Fall 12-20-2020

Malware Classification with Gaussian Mixture Model-Hidden Markov Models

Jing Zhao
San Jose State University

Malware Classification with Gaussian Mixture Model-Hidden Markov Models

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Jing Zhao

December 2020

The Designated Project Committee Approves the Project Titled

Malware Classification with Gaussian Mixture Model-Hidden Markov Models

by

Jing Zhao

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2020

Dr. Mark Stamp	Department of Computer Science
Dr. Teng Moh	Department of Computer Science
Samanvitha Basole	Department of Computer Science

ABSTRACT

Malware Classification with Gaussian Mixture Model-Hidden Markov Models

by Jing Zhao

Discrete hidden Markov models (HMM) are often applied to the malware detection and classification problems. However, the continuous analog of discrete HMMs, that is, Gaussian mixture model-HMMs (GMM-HMM), are rarely considered in the field of cybersecurity. In this study, we apply GMM-HMMs to the malware classification problem and we compare our results to those obtained using discrete HMMs. As features, we consider opcode sequences and entropy-based sequences. For our opcode features, GMM-HMMs produce results that are comparable to those obtained using discrete HMMs, whereas for our entropy-based features, GMM-HMMs generally improve on the classification results that we can attain with discrete HMMs.

ACKNOWLEDGMENTS

Dr. Mark Stamp has been mentoring me throughout my whole graduate study at San Jose State University. He is the professor that encouraged me to take the challenge of becoming a computer science graduate student, and guided me through the whole journey. As a major transferred students, without his encouragement and guidance, I would not have gone this far in the past three years. He is always patient with all my questions about this research. I am filled with gratitude to have Dr. Mark Stamp as my advisor.

I would also want to express my sincere gradidute to my committee members Dr. Teng Moh and Miss Samanvitha Basole. Dr. Teng Moh has played a vital role in inspiring me during the study of machine learning. The knowledge that I learned from his lectures has helped build the foundation of this study. Miss Samanvitha Basole has provided strong support throughout the whole thesis writing process. She has carefully reviewed all my writings and has always come up valuable suggestions for both research and career.

I cannot express enough thanks to my parents, my husband for their endless love and support. They are my ultimate motivation to face all the challenges.

My thankfulness also extends to all the professors, teachers and friends at SJSU. It was such an honor to study with all of you.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Due to COVID-19, businesses and schools have moved their work online and some explore the possibility of going online permanently [1]. This trend makes cybersecurity more important than ever before.

Malware, includes different types of malicious programs such as viruses, Trojan horses, worms, etc [2]. They are designed to "deliberately fulfills the harmful intent of an attacker" [3]. Those harmful actions include stealing private information, deleting sensitive data without consent, disrupt computer systems, etc [4].

The study of malware has been active for decades [5]. Malware detection and classification are fundamental topics in the study of malware. Traditionally, signature detection has been the most prevalent method for detecting malware, but recently, machine learning techniques have proven their worth, especially for dealing with advanced types of malware. Many machine learning techniques have been applied to the malware problem, including hidden Markov models (HMM) [6], k-nearest neighbors (KNN) [7], support vector machines (SVM) [8], and a wide variety of neural network based techniques [9]. Each machine learning technique has its own advantages and disadvantages. It is not the case that any one technique is best for detecting all types of malware---there are many different types of malware, and many different features that can be considered. Thus, it is useful to explore different techniques and algorithms in an effort to extend our knowledge base for effectively dealing with malware. In this paper, we consider Gaussian mixture model-hidden Markov models (GMM-HMMs), which can be viewed as the continuous analog of typical (discrete)

HMMs.

Discrete HMMs are well known for their ability to learn important statistical properties from a sequence of observations. For a sequence of discrete observations, such as the letters that comprise a selection of English text, we can train such a discrete HMM to determine the parameters of the (discrete) probability distributions that underlie the training data. However, some observation sequences are inherently continuous, such as a signal extracted from speech, as opposed to letters extracted from English text. In such cases, a discrete HMM is not directly applicable. While we can discretize a continuous signal, there will be some loss of information. As an alternative to discretization, we can attempt to model the continuous probability density functions that underlie continuous training data.

Gaussian mixture models (GMMs) are probability density functions that are represented by weighted sums of Gaussian distributions [10]. By varying the number of Gaussian components and the weights assigned to each, GMMs can approximately model any continuous probability distributions arbitrarily well [11]. It is possible to train HMMs to learn the parameters of GMMs, and the resulting GMM-HMM models are widely used in speech recognition [12, 13].

In the field of cybersecurity, GMMs have been used, for example, as a clustering method for malware classification [14]. However, to the best of our knowledge, GMM-HMMs have not been previously considered in the context of malware detection or classification. In this research, we apply GMM-HMMs to the malware classification problem, and we compare our results to discrete HMMs.

The remainder of this paper is organized as follows. In Chapter 2, we discuss relevant related work. Chapter 3 provides background on the various models considered,

namely, GMMs, HMMs, and GMM-HMMs. Malware classification experiments and results based on discrete features are discussed in Chapter 4. Since GMM-HMMs are more suitable for continuous observations, in Chapter 5 we present another set of malware classification experiments based on continuous entropy features. We provide our conclusions and we discuss possible directions for future work in Chapter 6.

CHAPTER 2

Related Work

A lot of work has been completed on exploring malware detection and classification. This chapter discusses about the previous work on malware analysis using different techniques. It also introduces the related work implemented by GMMs, HMMs and GMM-HMMs.

As mentioned in [15], the malware analysis can be categorized into static analysis, dynamic analysis or the hybrid. In the static analysis, the study is conducted without executing the malware. The features used in static analysis include byte sequence, opcodes, string signatures, etc. While in the dynamic analysis, the behavior of a malicious program is analyzed through executing the malware. The dynamic analysis requires a controlled environment to run executables and observe the outcome [16].

Using machine learning techniques for malware detection and classification is popular in both static and dynamic malware analysis. In [17], the author applied Naive Bayes and Multinominal Naive Bayes on the string data and the n-grams of byte sequences extracted from the executable files, obtaining results outperformed the traditional signature-based method. More machine learning techniques, such as SVM, Decision Tree, and boosted Decision Tree were explored by the authors in [18] and have further proved the effectiveness of those methods. Converting binary files into gray-scale images has opened up more opportunities for malware analysis [19]. In [7], they converted binary executables into images and used KNN to classify malware into different families. Using neural networks is another prominent technique for malware detection and classification. The author in [20] applied neural networks on a

large-scale malware classification system and reduced the error rate by 43% comparing to a previous technique using logistic regression.

A Gaussian mixture model (GMM) is one of a probability density models [21] that is a weighted sum of multiple Gaussian distributions. The advantage of a Gaussian mixture model is that it can describe variations of data distributions by changing the number of components and the values of weights [11]. GMM allows for modeling a more arbitrary distribution and has higher modeling capability than a single Gaussian function. Although the real distribution is not similar to a Guassian distribution, the combination of several Gaussians is able to make the model robust [22]. Due to the flexibility of GMM, it is capable of simulating all kinds of distributions. One of the main use of GMMs is distribution estimation in different fields such as wave elevation in Oceanography [23]. With a proper estimation of the distribution, the application can extend to anomaly detection [24], signal mapping, and positioning [25]. As a trade-off, the cost of calculation also increases while the model gets more complicated.

Besides distribution estimation, GMM is also a clustering method for classification [26]. Parameter fitting leads to a convergence in distribution which gives a natural process of clustering the data. The author in [27] used GMM as a classification method to segment brain lesions.

HMM is another probabilistic model to describe how events evolve by studying the sequence of observations of events. It describes the evolution of events by providing the probability of each observation at each state and by transitioning from one state to another. The event is considered as a Markov process, a process in which each event is only determined by the previous event, not other earlier events. Each event stays in some state that is unknown from the observer and each event can transit from one state to another with a certain probability. Due to the feature of a Markov

process, HMM has naturally been used in the signal processing area. One of a popular uses is speech recognition [28]. Due to its robustness and the efficiency, HMM is also widely used in medical area such as Sepsis detection [29] and human brain study through functional magnetic resonance imaging [30]. Motion recognition is another area where HMM plays a vital role, such as recognizing dancing moves [31] and 3D gestures [32]. HMMs have been applied on malware classification as well. In [33], the authors trained models and generated scores using HMMs, then applied KNN to cluster samples into classes. HMMs, as well as profile hidden markov models(PHMMs) have also been applied on dynamic analysis using API calls, and the result shows that the dynamic analysis outperforms the static analysis using static features, such as opcodes [34].

As an extension to HMM, GMM-HMM has also been widely used in classification problems. Given the flexibility of GMM, GMM-HMM is capable of studying more complex patterns underlying the sequence of observations. Yao et al. [35] used GMM-HMM to classify network traffic with different protocols. Moreover, GMM-HMM has also been used in motion detection. For more complex poses, GMM-HMM has performed better than a pure HMM [36].

CHAPTER 3

Background

This chapter introduces the mathematical definition for GMMs, HMMs and GMM-HMMs. We presented the model training procedure using HMMs and extends the technique to GMM-HMMs, as well as parameter re-estimation and scoring. Lastly, in order to validate the function of the model, we apply it to English text classification.

