randomly choose one dimension \( j \), where \( j \in \{1,...,D\} \).

2. Randomly choose one of the two moves: split or merge.
   - **Split**: and if \( C < D \), then set \( m'_j = C + 1 \), otherwise, set \( m'_j = m_j \).
   - **Merge**: and if \( C > 1 \), then set \( m'_j = r \), where \( r \in \{1,...,C\} \) but \( r \neq m_j \), otherwise \( m'_j = m_j \).

3. The proposed factorization is \( m' = (m_1,...,m_{j-1},m'_j,m_{j+1},...,m_D) \).

After proposing a new indicator \( m' \), the transition from the current factorization \( m \) to \( m' \) requires proposing new parameters. The idea is illustrated here by a 3-dimensional \( (D = 3) \) example. Let \( m = (m_1,m_2,m_3) = (1,1,2) \) indicate \( X = (Y_1,Y_2) \), which \( Y_1 = (X_1,X_2) \), and \( Y_2 = X_3 \), where \( X_i \) is the \( i \)th dimension of \( X \). Let the proposing \( m' \) be \( m' = (m'_1,m'_2,m'_3) = (1,3,2) \), which is a split move for \( Y_1 \), indicates the proposed factorization is \( X = (Y'_1,Y'_2,Y'_3) \), where \( Y'_1 = X_1, Y'_2 = X_2 \), and \( Y'_3 = X_3 \). The transition between \( m \) and \( m' \) is now the transition between \( Y_1 \) and \( \{Y'_1,Y'_2\} \). Following by (4.1), the model under factorization \( m \) has the form

\[
\begin{align*}
    f_1(Y_1 \mid g_1 = l) &= \phi_1(Y_1 \mid \mu_{l1}, \Sigma_1), \\
    f_2(Y_2 \mid g_2 = l) &= \phi_2(Y_2 \mid \mu_{l2}, \Sigma_2),
\end{align*}
\]

where \( \mu_{lr} \sim \phi_r(0,\Omega_r) \), and \( P(g_r = l) = w_l \), with \( r = 1,2 \). And the model under \( m' \) has the form

\[
\begin{align*}
    f'_1(Y'_1 \mid g'_1 = l) &= \phi'_1(Y'_1 \mid \mu'_{l1}, \Sigma'_1), \\
    f'_2(Y'_2 \mid g'_2 = l) &= \phi'_2(Y'_2 \mid \mu'_{l2}, \Sigma'_2), \\
    f'_3(Y'_3 \mid g'_3 = l) &= \phi'_3(Y'_3 \mid \mu'_{l3}, \Sigma'_3),
\end{align*}
\]

where \( \mu'_{lr} \sim \phi'_r(0,\Omega'_r) \), and \( P(g'_r = l) = w_l \), \( r = 1,2,3 \). First, the model for \( Y'_3 \) is the same with the model of \( Y_2 \). Also, we assign \( \mu_{l1} = (\mu'_{l1}, \mu'_{l2}) \), \( \Sigma_1 = (\Sigma'_1 0 0 \Sigma'_2) \), and...
\[ \Omega_1 = \begin{pmatrix} \Omega_1' & 0 \\ 0 & \Omega_2 \end{pmatrix} \]. The cluster labels \( g'_1 \) are proposed from the full-conditional distribution.

For the merge move, again assuming \( m = (m_1, m_2, m_3) = (1, 1, 2) \), but now with \( m' = (m'_1, m'_2, m'_3) = (1, 1, 1) \), which indicates a merge of \( Y_1 \) and \( Y_2 \). The model for \( Y'_1 = X \) has the form

\[
 f'_1(Y'_1 | g'_1 = l) = \phi'_1(Y'_1 | \mu'_l, \Sigma'_l),
\]

where \( \mu'_l \sim \phi'_1(0, \Omega'_1) \), and \( P(g'_1 = l) = w_l \). Similarly, we assign \( (\mu_{l1}, \mu_{l2}) = \mu'_l \), \( (\Sigma_1, 0) = \Sigma'_1 \), and \( (\Omega_1, 0) = \Omega'_1 \). The cluster label \( g'_1 \) is again proposed from the full-conditional distribution.

After proposing the parameters for factorization \( m' \), the acceptance probability of \( m' \) is \( \min\{1, \alpha(m, m')\} \), where

\[
 \alpha(m, m') = \frac{l_m \cdot \pi_m \cdot q_{m'} | m' \cdot p_{m' \rightarrow m}}{l'_m \cdot \pi'_m \cdot q_m | m \cdot p_{m \rightarrow m'}} |J|, 
\]

where \( l_m \) and \( \pi_m \) are the likelihood function and the prior distribution under factorization \( m \), \( q_{m'} | m \) is the candidate distribution of the parameters when proposing for factorization \( m \) under factorization \( m' \), \( p_{m' \rightarrow m} \) is the probability of proposing \( m' \) when the current indicator is \( m \), and \( |J| \) is the Jacobian formula for change-of-variable.

By the split-merge move introduced earlier in this chapter, the probabilities \( p_{m \rightarrow m'} \) and \( p_{m' \rightarrow m} \) are equal in the analysis.

### 4.4 Simulation study of low-dimensional data

The objective of this simulation study is to compare four different density estimation algorithms for different dimensions of \( X \) and different levels of factorization.

#### 4.4.1 Data Generation

We compare four different models under three different dimensions with four different levels of factorization. The data are generated as follow:

1. \( D = 2 \): \( X_1 \sim N(0, 1), X_2 | X_1 \sim N(bX_1^2, 1) \).
2. \( D = 3 \): The first and the second dimensions are generated as in the \( D = 2 \) case, and \( X_3 \sim N(0, 1) \) is independent of \( X_1 \) and \( X_2 \).

3. \( D = 4 \): The first two dimensions and the last two dimensions are generated independently, and each of those are following the same generation in \( D = 2 \) case.

