On a Bayesian Approach to Malware Detection and Classification through $n$-gram Profiles

José A. Perusquía$^1$, Jim E. Griffin$^2$ and Cristiano Villa$^3$

School of Mathematics, Statistics and Actuarial Science, University of Kent$^1$
Department of Statistical Science, University College London$^2$
School of Mathematics, Statistics and Physics, Newcastle University$^3$

Abstract

Detecting and correctly classifying malicious executables has become one of the major concerns in cyber security, especially because traditional detection systems have become less effective with the increasing number and danger of threats found nowadays. One way to differentiate benign from malicious executables is to leverage on their hexadecimal representation by creating a set of binary features that completely characterise each executable. In this paper we present a novel supervised learning Bayesian nonparametric approach for binary matrices, that provides an effective probabilistic approach for malware detection. Moreover, and due to the model’s flexible assumptions, we are able to use it in a multi-class framework where the interest relies in classifying malware into known families. Finally, a generalisation of the model which provides a deeper understanding of the behaviour across groups for each feature is also developed.

Key words: Bayesian nonparametrics, compound random measures, cyber security, feature selection, supervised learning.

1 Introduction

Malware is a computational term that is commonly used to describe any software specifically designed to disrupt, damage or gain access to a computer system. Twenty years ago, it was
remarked by [McGraw and Morrisett 2000] that dealing with malicious code was a rapidly increasing problem affecting individuals, organisations and governments equally. Nowadays, in a highly connected world, the fast and accurate detection of malware is one of the major concerns of cyber security. Traditionally, the use of antivirus software has been essential in order to detect malicious code and to keep the computer systems protected. Antivirus software usually makes use of the blacklisting method, where a new program is scanned in search of signatures of known malware and if found, the program is disabled and a warning is flagged. This approach is effective for detecting known threats; however, nowadays there are several types of malware, many variations of known threats and new malicious code can be made with slight modifications of the original code in order to avoid recognition ([McGraw and Morrisett 2000]). As a consequence, antivirus software has proved to be less effective when dealing with new threats and hence, there is a need of more flexible detection methods.

In recent years, machine learning and statistical approaches have been used as an alternative to the blacklisting method and several approaches have been proposed in order to detect malicious code. These detection procedures can work by either directly analysing the executable content through $n$-gram profiles (see e.g. [Kolter and Maloof 2004, 2006; Masud et al., 2007; Pektaş et al., 2011]) or dynamic traces (see e.g. [Storlie et al., 2014; Bolton and Heard, 2018; Kao et al., 2015]), or by analysing the network traffic or the packet payload content (see e.g. [Ahmed and Lhee, 2011; Prasse et al., 2017; Vidal et al., 2017]). Deciding which methodology to use will certainly depend on the problem at hand. However, it seems that a content-based analysis yields a more general detection procedure, while a behavioural approach might be better for specific situations. For example, [Ahmed and Lhee, 2011] proposed a model for detecting executable code by analysing the packet payload; this is certainly useful for network servers that do not expect that kind of code such as shopping stores or online streaming services.

It is important to remark that malware detection is not the only task required when dealing with malicious software. In order to understand their infectious process, their potential threat level and therefore, how to be well-protected against these malicious softwares, there is a need to correctly identify the family to which they belong. The accurate classification may also speed-up the process of reverse-engineering to fix computer systems that were infected as well as for developing security patches to prevent more computers to become infected. This task can also be done through an $n$-gram profile analysis since most of the binary classifiers used for malware detection can be extended to a multi-class setting. That is why we are mainly interested in directly analysing the executables’ content through $n$-gram profiles. In this approach it is assumed that the $n$-grams (that depending on the field are defined as a contiguous sequence of $n$ letters, numbers or in this case bytes) work as binary features that completely characterise each class. To the best of our knowledge, malware detection through $n$-gram profile analyses has been mainly done using discriminative classifiers, like support vector machine, decision trees and boosted
versions of them. These models have proved to be extremely accurate while keeping a low false positive rate, especially when there is a large amount of data available. Moreover, these classifiers usually outperform probabilistic models like naive Bayes, that is why in practice the latter have not been considered at all. However, probabilistic models are extremely useful because they allow us to understand the generative process of the data (and with it the opportunity to replicate it) and they can also be used for tasks such as clustering (unsupervised learning).

In this paper we present a novel supervised learning model for binary matrices and a generalised version of it that allows a straightforward feature selection step. Our approach is based on the underlying theory of compound random measures (CoRM’s) [Griffin and Leisen 2016], which allows us to define a hierarchy across groups controlled by a beta global distribution and a beta score distribution at a group level. The beta-CoRM approach proposed here is a flexible probabilistic model for which a slice sampling method for the posterior and predictive inference is also derived. CoRM’s are particularly attractive since they allow us to construct correlated measures that characterise and differentiate each group. This is especially useful in situations where it is desired to find from a set of common features the ones that are most influential for each group. Moreover, these features are conditionally independent and under certain prior specifications, the inference can be performed independently on each feature and hence the computations can be easily parallelised. These are attractive characteristics for \( n \)-grams since they are usually assumed independent and the set of distinct \( n \)-grams is usually a high-dimensional object.

The remainder of the paper is organised as follows: in Section 2 we provide an overview on how \( n \)-grams have been used for malware detection purposes. In Section 3 we describe the model and the classification procedure used. In Section 4 we describe the generalised model and the feature selection step. In Sections 5 and 6 we respectively provide illustrations and results of the proposed models on synthetic and real malware data sets, and compare them against some of the most commonly used supervised learning models. Lastly, in Section 7 we provide conclusions and future work.

2 Related work

In computer science, the byte is the basic unit of information for storage and processing, and it is most commonly represented by a sequence of 8 binary digits or bits. Every instruction given to a computer can be broken down into sequences of bytes, which form the instruction’s binary code. These binary sequences can be expressed in a more condensed form using the hexadecimal notation. That is, each byte is represented as the combination of two elements of the set \{0, 1, ..., 9, A, B, ..., F\}. For malware detection purposes these sequences of bytes in
hexadecimal notation are used to create a set of binary features that are assumed to characterise both benign and malicious programs. In order to create these features, one commonly used structure are the \( n \)-grams which may represent an instruction, a part of one or more instructions or even a string data inside the code (Masud et al., 2007).

Using the hexadecimal representation of the binary code and \( n \)-grams is not the only approach used in order to create the binary features. There is more information from the executables that can be retrieved and used, for example, Schultz et al. (2001) used three different feature extraction processes. As explained in Kolter and Maloof (2004), the first method used a list of Dynamically Linked Libraries (DLLs), function calls from these DLLs and the number of different system calls from within these DLLs. The second approach used the UNIX string command, which provided the printable strings in an object or binary file. And the third method used the hexadecimal representation of the executable content. It was further discussed that we should question the stability of the DLL names, the function names and string features since they could be easily modified or there could not be information of them at all. As for the hexadecimal code, a thorough study of \( n \)-grams as binary features is due to Kolter and Maloof (2004) and since then it has been proved to be an effective approach and, as explained in Raff et al. (2016), it is particularly attractive since it can also be applied to other file formats like PDF’s. However, there is an important consideration that needs to be discussed first. No matter which size of \( n \)-grams is used, the feature space is generally very large, and even for a small collection of benign and malicious executables the number of unique \( n \)-grams can reach the order of hundreds of millions, making it computationally and statistically unfeasible to consider them all. Therefore, a selection procedure is required as a first step.

In order to address the high-dimensionality issue, Kolter and Maloof (2004) proposed to use the \( M \) most important \( n \)-grams in terms of the information gain (IG), which is a measure that has been used in machine learning for variable selection in text categorisation. In Yang and Pedersen (1997) a complete study on the IG and other common feature selection approaches in text categorisation was developed. As for the IG, it was showed that it favours common terms, it considers the feature absence in its calculation, and most importantly, for scenarios with an extreme dimensionality reduction (up to 98 % of the feature space) the classifiers performed the best with the IG features. This makes the IG particularly attractive for malware detection through \( n \)-gram profiles due to the presence of millions of them. The IG for the \( j \)-th \( n \)-gram is calculated as

\[
IG(j) = \sum_{v_j \in \{0,1\}} \sum_{C_i} P(v_j, C_i) \log \left( \frac{P(v_j, C_i)}{P(v_j)P(C_i)} \right),
\]

where \( v_j \) is the indicator of a program \( v \) having or not the \( j \)-th \( n \)-gram and \( C_i \) the \( i \)-th class. It can be easily appreciated that the IG as defined in (1) is actually the Kullback-Leibler divergence.
between the joint distribution $P(v_j, C)$ and the distribution assuming independence $P(v_j)P(C)$. For its computation Kolter and Maloof (2004) used the empirical distributions as an estimate for the required probabilities, i.e., $P(v_j, C_i)$ is the proportion of the observations in class $C_i$ that contain (or not) the $j$-th feature, $P(C_i)$ is the proportion of the data belonging to class $C_i$ and $P(v_j)$ is the proportion of the training data containing (or not) the $j$-the feature.