3.1 Formulation of GMM

GMM is a probabilistic model which is combined by multiple Gaussian component distributions. Mathematically, the probability density function (PDF) of a GMM is a weighted sum of M Gaussian PDFs. As described in [37], a GMM is in the form of

$$p(x|\lambda) = \sum_{x=i}^{M} \omega_i g(x|\mu_i, \Sigma_i),$$

where x is a D dimension vector; ω_i is the weight of each Gaussian component; μ_i and Σ_i are the mean and the covariance matrix of the ith component of GMM respectively. The sum of the mixture weights should be 1. Each component of GMM is a multivariate Gaussian distribution function given by

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left\{ -\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i)\right\}. \quad (1)$$

3.2 Formulation of HMMs

To better understand the HMM, we use mathematical notations to define the model as described in [6]. The notations of the model is given by

T: number of observations in the observation sequences

$\mathcal{O} : \{\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_{T-1}\}$, the observation sequence

K: number of unique symbols in the observation sequence

N: number of states

π: initial state distribution

A: states transition matrix with size $N \times N$

B: the probability of each unique observation symbol at each state

The HMM is therefore denoted by $\lambda = (A, B, \pi)$. The component a_{ij} of A matrix is given by

$$a_{ij} = P(\text{state } q_j \text{ at } t+1 | \text{ state } q_i \text{ at } t).$$

In a discrete HMM, B matrix is of size $N \times K$. Each row of B matrix represents the probability distribution of the observation symbol at that state. Specifically, each element of B matrix is given by

$$b_i(\mathcal{O}) = P(\text{observation } \mathcal{O} \text{ at } t | \text{ state } q_i \text{ at } t).$$

HMM is able to answer the following three problems [6]:

1. Given a sequence of observations and the model $\lambda = (\pi, A, B)$, calculate the probability of the observing sequence.

2. Given the model and the observation sequences, find the optimum states sequences. That is, find out the most possible state of each observation in the sequence.

3. Find out the model $\lambda = (A, B, \pi)$ which gives the highest possibility for a given observation sequence with a predefined dimension for N and K.

Solutions to the three problems provide methods for model construction, parameters re-estimation and model scoring.

3.3 Formulation of GMM-HMM

The overall model structure of a GMM-HMM is similar to a HMM. In a GMM-HMM, the difference happens to be the B matrix. In a discrete HMM, the B matrix is a discrete probability distribution for each state given the number of symbols. In a GMM-HMM, the probability of each observation at a certain state is determined by the PDF, which is described by a GMM. Specifically, the PDF of having each observation at state q_i at time t is in the form

$$p_i(\mathcal{O}_t) = \sum_{m=1}^{M} c_{im} g(\mathcal{O}_t|\mu_{im}, \Sigma_{im}), \quad 1 \leq i \leq N, \quad 0 \leq t \leq T-1, \tag{2}$$

where

$$\sum_{m=1}^{M} c_{im} = 1, \quad 1 \leq i \leq N.$$

The parameter M is the total number of Gaussian mixtures components; c_{im} is the mixture coefficient or the weight of mth Gaussian mixture at state q_i; μ_{im} and Σ_{im} are the mean vector and covariance matrix for the mth Gaussian mixture at state q_i. As a modification for equation (1), $g(\mathcal{O}_t|\mu_{im}, \Sigma_{im})$ is in the form

$$g(\mathcal{O}_t|\mu_{im}, \Sigma_{im}) = \frac{1}{(2\pi)^{D/2}|\Sigma_{im}|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathcal{O}_t - \mu_{im})'\Sigma_{im}^{-1}(\mathcal{O}_t - \mu_{im}) \right\}, \tag{3}$$

where D is the dimension of each observation. The A matrix and π remain the same with the HMM.

In a GMM-HMM, the notations are defined by

N: number of states

M: number of Gaussian mixture components

D: dimension of each observation

$\mathcal{O}$: $(\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_{T-1})$ the observation sequence

T: number of observations in a observation sequence

π: initial distribution

A: state transition matrix, size $N \times N$

c: weight of each Gaussian mixture at each state, size $N \times M$

μ: mean value of each Gaussian mixture component at each state, size $N \times M \times D$

Σ: co-variance matrix of M Gaussian mixture components at each state, size $N \times M \times D \times D$

As GMM-HMMs inherently follow the structure of HMMs, a portion of notations share the same meanings with HMMs, such as the number of states N, total observations T, the initial state π and the state transitioin matrix A. The difference is that the underlying distribution of the observation data is characterized by a GMM. Therefore, a GMM-HMM is defined by $\lambda = (A, \pi, c, \mu, \Sigma)$, where c, μ and Σ uniquely defines the PDF listed in (2). The dimensions of data is given by D. In this study, the dimension of the experiment data is 1. Similar to HMMs, we need to get solutions for the three problems mentioned in Section 3.2.

3.4 GMM-HMM Parameters Re-estimation and Scoring

To use GMM-HMM as a classification technique to classify malware samples, we need to solve problem 3 and problem 1 as mentioned in 3.2. Given a malware training sample, we use problem 3 to obtain the parameters to describe the malware. For testing, we use the solution to problem 1 to score each sample.

3.4.1 Solution to problem 1

Given a model $\lambda = (A, \pi, c, \mu, \Sigma)$ and a sequence of observations $\mathcal{O} = \{\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_{T-1}\}$, the probability of the likelihood of the observing sequence is obtained by $P(\mathcal{O}|\lambda)$. The forward algorithm, or the alpha pass is used to find out

$P(\mathcal{O}|\lambda)$ [6].

Similar to a discrete HMM mentioned in [6], in the alpha pass algorithm, we have $\alpha_t(i) = P(\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_t, x_t = q_i|\lambda)$, which gives the probability of a sequence of observation up to time t with the state of time t being q_i. The probability can be reformed to be

$$P(\mathcal{O}|\lambda) = \sum_{i=0}^{N-1} \alpha_{T-1}(i) \tag{4}$$

where

$$\alpha_t(i) = \left[\sum_{j=0}^{N-1} \alpha_{t-1}(i) a_{ji}\right] b_i(\mathcal{O}_t). \tag{5}$$

At $t = 0$, $\alpha_0(i) = \pi_i b_i(\mathcal{O}_0)$.

In a discrete HMM, $b_i(\mathcal{O}_t)$ gives the probability of observing $\mathcal{O}_t$ at time t in state i. In a GMM-HMM, however, simply replacing $b_i(\mathcal{O}_t)$ in (5) by the PDF from (2) gives the likelihood but not the exact probability. To obtain the probability we need to take an integral of a small region around observation $\mathcal{O}_t$ [38].

$$b_i(\mathcal{O}_t) = \int_{\mathcal{O}_t-\epsilon}^{\mathcal{O}_t+\epsilon} p_i(\mathcal{O}_t|\theta_i)\, do, \tag{6}$$

with θ_i being the probabilistic parameters of c_i, μ_i and Σ_i. ϵ is a small range, which demonstrates the continuity of observations.

3.4.2 Solution to Problem 3

The alpha pass calculates the probability of observing the sequence from the beginning up to time t. The beta pass algorithm [6] is used to describe the other half of the observation from time $t+1$ to the end. In the beta pass algorithm, we have $\beta_t(i) = P(\mathcal{O}_{t+1}, \mathcal{O}_{t+2}, \ldots, \mathcal{O}_{T-1}|x_i = q_i, \lambda)$. At $t = T-1$, $\beta_t(i) = 1$. Therefore we

have

$$\beta_t(i) = \sum_{j=0}^{N-1} a_{ij} b_j(\mathcal{O}_t)\beta_{t+1}(j). \tag{7}$$

In a HMM, to estimate A, a "di-gamma" [6] is defined as follows.

$$\gamma_t(i,j) = P(x_t = q_i, x_{t+1} = q_j|\mathcal{O}, \lambda).$$

It defines the probability of the observation $\mathcal{O}$ being at state q_i at time t and transiting to state q_j at time $t+1$. Using the alpha pass and the beta pass, we can obtain the result for $\gamma_t(i,j)$ [6]. The sum of "di-gamma" with respect to the transiting states gives the probability of the observation being at state q_i at time t. The representation is

$$\gamma_t(i) = \sum_{j=1}^{N} \gamma_t(i,j).$$

Therefore, the transiting probability of a_{ij} in A is given by

$$a_{ij} = \frac{\sum_{t=0}^{T-2} \gamma_t(i,j)}{\sum_{t=0}^{T-2} \gamma_t(i)}$$

To solve problem 3 for GMM-HMM, we use a similar strategy as the one used in HMM. We define a "gamma GMM-HMM", which is a generalization of the "gamma" from HMM [6]. The form is given by

$$\gamma_t(j,k) = P(x_t = q_j|k, \mathcal{O}, \lambda), \tag{8}$$

for $t = 0, 1, \ldots, T-2$, $j = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, M$. It represents the probability of being state q_j at time t given by the kth Gaussian mixture accounting for $\mathcal{O}_t$. According to [12], the form of $\gamma_t(j,k)$ is given by

$$\gamma_t(j,k) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \cdot \frac{c_{jk} N(\mathcal{O}_t|\mu_{jk}, \Sigma_{jk})}{\sum_{m=1}^{M} c_{jm} N(\mathcal{O}_t|\mu_{jm}, \Sigma_{jm})},$$

where $\alpha_t(j)$ is the alpha pass defined in equation (5) and $\beta_t(j)$ is from (7); c_{jk} is the weight of the kth mixture component.

The re-estimation for weight c for each Gaussian mixture is given by

$$\hat{c}_{jk} = \frac{\sum_{t=0}^{T-1} \gamma_t(j,k)}{\sum_{t=0}^{T-1} \sum_{k=1}^{M} \gamma_t(j,k)},$$

for $j = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, M$ [39, 12]. The numerator can be interpreted as the expected number of transitions from q_j determined by the kth Gaussian mixtures and the denominator can be seen as the expected transitions from state q_j given by M Gaussian mixtures.

Accordingly, the re-estimation for μ_{jk} and Σ_{jk} are in the form

$$\hat{\mu}_{jk} = \frac{\sum_{t=0}^{T-1} \gamma_t(j,k)\mathcal{O}_t}{\sum_{t=0}^{T-1} \gamma_t(j,k)}$$

and

$$\hat{\Sigma}_{jk} = \frac{\sum_{t=0}^{T-1} \gamma_t(j,k)(\mathcal{O}_t - \mu_{jk})(\mathcal{O}_t - \mu_{jk})'}{\sum_{t=0}^{T-1} \gamma_t(j,k)},$$

for $i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, M$.