In each case, the sample size is \( n = 200 \), and the strength of dependence, which is controlled by \( b \), is varied from \( b \in \{0, 0.3, 0.5, 0.8\} \). For each scenario we generate 100 data sets.

### 4.4.2 Methods for comparisons

We compared four different methods in this simulation study: the independent DPM model (I), which estimates the density of each dimension of separately \( X \) using the univariate DPM model. The joint DPM model (J), which estimates the density of \( X \) using the \( D \)-dimensional DPM model. The joint kernel density estimation (KDE) (Hastie et al., 2001), which estimates the density \( f(X) \) jointly by the formula

\[
\hat{f}(X) = \frac{1}{n} \sum_{i=1}^{n} K_H(X - X_i),
\]

where \( X_i \) is the \( i^{th} \) observation of \( X \), and \( K_H \) is the specified kernel function with bandwidth matrix \( H \). In this chapter, \( K_H \) is the Gaussian kernel and bandwidth matrix \( H \) is the corresponding covariance matrix. The KDE method is implemented in \( R \) by the function \textit{kde} in \textit{library(ks)}. The bandwidth matrix \( H \) is selected using the method introduced by Wand and Jones (1994). The fourth method is the proposed method (RJ).

The Kullback-Leibler distance (KLD) (Kullback and Leibler, 1951) and the Hellinger distance (HD) (Hellinger 1909) are calculated to measure the distances from each estimated density function \( \hat{f} \) to the true density function \( f_0 \). Both of the measures are integrals, which are approximated by a sum of \( n_g \) grid points \( x_i \), i.e.,

\[
\text{KLD}_{h,t} = \sum_{i=1}^{n_g} \left\{ \ln \left( \frac{f_0(x_i)}{\hat{f}(x_i)} \right) \right\} f_0(x_i),
\]

\[
\text{HD}_{\hat{f},f_0} = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{n_g} \left[ \sqrt{f_0(x_i)} - \sqrt{\hat{f}(x_i)} \right]^2}.
\]
For comparison, all densities (\( \hat{f} \) and \( f_0 \)) are renormalized to sum to one over the grid points. We use \( 30 \times 30 \) grid points for \( D = 2 \), and 10,000 randomly generated grid points for \( D > 2 \) scenarios for calculating the KLD and the HD. The posterior probability of the factorizing indicator equaling to the true factorization is also calculated under each scenario.

4.4.3 Simulation results

The results of 100 simulated data sets are given in Figures 4.8–4.10, with dimensions \( D = 2, 3 \) and 4. The KLD and the HD results are similar, so we only present the KLD. For \( D = 2 \) and \( b = 0 \) (Figure 4.8), the two responses are independent normals, and all four methods perform similarly. As the strength of dependence \( b \) increases, the independent DPM model (I) starts to have larger KLD values, and the other methods all have smaller values indicating that in 2-dimensions, the Joint DPM and proposed methods perform well.

Under \( D = 3 \) and 4 (Figure 4.9 and 4.10, respectively), the proposed method outperforms both the joint DPM model and KDE, especially as \( b \) increases. This indicates that when there are more factorized components and the associations within the components increase, the proposed model is able to capture the factorization and gives more accurate density estimates than the other methods.

4.5 Simulation study of high-dimensional data

4.5.1 Simulation study of high-dimensional data with paired associations

We first extend the simulation study to \( D = 10 \) and 15 with paired associations. In \( D = 10 \), we assign five pairs of dimensions to have joint structures. In \( D = 15 \), we assign seven pairs of dimensions have joint structures and the remaining one is generated independently. For all scenarios, each dependent pair has the same joint distribution as the \( D = 2 \) case in Chapter 4.4. The level of association is fixed at \( b = 0.8 \). For each scenario, 100 data sets are generated with sample size \( n = 300 \). The proposed model is compared only with the independent (I) and joint (J) DPM methods, as the KDE method cannot be implemented with dimensions this large.
The results are in Figures 4.1a and 4.1b. The proposed model has significantly smaller KLD than the independent (I) and joint (J) DPM methods under both 10 and 15-dimensional data sets. The average probabilities of the true factorizations over 100 data sets are 0.960 and 0.957, respectively for each scenario, which means that even in high dimensions, if the data can be factorized into several independent low-dimensional structures, the proposed model has an advantage in estimating the density by factorizing the joint density.

4.5.2 Simulation study of high-dimensional data with multidimensional associations

Instead of the paired association, we examine the case where \( D = 10 \) with two groups of 5-dimensional joint structures: 
\[
X_1 \sim N(0, 1), \quad X_j \mid X_{j-1} \sim N(bX_{j-1}^2, 1), \quad \text{for } j = 2, \ldots, 5,
\]
and 
\[
X_6 \sim N(0, 1), \quad X_l \mid X_{l-1} \sim N(bX_{l-1}^2, 1), \quad \text{for } l = 7, \ldots, 10.
\]
We fixed \( b = 0.8 \), \( n = 300 \) and compare with the independent (I) and joint (J) DPM methods. The results are presented in Figure 4.1c. In this case when the factorized components have high-dimensional structures, the proposed method still outperforms the independent DPM method (I) but is not significantly different from the joint DPM method (J).

![Box plots](image)

(a) \( D = 10 \), paired associations  
(b) \( D = 15 \), paired associations  
(c) \( D = 10 \), 5-dimensional assoc.

Figure 4.1: KLD under high-dimensional data.

To explore this further, we examine four simulated data sets analyzed with the pro-
posed method. The factorization probabilities for the four simulated data sets are given in Figure 4.2. The proposed method still hints at the true factorization into two 5-dimensional densities, but the probability of the correct model is not as high as in the paired association in Chapter 4.5.1. The reason is that under this setting, the association between $X_1$ and $X_k$ decays while $k$ increases. For instance, $E(X_2 \mid X_1) = bX_1^2$, $E(X_3 \mid X_1) = b^3X_1^4 + b$, and $E(X_4 \mid X_1) = b^7X_1^8 + 6b^5X_1^4 + b^3 + b$. In other words, the relationship between $X_k$ and $X_1$ becomes weaker and more complicated when $k$ is large. As the model fails to identify the correct factorization, it is outperformed by the joint DPM model. This illustrates a limitation of the proposed method. Namely, the proposed method is inefficient when the true factorization includes a large group of variables with complex dependence structure.