In practice, this feature extraction and selection approach requires the prior specification of the size of $n$-grams and the number of features to be considered. In order to know the best possible combination Kolter and Maloof (2004) performed pilot studies where they considered different values for $n$ and $M$. These combinations were tested in a subset of the data and compared through the performance of the classifiers used for the detection phase which included naive Bayes, decision trees, support vector machine and boosted versions of them. In these pilot studies it was also tested the number of bytes to be considered, and it was showed that the best results were obtained by using single bytes and setting $n = 4$ and $M = 500$. Since its introduction, most of the posterior research on malware detection through $n$-grams profiles has used this approach as the basis for the feature selection process. For example, some authors like Masud et al. (2007) have proposed hybrid detection models where the features are obtained using both $n$-grams and DLL, while others like Pektaş et al. (2011) have only used $n$-grams in their methodologies. As an example of the kind of data considered, Figure 1 illustrates a collection of 72 benign and 301 malicious executables for which 503 different $n$-grams have been selected. This data set can be found at the University of California Irvine Machine Learning repository (Rumao, 2016).

Although widely used, the approach described by Kolter and Maloof (2004) might not be the best one overall and further improvements could be made. For example, in the experiments performed by Raff et al. (2016), it was showed that the number of unique $n$-grams can reach the order of billions. As explained by the authors, working with these amount of features and calculating the IG for all of them is computationally expensive in both time and space. It was also discussed that determining the actual number of features used is also an expensive process that is not well addressed in Kolter and Maloof (2004) since they only used a subset of the data in order to fix its value. In order to address these issues Raff et al. (2016) proposed a three step feature selection process. The first step is to consider only the $n$-grams that appear in at least 1% of the observations. This step is justified by the fact that there is a need to select features that appear frequently enough to observe them in new data. By doing so the authors achieved to reduce the number of $n$-grams from 36 billions to 1.6 millions. The second step is to consider a measure like the IG in order to further reduce the number of features since having millions of them is still a computational burden. In their experiments they selected the 200 thousand most important ones. Finally, the third step is to use a learning model able to do feature selection at
As for this paper, its main contribution is a new Bayesian supervised learning model (and its generalised version) specifically designed for binary matrices. This probabilistic approach can be used in data sets with two classes, like the malware detection data set illustrated in Figure 1, or in a multi-class setting, like the one found in the Microsoft Malware Classification Challenge (Ronen et al., 2018). For this multi-class data set we further explore using only the 4-grams that appear at least once in each class, contrary to the IG gain related approaches. This first feature selection process drastically reduces the feature space and the computational resources required, and promising accuracy results are obtained with the beta-CoRM models and the other commonly used classifiers.
3 Discrete beta compound random measure

In this section, we present a novel Bayesian nonparametric approach to supervised learning that builds on a special type of \( d \)-dimensional vectors of completely random measures (CRM’s) (Kingman, 1967), known as compound random measures (CoRM’s) (Griffin and Leisen, 2016).

Since their introduction in Kingman (1967), CRM’s have become essential for most Bayesian nonparametric models. One of their key properties is their almost sure discreteness, which means that their realisations are discrete with probability 1. This characteristic allows us to use CRM’s to model data generated by a discrete distribution or to use them as the basic building block in mixture models. A useful representation of a CRM is due to Kingman (1967) where it is showed that if \( X \) is a CRM then,

\[
X = \sum_{i=1}^{\infty} J_i \delta_{x_i},
\]

where both the locations, \( x_i \)’s, and the jumps, \( J_i \)’s, are random. Finally, \( X \) is characterised by the Lévy measure \( \nu \) that contains all the information about the distribution of the \( x_i \)’s and the \( J_i \)’s. As for the CoRM’s, the basic idea is that if we consider a CRM as in (2) then we can define \( d \) correlated measures by perturbing the jumps, that is, if \( \mu_j \) represents the \( j \)-th random measure then,

\[
\mu_j = \sum_{i=1}^{\infty} m_{ji} J_i \delta_{x_i}, \quad m_{1i}, ..., m_{di} \sim h,
\]

where the \( m_{ji} \)’s are the perturbation coefficients that identify specific features on the \( j \)-th random measure and \( h \) is the score distribution. Therefore, CoRM’s are completely characterised through the distribution \( h \) and the directing Lévy measure \( \nu \) of the CRM. For a complete theoretical study on CoRM’s and some of their applications the reader can refer to Griffin and Leisen (2016, 2018).

3.1 Construction

From the broad description given above, it can be easily argued that compound random measures are particularly attractive for any kind of grouped data and hence, for supervised learning. In particular, for grouped binary matrices like Figure 1 where the main task is the detection of new malware. Since our interest relies on the probabilistic modelling of this kind of matrices, a natural approach would be to consider a beta-Bernoulli model. In a Bayesian nonparametric framework this can be achieved by choosing a beta process (BP) \( B \), as the directing CRM on a suitable space \( \Omega \). One of the main features of this stochastic process is that it concentrates the jumps in \((0, 1)\) and hence they can be used as the parameters for Bernoulli random variables. As
a CRM, the beta process is completely characterised by its Lévy measure given by

\[ \nu(d\omega, dp) = c(\omega)p^{-1}(1 - p)^{c(\omega) - 1}dpB_0(d\omega), \]

where \( c(\omega) \) is a concentration function and \( B_0 \) is a finite fixed measure on \( \Omega \). In practice, it is usual to consider \( c(\omega) = c \), so that \( c \) is a concentration parameter. As for \( B_0 \) this measure can be continuous, discrete or a mix of both types. In the Bayesian nonparametric literature the most common choice for this base measure is to be absolutely continuous. This choice is particularly useful for factorial models like the Indian buffet process (and related models) where the interest relies on the inference of the possible infinite number of unknown latent factors (see e.g. Griffiths and Ghahramani, 2005, 2011; Thibaux and Jordan, 2007).

For an \( n \)-gram profile analysis for malware detection a factorial model could also be used; however, in this paper we are interested in modelling the data directly. In order to do so, a discrete base measure on the set of unique \( n \)-grams might be a more appropriate choice. That is,

\[ B_0 = \sum_{i=1}^{\infty} q_i \delta_{\omega_i}, \tag{3} \]

where the set of \( \omega_i \)'s just work as labels in order to distinguish the \( n \)-grams, contrary to factorial models where each \( \omega_i \) usually represents a latent factor that needs to be inferred. This is a particularly interesting approach since the beta process will share the same atoms as \( B_0 \) with corresponding jumps \( p_i \) sampled from a beta distribution \((cq_i, c(1 - q_i))\) and hence, \( B \) has the following representation

\[ B = \sum_{i=1}^{\infty} p_i \delta_{\omega_i}. \tag{4} \]

A beta process with discrete base measure (3) is going to be the directing CRM used for the discrete beta-CoRM model we develop in this paper. By doing so, the set of jumps \( p_i \) can be thought as the probability that an executable regardless of the class has the corresponding \( n \)-gram, and for each of the \( d \) correlated groups these weights are perturbed by the scores \( m_{ji} \). The perturbed coefficients \( m_{ji}p_i \) represent the probability of observing each \( n \)-gram for each of the \( d \) groups; therefore, we need to ensure that they remain in the interval \((0, 1)\), which can be done by selecting a score distribution defined on this interval, for example, a beta distribution. In this way, for the \( j \)-th group we have a marginal process given by

\[ B_j = \sum_{i=1}^{\infty} m_{ji}p_i \delta_{\omega_i} \quad m_{ji} \overset{\text{ind}}{\sim} \text{beta}(a, 1). \tag{5} \]

The choice of this beta \((a,1)\) distribution for the scores was first proposed in Griffin and Leisen (2016) where a slightly different version of a beta-CoRM was considered. In its original formulation, the marginal processes of the beta-CoRM were fixed to be beta processes and using the beta
(a,1) score distribution it was proved that the directing process was the sum of the original beta process and a compound Poisson process with beta-distributed jumps.

Finally, the generative process is fully described by assuming that each observation $X_{kj}$ in group $j$ follows a Bernoulli process with corresponding base measure $B_j$, so that,

$$X_{kj} = \sum_{i=1}^{\infty} x_{kji} \delta_{\omega_i}, \quad x_{kji} \sim \text{Ber}(m_{ji}p_i).$$  \hspace{1cm} (6)

We believe that this Bayesian nonparametric model is an interesting approach to malware detection and classification through an $n$-gram analysis due to several reasons. First of all, Bayesian nonparametric models have an inherent flexibility that allows us to control the actual number of features considered. For a beta process with a continuous base measure this is achieved through the directing Lévy measure since $\int \nu(d\omega, dp) = \infty$, which means that there are an infinite number of small jumps. For a discrete base measure this can be done by directly specifying for which locations $\omega_i$ their respective jumps $q_i$’s are non-zero. More specifically, a CoRM approach allows us to analyse each feature independently through the perturbation coefficients which is useful in situations where we expect to differentiate groups through differences found in some of the features.