3.4.3 GMM-HMM Scoring

To score an observation sequence means that we want to get the $P(\mathcal{O}|\lambda)$. As a GMM-HMM is an extension to an HMM, the scoring process is the same as an HMM except that we will use a GMM to calculate the probability for the presence of each observation. Once the parameter re-estimation process completes, the probability of observing a certain certain sequence is obtained through the alpha pass, according to (4). A more efficient calculation method -- HMM scaling [6] is used.

3.5 Model Performance Evaluation

A receiver operating characteristics (ROC) is a two dimensional graph that illustrates the trade-offs between true positive rate and false positive rate [40]. In a good classifier, the true positive rate should be close to 1 while the false positive rate remains the minimum. Therefore, we use the area under a receiver operating characteristics curve (AUC) as the measurement for evaluating models.

3.6 A Simple Example of GMM-HMM

As an example to test the GMM-HMM model, we experiment on real and fake English text classification. The English text experiment has been conducted using HMM in [41], and the model had successfully differentiated the vowels from consonants by giving a sequence of English text. We present the same experiment using a GMM-HMM to see how a GMM-HMM performs compared to an HMM. We also generate some fake English texts to test if the trained GMM-HMM model is able to differentiate them. The AUC is used to measure the performance of each model. As we use discrete observations in this experiment, instead of taking the integral of the probability density function to get probabilities, we simply use the probability density function itself in the training process. The same is happening at the scoring phase.

The English training data is from the "Brown corpus" [42]. To make the context ready for training, we convert all context to lowercase and remove punctuation, with only 26 English alphabets and spaces left. The total 27 symbols are encoded by integers from 0 to 26. As each observation is an encoded number, the dimension of each observation is 1. We set $N = 2$, $M = 6$ (6 Gaussian mixture models) and T = 50000. A is a 2×2 matrix with each row being stochastic. π is also a row stochastic matrix with a size of 1×2. The weights of mixture components c is initialized with

row stochastic values. We use the global mean value (the mean of all observations) and global variance (the variance of all observations) as the initialization value for the μ and Σ. In the beginning, each component of the Gaussian mixtures has the same mean value and the variance.

A GMM-HMM model is trained with 100 random restarts. As the observations are discrete symbols, the probability of observing each observation in state i at time t is estimated by the probability density function, given by equation (2). The trained result shows that GMM-HMM has successfully identified the vowels in one state. Figure 1 illustrates the distribution of observations in each state. Space is represented by the capitalized letter "S".

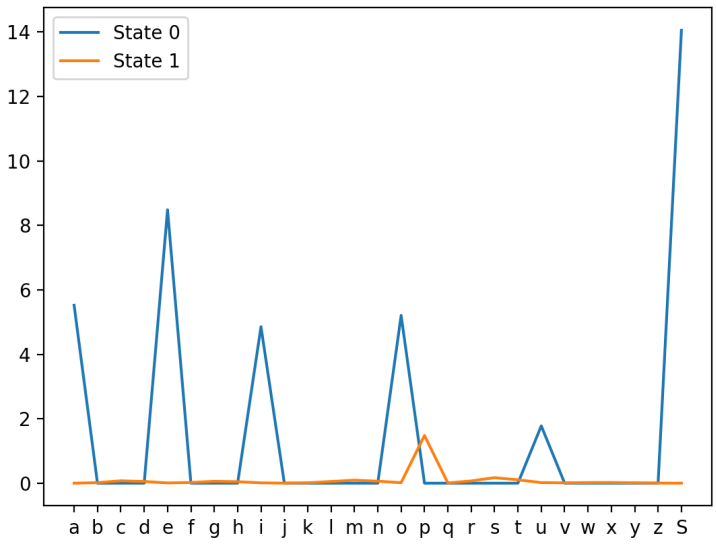


Figure 1: English letter distributions in each state

As Figure 1 shows, all vowels have been successfully selected in State 0. Table 1 lists the trained mean value for each Gaussian mixture. The mean value of each Gaussian mixture component corresponds to the encoded value of each vowel--0, 4, 8,

States	GM1	GM2	GM3	GM4	GM5	GM6
0	26.000000	14.000000	8.000000	4.000000	20.000000	0.000000
1	22.603863	6.308116	15.000000	12.080905	2.313083	18.140713

Table 1: The mean value of each Gaussian mixture in each state

14 and 20 represent letter "a", "e", "i", "o", "u" respectively. The space is encoded by 26. The GMM-HMM has separated the vowels and the space into one state by each Gaussian mixture component representing one symbol, and the rest into another state.

To compare the GMM-HMM and the HMM, we trained an HMM using the same English text dataset with $N = 2$, $M = 27$. The total observation we used for training is $T = 50000$. We use AUC to measure the performance of each model while testing with fake English text samples. The total testing samples include alphabets sequences from an English article and 50 fake English text by exchanging certain alphabets, such as changing "a" to "n", "e" to "z", etc.

Figure 2 presents the AUC of 5 testing examples generated by HMM and GMM-HMM. When converting the letter "e" to "z", both models successfully differentiated the fake samples from the real sample as letter "e" is much more frequently used than the letter "z". The probability of observing "e" is much higher than observing "z". Switching these two letters will make it easy for both models to make decisions. The performance of these two models, however, varies when the English letters are replaced using different rules. The experiments results have shown that GMM-HMM and HMM are sensible to the fake level of English text as the performance changes with different testing samples. It is hardly possible to test all cases with these two models, but based on the experiments we conducted, the performance of a GMM-HMM is

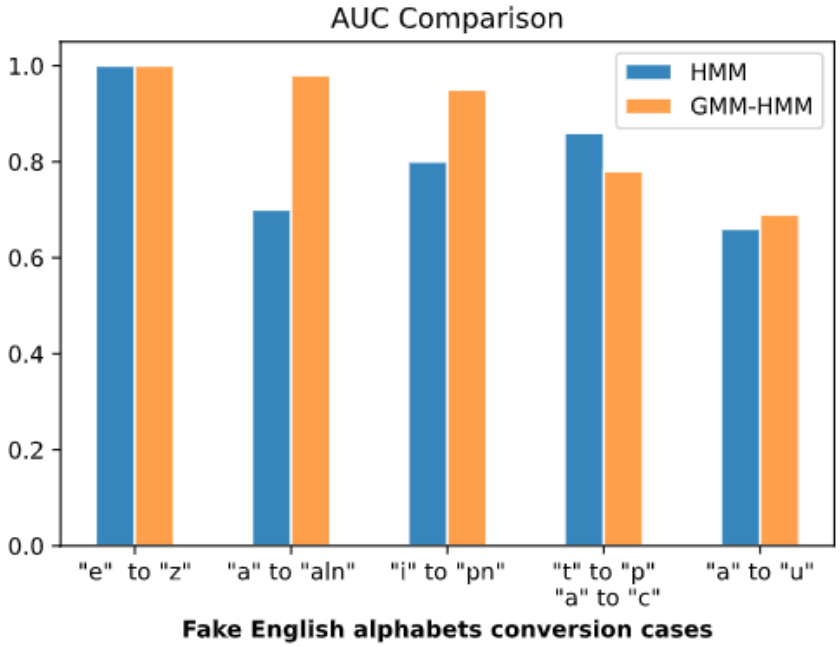


Figure 2: ROC comparison by HMM and GMM-HMM

comparable to an HMM in differentiating fake English text.

Since we know the number of vowels beforehand, it is not reasonable to set the number of Gaussian mixture components other than 6 (5 vowels and the space). This is also the key factor result in successfully differentiating the vowels from consonants in this experiment. As the observations are discrete, we are essentially a GMM-HMM to simulate the discrete probability density function. The more closer the number of "peaks" we select to that in a discrete probability density distribution, the similar the performance we obtain to that from a discrete HMM.

CHAPTER 4

Malware Classification Using Opcodes

The success on the English alphabets example has validated the GMM-HMM. In this chapter, we apply GMM-HMMs on three malware families using opcodes as features. In addition, we compare the results of GMM-HMM with the model trained with HMMs.

4.1 Introduction to the dataset

The training data we use belongs to the Malicia dataset [43]. It comprises 19 families and total 11,688 binary files. We use the top three largest families in this dataset, including Winwebsec, Zbot and Zeroaccess.

Winwebsec is a type of Trojan horse running in the Windows operating system. It attempts to install malicious programs by displaying fake links to bait users [44]. Zbot is another type of Trojan family that tries to steal users' core information by attaching executable files through spam email messages [45]. Zeroaccess attempts to store core information or components by creating hidden files. It can cause other malicious actions such as downloading malware or opening backdoor [46]

Table 2 lists the number of samples in each malware category. To encode the input, we extracted the top 30 opcodes of each malware, labeled them from 1 to 30, and labeled the rest of the opcodes as 31. We pick the top 30 opcodes as they represent more than 90% of all opcodes, as shown in Table 3.

The samples of each malware are split into 80% for training and 20% for testing. The training samples are concatenated into one file for training after encoding. T

Family	Samples
Winwebsec	4360
Zeroaccess	2136
Zbot	1305
Total	7801

Table 2: The number of samples in each malware family

Malware	% of top 30 opcodes
Winwebsec	96.9%
Zeroaccess	95.8%
Zbot	93.4%

Table 3: The percentage of Top 30 opcodes

observations will be used as the input to train models. The testing samples are also encoded to be prepared for testing against each model.

We use the averaged AUC from a 5 fold-cross validation to measure the performance of models and compare the results. To compare the performance of HMM and GMM-HMM, we also train a HMM for each malware family.