(a) Example 1  
(b) Example 2  
(c) Example 3  
(d) Example 4

Figure 4.2: Factorization probability (that is, the posterior probability that each pair of variables are in the same cluster) plots for four simulated data sets.
4.6 Analysis of tropical storm tracks

To illustrate the proposed method, we studied tropical storm tracks in the Atlantic Ocean for the years 1981-2012. The HURDAT data are downloaded from the National Hurricane Center (http://www.nhc.noaa.gov/data/hurdat) and plotted in Figures 4.3 and 4.4. For each storm, we extract the initial (ilong, ilat) and final (flong, flat) spatial locations, as well as the year of the storm. Our objective was to estimate the spatial distribution of tropical storms, and the evolution of the spatial distribution over time. By including year in this way, we test for climate change effect in a very flexible way, without assuming normality or linearity in the statistical model.

Figure 4.3: Tracks of 79 tropical storm tracks after year 2005.

We first applied the DPM test of dependence introduced in Chapter 2 for each pair of variables. The posterior probabilities of pairwise dependence are given in Table 4.1.
Figure 4.4: Pairwise scatter plots for the tropical storm track data.

The results show that the pairs (iLat, iLong), (iLong, fLong), (fLong, fLong), and (iLat, fLong) are associated, which agrees with the scatter plots in Figure 4.4.

The factorization probability plot calculated by the proposed model is presented in Figure 4.5. Year has very low probabilities of being grouped with the spatial variables. We therefore conclude the data provide little evidence of a change in track distribution over the course of the study period. Except for the initial longitude and final latitude, other pairs all present some joint structures, especially (iLat, iLong) and (iLong, fLong). The factorization probability is concordant with the DPM test that the pairs with high probabilities in the DPM tests are tend to be factorized together in the model.

For measuring the performance of the density estimation, we first implement the 10-fold cross validation. The deviance of the testing samples for the methods compared in the simulation study in Chapter 4.5 are presented in Table 4.2. The proposed method has smaller deviance than both the independent and joint DPM methods, and the differences
Table 4.1: Posterior probabilities of pairwise associations.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Year</th>
<th>iLat</th>
<th>iLong</th>
<th>fLat</th>
<th>fLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>-</td>
<td>0.2398</td>
<td>0.3858</td>
<td>0.3038</td>
<td>0.4630</td>
</tr>
<tr>
<td>iLat</td>
<td>-</td>
<td>-</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9552</td>
</tr>
<tr>
<td>iLong</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.5348</td>
<td>1.0000</td>
</tr>
<tr>
<td>fLat</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.0000</td>
</tr>
<tr>
<td>fLong</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 4.5: Factorization probability plot of the 344 tropical storm tracks between methods are all statistically significant using a paired t-test.

We display the estimated density for several low-dimensional marginal and conditional distributions. For pairs (iLong, iLat) and (fLong, fLat), we calculate the estimated marginal densities $\hat{f}(iLong, iLat)$ and $\hat{f}(fLong, fLat)$ by the form

$$\hat{f}(iLong, iLat) = \frac{1}{G} \sum_{i=1}^{G} \hat{f}(year_i, iLong, iLat, fLong_i, fLat_i),$$

$$\hat{f}(fLong, fLat) = \frac{1}{G} \sum_{i=1}^{G} \hat{f}(year_i, iLong_i, iLat_i, fLong, fLat),$$
Table 4.2: 10-fold cross validation of deviances for the independent DPM (I), joint DPM (J), and the proposed RJ method.

<table>
<thead>
<tr>
<th>Methods</th>
<th>I</th>
<th>J</th>
<th>RJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>556.4</td>
<td>530.7</td>
<td>525.5</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>11.2</td>
<td>20.7</td>
<td>14.6</td>
</tr>
</tbody>
</table>

where \((\text{year}_i, \text{flong}_i, \text{flat}_i)\) and \((\text{year}_i, \text{ilong}_j, \text{ilat}_j)\) are randomly generated points. In this analysis, we generate \(G = 10\) points, and the results are plotted in Figure 4.6 with 20 \times 20 grids.

Figure 4.6: Estimated density function of initial (left) and final (right) spatial locations.

We also calculate the estimated conditional density function \(\hat{f}(\text{flong, flat} \mid \text{ilong}_j, \text{ilat}_j)\) in the same manner

\[
\hat{f}(\text{flong, flat} \mid \text{ilong}_j, \text{ilat}_j) = \frac{1}{G} \sum_{i=1}^{G} \hat{f}(\text{year}_i, \text{ilong}_j, \text{ilat}_j, \text{flong}, \text{flat}).
\]
The density of the final position given four initial locations are plotted in Figure 4.7. The highest estimated densities in all four examples are in the Northern Atlantic, which is also the highest marginal density in Figure 4.6. However, there is some variation in the density depending on the initial position. When the storm starts of the African Coast as in Example 1, the Southeastern US has higher estimated density. On the other hand, when the storm starts in the middle of the Atlantic Ocean as in Example 2, the density increases in the Midwest. In Example 3, the initial location is in the Caribbean which gives higher estimated density of the final location in the Southeast. In Example 4, the initial location is near Florida and there is also high density in the Southeast.