3.2 Properties

Now that the model has been fully described it is important to analyse its properties in order to fully understand the generative process and the role of the hyperparameters in the learning process. For the directing discrete beta process we can obtain both the expectation and the variance, this will provide an insight into the role of the jumps $q_i$ and the concentration parameter $c$.

**Proposition 1.** Let $B$ be a beta process with discrete base measure as in (4) and (3) respectively. Then

1. $\mathbb{E}(B) = B_0$ and
2. $\text{Var}(B) = \frac{1}{c+1} \sum_{i=1}^{\infty} q_i (1 - q_i)$.

The proof of the Proposition 1 is straightforward and can be found in the Appendix A. The properties stated in Proposition 1 certainly provide important information about the parameters of the directing beta process and the role they have in the generative process. For instance, $q_i$ represents our prior knowledge on the global probabilities, and the concentration parameter $c$, controls the similarity between the $p_i$’s and the $q_i$’s. As $c \to \infty$, $\text{Var}(p_i) \to 0$ and hence, $p_i \overset{a.s.}{\to} q_i$, therefore, large values of $c$ should be used when there is a strong prior belief that the $q_i$’s are...
good estimates of the $p_i$'s. On the other hand, for values of $c$ close to 0 then $p_i$ will be either close to 1 or 0 with probabilities $q_i$ and $(1 - q_i)$ respectively.

For the correlated measures $B_j$'s, that characterise each group, useful properties can also be derived, like their expectation and their variance. Moreover, since the shared jump $p_i$ introduces dependence between the jump heights in each measure, the covariance and the correlation at each location $\omega_i$ can also be obtained. All this information is grouped in the following proposition, which proof can be found in the Appendix A.

**Proposition 2.** Let $B$ a beta process defined as in Proposition 1 and $B_j$ and $B_k$ denote the $j$-th and the $k$-th measure defined as in (5), then

1. $\mathbb{E}(B_j|B) = \frac{a}{a+1} B$ and hence $\mathbb{E}(B_j) = \frac{a}{a+1} B_0$.

2. For $i \neq l$ then $\text{Cov}(B_j(d\omega_i), B_j(d\omega_l)) = 0$.

3. For a fixed feature $\omega_i$

   $$ \text{Var}(B_j(d\omega_i)) = \left( \frac{aq_i}{a+2} \right) \left( \frac{(1 - q_i)(a+1)^2 + q_i(c+1)}{(c+1)(a+1)^2} \right) $$

   and for $j \neq k$,

   $$ \text{Cov}(B_j(d\omega_i), B_k(d\omega_i)) = \left( \frac{a}{a+1} \right)^2 \frac{q_i(1 - q_i)}{c+1} $$

   and

   $$ \text{Corr}(B_j(d\omega_i), B_k(d\omega_i)) = \frac{a(a+2)(1 - q_i)}{(a+1)^2(1 - q_i) + q_i(c+1)}. $$

From these properties we can immediately obtain the probability of an observation having the feature $\omega_i$, which does not depend on $c$ and is given by

$$ \mathbb{P}(x_{kji} = 1|a, q_i) = \mathbb{E}(B_j(d\omega_i)) = \frac{a}{a+1} q_i. $$

It is also interesting to notice that the covariance is the difference between the joint probability of two observations in different groups having the feature $\omega_i$ and the distribution assuming independence, that is, for the $n$-th and $m$-th observation in the $j$-th and $k$-th group respectively,

$$ \text{Cov}(B_j(d\omega_i), B_k(d\omega_i)) = \mathbb{P}(x_{nji} = 1, x_{mkj} = 1|a, c, q_i) - \mathbb{P}(x_{nji} = 1|a, q_i)\mathbb{P}(x_{mkj} = 1|a, q_i) $$

$$ = \left( \frac{a}{a+1} \right)^2 \left( \frac{c q_i^2 + q_i}{c+1} \right) - \left( \frac{a}{a+1} \right)^2 q_i^2. $$

The joint distribution can be further generalised in order to consider all groups by simply obtaining the $d$-th moment of a beta distribution with parameters $(cq_i, c(1 - q_i))$, yielding

$$ \mathbb{P} \left( \prod_{j=1}^d x_{kji} = 1|a, c, q_i \right) = \mathbb{E} \left( \prod_{j=1}^d B_j(d\omega_i) \right) = \left( \frac{a}{a+1} \right)^d \mathbb{E}(p_i^d) = \left( \frac{a}{a+1} \right)^d \prod_{j=0}^{d-1} \frac{cq_i + j}{c + j}. $$
A final remark on the expression for the covariance between the jump heights is the fact that \( a \) and \( c \) have an opposite effect on it. That is, for a fixed \( a \) the covariance decreases as \( c \) increases and for a fixed \( c \) the covariance decreases as \( a \) decreases.

### 3.3 Posterior inference

Due to the discrete nature of the model, the joint posterior distribution of the correlated random measures \( \{B_j\}_j \) and the directing beta process \( B \) is the product of the posterior distribution of the random variables associated to each atom, that is, \( p_i \) and the set of scores \( m_{1i}, ..., m_{di} \). Therefore, we can analyse the posterior density on each atom. So, if we consider the feature whose label is \( \omega_i \), the posterior density (up to proportionality) is given by

\[
 f(\{m_{ji}\}_j, p_i|\{x_{kji}\}_{kj}) \propto p_i^{c q_i + x_{..i} - 1}(1-p_i)^{c(1-q_i) - 1} \prod_{j=1}^{d} (m_{ji}^{x_{ji} + a - 1}(1-p_i m_{ji})^{n_j - x_{ji}}),
\]

where \( x_{..i} = \sum_{k=1}^{n_i} x_{kji} \) and \( x_{..i} = \sum_{j=1}^{d} x_{ji} \). From (7) we can immediately appreciate that due to the presence of the \( d \) terms \((1-m_{ji}p_i)\), the joint and the conditional distributions do not have a known form. Therefore, a Gibbs sampling algorithm cannot be applied directly to this distribution. One way to address this issue is with the introduction of a set of latent variables \( \{y_{kji}\} \) that allows us to define an artificial measure \( B_{kj} \) as the base measure for the \( k \)-th observation in the \( j \)-th group, that is,

\[
 B_{kj} = \sum_i y_{kji} p_i \delta_{\omega_i} \quad \text{and} \quad y_{kji} = 1(u_{kji} < m_{ji}) \sim \text{Bernoulli}(m_{ji})
\]

\[
 X_{kj} = \sum_i x_{kji} \delta_{\omega_i} \quad \text{and} \quad x_{kji} \sim \text{Ber}(y_{kji} p_i).
\]

This approach is based on the idea of slice sampling ([Damien et al., 1999](#)), where a set of latent variables that preserve the marginal distribution are introduced. Slice sampling schemes have become widely used in Bayesian nonparametric models since they yield efficient computational methods for the infinite dimensional objects that are at their core. The reader can refer to [Griffin and Holmes (2010)](#) and the references therein for an overview on the computational issues found in some nonparametric models and the approaches used to address them. As for the normalised completely random measures, the slice sampling technique is useful in order to introduce a random truncation point and hence, consider only a random finite number of jumps. In our case the slice sampling approach that we propose allows us to create these infinite activity measures \( \{B_{kj}\}_{k,j} \) that yield a suitable augmented likelihood from which we can recover the original one by integrating out the latent variables as stated in the following Lemma.