4.2 GMM-HMM on malware classification using opcodes

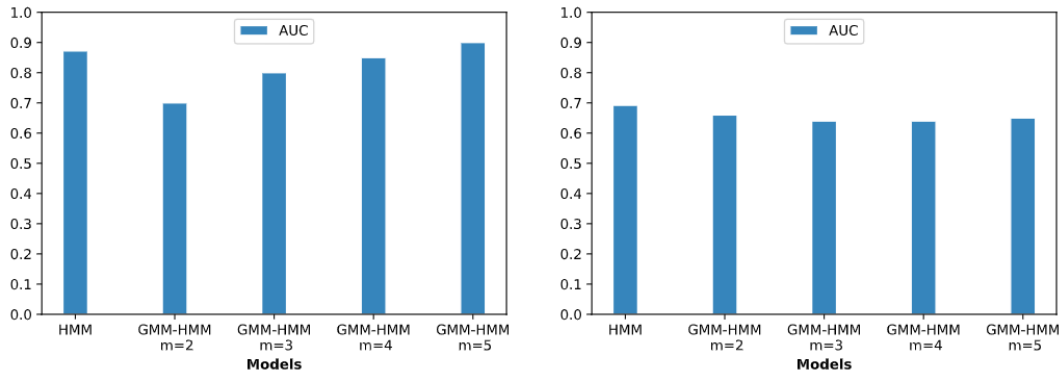
We set the length of observation $T = 100000$ for training, with $N = 2$. As malware classification is not as straightforward as the English text example, the best choice for the number of M, i.e., the number of Gaussian mixtures, is unknown. In general, the larger the M, the more complex the model becomes, the longer computation time it needs. Considering the computation efficiency, we experiment from $M = 2$ to $M = 5$. We train a model with one malware family and test with the other two malware

families. To test each model's performance, we use one hundred samples from the original malware family and another hundred instances from the testing malware family.

For initialization, π and A are row stochastic matrices. The mean values and the covariance are initialized with the global mean value and the global covariance of all training input data.

4.3 Results and discussion

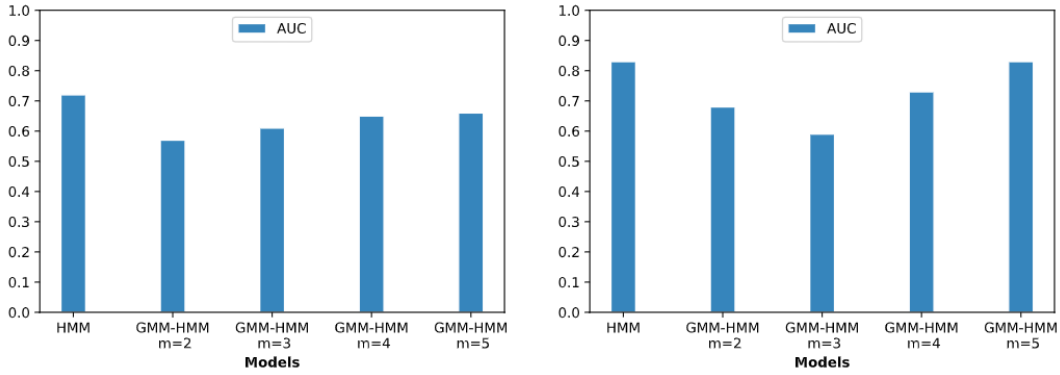
Figure 3 to Figure 5 give the average AUC for models trained with discrete HMMs and GMM-HMMs with different values for m. For most of the models, the GMM-HMM is able to obtain comparable results to the discrete HMM. The GMM-HMM outperforms the HMM in some cases, but the improvement is slight. There are cases that the larger the number for m, the better the performance of the model. For the rest of the models, the differences of each model are relatively small. It is possible that the training iterations are not large enough, or the opcodes sequences of malware pairs are challenging for both HMMs and GMM-HMMs.



(a) Test with Zeroaccess (b) Test with Winwebsec

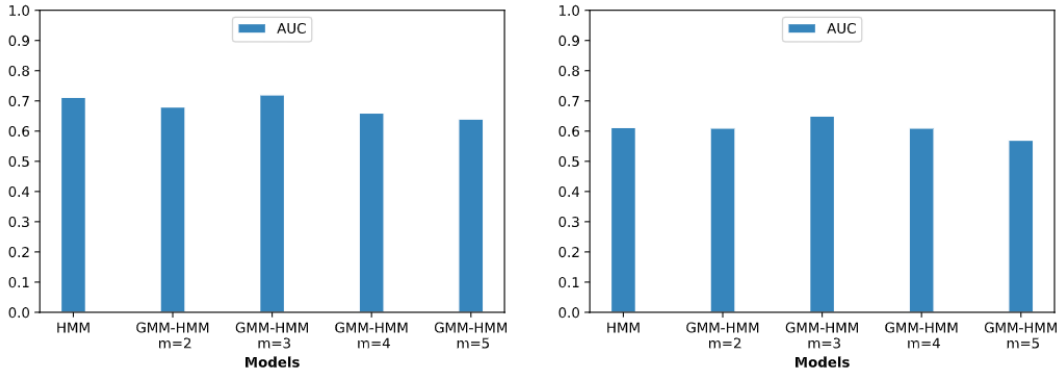
Figure 3: HMM versus GMM-HMMs --- Zbot

It is worth mentioning that, during the experiments, we have observed that the



(a) Test with Zbot (b) Test with Zeroaccess

Figure 4: HMM versus GMM-HMMs --- Winwebsec



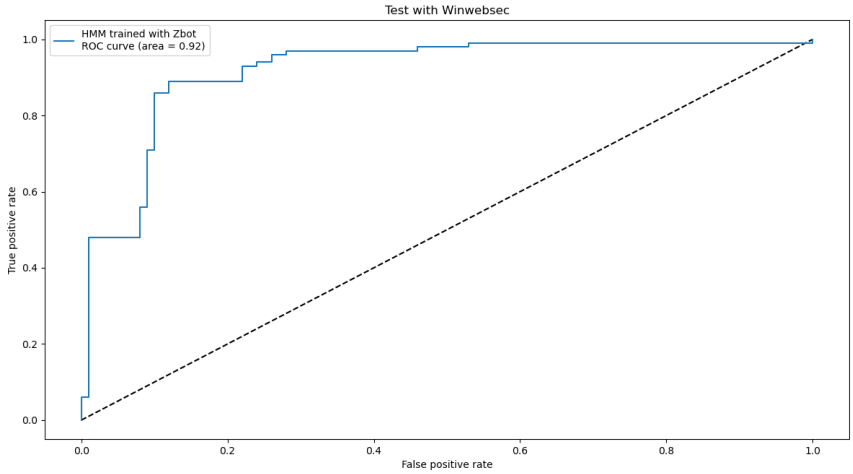
(a) Test with Zbot (b) Test with Winwebsec

Figure 5: HMM versus GMM-HMMs --- Zeroaccess

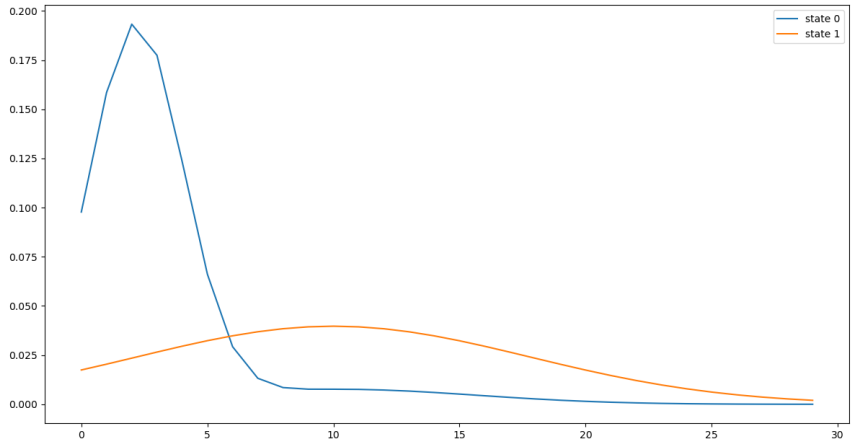
model given by the highest training score does not necessarily become the best model. An example is given by Figure 6 and 7. These figures exhibit the testing results for models trained with Zbot and tested with Winwebsec with $m = 2$ within one training process. The model obtained a lower score performs better than the highest score model in the testing phase. We can see that the observation distribution is highly concentrated around symbol near 0 for the model with the highest training score.This is more likely to happen when the number of Gaussian mixture components is small. While on the other hand, the model with a lower score seems to be able to capture more dynamics of the data distribution.

Receiving a high training score but performing worse during testing is a typical sign for over-fitting. As mentioned by the author in [47], GMMs have a singularity issue when the data is highly collapsed on a single value, with a corresponding covairance falling to 0. A more serious problem with the covariance being 0 is that the it would cause numerical error while calculating the likelihood given by (3). A quick fix to this issue is adding a small value of *err* when the covariance gets close to 0. The *err* is chosen to be 0.001 in this study. However, this fix does not avoid receiving a high value from the probability density function, which results in a high score using our scoring metric. Since we use discrete opcodes as the training features, this issue is easily to occur. For a discrete HMM, the sum of the probability of all the observations in each state is 1, which is guaranteed in the training process. For a GMM-HMM, the value obtaining from the probability density function for each observation is proportional to the probability, but there is no such a limit to restrain the sum of probability to be 1. This will cause problem especially when observations collapse. Therefore, for discrete features, using HMMs is a advantageous method.