4.7 Conclusion

In this chapter, we propose an efficient Bayesian multidimensional density estimation method by factorizing the original data to several lower-dimensional components. The reversible jump MCMC method is implemented for factorization, and the DPM method is used to estimate each factorized component. In the simulation study where each factorized component is paired-associated, the proposed method outperforms the kernel density estimation, the independent DPM model, and the joint DPM model. We also examine the case where each factorized component has a 5-dimensional joint structure. In this scenario, the proposed method still outperforms the independent DPM model, but is just competitive with the joint DPM model.

In the real data analysis, we implement the proposed method to the tropical storm data with five variables. The results show that the factorization probability among the five variables is concordant to the DPM test of pairwise associations. This indicates that the factorization in the proposed method did gather the variables which are associated, and separate variables which are not. Cross validation shows that our model is preferred to both the independent and joint DPM methods.
Figure 4.7: Estimated conditional density function of the final location given the initial spatial location (denoted as a solid triangle).
Figure 4.8: 2-dimensional data with $b = 0, 0.3, 0.5, 0.8$. Panel (a) plots the KLD for the 100 simulated data sets, and Panel (b) plots the posterior probabilities of the true factorization under the RJ model for the 100 data sets.

Figure 4.9: 3-dimensional data with $b = 0, 0.3, 0.5, 0.8$. Panel (a) plots the KLD for the 100 simulated data sets, and Panel (b) plots the posterior probabilities of the true factorization under the RJ model for the 100 data sets.
Figure 4.10: 4-dimensional data with $b = 0, 0.3, 0.5, 0.8$. Panel (a) plots the KLD for the 100 simulated data sets, and Panel (b) plots the posterior probabilities of the true factorization under the RJ model for the 100 data sets.
Chapter 5

Future work

In Chapter 2, the DPM model is based on the first-order Markov assumption. Higher-order Markov model could also be applied, however the number of parameters in the model will increase geometrically. The proposed model also assumes the scale parameter $\sigma_i$ is identical among all clusters in class $i$, however, this assumption can be relaxed by letting each cluster $k$ have its own scale parameter $\sigma_{ik}$ in (2.13). The number of instructions $M$ in the real data analysis is 8, but $M$ can be greater than 50 in practice. Larger $M$ will likely result in slightly better performance, but there are computational limitations to the size of $M$ with this approach. Improving the computational efficiency such as parallel computing of the MCMC algorithm will help in applying this approach to larger $M$ and larger $K$. The proposed nonparametric Bayesian model can also be applied to other information-based analysis such as the static binary string analysis. Finally, to implement the method in real time may require faster computation. Sequential Monte Carlo method may prove useful for this application.

The proposed test in Chapter 3 is based on the DPM prior with the same covariance structures across the mixtures. The model can be extended to more complex structures such as different covariance matrices for each mixture. However, both extensions will increase the computing load may lead to slower MCMC convergence. The RJMCMC algorithm can be improved by constructing a more efficient proposal distributions for the mean vectors and the covariance matrices. Also, future work includes studying the theoretical properties of the proposed test. Given the current literature on convergence of DPM models, we make the conjecture that the test is consistent, meaning as the sample size increases the posterior probability of the correct hypothesis tends to be one.
The proposed density estimation method in Chapter 4 can be improved in several aspects. First, the single mixture weights vector $w$ and proportion of the variance scalar $r$ can all be extended to vary across the $C$ components, and constructing more efficient way for proposing the mean vector $\mu$ while transition between factorizations. Second, the diagonal covariance matrix $S$ can be extended to non-diagonal structure. It is likely that these changes would improve the fit for large problems with complex densities. However, both improvements will increase the computing load and the complexity of the model. When the dimensions of the data becomes large, the convergence of the parameters can be another concern for these changes. Third, the split and merge move in the reversible jump algorithm is restricted to choose one dimension at a time for each MCMC step. A modification such as proposing a random factorization along with the split and merge move will allow the RJMCMC algorithm to become more flexible and efficient.
REFERENCES


A. Jiang, X. Huang, Z. Zhang, J. Li, Z. Zhang, and H. Hua. Mutual information algo-


Appendix A

Supplementary materials for malware detection using nonparametric Bayesian clustering and classification techniques

A.1 Derivation of Matrix Multivariate Pólya (MMP) distribution

The derivation of the MMP distribution is following by the derivation of the Multivariate Pólya distribution (Madsen et al., 2005; Minka, 2003). Let $Z$ be the transition matrix and $P$ be the corresponding transition probability matrix with $M$ instructions and cluster label $g$. The distribution of $r^{th}$ row of $Z$ has the form

$$f(Z_r | P_r) = \prod_{c=1}^{M} P_{rc}^{Z_{rc}}.$$

And the distribution of $r^{th}$ row of $P$ is

$$P_r | \lambda, \sigma, g = k \sim \text{Dirichlet}(\sigma \lambda_{kr}),$$

72
where $\lambda$ and $\sigma$ are defined in the Chapter 1. We derive $f(Z_r | \lambda, \sigma, g)$ by

$$
f(Z_r | \lambda, \sigma, g) = \int_{P_r} f(Z_r | P_r) \cdot f(P_r | \lambda, \sigma, g = k) dP_r
$$

$$
= \int_{P_r} \left( \prod_{c=1}^{M} P_{Z_{rc}}^{Z_{rc}} \right) \frac{\Gamma(\sum_{c=1}^{M} \sigma_{krc})}{\prod_{c=1}^{M} \Gamma(\sigma_{krc})} \prod_{c=1}^{M} P_{Z_{rc}}^{\sigma_{krc} - 1} dP_r
$$

$$
= \frac{\Gamma(\sum_{c=1}^{M} \sigma_{krc})}{\prod_{c=1}^{M} \Gamma(\sigma_{krc})} \int_{P_r} \prod_{c=1}^{M} P_{Z_{rc} + \sigma_{krc} - 1}^{Z_{rc}} dP_r
$$

$$
= \frac{\Gamma(\sum_{c=1}^{M} \sigma_{krc})}{\prod_{c=1}^{M} \Gamma(\sigma_{krc})} \frac{\Gamma(Z_{rc} + \Lambda_{krc})}{\Gamma(\Lambda_{krc})}
$$

where (A.1) to (A.2) is by seeing that $P_{Z_{rc} + \sigma_{krc} - 1}$ is an unnormalized version of Dirichlet distribution. By assuming row-independence, we have

$$
f(Z | \lambda, \sigma, g = k) = \prod_{r=1}^{M} \left\{ \frac{\Gamma(A_{kr})}{\Gamma(N_{sr} + A_{kr})} \prod_{c=1}^{M} \frac{\Gamma(Z_{rc} + \Lambda_{krc})}{\Gamma(\Lambda_{krc})} \right\},
$$

which is the probability density function of MMP.