**Lemma 1.** The discrete beta-CoRM defined by equations (4), (5) and (6) is equivalent to the augmented model in (8).
In order to prove Lemma 1 it is sufficient to note that the likelihood for a specific observation \(x_{kji}\) is a mixture of a degenerate distribution and a Bernoulli distribution with parameter \(p_i\) with respective weights \((1 - m_{ji})\) and \(m_{ji}\). The complete details can be found in the Appendix A. Now, with this form of the likelihood, the complete augmented posterior (up to proportionality) is given by

\[
\prod_{j=1}^{d} \prod_{k=1}^{n_j} \left( \delta^0_{0j} (1-y_{kji}) p_i x_{kji} y_{kji} + (1-x_{kji}) (1-y_{kji}) \right) \prod_{j=1}^{d} \prod_{k=1}^{n_j} \left[ m_{ji}^{a-1} m_{ji}^{y_{kji}} (1-m_{ji})^{1-y_{kji}} \right] \times \left( p_i q_i^{1-1} (1-p_i)^{c(1-q_i)} - 1 \right). \tag{9}
\]

From (9) we can immediately notice that the conditional posterior distributions are

\[
p_i|\{x_{kji}\}, \{y_{kji}\} \sim \text{beta} \left( \sum_{j=1}^{d} \sum_{k=1}^{n_j} x_{kji} y_{kji} + cq_i, \sum_{j=1}^{d} \sum_{k=1}^{n_j} (1-x_{kji}) y_{kji} + c(1-q_i) \right)
\]

\[
m_{ji}|\{y_{kji}\} \sim \text{beta} \left( a + \sum_{k=1}^{n_j} y_{kji}, 1 + n_j - \sum_{k=1}^{n_j} y_{kji} \right)
\]

\[
y_{kji}|x_{kji}, m_{ji}, p_i \sim \begin{cases} 
\delta_1 & \text{if } x_{kji} = 1 \\
\text{Ber} \left( \frac{(1-p_i)m_{ji}}{1-p_i m_{ji}} \right) & \text{if } x_{kji} = 0
\end{cases} \quad k \in \{1, \ldots, n_j\}.
\]

With these conditional distributions a Gibbs sampling procedure can be used in order to obtain samples from the posterior distribution. Finally, once a new observation, \(Y\), is available the classification procedure consists on computing the posterior probability of having such observation for each of the possible groups and assigning the observation to the group that has the highest probability. That is, for all \(j\) we need to obtain \(P(X_{n_j+1,j} = Y|X)\), which is the product of the posterior probability on each atom \(\omega_i\) given by

\[
P(x_{n_j+1,j,i} = 1|X) = \int P(x_{n_j+1,j,i} = 1|\{m_{ji}\}_j, p_i) f(\{m_{ji}\}_j, p_i|X) \approx \frac{1}{T} \sum_{t=1}^{T} m_{ji}^{(t)} p_i^{(t)}. \tag{10}
\]

And by the same reasoning \(P(x_{n_j+1,j,i} = 0|X) \approx 1 - \frac{1}{T} \sum_{t=1}^{T} m_{ji}^{(t)} p_i^{(t)}\). Where \(m_{ji}^{(t)}\) and \(p_i^{(t)}\) are obtained through the Gibbs sampling procedure on the augmented model.

### 4 Beta-CoRM with feature selection

Now that the discrete beta-CoRM approach for grouped binary matrices has been fully described, in this section we present a straightforward generalisation of the model that provides a
deeper understanding of the features and their respective behaviour across groups. This generalised beta-CoRM model yields a natural feature selection procedure which will be particularly useful for situations where the feature space is a high-dimensional object and there is a need to learn which are the best discriminative features. This approach arises naturally by noting that the density of the beta\((a,1)\) score distribution can be written for small \(x_0\) as \(1 - w) f(x) + wg(x)\), where \(f(x)\) and \(g(x)\) are truncated beta distributions on \((0,x_0)\) and \((x_0,1)\), respectively. This representation mimics the form of a spike-and-slab prior (see e.g. [Mitchell and Beauchamp, 1988; Ishwaran and Rao, 2005]) with \(w\) the probability of “including” a variable and \(g(x)\) the slab distribution. As for \(g(x)\) its cumulative distribution function (c.d.f.) is

\[
G(x) = \frac{x^a - x_0^a}{1 - x_0^a}.
\]

Since \(a = \log(1 - w)/\log(x_0)\) as \(w \downarrow 0\), \(a \downarrow 0\) and so we are interested in the limit of the above c.d.f. as \(a \downarrow 0\), which, using L’Hôpital’s rule, is

\[
\lim_{a \downarrow 0} G(x) = \frac{\log(x_0) - \log(x)}{\log(x_0)},
\]

and the corresponding probability density function (p.d.f.) \(g(x)\) is

\[
g(x) = \frac{1}{\log(1/x_0)} \frac{1}{x}, \quad x > x_0.
\]

Therefore, we can understand the prior distribution as a spike-and-slab prior where \(a\) controls the size of the spike and, if \(a\) is close to zero, the p.d.f. of the slab is approximately \(g(x)\). In the beta-CoRM, the beta\((a,1)\) random variables moderate \(p_i\), and so, for small \(a\), the prior expects some of the products \(m_{ji}p_i\) to be close to zero with \(w\) controlling the proportion close to “zero”.

With this interpretation in mind, the generalised beta-CoRM is fully specified by giving to each feature \(\omega_i\) an individual score distribution beta\((a_i,1)\), that is, \(m_{ji} \sim \text{beta}(a_i,1)\). For this generalised model, the slice sampling approach developed in Section 3.3 is still valid, therefore, we only need to provide the details for the posterior inference on the \(a_i\)’s. In order to do so, we first notice that for the beta\((a_i,1)\) distribution a gamma\((\alpha,\beta)\) distribution on \(a_i\) yields a conjugate model, and hence the posterior of each \(a_i\) is gamma\((\alpha + d, \beta - \sum_j \log(m_{ji}))\). For the hyperparameters, we assign gamma vague priors which yield a conjugate model for \(\beta\), and for \(\alpha\) we suggest the adaptive random walk Metropolis Hasting (Atchadé and Rosenthal, 2005) on \(\phi = \log \alpha\). Under this transformation it can be seen that

\[
f(\phi|\{a_i\}, \beta) \propto \frac{\exp(\phi)^{0.001}}{\Gamma(\exp(\phi))^{M}} \exp \left[ -\exp(\phi) \left( 0.001 - \sum_{i=1}^{M} \log(\beta a_i) \right) \right],
\]

and since \(\phi \in \mathbb{R}\) we use a \(N(x, \sigma^2)\) as the proposal distribution and accept the move with probability \(\gamma(x_n, y_{n+1}) = \min \left\{ 1, \frac{\pi(y_{n+1})}{\pi(x_n)} \right\}\), where \(\pi\) is the target distribution (11).
Once posterior estimates of the $a_i$’s are obtained a feature selection procedure can be used. In order to know which features are the “best” ones we can recall from the spike-and-slab interpretation that small values of the parameters $a_i$’s should be preferred since larger values will make the scores across groups very similar among each other and hence their discriminative “power” will decrease. A natural approach to find the optimum threshold $T_{\text{opt}}$ would be to define a grid on $\Delta = (\min \{\hat{a}_i\}, \max \{\hat{a}_i\})$ and for different values $T \in \Delta$ perform the classification procedure using the features whose score $\hat{a}_i \leq T$ and set $T_{\text{opt}} = T$ for the $T$ that yields the best classification accuracy. In the examples considered in the following sections, this procedure was realised directly in the test set since we had access to the real groups. Otherwise, this procedure should be performed on a validation set.

5 Synthetic data

In this section we present and describe some important results of the proposed models on synthetic data, a complete version of these simulations including some relevant illustrations is provided in the Appendix B. In order to have a better understanding on its performance, we divided this section into two parts. For the first one, we considered a scenario where the data is divided into three groups with non-overlapping score distributions, and for the second one, we considered a more complex scenario where the data is comprised of five groups with some overlapping score distributions. The main objectives of this section are to analyse the posterior inference, the effect the hyperparameters have on the beta-CoRM models and to compare their performance against other commonly used supervised learning algorithms such as, naive Bayes, multinomial logistic model and decision trees with their adaptive, gradient and extreme gradient boosted versions.

5.1 Three non-overlapping groups

For this section we simulated a data set comprised of 100 observations divided into three groups with a balanced number of observations and non-overlapping score distributions. That is, the scores were sampled from uniform distributions on the intervals $(0.6, 1)$, $(0.4, 0.6)$ and $(0, 0.4)$. The global probabilities $p_i$ were also sampled from a uniform distribution with support on the interval $(0.3, 1)$. This ensured that there was a clear distinction among the groups.

The posterior inference was performed using the MCMC procedure described in Section 3 and consisted of 55,000 iterations. The first 10,000 were used as the burning period and a thinning of 15 was used in the remaining 45,000, yielding an effective sample size of 3,000 iterations. As for the hyperparameters, we fixed $q_i = \max_j \{\sum_{k=1}^{n_j} \delta_{\frac{x_{kji}}{n_j}} / n_j\} \forall i$, and we tried several values for
and $a$. This was due to the fact that, in this model the parameters are non-identifiable, which can be directly seen if we analyse the likelihood for a specific feature $\omega_i$ given by

$$\mathcal{L}(\{m_{ji}\}_j, p_i) = \prod_{j=1}^{d} (p_i m_{ji})^{x_{ji}} (1 - p_i m_{ji})^{n_j - x_{ji}},$$

(12)

where different values for the parameters will yield the same likelihood. That is why the posterior inference for these models heavily relies on the choice of the hyperparameters, especially on $c$ and $a$. However, it is also important to remark that although the $m_{ji}'s$ and the $p_i's$ are useful for building a structure for modelling the variability across groups and features, in this case we are not interested in their individual estimation since the classification task only relies on the joint distribution of the $X$'s that is induced.