However, this does not mean that a GMM-HMM is not as good as a HMM. One advantage of GMM-HMM is that it gives the flexibility to modify the complexity of the model. For a discrete HMM, once training completes, there is not much room to make modifications. For a GMM-HMM, we can extend the model's capability by changing the number of Gaussian mixture components. However, the trade-off is that the computation cost increases while adding more mixture components. Having a better understanding of the number of mixture components is helpful to select the best model.

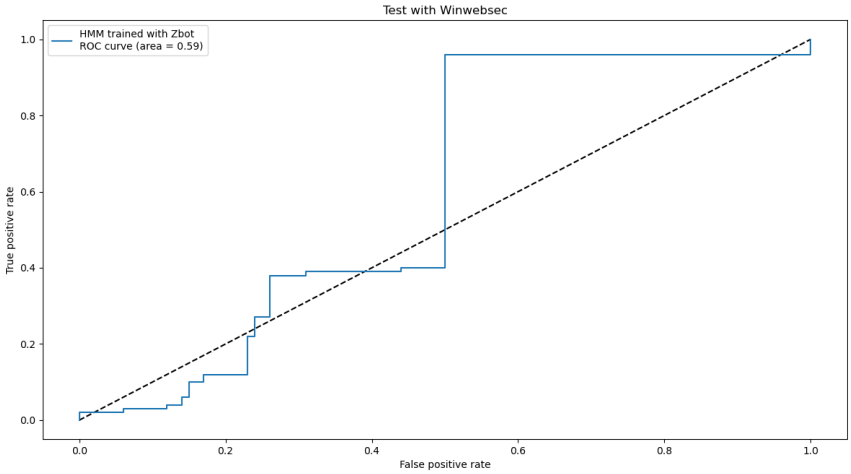


(a) The AUC of a model with lower score

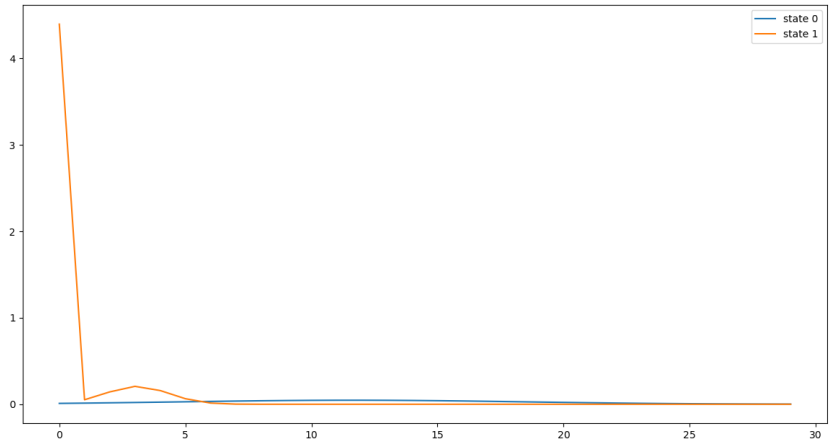


(b) The observation distribution of the model with a lower score

Figure 6: A model with a lower training score



(a) The AUC of a model with the highest score



(b) The observation distribution of the model with the highest score

Figure 7: A model with the highest training score

CHAPTER 5

Malware Classification Using Entropy

A GMM-HMM, by nature, is designed for continuous data, as opposed to discrete features, such as opcodes. To fully take the advantage of the model, we used malware binary files' entropy as features to train models and evaluate the performance.

5.0.1 Entropy of malware

We have used a similar feature-extracting method as [48]. Specifically, we first define a window size as the range to obtain entropy, then we move the window size by a fixing sliding step till the end of the file. Both the window size and the sliding step are the parameters that need to be tuned to get the model's best performance. In general, the sliding step needs to be smaller than the window size to ensure all the information is captured. According to Shannon's formula [49], the entropy(E) is obtained by

$$E = -\sum_{x \in W_i} p(x) \log_2 p(x),$$

where W_i is the ith window, and $p(x)$ is the frequency of the of byte x within that window range.

5.0.2 Parameters selection

Similar to [48], we use half of the window size as the sliding step. The entropy gets smoothed out with a large window size. It removes noises but might fail in capturing details. On the other hand, with a smaller window size, the number of observations of each malware sample increases, which results in higher computation costs. An example of entropy plot is given in Figure 8. To select the best values for

Window size	512	256	128
Sliding step	256	128	64

Table 4: The window size and its corresponding sliding step

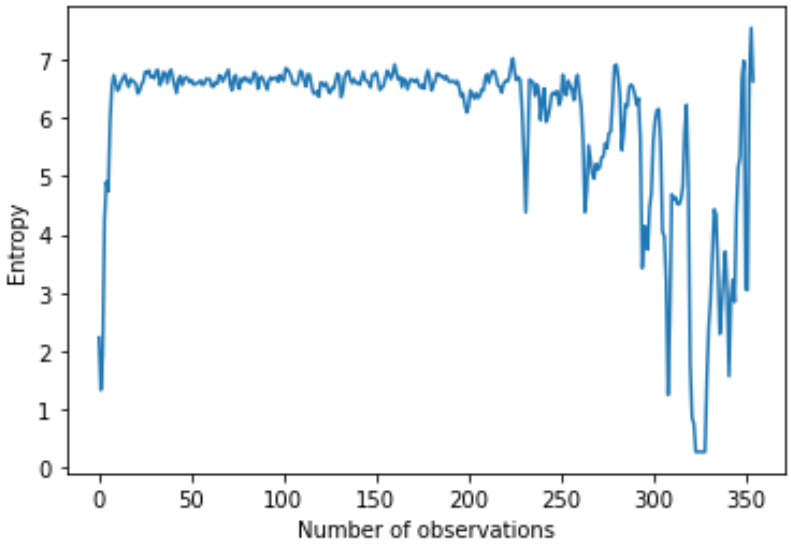
these parameters, we conduct experiments with different window sizes, as listed in Table 4.

Similar to experiments with the opcodes, there are two states for all the models. The number of Gaussian mixtures components are from 2 to 5. We train a model with one malware family and test it against the other two families. To evaluate the performance, we again used the ROC curve with 5-fold cross-validation.

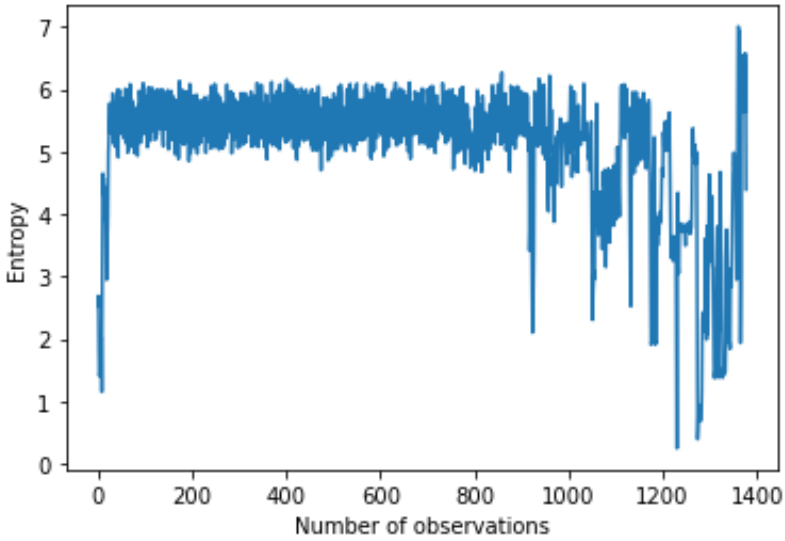
Figure 9 presents the averaged AUC with 5-fold cross-validation for models trained with Zbot. Each model is tested with 50 Winwebsec and Zeroaccess samples. As shown in Figure 9, the performance has improved from the result using HMM. Moreover, training samples with window size 256 and 128 have obtained better results than models with window size 512. Finding the value for M and selecting a proper window size is a part of the fine-tuning process.

We have conducted similar experiments for Winwebsec and Zeroaccess. Varying the window size generates similar results for models trained with Winwebsec. In this case, a larger window size is appropriate, as it provides lower computation cost. The best models for Zeroaccess are obtained at window size 128 and 512 for testing Zbot and Winwebsec respectively.

The ϵ in (6) gives the range which is used to approximate the probability under the continuous PDF. Specifically, the probability is given by the difference of the cumulative distribution fuction (CDF) under the range of $2 \times \epsilon$. In theory, the smaller



(a) Entropy with window size 512



(b) Entropy with window size 128

Figure 8: Entropy in different window size

the range, the more precise the probability we receive. However, one observation we have met during the experiment is that the approximate probability gets close to 0 if the ϵ is set too small, leading to an error in the scoring process. We take experiments with the ϵ to be $1e{-}1, 1e{-}3$ and $1e{-}6$ and select the one resulting in the

Options for ϵ	1e−1	1e−3	1e−6
AUC for testing Zeroaccess	0.87	0.86	0.87
AUC for testing Winwebsec	0.96	0.94	0.92

Table 5: AUC of models trained with Zbot

Options for ϵ	1e−1	1e−3	1e−6
AUC for testing Zbot	0.99	0.71	0.77
AUC for testing Zeroaccess	0.98	0.83	0.86

Table 6: AUC of models trained with Winwebsec

best performance for training models.

The model performance does not vary much for Zbot using different options for ϵ, which is listed in Table 5. For maintaining a balance between the precision and the performance, we let ϵ be 1e−6 for training the Zbot family. Whereas, for the Winwebsec family, 1e−1 seems a better option in terms of the model performance. Selecting the ϵ for Zeroaccess is a little bit tricky. A larger ϵ improves the AUC while testing with Zbot, but it sacrifices the performance with Winwebsec. Considering a smaller ϵ gives a better precision, we choose to stick with $\epsilon =$ 1e−6.