### A.2 DPM model setting details

The prior mean matrix $\tilde{P}_i$ ($i = 0, 1$) for each benign and malicious model are using the sample proportions $\tilde{P}_{irc} = \frac{\sum_{s=1}^{n_h} Z_{src} I(\xi_s = i)}{\sum_{s=1}^{n_h} \sum_{c=1}^{M} Z_{src} I(\xi_s = i)}$. Priors of $\sigma_i$ and $\gamma_i$ ($i = 0, 1$) are $\sigma_i, \gamma_i \sim \text{Gamma}(0.01, 0.01)$. Also, the prior probability of being malicious $\psi = 0.5$ for both simulation and real data analysis. The parameters in the proposed model are updated by the Metropolis-Hastings algorithm, in Chapter 1, there are 5,000 burn-in for choosing the proposal density and another 20,000 burn-in for a total of 30,000 MCMC samples.

The truncation number $(K_0, K_1)$ of the simulation study is set to be (10, 10) as the true number of clusters is known to be 5. The tuning parameters $(a, b)$ of the prior distribution of $\alpha_i$, ($i = 0, 1$) in the simulation study is set to be (1.7, 1.1), with results of a 95% confidence interval of the number of clusters between [1, 10]. For the real data analysis,
under \((K_0, K_1) = (150, 200), (a_0, b_0) = (1.2, 12)\) and \((a_1, b_1) = (1.2, 23)\) with results of a 95% confidence interval of the number of clusters between [3, 72] for \(K_0\), and [3, 151] for \(K_1\).

### A.3 Approximation of the posterior probability of being a malware

\[
P(\xi_s = 1 | Z_s^*, Z) = \frac{f(Z_s^* | \xi_s = 1, Z)}{f(Z_s^* | \xi_s = 1, Z) + f(Z_s^* | \xi_s = 0, Z)}
\]

\[
\approx \frac{A_1 + A_2}{\sum_{l=1}^{M} f(Z_s^* | \xi_s = 1, Z, \Theta_1^{(l)})} \sum_{l=1}^{M} f(Z_s^* | \xi_s = 0, Z, \Theta_0^{(l)}) = U_{s^*},
\]

where \(A_1 = \int f(Z_s^* | \xi_s = 1, Z, \Theta_1) \cdot f(\Theta_1 | Z) \, d\Theta_1\) and \(A_2 = \int f(Z_s^* | \xi_s = 0, Z, \Theta_0) \cdot f(\Theta_0 | Z) \, d\Theta_0\).

### A.4 ENL model for classification

First, we assume

\[
H[P(\xi_s = 1 | \beta)] = \varrho_s = \sum_{r=1}^{M} \sum_{c=1}^{M-1} P_{src} \beta_{rc},
\]

(A.3)

where \(H\) is the logistic link function. The log-likelihood function of \(\beta\) has the form

\[
l(\beta | \xi_s, P_s : s = 1, ..., n_h) = \sum_{s=1}^{n_h} \{\xi_s \ln[H^{-1}(\varrho_s)] + (1 - \xi_s) \ln[1 - H^{-1}(\varrho_s)]\},
\]

where \(P_s\) is unknown in practice but can be estimated from \(Z_s\) by \(\widehat{P_{src}} = \frac{\sum_{s=1}^{n_h} Z_{src}}{\sum_{s=1}^{n_h} \sum_{c=1}^{M} Z_{src}}\). Then the elastic net regularization combines the \(L1\) and \(L2\) penalties from LASSO (Tibshirani, 1996) and ridge methods (Hoerl and Kennard, 1970) in estimating \(\beta\).
Algorithm minimizes the penalized negative log-likelihood function below

\[ l_{EL}(\beta, \theta, \varphi) = -l(\beta \mid \xi_s, \hat{P}_s : s = 1, ..., n_h) + \varphi\left\{ \frac{1 - \theta}{2} \|\beta\|^2_2 + \theta \|\beta\|_1 \right\}, \]

where \( \varphi \) is the regularization parameter, and \( \theta \) is the elastic net mixing parameter: \( \theta = 1 \) is the lasso penalty, and \( \theta = 0 \) is the ridge penalty. Once we have the estimates of \( \hat{\beta} \), we then calculate the probability of being malicious. We use the function \( \text{cv.glmnet} \) in R to implement the algorithm. The tuning parameters \( \varphi \) and \( \theta \) are chosen simultaneously by grid-searching and 10-fold cross validation among 100 values each by AUC criterion. We also control the FDR in \( \alpha = 0.1 \) level to determine the threshold for classification. For simplicity, we control the FDR under testing samples directly for the ENL model in our study.

A.5 SVM for classification

Support vector machines (SVM) is a classifier that search for a hyperplane in the feature space, and separates the target samples to two classes with a maximal margin (Hastie et al., 2001). In this analysis, a soft-margin SVM (C-SVM) (Cortes and Vapnik, 1995) is used to compare with the ENL and the DPM model. Parameter \( C \) is a tuning parameter which allows some of the samples lying between the margins. The \( C \) is estimated through the grid-searching with the candidate values being 0.1, 1, 10, 100, 1000. The kernel function in the SVM plays the major role by projecting the data from the original space to a higher dimensional space, where a linear classification is achievable. The Gaussian kernel is applied in this analysis as suggested by the previous work (Anderson et al., 2011). The \( \text{ksvm} \) function in R is used, and the hyperparameter in the Gaussian kernel is tuning by the grid-searching under the values from 0.01 to 0.1.