In these pilot studies we found that for small values of $a$ there is a clear difference in the posterior scores across groups that vanishes as $a$ increases, and this effect is more pronounced when $c$ is also large. As for $c$, we already knew from Proposition 1 that it controls the closeness of the $\hat{p}_i$’s and $q_i$’s. A possible approach to find the best values for $a$ and $c$ is to run the model for different combinations and perform the classification procedure on the training/validation set or the test set (if applicable), and select the ones that have the best posterior performance. For the beta-CoRM we found that the best performance was achieved when both $a$ and $c$ were close to one. Furthermore, the size of $a$ (being small) was more important compared to the size of $c$.

Finally, for the generalised version we fixed $c = 1$ and used $\alpha = \beta = \sigma = 1$ as the initial values for the MCMC scheme with the same number of iterations as for the beta-CoRM. In this case, we found that there does not seem to be a big difference among the posterior estimates of the score parameters $\hat{a}_i$’s. Something that does not surprised us since the data was randomly generated. The best performance was found to be 97% with $T_{opt} = 1.392$ which yielded 143 features considered for the classification.

### 5.2 Five overlapping groups

In this section the data used was comprised of five groups, where the score distribution for each group overlapped with the previous and the next one. That is, the scores were sampled from uniform distributions on the intervals $(0.7, 1), (0.5, 0.8), (0.4, 0.6), (0.2, 0.5)$ and $(0, 0.3)$, the global probabilities were sampled from a uniform distribution on $(0.3, 1)$ and we tested the model’s performance in different kind of circumstances by having balanced and unbalanced number of observations and by introducing more observations and features.

In the balanced scenario, all the groups contained the same number of observations for both the training and the test set. We used 150, 200 and 250 observations and 100, 200 and 300 features
and compared the beta-CoRM ($a = 1$, $c = 1$), the generalised beta-CoRM ($c = 1$), naive Bayes, multinomial logistic model and decision trees with its boosted versions. The results obtained showed that the proposed models had the best performance, with the generalised approach having a slightly better classification accuracy for most cases.

Finally, for the unbalanced scenario the middle group had twenty observations for all the cases contrary to the other groups that had more as we increased the number of total observations. We started with 150 and increased to 200 and 250, with respective partitions (30, 40, 20, 30, 30), (45, 50, 20, 45, 40) and (55, 70, 20, 55, 50). We used and compared the same models as before and found that the best results were achieved with the beta-CoRM models, and with the generalised version having the best performance overall.

### 6 Malware detection and classification

In this section we present results for both the malware detection and classification into known families problems. It is important to remark that we used two different data sets obtained from different sources. However, both the detection and classification could be done at the same time if the data contained benign and family-identified malicious executables.

#### 6.1 Detection

For the malware detection task we used the data set found in the University of California Irvine Machine Learning repository by Rumao (2016) (Figure 1). This data set is originally comprised of 72 benign and 301 malicious executable programs and 531 features comprised of 503 different n-grams and 28 DDL features. For our purposes we only considered the 503 n-grams which were obtained according to the author, following the assumptions and procedures described by Kolter and Maloof (2004), so we believe that $n = 4$ although it is not directly specified. In order to do a classification analysis we split the binary matrix into a training and a test set. This procedure was done several times to analyse the impact on the classification performance when the number of observations in the training set decreased to the point where there were more observations in the test set. First, we (randomly) selected 90 percent of the data to be in the training set and continued to decrease this percentage by 10 points each time until we reached the scenario where 30% of the data was used as the training set and the remaining 70% as the test set. The performance of the beta-CoRM with parameters $a = c = 1$, the generalised beta-CoRM with $c = 1$ and the other classifiers can be compared in Table 1.
Table 1: Classification accuracy comparison (in %) of the beta-CoRM models against commonly used supervised learning algorithms for the malware data set illustrated in Figure 1, with a decreasing number of observations in the training set.

| Ratio   | beta-CoRM | Gbeta-CoRM | nBayes | Tree | ABoost | Gboost | XGBoost |
|---------|-----------|------------|--------|------|--------|--------|---------|
| 90-10   | 100.00    | 100.00     | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 80-20   | 98.67     | 100.00     | 76.00  | 98.67 | 98.67  | 98.67  | 98.67  |
| 70-30   | 99.11     | 100.00     | 77.68  | 99.11 | 99.11  | 99.11  | 99.11  |
| 60-40   | 99.33     | 100.00     | 79.19  | 97.99 | 99.33  | 99.33  | 99.33  |
| 50-50   | 98.93     | 99.47      | 80.75  | 98.93 | 99.47  | 98.40  | 99.47  |
| 40-60   | 99.11     | 99.55      | 80.36  | 98.66 | 99.55  | 98.21  | 99.11  |
| 30-70   | 99.23     | 99.62      | 80.84  | 97.32 | 99.62  | 99.23  | 98.47  |

Interesting conclusions can be obtained from these results. First of all, it is clear that the beta-CoRM models have an impressive performance for this data set. Moreover, we can immediately appreciate that the generalised beta-CoRM outperformed or at least had the same accuracy as the boosted algorithms which are specifically built to provide high accuracy results while keeping a low false positive rate. We can also appreciate how naive Bayes, which is the other probabilistic classifier, had a poor performance compared to the beta-CoRM models. This is due to the fact that naive Bayes classified all the observations as malware as soon as we reduced the number of observations in the training set. Finally, it is also worth mentioning that we also tried to use a multinomial logistic model, however, since in all the scenarios considered the training data contained several features with no variation it resulted in an error while running the multinom function in the nnet R package (Venables and Ripley, 2002).

6.2 Classification

The data we used for the classification task was released as part of the Microsoft Malware Classification Challenge (Ronen et al., 2018) hosted at Kaggle in 2015. The training data set is comprised of almost 11,000 malware representing a mix of nine different families. For each of these malicious executables the data provides the label representing the true family, a file with the hexadecimal representation of the binary code and a metadata manifest containing information extracted from the binary. The reader can refer to Ronen et al. (2018) for a complete description of the data including the names of the families, the number of observations in each one of them and their type.
Since one of the families only contains 42 observations we decided to work with a random sample of 842 malware (100 observations per group plus the 42), which were further split into a training set comprised of 590 observations and a test set of 292 elements. Following the procedure described in Section 2, we obtained the unique 4-grams that appeared at least once in each family, yielding a total of 826 4-grams. The graphical representation of the binary matrix can be seen in Figure 2. In this case, and contrary to the malware detection application, we can observe that there is not a clear pattern in some of the groups and there is also a large overlapping across several of them.

![Graphical representation of the data set](image)

Figure 2: Graphical representation of the data set comprised of 590 malware (rows) of 9 families separated with the solid horizontal lines and 826 4-grams (columns). The dots represent the features that appear in each executable.

As for the classification is concerned, we used the beta-CoRM model with parameters $a = c = 1$, the generalised beta-CoRM with parameter $c = 1$, the supervised learning algorithms considered in the previous section and the multinomial logistic model. Table 2 shows the classi-
fication performance for all these models.

Table 2: Classification accuracy comparison (in %) for the beta-CoRM models against commonly used supervised learning algorithms for the malware data set illustrated in Figure 2.

| beta-CoRM | Gbeta-CoRM | nBayes | Tree | ABoost | GBoost | XGBoost | ML |
|-----------|------------|--------|------|--------|--------|---------|----|
| 80.95     | 84.92      | 77.38  | 80.15| 94.05  | 91.67  | 91.27   | 84.92|

In this case the adaptive algorithms outperformed the beta-CoRM models proposed; nevertheless, promising results and interesting conclusions can be obtained. First of all, we can recall that we are only considering the 4-grams that appear at least once in each family. This drastic dimensionality reduction technique has been more than appropriate choice since an outstanding accuracy can be obtained with the boosted algorithms. This is certainly promising for malware-related tasks where a priori we can have billions of different binary features to consider. As for the proposed models, we can appreciate that (as expected) the generalised beta-CoRM had a better performance. In this direction it is interesting to analyse the posterior estimates of the score parameters (Figure 3a) and the classification accuracy for different thresholds (Figure 3b). From these results the optimum threshold was found to be 0.604, which yielded only 93 4-grams used for the classification (certainly a drastic reduction from the original 826 4-grams considered).

(a) Posterior mean of the score parameters.  
(b) Accuracy for different thresholds.

Figure 3: Generalised beta-CoRM posterior results on the malware data set illustrated in Figure 2.

It was without a doubt interesting to see that, contrary to the malware detection task, the beta-CoRM models were outperformed by the boosted algorithms. In order to had a better un-
derstanding on why this happened, we directed our attention to the misclassified observations under the generalised beta-CoRM model. In Figure 4 we present the test set split into correctly classified and misclassified. As previously thought, the model is able to capture the general structure of each group; however, the overlapping in these groups was more pronounced than before and the misclassified observations were assigned to a group with a more similar structure.