A consequential effect is that, the AUC of the model trained with Zeroaccess is unsatisfactory, which can also be seen from Figure 11, especially with a larger

Options for ϵ	1e−1	1e−3	1e−6
AUC for testing Winwebsec	0.61	1.00	1.00
AUC for testing Zbot	0.89	0.54	0.54

Table 7: AUC of models trained with Zeroaccess

window size. The model trained with Zeroaccess generates similar scores for samples in Zbot family. To better understand the reason behind it, we use Kullback–Leibler divergence (KL divergence) [50] to compare the models trained with these two families by assessing observation distributions obtained from the training process, as the score is mostly determined by the probability of each observation.

The KL divergence is given as follows:

$$\mathrm{KL}(p\|q) = \int_{-\infty}^{\infty} p(x)\log\frac{p(x)}{q(x)}, \tag{9}$$

where p and q are probability density functions.

As each model have two states, there are total 4 combinations. The PDF p and q in (9) are not interchangeable, thus, we take the average value of the KL divergence by swapping the position of p and q as the distance measurement for each combination. Therefore, we define

$$\mathrm{KL}(\mathrm{model}_1, \mathrm{model}_2) = \frac{\mathrm{KL}(\mathrm{model}_1\|\mathrm{model}_2) + \mathrm{KL}(\mathrm{model}_2\|\mathrm{model}_1)}{2}$$

Among the 4 combinations, we select the minimum value as the distance for measuring two models. The comparisons of difference models are listed in Table 8. As the KL divergence is not symmetric, the ingredients of the final KL divergence are listed in Table 9. The probability density functions are obtained from the models with window size 128. It can be seen that Zbot and Zeroaccess are considered "similar" according to the KL divergence result when compared to other model groups, as the score of Zbot and Zeroaccess is lowest. However, this might only explain part of the reason as other components, as the probabilities of transiting between states would also affect the scoring process.

In this experiment, using entropy as features works well in malware classification.

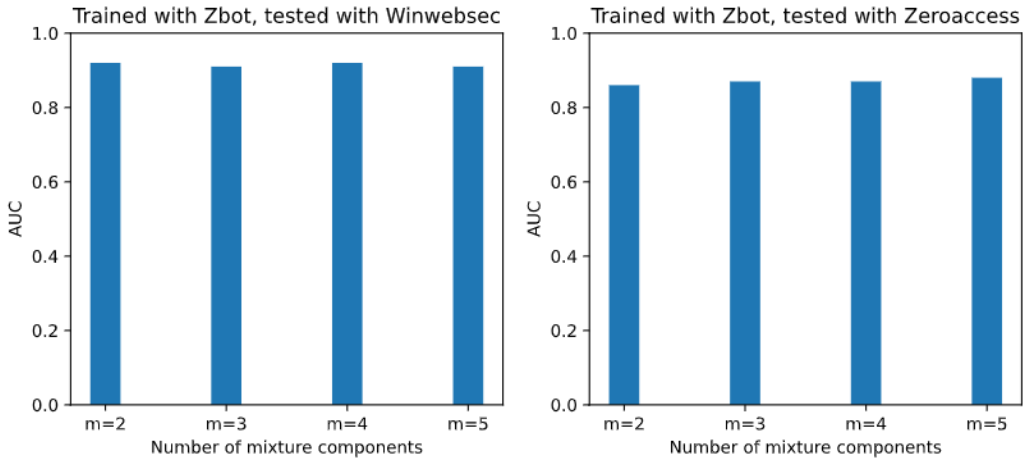
Models	**KL divergence**
KL(Zbot, Zeroaccess)	611.58
KL(Zbot, Winwebsec)	1438.42
KL(Zeroaccess, Winwebsec)	1295.79

Table 8: The KL divergence of different models

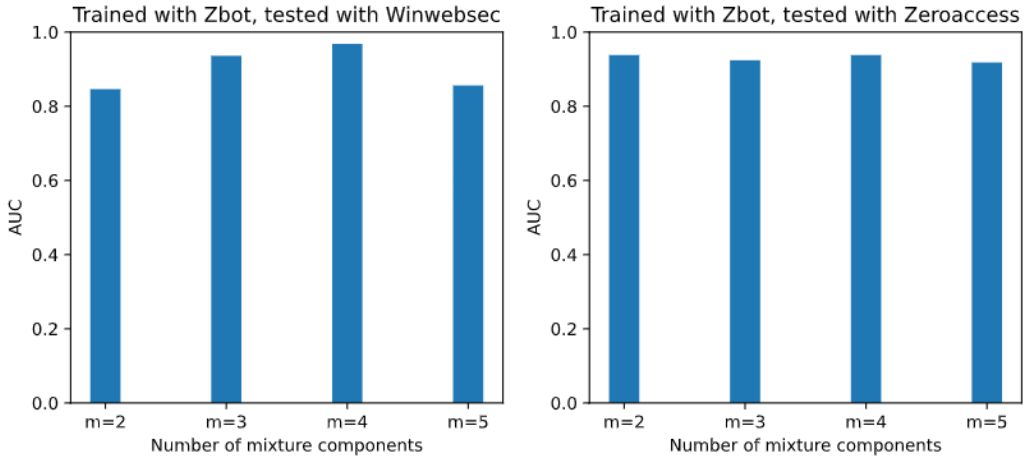
Models	$\mathbf{KL(model_1 \| model_2)}$	$\mathbf{KL(model_2 \| model_1)}$
Zbot versus Zeroaccess	504.54	718.60
Zbot versus Winwebsec	864.97	2011.86
Zeroaccess versus Winwebsec	1797.33	53.79

Table 9: The KL divergence of different models in detail

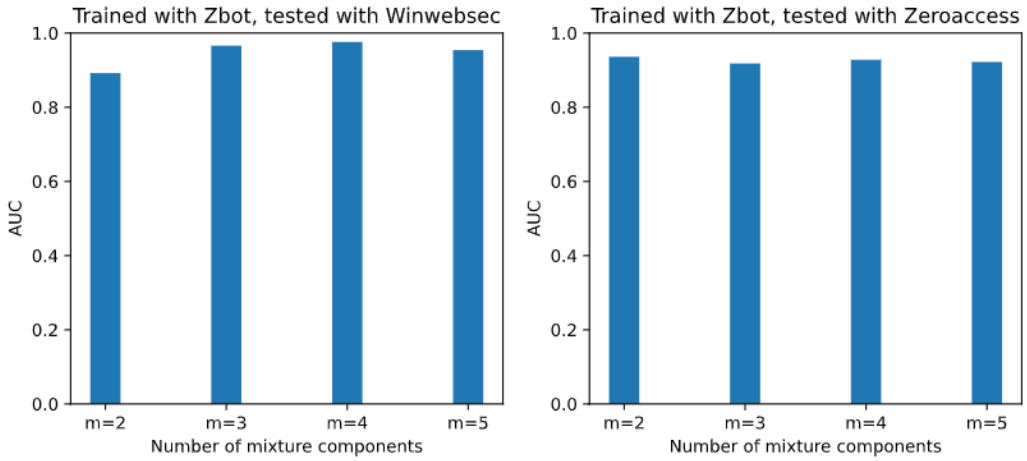
Nevertheless, there is a potential problem with entropy-based method. The value of entropy is solely calculated based on the frequency of bytes. The content, or the meaning of each byte does not affect the value of entropy. With an intricate manipulation, it might be possible that two different sequences in terms of the context receive similar scores due to a matching symbols appearance pattern.