A.6 Spectral clustering method for clustering

The spectral clustering algorithm is described as follow under malicious programs: For each observation, an adjacency matrix \( K \) is calculated by the Radial Basis Function (RBF) kernel, \( K_{ij} = \exp(-\kappa \cdot \|\hat{P}_i - \hat{P}_j\|^2_2) \), where \( \hat{P}_i \) is the \( i^{th} \) row of the estimated transition probability matrix \( \hat{P} \), and \( \kappa \) is a tuning parameter. Let \( D \) be a diagonal matrix with
\(d_{ii} = \sum_j K_{ij}\). The normalized graph Laplacian \(L_{\text{norm}}\) is calculated as \(L_{\text{norm}} = I - D^{-1}K\). \(L_{\text{norm}}\) now is a positive semi-definite matrix. Then we calculate the eigen decomposition of \(L_{\text{norm}}\), and extract the eigenvectors corresponding to the last \(b\) eigenvalues. K-means clustering method with squared Euclidean distance metric is then implemented on those vectors to cluster observations. In Chapter 1, the number of clusters in K-means clustering is set to be the true values \(K^*_1\). The number of eigenvalues \(b\) is varied from 1 to \(M-1\), and we let the free parameter \(\kappa\) in RBF kernel vary over a sequence of values from 0.01 to 1. Both parameters are chosen by grid search for the largest pairwise clustering accuracy.
Appendix B

Supplementary materials for nonparametric Bayesian test of independence

B.1 Full conditional distributions

B.1.1 Full conditionals for the independent DPM prior

Let $X_j = \{X_{ij} : i = 1, \ldots, N\}$, where $X_{ij}$ is the $i^{th}$ observation of $X_j$ and $N$ is the number of observations. The prior of $S_j$ is $S_j \sim IW_{D_j}(\rho_j, W_j)$. The full conditional
distribution for each parameters under the independent DPM prior of $X_j$ are

$$
\mu_{lj} \mid \text{rest} \sim \text{MVN}_{D_j}[(n_{lj}\Sigma_j^{-1} + \Omega_j^{-1})^{-1}\sum_{i:j_i=l}X_{ij}, (n_{lj}\Sigma_j^{-1} + \Omega_j^{-1})^{-1}]
$$

$$
S_j \mid \text{rest} \sim \text{IW}[N + K + \rho_j, A]
$$

$$
P(r = r_m \mid \text{rest}) = \frac{\prod_{i=1}^{n_r} \phi_j(X_{ij} \mid \mu_{g_{ij}}, r_m S_j) \prod_{l=1}^{K} \phi_j(\mu_{lj} \mid 0, (1 - r_m)S_j)^{1 \mid n_r}}{\sum_{l=1}^{K} \phi_j(X_{ij} \mid \mu_{g_{ij}}, r_q S_j) \prod_{l=1}^{K} \phi_j(\mu_{lj} \mid 0, (1 - r_q)S_j)^{1 \mid n_r}}
$$

$$
P(g_{ij} = l \mid \text{rest}) = \frac{\phi_j(X_{ij} \mid \mu_{lj}, \Sigma_j) w_{lj}}{\sum_{s=1}^{K} \phi_j(X_{ij} \mid \mu_{sj}, \Sigma_j) w_{sj}}
$$

$$
v_{lj} \mid \text{rest} \sim \text{Beta} \left[ \sum_{i=1}^{N} I(g_{ij} = l) + 1, \sum_{i=1}^{N} I(g_{ij} > l) + d_j \right]
$$

$$
d_j \mid \text{rest} \sim \text{Gamma} \left[ K + a - 1, b - \sum_{l=1}^{K-1} \log(1 - v_{lj}) \right],
$$

where $l = 1, ..., K$, $m = 1, ..., n_r$, $i = 1, ..., N$, $A = r^{-1} \sum_{i=1}^{N} (X_{ij} - \mu_{g_{ij}})(X_{ij} - \mu_{g_{ij}}) + (1 - r)^{-1} \sum_{l=1}^{K} \mu_{lj}\mu_{lj}^T + \rho_j W_j$, $n_{lj} = \sum_{i=1}^{N} I(g_{ij} = l)$, $g_{ij}$ is the cluster label of the $i^{th}$ observation, $n_r$ is the number of discrete $r$ values, $\phi_j$ is the $D_j$-dimensional multivariate normal density function, and $(a,b)$ is the tuning parameter of the stick-breaking algorithm.

### B.1.2 Full conditionals for the joint DPM prior

Let $X = \{X_i : i = 1, ..., N\}$, where $X_i$ is the $i^{th}$ observation of $X$ and $N$ is the number of observations. The prior of $S$ is $S \sim \text{IW}_{D_1 + D_2}({\rho, W})$. The full conditional distribution
for each parameters under the joint DPM prior of $X$ are

$$\mu_i \mid \text{rest} \sim \text{MVN}_{D_1+D_2}[(n_l\Sigma^{-1} + \Omega^{-1})^{-1}(\sum_{i:g_i=l} X_i), (n_l\Sigma^{-1} + \Omega^{-1})^{-1}]$$

$$S \mid \text{rest} \sim \text{IW}[N + K + \rho, B]$$

$$P(r = r_m \mid \text{rest}) = \frac{\prod_{i=1}^{n_r} \phi(X_i \mid \mu_{g_i}, r_m S) \prod_{l=1}^{K} \phi(\mu_l \mid 0, (1 - r_m)S)^{\frac{1}{n_r}}}{\sum_{q=1}^{n_r} \prod_{i=1}^{N} \phi(X_i \mid \mu_{g_i}, r_q S) \prod_{l=1}^{K} \phi(\mu_l \mid 0, (1 - r_q)S)^{\frac{1}{n_r}}}$$