![Figure 4: Malware test set comprised of executables of 9 families separated by the solid horizontal lines and split into correctly classified (right) and misclassified observations (left).](image)

7 Conclusions

In this paper we have presented a novel Bayesian supervised learning model for grouped binary matrices known as beta-CoRM and a straightforward generalisation that further allowed us to incorporate a feature selection step in the learning process. These models are built up on the underlying theory of compound random measures that belong to a wider class of Bayesian nonparametric discrete priors. For both models and due to the posterior intractability, a slice sampling algorithm was designed that allowed a fast update of the variables, parameters and hyperparameters involved. These two models were compared against some commonly used supervised learning algorithms on synthetic and real malware data sets. For data with “well-defined” groups, the beta-CoRM models had the best classification performance. However, in the malware classification task they were outperformed by the boosted algorithms, due to the fact that there was a clear overlapping in some of the groups that made some observations to be classified in groups with a more similar pattern. Further interesting results in the classification
were obtained. First of all, we need to remark that we considered a drastic dimensionality reduction approach by using only the \( n \)-grams that appeared at least once in each group. This allowed us to save both computational time and space and for which promising results were obtained through the boosted algorithms. Finally, the generalised beta-CoRM approach further allowed us to have a deeper understanding of the importance of each of the features and as expected had a better performance than the original beta-CoRM.

What we have presented in this paper opens to some interesting further directions of research. First of all, there are more malicious executables that we could use and from which more \( n \)-grams could be obtained. This is an especially interesting direction now that the generalised beta-CoRM allows us to choose an optimum threshold and hence, extract the features with the better discriminative power. With this in mind, a not so restrictive first dimensionality reduction approach could be used on the feature space, since we might be leaving out \( n \)-grams that could potentially improve the classification accuracy. Moreover, further research about the optimum threshold also needs to be done. Finally, another possible alternative for a better classification accuracy would be the introduction of dependence across features at a group level, where factorial models might be an interesting approach to follow.

### Appendix A  Proofs

**Proof of Proposition 1.** Proving the properties established in Proposition 1. only requires to remark that \( p_i \) are beta distributed with parameters \( (cq_i, c(1-q_i)) \). Therefore \( \mathbb{E}(p_i) = q_i \), and using the monotone convergence theorem we get

\[
\mathbb{E}(B) = \mathbb{E}\left( \sum_{i=1}^{\infty} p_i \delta_{\omega_i} \right) = \sum_{i=1}^{\infty} \mathbb{E}(p_i) \delta_{\omega_i} = \sum_{i=1}^{\infty} q_i \delta_{\omega_i} = B_0.
\]

Following the same monotone convergence reasoning and using the fact that \( \mathbb{E}(p_i^2) = \frac{q_i(1-q_i)}{c+1} + q_i^2 \)

it can be shown that

\[
\mathbb{E}(B^2) = B_0^2 + \frac{1}{c+1} \sum_{i=1}^{\infty} q_i(1-q_i).
\]

From which the variance can be obtained directly.

**Proof of Proposition 2.** The first property is straightforward since it follows the same reasoning as in Proposition 1, and the second property is a direct consequence of the scores and jumps being
mutually independent. Now, for the variance,

\[
\text{Var}(B_j(d\omega_i)) = \mathbb{E}(B_j^2(d\omega_i)) - \mathbb{E}(B_j(d\omega_i))^2 = \mathbb{E}(m_{ji}^2 p_i^2) - \mathbb{E}(m_{ji} p_i)^2
\]

\[
= \left(\frac{a}{(a+1)^2(a+2)} + \frac{a^2}{(a+1)^2}\right) \left(\frac{q_i(1-q_i)}{c+1} + q_i^2\right) - \left(\frac{a}{a+1}\right)^2 q_i^2
\]

\[
= \left(\frac{a}{(a+1)^2(a+2)} + \frac{a^2}{(a+1)^2}\right) \left(\frac{q_i(1-q_i)}{c+1}\right) + \left(\frac{a}{(a+1)^2(a+2)}\right) q_i^2
\]

\[
= \left(\frac{aq_i}{a+2}\right) \left(\frac{(1-q_i)(a+1)^2 + q_i(c+1)}{(c+1)(a+1)^2}\right).
\]

From the previous expressions it can be clearly appreciated that for a fixed feature, the variance is the same across families, that is \(\text{Var}(B_j(d\omega_i)) = \text{Var}(B_k(d\omega_i))\). This will be useful in order to obtain the correlation. But first, for the covariance between \(B_j(d\omega_i)\) and \(B_k(d\omega_i)\) we have from the first property in this proposition that

\[
\mathbb{E}(B_j(d\omega_i)) = \left(\frac{a}{a+1}\right) B_0(d\omega_i) = \left(\frac{a}{a+1}\right) q_i = \mathbb{E}(B_k(d\omega_i)),
\]

and

\[
\mathbb{E}(B_j(d\omega_i)B_k(d\omega_i)) = \mathbb{E}(m_{ji}m_{ki}p_i^2) = \left(\frac{a}{a+1}\right)^2 \mathbb{E}(p_i^2) = \left(\frac{a}{a+1}\right)^2 \left(\frac{q_i(1-q_i)}{c+1} + q_i^2\right),
\]

therefore,

\[
\text{Cov}(B_j(d\omega_i), B_k(d\omega_i)) = \mathbb{E}(B_j(d\omega_i)B_k(d\omega_i)) - \mathbb{E}(B_j(d\omega_i))\mathbb{E}(B_k(d\omega_i))
\]

\[
= \left(\frac{a}{a+1}\right)^2 \left(\frac{q_i(1-q_i)}{c+1} + q_i^2\right) - \left(\frac{a}{a+1}\right)^2 q_i^2
\]

\[
= \left(\frac{a}{a+1}\right)^2 \left(\frac{q_i(1-q_i)}{c+1}\right).
\]

Hence, the correlation is given by

\[
\text{Corr}(B_j(d\omega_i), B_k(d\omega_i)) = \frac{\text{Cov}(B_j(d\omega_i), B_k(d\omega_i))}{\text{Var}(B_j(d\omega_i))}
\]

\[
= \left(\frac{a}{a+1}\right)^2 \left(\frac{q_i(1-q_i)}{c+1}\right) \left(\frac{a+2}{aq_i}\right) \left(\frac{(c+1)(a+1)^2}{(1-q_i)(a+1)^2 + q_i(c+1)}\right)
\]

\[
= \frac{a(a+2)}{(1-q_i)(a+1)^2 + q_i(c+1)}.
\]

\(\Box\)
Proof of Lemma 1. Let us consider first the augmented model. In this case it is straightforward to see that conditioned on \( y_{kji} = 0 \) we have that \( x_{kji} \overset{\text{a.s.}}{=} 0 \) and conditioned on \( y_{kji} = 1 \) we have that \( x_{kji} \sim \text{Ber}(p_i) \). With this in mind, the augmented likelihood is

\[
\prod_{j=1}^{d} \prod_{k=1}^{n_j} (\delta_0^{x_{kji}} (1-y_{kji})) \left( p_i^{x_{kji}} (1-p_i)^{(1-x_{kji})} \right)^{y_{kji}},
\]

and the posterior distribution is proportional with respect to the latent variables to

\[
\prod_{j=1}^{d} \prod_{k=1}^{n_j} (\delta_0^{x_{kji}} (1-y_{kji})) \left( p_i^{x_{kji}} (1-p_i)^{(1-x_{kji})} \right)^{y_{kji}} m_{ji}^{y_{kji}} (1 - m_{ji})^{(1-y_{kji})}.
\]

Integrating out the latent variables yields,

\[
\prod_{j=1}^{d} \prod_{k=1}^{n_j} \delta_0^{x_{kji}} (1 - m_{ji}) + m_{ji} p_i^{x_{kji}} (1 - p_i)^{(1-x_{kji})}.
\]

Therefore, we can appreciate that the marginal posterior is the product of the mixture of a degenerate distribution and a Bernoulli distribution with corresponding weights \((1 - m_{ji})\) and \(m_{ji}\). This expression at first sight does not resemble the posterior distribution for the original beta-CoRM model. However, it is sufficient to notice that from (13) we obtain

\[
\delta_0^{x_{kji}} (1 - m_{ji}) + m_{ji} p_i^{x_{kji}} (1 - p_i)^{(1-x_{kji})} = \begin{cases} 1 - m_{ji} + m_{ji} (1 - p_i) = 1 - m_{ji} p_i & \text{if } x_{kji} = 0 \\ m_{ji} p_i & \text{if } x_{kji} = 1. \end{cases}
\]

Hence, we recover the original posterior distribution. \(\Box\)

Appendix B  Pilot Studies

In this appendix we provide a more complete version of the pilot studies discussed in Section 5. The following scenarios were chosen in order to have a better understanding of the models’ posterior and classification performance and to compare them against other commonly used supervised learning classifiers such as naive Bayes, multinomial logistic model, and decision tree. For the decision tree we also considered the adaptive, gradient and extreme gradient boosted versions.