(a) Window size = 512

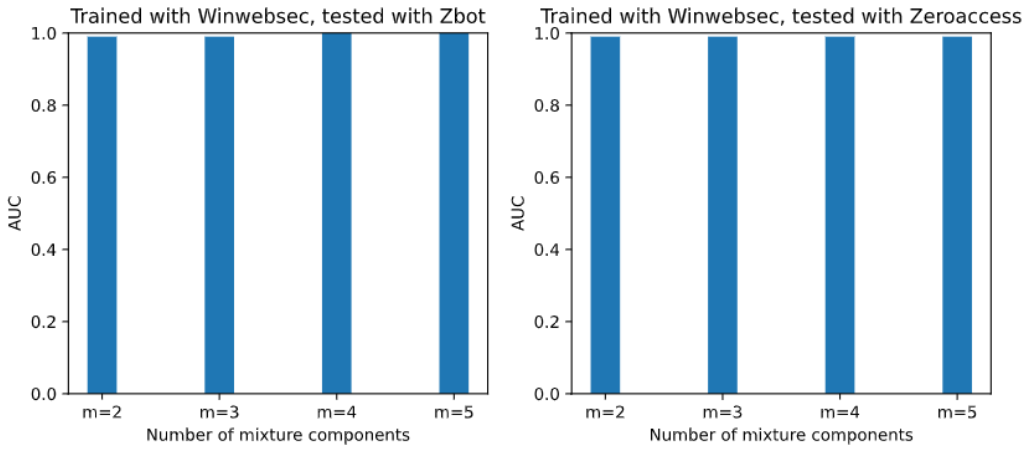


(b) Window size = 256

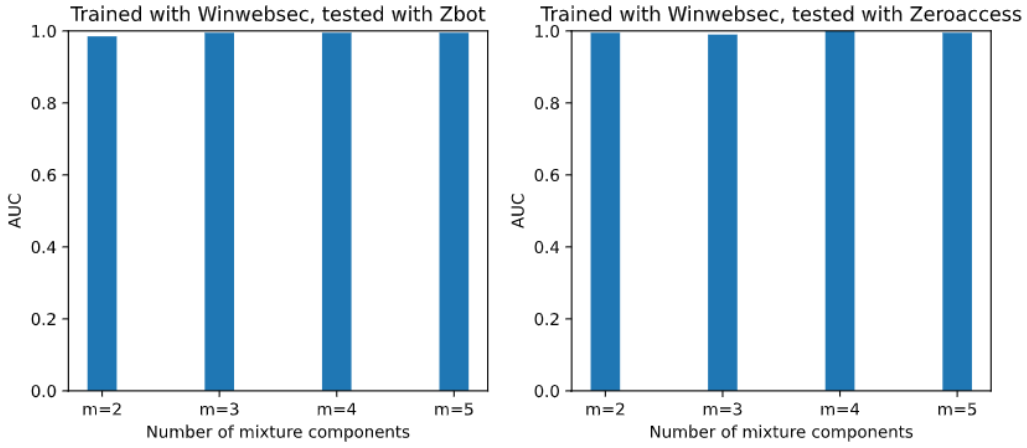


(c) Window size = 128

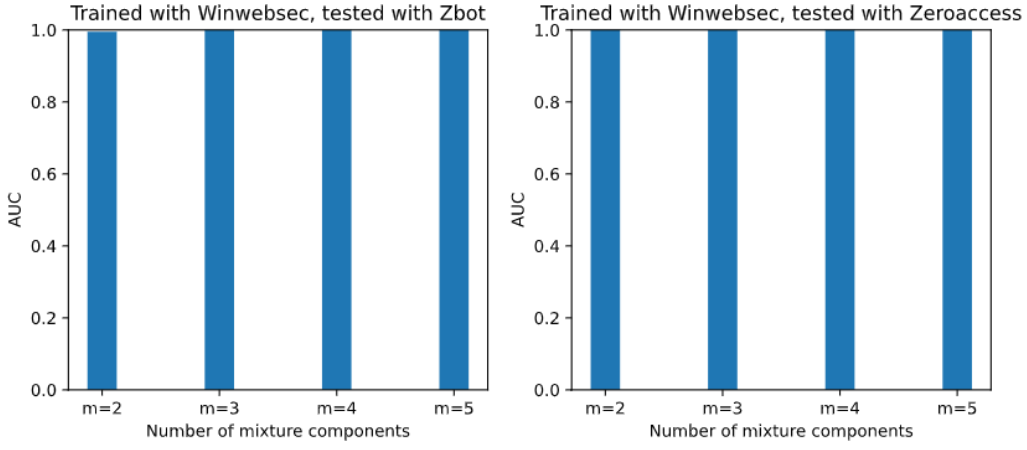
Figure 9: Entropy versus window size --- Zbot



(a) Window size = 512

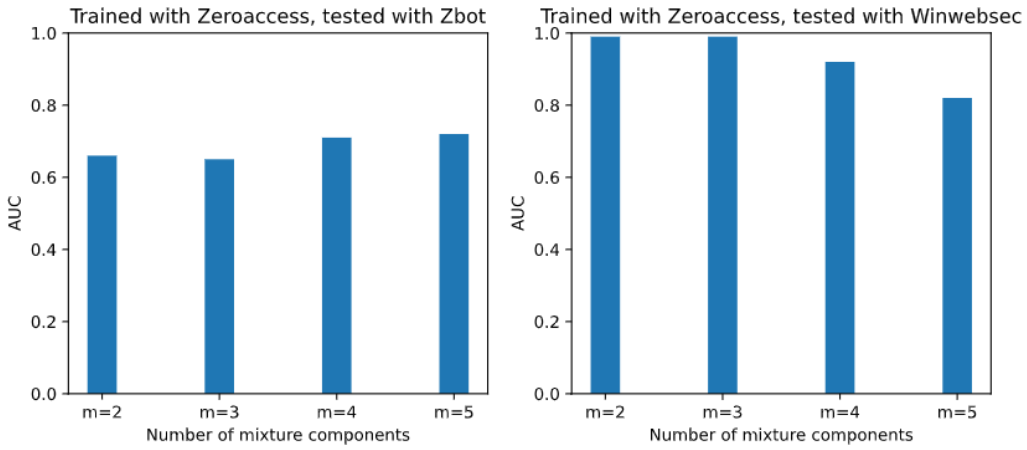


(b) Window size = 256

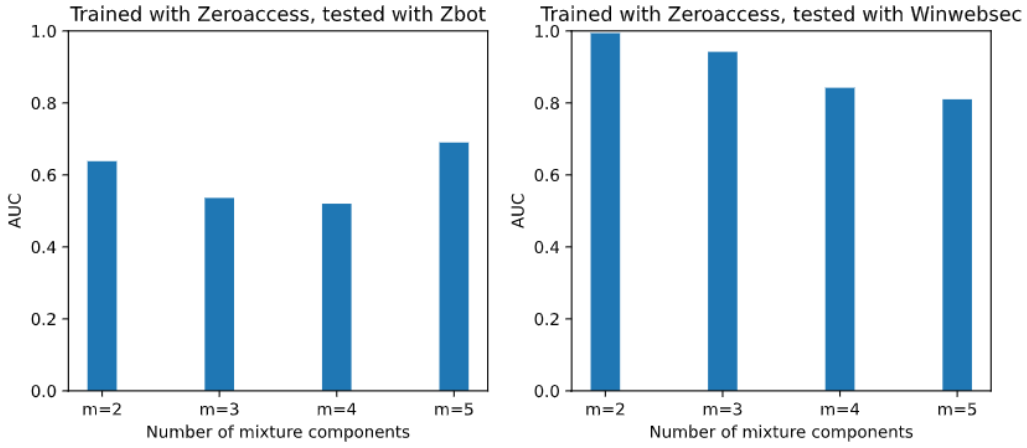


(c) Window size = 128

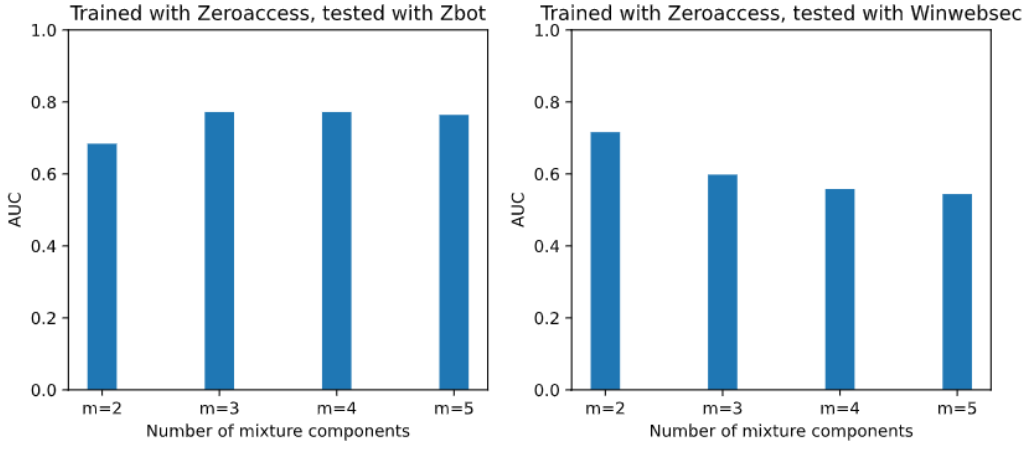
Figure 10: Entropy versus window size --- Winwebsec



(a) Window size = 512



(b) Window size = 256



(c) Window size = 128

Figure 11: Entropy versus window size --- Zeroaccess

CHAPTER 6

Conclusion and Future Work

In this work, we have explored the algorithm of building a GMM-HMM and have applied GMM-HMMs on malware classification using opcodes and entropy. As an extension to the discrete HMM, GMM-HMMs share a lot in common with HMMs, including the model building procedure and the scoring process. The difference is that GMM-HMMs use GMMs to characterize the distribution of the underlying data. This provides benefits to this model as well as some drawbacks.

For discrete features, such as the sequence of English text, or malware opcodes, building a GMM-HMM is essentially a process of simulating the discrete probability distribution with GMMs. As GMM-HMMs has strong flexibility by varying the number of components and adjusting the weights of each, it is able to capture the discrete distribution of observing symbols as HMMs, resulting in a comparable model performance. However, GMM-HMMs doesn't perform well when the number of the mixture components is not selected properly. For instance, if there are not enough mixture components, the model may fail in capturing the full picture of the distributions. Finding a proper number of mixtures needs prior understanding of the data. Otherwise, it is more of a tuning process to figure out the best choice. Nevertheless, HMMs can play a role as a benchmark providing a reference to the result. Another issue for GMM-HMMs is that it can easily get over-fitted when the observations are highly collapsed. This type of data distribution holds a small covariance, which results in obtaining a large value in the PDF. When this happens, a model receiving a high scores does not ensure to generalize well from training to testing. As part of the future work, a proper over-fitting handling is needed for training with

discrete features.

On the other hand, GMM-HMMs has shown its power for entropy-based features while doing malware classification. GMM-HMMs has the advantage of better capturing the dynamics of continuous data. Moreover, we can easily examine the data distribution in different states and use that to compare GMM-HMMs using KL divergence. Despite of the flexibility of GMM-HMMs, it takes more efforts to select proper parameters such as the number of mixture components. Having an insight of the data distribution in advance would be beneficial for improving efficiency. One thought that is worth trying in the future work is clustering the sample data to receive an overview estimation of the distribution. Although it cannot inform which states the cluster might belong to, it provides a meaningful estimation to initialize the number of mixtures, and even indicate the mean values.

To fully take the advantage of GMM-HMMs, selecting proper features is essential. In this study, we use entropy as features to train models. One potential issue about entropy is that it only considers the frequency of bytes in a fixed window size, no matter what content it represents. As a byproduct of using entropy, we need to experiment extensively to select the best choice for window size, the sliding step, etc, to figure out the best combinations. In the future work, we can explore other continuous features.