$$P(g_{ij} = l \mid \text{rest}) = \frac{\phi_j(X_{ij} \mid \mu_{ij}, \Sigma_j)w_{ij}}{\sum_{s=1}^{K} \phi_j(X_{ij} \mid \mu_{sj}, \Sigma_j)w_{sj}}$$

$$v_l \mid \text{rest} \sim \text{Beta} \left[ \sum_{i=1}^{N} I(g_i = l) + 1, \sum_{i=1}^{N} I(g_i > l) + d \right]$$

$$d \mid \text{rest} \sim \text{Gamma} \left[ K + a - 1, b - \sum_{l=1}^{K-1} \log(1 - v_l) \right],$$

where $l = 1, ..., K$, $m = 1, ..., n_r$, $i = 1, ..., N$, $B = r^{-1} \sum_{i=1}^{N} (X_i - \mu_{g_i})(X_i - \mu_{g_i})^T + (1 - r)^{-1} \sum_{l=1}^{K} \mu_l\mu_l^T + \rho W$, $n_l = \sum_{i=1}^{N} I(g_i = l)$, $g_i$ is the cluster label of the $i^{th}$ observation, $n_r$ is the number of discrete $r$ values, $\phi$ is the $D$-dimensional multivariate normal density function, and $(a, b)$ is the tunning parameter of the stick-breaking algorithm.

### B.2 Test of independence by E-statistics

The test of independence by E-statistics, which calculates the distance covariance measures (dCov), was first introduced by Szekely et al. (2007). The dCov between two random variables (or vectors) $X_1 \in \mathbb{R}^p$ and $X_2 \in \mathbb{R}^q$ with finite first moments is the nonnegative number defined as

$$\gamma^2(X_1, X_2) = \|f(X) - f_1(X_1)f_2(X_2)\|_w^2,$$  \hspace{1cm} (B.1)

where $\| \cdot \|_w^2$ is the $L_2$-norm with weight function $w$. The $w$ is described more details in Szekely et al. (2007), and in Chapter 2, we use the identical $w$ as suggested. The empirical distance covariance of $n$ observed samples $\gamma^2_n(X_1, X_2)$ is also defined in Szekely et al. (2007). A test statistic $T(X_1, X_2, p, n)$ that rejects the null hypothesis that two random
variables (or vectors) if
\[
\frac{n\nu^2_n(X_1, X_2)}{S_2} > (\Phi^{-1}(1 - \alpha/2))^2
\]
has an asymptotic significance level at most \(\alpha\), and
\[
S_2 = \frac{1}{n^2} \sum_{k,l=1}^{n} |X_{1k} - X_{1l}|_p \frac{1}{n^2} \sum_{k,l=1}^{n} |X_{2k} - X_{2l}|_q,
\]
where \(|\cdot|_r\) is the \(L_r\)-norm, and \(\Phi\) is the standard normal distribution. However, the test decision based on \(\Phi\) is quite conservative for many distributions, so the testing decision in Chapter 2 is determined by 300 permutation samples under the null hypothesis with Type I error rate \(p = 0.05\) level under each data set. The R package "energy" with function "indep.test" is used in the analysis.

A distribution-free version of distance covariance was also introduced in (Szekely and Rizzo, 2009), which uses the ranks of the observations instead of the values. In Chapter 2, the distribution-free version of dCov performs similar to the original version, so we only present the original version of the dCov.

**B.3 Heller-Heller-Gorfine test of association based on Euclidean distance metric**

This test was first introduced by Heller et al. (2013). The test is based on the pairwise distances within \(X_1\) and \(X_2\) respectively. Let the pairwise distances within \(X_j\), \(j = 1, 2\), denoted as \(\{d(X_{ij}, X_{i'j}) : i, i' \in \{1, ..., n\}\}\), where \(X_{ij}\) is the \(i^{th}\) observation in \(X_j\), and \(d(\cdot, \cdot)\) is assigned to be the Euclidean distance metric in Chapter 2. The idea is to first randomly select two samples \(i\) and \(i'\) in each of \(X_1\) and \(X_2\), and then use the distances \(d(X_{i1}, X_{i'1})\) and \(d(X_{i2}, X_{i'2})\) as the references to construct a \(2 \times 2\) contingency table among the remaining \(n - 2\) samples. Then the likelihood ratio test of independence for summarizing this table denoted as \(S(i, i')\) gives test statistic
\[
T = \sum_{i=1}^{n} \sum_{i' = 1, i' \neq i}^{n} S(i, i').
\]
The 0.05 Type I error rate is controlled by 300 permutation samples under the null hypothesis of each data set. The R package ”HHG” with function ”hhg.test” is used in the analysis.

A distribution-free version of HHG test was suggested in (Heller et al., 2013) for comparison. We found that in Chapter 2 the results are similar to the original version of HHG test. Therefore, we only present the original version of the HHG results in Chapter 2.