B.1 Three non-overlapping groups

In this scenario the data is comprised of 100 observations divided into three groups with non-overlapping score distributions and 33, 34 and 33 observations, respectively. The global probabilities \(p_i\) were sampled from a uniform distribution on the interval \((0.3, 1)\), and for each group
the set of scores were sampled from uniform distributions on the intervals $(0.6, 1)$, $(0.4, 0.6)$ and $(0, 0.4)$, respectively. Figure 5 is the graphical representation of the binary matrix, from which it is immediate to appreciate that there is a significant difference in the amount of features present in each group (dots).

Figure 5: Synthetic binary matrix comprised of 150 features (columns) and 100 observations (rows) divided into three well-defined groups separated by the horizontal solid lines. The dots represent the features present for each observation.

For the posterior inference, we used the MCMC scheme described on Section 3. 55,000 iterations were obtained for which the first 10,000 were used as the burning period and a thinning lag of 15 was used in the remaining 45,000 ones, yielding an effective sample size of 3,000 iterations. The $q_i$’s were fixed to be the maximum across groups of the proportions of the observations having the corresponding feature $\omega_i$, that is, $q_i = \max_j \{ \sum_{k=1}^{n_j} \delta_1^{(x_{kji})}/n_j \}$, and $c = a = 1$. Figure 5 illustrates the posterior estimates obtained from these simulations. The upper graph shows the estimates (dots) of the global probabilities (spikes), the first heat map is the graphical represen-
tation of the real scores \{m_{ji}\}, and the second heat map shows their posterior estimates, \{\hat{m}_{ji}\}.

Figure 6: Posterior inference results of the beta-CoRM model for the data illustrated in Figure 5. Upper graph: real probabilities (spikes) and posterior estimates (dots). Middle graph: real scores. Bottom graph: posterior estimates of the scores.

It can be appreciated that for \( c = a = 1 \) good estimates of the real parameters are obtained. However, it also important to notice that the parameters in this model are non-identifiable due to the presence of the product of the \( p_i \) and the \( m_{ji} \)'s in each term of the likelihood, given by

\[
\mathcal{L}(\{m_{ji}\}, p_i) = \prod_{j=1}^{d} (p_i m_{ji})^{x_{ij}} (1 - p_i m_{ji})^{n_j - x_{ij}}. \tag{14}
\]

That is why for this model the posterior inference heavily relies on the choice of \( c, a \) and the \( q_i \)'s (Figure 7). However, it is important to remark that although the \( m_{ji}'s \) and the \( p_i \)'s are useful for building a structure for modelling the variability across groups and features, in this case we are
not interested in their individual estimation since the classification task only relies on the joint distribution of the X’s that is induced.

Figure 7: Posterior inference results for the beta-CoRM model with different choices of $c$ and $a$ for the data illustrated in Figure 5. For each subplot: Upper graph: real probabilities (spikes) and posterior estimates (dots). Middle graph: real scores. Bottom graph: posterior estimates of the scores.

In Figure 7, it can be appreciated the effects of different choices of $a$ and $c$, while keeping the $q_i$’s fixed as the maximum proportion across groups. As expected by the spike-and-slab interpretation given in Section 4, for small values of $a$ (Figure 7a and Figure 7c) there is a clear difference in the posterior scores across groups that vanishes as $a$ increases (Figure 7a) and (Figure 7d). Furthermore, it can be appreciated that this effect occurs independently of the value of $c$. Hence, it can be easily argued that smaller values of $a$ should be preferred. Otherwise, it would be like
randomly assigning a group to each new observation. To appreciate this effect more carefully, in Figure 8 we illustrate the effect of a small and a large value of \( a \) for a fixed value of \( c \) on the posterior predictive probabilities.

![Figure 8: Posterior predictive probabilities comparison between the beta-CoRM with a small \( a \) (left) and a large \( a \) (right) for the data illustrated in Figure 5.](image)

(a) \( c = 10 \) and \( a = 0.01 \)  
(b) \( c = 10 \) and \( a = 10 \)

Finally, for sake of completeness Figure 9 illustrates how \( c \) controls the closeness of the posterior estimates of the \( p_i \)'s (points) and \( q_i \)'s (spikes) for fixed \( a = 1 \).

![Figure 9: Posterior estimates of \( p_i \)'s (dots) and \( q_i \)'s (spikes) comparison between the beta-CoRM model for small \( c \) (left) and large \( c \) (right) and fixed \( a \) for the data illustrated in Figure 5.](image)

(a) \( c = 0.01 \) and \( a = 1 \).  
(b) \( c = 10 \) and \( a = 1 \).

Figure 9: Posterior estimates of \( p_i \)'s (dots) and \( q_i \)'s (spikes) comparison between the beta-CoRM model for small \( c \) (left) and large \( c \) (right) and fixed \( a \) for the data illustrated in Figure 5.
In order to find the best values for $c$ and $a$ one possibility would be to run the model for different values and perform the classification procedure in the training/validation set or the test set (if applicable), and select the ones that have the best posterior predictive performance. As an example, for the data illustrated in Figure 5, Table 3 shows the classification accuracy on the training and the test set for different values of $c$ and $a$. It can be appreciated that the best performance was achieved when both $c$ and $a$ were small. Also, it seems that the size of $a$ (being small) is more important compared to the size of $c$. In fact, for $c = 10$ and $a = 0.01$, we still had a decent result.

Table 3: Classification accuracy comparison (in %) of the beta-CoRM model with different values of $c$ and $a$ using the synthetic binary data illustrated in Figure 5.

| $c$   | $a$  | Accuracy on training set | Accuracy on test set |
|-------|------|--------------------------|----------------------|
| 0.01  | 0.01 | 100                      | 96                   |
| 1     | 1    | 100                      | 96                   |
| 2     | 0.5  | 100                      | 97                   |
| 0.01  | 10   | 92                       | 90                   |
| 10    | 0.01 | 99                       | 96                   |
| 10    | 10   | 91                       | 89                   |

Finally, using the generalised version of the beta-CoRM with parameter $c = 1$, and initial values $\alpha = \beta = \sigma = 1$ for the adaptive Metropolis-Hastings within the slice sampling algorithm described in Section 4 (with the same burning period, total number of simulations and thinning as before) we obtained the posterior estimates of the score parameters shown in Figure 10a and the classification accuracy for different thresholds illustrated in Figure 10b.
It can be immediately appreciated that there does not seem to be a big difference among the \(a_i\)'s. Something that did not surprise us since the data was randomly generated. However, it was interesting to see that we achieved an accuracy of 97\% when we considered \(T_{opt} = 1.392\) which yielded a total number of 143 features used for the classification.

### B.2 Five overlapping groups

For this subsection we considered a more complex scenario with five groups with overlapping score distributions, and tested the models’ performance in different kinds of circumstances by having a balanced and unbalanced number of observations and by introducing more observations and features. The data was generated by sampling the global probabilities from a uniform distribution on \((3, 1)\) and for each of the five groups we sampled the scores from uniform distributions on the intervals \((0.7, 1)\), \((0.5, 0.8)\), \((0.4, 0.6)\), \((0.2, 0.5)\) and \((0, 0.3)\), respectively. For both the balanced and unbalanced scenarios we tried with 150, 200 and 250 observations and 100, 200 and 300 features. For the beta-CoRM several combinations of \(c\) and \(a\), with the best results achieved when both \(a\) and \(c\) were close to one. It was interesting to notice that the accuracy decreased if both values were close to zero at the same time. For illustrative purposes, the results provided in the following subsections were obtained with \(c = a = 1\). Finally, for the generalised beta-CoRM we used \(c = 1\) and the same initial values for the MCMC scheme as in the previous subsection.

![Figure 10: Generalised beta-CoRM posterior results on the data illustrated in Figure 5.](image-url)
B.2.1 Balanced groups

In this scenario, all the groups contained the same number of observations for both the training and the test set. The accuracy performance of the discrete beta-CoRM (b-CoRM), the generalised beta-CoRM (Gb-CoRM), naive Bayes (nB), multinomial logistic (ML) and decision trees with their adaptive (ABoost), gradient (GBoost) and extreme gradient (XGboost) boosted versions can be seen and compared in Table 4. It can be immediately appreciated that the proposed models had the best performance with the generalised approach having a slightly better performance for most scenarios.