The purpose of experimenting with GMM-HMMs is to extend the application of HMMs to data with more complexity. For techniques that use HMMs as part of the scoring method, or training method, they can replace HMMs by GMM-HMMs to see if it can boost the performance. In addition, combining GMM-HMMs with neural networks is another plausible technical trend, as GMM-HMMs are capable of handling continuous data type.

LIST OF REFERENCES

[1] ''How much will remote work continue after the pandemic? - harvard business school working knowledge,'' https://hbswk.hbs.edu/item/how-much-will-remote-work-continue-after-the-pandemic, (Accessed on 11/25/2020).

[2] J. Aycock, *Computer Viruses and Malware*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[3] U. Bayer, A. Moser, C. Kruegel, and E. Kirda, ''Dynamic analysis of malicious code,'' *Journal in Computer Virology*, vol. 2, pp. 67--77, 08 2006.

[4] ''Casestudy-002-v2.pub,'' https://us-cert.cisa.gov/sites/default/files/recommended_practices/CaseStudy-002.pdf, (Accessed on 11/25/2020).

[5] N. Milošević, ''History of malware,'' https://arxiv.org/abs/1302.5392, 2013.

[6] M. Stamp, ''A revealing introduction to hidden Markov models,'' https://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf, 2018.

[7] I. Ben Abdel Ouahab, M. Bouhorma, A. A. Boudhir, and L. El Aachak, ''Classification of grayscale malware images using the k-nearest neighbor algorithm,'' in *Innovations in Smart Cities Applications*, 3rd ed., M. Ben Ahmed, A. A. Boudhir, D. Santos, M. El Aroussi, and İ. R. Karas, Eds. Springer, 2020, pp. 1038--1050.

[8] M. Kruczkowski and E. N. Szynkiewicz, ''Support vector machine for malware analysis and classification,'' in *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, ser. WI-IAT '14, 2014, pp. 415--420.

[9] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, ''Malware classification with deep convolutional neural networks,'' in *2018 9th IFIP International Conference on New Technologies, Mobility and Security*, ser. NTMS, 2018, pp. 1--5.

[10] D. Reynolds, ''Gaussian mixture models,'' in *Encyclopedia of Biometrics*, S. Z. Li and A. K. Jain, Eds. Springer, 2015, pp. 827--832.

[11] G. J. McLachlan and D. Peel, *Finite mixture models*. New York: Wiley Series in Probability and Statistics, 2000.

[12] L. R. Rabiner, ''A tutorial on hidden Markov models and selected applications in speech recognition,'' *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257--286, 1989.

[13] P. Bansal, A. Kant, S. Kumar, A. Sharda, and S. Gupta, ‘‘Improved hybrid model of hmm/gmm for speech recognition,’’ in *International Conference on Intelligent Information and Engineering Systems*, ser. INFOS 2008, 2008.

[14] A. M. Interrante-Grant and D. Kaeli, ‘‘Gaussian mixture models for dynamic malware clustering,’’ https://coe.northeastern.edu/wp-content/uploads/pdfs/coe/research/embark/4-interrante-grant.alex_final.pdf, 2018.

[15] E. Gandotra, D. Bansal, and S. Sofat, ‘‘Malware analysis and classification: A survey,’’ *Journal of Information Security*, vol. 05, pp. 56--64, 01 2014.

[16] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, ‘‘A survey on automated dynamic malware-analysis techniques and tools,’’ *ACM Comput. Surv.*, vol. 44, no. 2, Mar. 2008. [Online]. Available: https://doi.org/10.1145/2089125.2089126

[17] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, ‘‘Data mining methods for detection of new malicious executables,’’ in *Proceedings 2001 IEEE Symposium on Security and Privacy. S P 2001*, 2001, pp. 38--49.

[18] J. Z. Kolter and M. A. Maloof, ‘‘Learning to detect malicious executables in the wild,’’ in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’04. New York, NY, USA: Association for Computing Machinery, 2004, p. 470–478. [Online]. Available: https://doi.org/10.1145/1014052.1014105

[19] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, ‘‘Malware images: Visualization and automatic classification,’’ in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, ser. VizSec ’11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2016904.2016908

[20] G. Dahl, J. Stokes, L. Deng, and D. Yu, ‘‘Large-scale malware classification using random projections and neural networks,’’ in *Proceedings IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013. [Online]. Available: https://www.microsoft.com/en-us/research/publication/large-scale-malware-classification-using-random-projections-and-neural-networks/

[21] G. McLachlan and D. Peel, *Finite Mixture Models*. Wiley, 2004.

[22] M. Alfakih, M. Keche, H. Benoudnine, and A. Meche, ‘‘Improved Gaussian mixture modeling for accurate Wi-Fi based indoor localization systems,’’ *Physical Communication*, vol. 43, 2020.

[23] Z. Gao, Z. Sun, and S. Liang, ‘‘Probability density function for wave elevation based on Gaussian mixture models,’’ *Ocean Engineering*, vol. 213, 2020.

[24] Y. Chen and W. Wu, ‘‘Separation of geochemical anomalies from the sample data of unknown distribution population using gaussian mixture model,’’ *Computers & Geosciences*, vol. 125, pp. 9--18, 2019.

[25] M. Raitoharju, A. García-Fernández, R. Hostettler, R. Piché, and S. Särkkä, ‘‘Gaussian mixture models for signal mapping and positioning,’’ *Signal Processing*, vol. 168, p. 107330, 2020.

[26] J. Gallop, ‘‘Facies probability from mixture distributions with non-stationary impedance errors,’’ in *SEG Technical Program Expanded Abstracts 2006*. Society of Exploration Geophysicists, 2006, pp. 1801--1805.

[27] J. Qiao, X. Cai, Q. Xiao, Z. Chen, P. Kulkarni, C. Ferris, S. Kamarthi, and S. Sridhar, ‘‘Data on MRI brain lesion segmentation using k-means and Gaussian mixture model-expectation maximization,’’ *Data in Brief*, vol. 27, 2019.

[28] Guoning Hu and DeLiang Wang, ‘‘Monaural speech segregation based on pitch tracking and amplitude modulation,’’ *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1135--1150, 2004.

[29] I. Stanculescu, C. K. I. Williams, and Y. Freer, ‘‘Autoregressive hidden Markov models for the early detection of neonatal sepsis,’’ *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 5, pp. 1560--1570, 2014.

[30] S. Dang, S. Chaudhury, B. Lall, and P. K. Roy, ‘‘Learning effective connectivity from fMRI using autoregressive hidden Markov model with missing data,’’ *Journal of Neuroscience Methods*, vol. 278, pp. 87--100, 2017.

[31] S. Laraba and J. Tilmanne, ‘‘Dance performance evaluation using hidden Markov models,’’ *Computer Animation and Virtual Worlds*, vol. 27, no. 3-4, pp. 321--329, 2016.

[32] A. Truong and T. Zaharia, ‘‘Laban movement analysis and hidden Markov models for dynamic 3D gesture recognition,’’ *EURASIP Journal on Image and Video Processing*, vol. 2017, 2017.

[33] C. Annachhatre, T. H. Austin, and M. Stamp, ‘‘Hidden markov models for malware classification,’’ *Journal of Computer Virology and Hacking Techniques*, vol. 11, pp. 59--73, 2014.

[34] ‘‘[1901.07312] malware detection using dynamic birthmarks,’’ https://arxiv.org/abs/1901.07312?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+arxiv%2FQSXk+%28ExcitingAds%21+cs+updates+on+arXiv.org%29, (Accessed on 11/27/2020).

[35] Z. Yao, J. Ge, Y. Wu, X. Lin, R. He, and Y. Ma, ''Encrypted traffic classification based on Gaussian mixture models and hidden Markov models,'' *Journal of Network and Computer Applications*, vol. 166, 2020.

[36] F. Zhang, S. Han, H. Gao, and T. Wang, ''A Gaussian mixture based hidden Markov model for motion recognition with 3D vision device,'' *Computers & Electrical Engineering*, vol. 83, 2020.

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

[38] L. Nguyen, ''Continuous observation hidden Markov model,'' *Revista Kasmera*, vol. 44, no. 6, pp. 65--149, 2016.

[39] B. Juang, ''Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains,'' *AT&T Technical Journal*, vol. 64, no. 6, pp. 1235--1249, 1985.

[40] A. P. Bradley, ''The use of the area under the ROC curve in the evaluation of machine learning algorithms,'' *Pattern Recognition*, vol. 30, no. 7, pp. 1145--1159, 1997.

[41] R. L. Cave and L. P. Neuwirth, ''Hidden Markov models for English,'' in *Hidden Markov Models for Speech*, J. D. Ferguson, Ed., 1980.

[42] ''The Brown corpus of standard American English,'' http://www.cs.toronto.edu/~gpenn/csc401/a1res.html.

[43] A. Nappa, M. Z. Rafique, and J. Caballero, ''The MALICIA dataset: Identification and analysis of drive-by download operations,'' *International Journal of Information Security*, vol. 14, no. 1, pp. 15--33, 2015.

[44] ''Win32/winwebsec threat description - Microsoft security intelligence,'' https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win32/Winwebsec, 2017.

[45] ''Pws:win32/zbot threat description - Microsoft security intelligence,'' https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=PWS%3AWin32%2FZbot, 2017.

[46] A. Neville and R. Gibb, ''ZeroAccess Indepth,'' https://docs.broadcom.com/doc/zeroaccess-indepth-13-en, 2013.

[47] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[48] D. Baysa, R. Low, and M. Stamp, ‘‘Structural entropy and metamorphic malware,’’ *Journal of Computer Virology and Hacking Techniques*, vol. 9, no. 4, pp. 179--192, 2013.

[49] R. Togneri and C. J. S. DeSilva, *Fundamentals of Information Theory and Coding Design*. CRC Press, 2003.

[50] J. M. Joyce, ‘‘Kullback-Leibler divergence,’’ in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Springer, 2011, pp. 720--722.