B.4 Distribution-free tests of association based on data derived partitions

This test was first introduced in (Kaufman et al., 2013), which is designed for testing two univariate random variables (i.e. $D_1 = D_2 = 1$). The idea follows the HHG test but with different ways of forming the contingency tables. The data values are now used directly instead of using the distances. In forming a $2 \times 2$ contingency table, one sample point is randomly selected as the reference, and then a $2 \times 2$ contingency table can be constructed and a test statistic of this table is calculated. The same procedure can be applied to form $m \times m$ contingency tables ($m > 2$) with randomly selected $m - 1$ data values as references. More specifically, the $m \times m$ contingency table is defined by the range $(-\infty, X_{1(1)}^*), (X_{1(2)}^*, X_{1(3)}^*), ..., (X_{1(m-1)}^*, \infty)$ in $X_1$, and $(-\infty, X_{2(1)}^*), (X_{2(2)}^*, X_{2(3)}^*), ..., (X_{2(m-1)}^*, \infty)$ in $X_2$, where $X_{j(r)}^*$ is the $r^{th}$ ordered selected observation in $X_j$, $j = 1, 2$. In Chapter 2, the summation of the likelihood ratio test statistics with each $3 \times 3$ ($m = 3$) contingency table is used as the test statistics. This setting was shown to perform the best in most of the scenarios in (Kaufman et al., 2013). The testing decision is again based on 300 permutation samples under the null hypothesis for each data sets controlled under 0.05 Type I error rate in this study. The R package ”HHG” with function ”xdp.test” is used in Chapter 2.
B.5 Maximal information coefficient for measuring dependence of two variables

The Maximal Information Coefficient (MIC) method is first introduced by Reshef et al. (2011). The intuition is that if a relationship exists between two univariate random variables, then a grid (a square) can be drawn on the scatter-plot of these two variables which can partition the data to capture the relationship. The method explores all size of grids up to a maximal grid resolution. For grid size $x$-by-$y$, the largest achievable normalized mutual information (MI) is denoted as $m_{xy}$

$$m_{xy} = \max\{I_{xy}\}/\log(\min\{x,y\}),$$

where computation of $I_{xy}$ can be found in (Jiang et al., 2010), and the MIC is the maximum of $m_{xy}$ over all pair $(x, y)$ such that $xy < B$, where $B$ depends on the sample size $n$. In Chapter 2, we use $B = n^{0.6}$ as suggested in Reshef et al. (2011), and the p-value is calculated from the p-value table given in www.exploredat.net/Downloads/P-Value-Tables. The R package "minerva" with function "mine" is used in Chapter 2.
Appendix C

Supplementary materials for efficient Multidimensional Bayesian density estimation using factored densities

C.1 Full conditionals for the DPM model under single factorized component

Let $Y_i$ be the $i^{th}$ factorized component and $Y_s = \{Y_{is} : s = 1, ..., N\}$, where $Y_{is}$ is the $s^{th}$ observation of $Y_i$ and $N$ is the number of observations. Recall that $S_i$ is a diagonal matrix, and the prior of the $j^{th}$ diagonal element $S_{ij}$ is $S_{ij} \sim \text{InvGamma}(a_1, b_1)$. The concentration parameter of the stick-breaking algorithm $d$ has prior distribution $d \sim \text{Gamma}(a_2, b_2)$. The full conditional distribution for each parameters under the
DPM model of $Y_i$ are

$$
\mu_{li} \mid \text{rest} \sim \text{MVN}_{q_i}((n_{li}\Sigma_i^{-1} + \Omega_i^{-1})^{-1}\Sigma_i^{-1}(\sum_{s:g_{is}=l} Y_{is}), (n_{li}\Sigma_i^{-1} + \Omega_i^{-1})^{-1}]
$$

$$
S_{ij} \mid \text{rest} \sim \text{IG}\left\{a_1 + \frac{K}{2} + \frac{N}{2}, b_1 + \frac{1}{2}\left[\frac{1}{1-r}\sum_{l=1}^{K}(\mu_{lij})^2 + \frac{1}{r}\sum_{s=1}^{N}(Y_{ij}s - \mu_{g_{is}ij})^2\right]\right\}
$$

$$
P(g_{is} = l \mid \text{rest}) = \frac{\phi_i(Y_{is} \mid \mu_{li}, \Sigma_i)w_{li}}{\sum_{h=1}^{K}[\phi_i(Y_{is} \mid \mu_{hi}, \Sigma_i)w_{hi}]}
$$

$$
v_l \mid \text{rest} \sim \text{Beta}\left[\sum_{i=1}^{C}\sum_{s=1}^{N}I(g_{is} = l) + 1, \sum_{i=1}^{C}\sum_{s=1}^{N}I(g_{is} > l) + d\right]
$$

$$
d \mid \text{rest} \sim \text{Gamma}\left[K + a_2 - 1, b_2 - \sum_{l=1}^{K-1}\log(1 - v_l)\right],
$$

where $l = 1, \ldots, K$, $s = 1, \ldots, N$, $n_{li} = \sum_{s=1}^{N}I(g_{is} = l)$, $g_{is}$ is the cluster label of the $s^{th}$ observation, $\mu_{lij}$ is the $j^{th}$ element of $\mu_{li}$, $v_l$ is the latent variable of $w_l$ when using the stick-breaking algorithm, $\phi_i$ is the $q_i$-dimensional multivariate normal density function.

The parameter $r$ is not conjugate in this model, and the Metropolis-Hasting algorithm is applied. The proposed value $r'$ is proposed from $r' \sim \text{Beta}\left(\frac{1}{E}, \frac{1-r}{rE}\right)$, where $E$ is tuned to control the acceptance rate to be within 0.3 to 0.5. The acceptance probability $\alpha(r, r')$ is calculated as

$$
\alpha(r, r') = \frac{\prod_{i=1}^{C}\prod_{s=1}^{N} \phi_i(Y_{is} \mid \mu_{gis}, r'S_i) \prod_{i=1}^{C}\prod_{l=1}^{K} \phi_i[\mu_{li} \mid 0, (1-r'S_i)f_r(r')] \prod_{i=1}^{C}\prod_{s=1}^{N} \phi_i(Y_{is} \mid \mu_{gis}, rS_i) \prod_{i=1}^{C}\prod_{l=1}^{K} \phi_i[\mu_{li} \mid 0, (1-r'S_i)f_r(r')]}{\prod_{i=1}^{C}\prod_{s=1}^{N} \phi_i(Y_{is} \mid \mu_{gis}, rS_i) \prod_{i=1}^{C}\prod_{l=1}^{K} \phi_i[\mu_{li} \mid 0, (1-r'S_i)f_r(r')]},
$$

where $f_r$ is the prior distribution of $r$. 