Table 4: Classification accuracy comparison (in \%) of the beta-CoRM models against commonly used supervised learning algorithms for the balanced groups scenario with varying number of observations and features.

| Obs | Feat | b-CoRM | Gb-CoRM | nB  | Tree | ABoost | GBoost | XGBoost | ML  |
|-----|------|--------|---------|-----|------|--------|--------|---------|-----|
| 150 | 100  | 80.67  | 84.67   | 66.00| 49.33| 63.33  | 66.00  | 68.00   | 51.33|
| 150 | 200  | 92.67  | 92.67   | 70.00| 44.67| 70.67  | 71.33  | 69.33   | 46.67|
| 150 | 300  | 97.33  | 98.00   | 71.33| 50.67| 74.00  | 69.33  | 70.00   | 50.67|
| 200 | 100  | 81.00  | 81.00   | 66.50| 44.50| 68.00  | 65.00  | 69.00   | 50.50|
| 200 | 200  | 92.50  | 93.00   | 83.00| 48.50| 76.50  | 75.50  | 73.50   | 50.50|
| 200 | 300  | 97.50  | 98.00   | 79.00| 56.00| 75.00  | 79.00  | 81.00   | 55.00|
| 250 | 100  | 82.40  | 83.20   | 77.60| 40.40| 66.00  | 68.00  | 70.00   | 48.00|
| 250 | 200  | 94.40  | 94.80   | 85.60| 52.00| 81.60  | 80.80  | 83.20   | 61.60|
| 250 | 300  | 96.00  | 96.40   | 83.20| 50.40| 76.80  | 75.60  | 82.80   | 58.80|

B.2.2 Unbalanced groups

For the unbalanced scenario the middle group had twenty observations for all the cases contrary to the other groups that had more as we increased the number of total observations. We started with 150 and increased to 200 and 250 total observations, with respective partitions \((30, 40, 20, 30, 30), (45, 50, 20, 45, 40)\) and \((55, 70, 20, 55, 50)\). The classification accuracy of all the classifiers can be seen and compared in Table 5. It can be immediately appreciated that just as for the balanced scenario, the beta-CoRM models had the best performance, with the generalised version having the best performance overall.
Table 5: Classification accuracy comparison (in %) of the beta-CoRM models against commonly used supervised learning algorithms for the unbalanced groups synthetic data with varying number of observations and features.

| Obs | Feat | b-CoRM | Gb-CoRM | nB | Tree | ABoost | Gboost | XGBoost | ML |
|-----|------|--------|---------|----|------|--------|--------|---------|----|
| 150 | 100  | 86.00  | 88.67   | 64.67 | 36.67 | 68.67  | 61.33  | 66.67   | 38.67 |
| 150 | 200  | 93.33  | 94.00   | 79.33 | 53.33 | 72.67  | 72.00  | 75.33   | 52.00 |
| 150 | 300  | 94.00  | 95.33   | 74.67 | 60.00 | 76.67  | 78.00  | 78.67   | 48.67 |
| 200 | 100  | 90.50  | 91.00   | 75.50 | 59.00 | 76.00  | 76.50  | 80.00   | 57.50 |
| 200 | 200  | 95.50  | 95.50   | 83.50 | 57.00 | 78.50  | 76.50  | 79.00   | 58.00 |
| 200 | 300  | 97.50  | 98.00   | 78.50 | 57.50 | 82.00  | 79.50  | 80.50   | 56.50 |
| 250 | 100  | 89.20  | 81.20   | 69.20 | 68.40 | 83.60  | 81.20  | 82.00   | 66.40 |
| 250 | 200  | 92.80  | 92.80   | 87.20 | 64.40 | 82.80  | 77.20  | 82.80   | 54.40 |
| 250 | 300  | 97.20  | 97.20   | 82.40 | 60.80 | 84.80  | 82.40  | 85.20   | 61.60 |

References

Ahmed, I. and Lhee, K. (2011). Classification of packet contents for malware detection. *Journal in Computer Virology, 7*(4):279 – 295.

Atchadé, Y. F. and Rosenthal, J. S. (2005). On adaptive Markov chain Monte Carlo algorithms. *Bernoulli, 11*(5):815 – 828.

Bolton, A. and Heard, N. (2018). Malware Family Discovery Using Reversible Jump MCMC Sampling of Regimes. *Journal of the American Statistical Association, 113*(524):1490 – 1502.

Damien, P., Wakefield, J., and Walker, S. (1999). Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 61*(2):331 – 344.

Griffin, J. and Holmes, C. (2010). *Computational issues arising in Bayesian nonparametric hierarchical models*, pages 208 – 222. Cambridge University Press.

Griffin, J. and Leisen, F. (2018). Modelling and Computation Using NCoRM Mixtures for Density Regression. *Bayesian Analysis, 13*(3):897–916.

Griffin, J. E. and Leisen, F. (2016). Compound Random Measures and their use in Bayesian nonparametrics. *Journal of the Royal Statistical Society Series B-Statistical Methodology, 79*(2):525 – 545.
Griffiths, T. L. and Ghahramani, Z. (2005). Infinite Latent Feature Models and the Indian Buffet Process. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS’05, pages 475 – 482, Cambridge, MA, USA. MIT Press.

Griffiths, T. L. and Ghahramani, Z. (2011). The Indian Buffet Process: An Introduction and Review. *Journal of Machine Learning Research*, 12(32):1185 – 1224.

Ishwaran, H. and Rao, J. S. (2005). Spike and slab variable selection: Frequentist and Bayesian strategies. *Annals of Statistics*, 33(2):730 – 773.

Kao, Y., Reich, B., Storlie, C., and Anderson, B. (2015). Malware Detection Using Nonparametric Bayesian Clustering and Classification Techniques. *Technometrics*, 57(4):535 – 546.

Kingman, J. F. C. (1967). Completely Random Measure. *Pacific Journal of Mathematics*, 21(1):59 – 78.

Kolter, J. Z. and Maloof, M. A. (2004). Learning to detect malicious executables in the wild. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’04, pages 470 – 478, New York, NY, USA. Association for Computing Machinery.

Kolter, J. Z. and Maloof, M. A. (2006). Learning to Detect and Classify Malicious Executables in the Wild. *Journal of Machine Learning Research*, 7:2721 – 2744.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79 – 86.

Masud, M., Khan, L., and Thuraisingham, B. (2007). A Hybrid Model to Detect Malicious Executables. In *2007 IEEE International Conference on Communications*.

McGraw, G. and Morrisett, G. (2000). Attacking Malicious Code: A Report to the Infosec Research Council. *IEEE Software*, 17(5):33 – 41.

Mitchell, T. J. and Beauchamp, J. J. (1988). Bayesian Variable Selection in Linear Regression. *Journal of the American Statistical Association*, 83(404):1023 – 1032.

Pektaş, A., Eris, M., and Acarman, T. (2011). Proposal of n-gram based algorithm for malware classification. *SECUWARE 2011 - 5th International Conference on Emerging Security Information, Systems and Technologies*, pages 14 – 18.

Prasse, P., Machlica, L., Pevný, T., Havelka, J., and Scheffer, T. (2017). Malware Detection by Analysing Network Traffic with Neural Networks. In *2017 IEEE Security and Privacy Workshops (SPW)*, pages 205 – 210.
Raff, E., Zak, R., Cox, R., Sylvester, J., Yacci, P., Ward, R., Tracy, A., Mclean, M., and Nicholas, C. (2016). An investigation of byte n-gram features for malware classification. *Journal of Computer Virology and Hacking Techniques*, pages 1 – 20.

Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., and Ahmadi, M. (2018). Microsoft Malware Classification Challenge. arXiv:1802.10135.

Rumao, P. (2016). Detect Malicious Executable (AntiVirus). Available at: http://archive.ics.uci.edu/ml/datasets/Detect+Malicious+Executable(AntiVirus). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science.

Schultz, M. G., Eskin, E., Zadok, F., and Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables. In *Proceedings 2001 IEEE Symposium on Security and Privacy. SP 2001*, pages 38 – 49.

Storlie, C., Anderson, B., Vander Wiel, S., Quist, D., Hash, C., and Brown, N. (2014). Stochastic identification of malware with dynamic traces. *The Annals of Applied Statistics*, 8(1):1 – 18.

Thibaux, R. and Jordan, M. I. (2007). Hierarchical Beta Processes and the Indian Buffet Process. In Meila, M. and Shen, X., editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 564 – 571, San Juan, Puerto Rico. PMLR.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition.

Vidal, M., Orozco, J., Orozco, S., Lucila, A., Villalba, G., and Javier, L. (2017). Alert Correlation Framework for Malware Detection by Anomaly-Based Packet Payload Analysis. *Journal of Network and Computer Applications*, 97(C):11 – 22.

Yang, Y. and Pedersen, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning, ICML ’97*, pages 412 – 420. Morgan Kaufmann Publishers Inc.