Spring 5-25-2015

A Comparison of Clustering Techniques for Malware Analysis

Swathi Pai
San Jose State University

A Comparison of Clustering Techniques for Malware Analysis

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

Swathi Pai

May 2015

The Designated Project Committee Approves the Project Titled

A Comparison of Clustering Techniques for Malware Analysis

by

Swathi Pai

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2015

Dr. Mark Stamp	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science
Fabio Di Troia	Università del Sannio

ABSTRACT

A Comparison of Clustering Techniques for Malware Analysis

by Swathi Pai

In this research, we apply clustering techniques to the malware detection problem. Our goal is to classify malware as part of a fully automated detection strategy. We compute clusters using the well-known K-means and EM clustering algorithms, with scores obtained from Hidden Markov Models (HMM). The previous work in this area consists of using HMM and K-means clustering technique to achieve the same. The current effort aims to extend it to use EM clustering technique for detection and also compare this technique with the K-means clustering.

ACKNOWLEDGMENTS

I am highly indebted to Dr. Mark Stamp for his continuous guidance and constant supervision during my tenure as a student at SJSU, and specially while working on this project. His patience, right resolutions to questions and the very fact of he being there for you, helped me complete this project.

I would like to express my special gratitude and thanks to Fabio Di Troia for his attention and time. His help with disassembling of malware executables and extracting op codes from disassembled files helped me tremendously.

I would also like to thank Dr. Thomas Austin for his valuable inputs and guidance.

Finally, I would like to thank all of my family and friends, who in many ways helped me in my journey of Masters.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Malware is short for malicious software. Malware can perform malicious activities ranging from interfering in computer operation, gathering unauthorized confidential data and breaking into a private network. Malware are categorized by their intent and generally do not include software that accidentally perform malicious acts.

History of malware can be back tracked as far as 1949 when an underlying theory of self-reproducing automatons was first developed by John von Neuman [41, 18]. It was not technically exploited to a great extent in those days due to various limitations. However, in the latter half of the century, this theory was implemented by many, in both ethical and un-ethical ways. This concept was further developed to explore space design as well as help many space missions. On the flip side, it was negatively exploited by many to perform malicious acts, which is an underlying concept for malware.

According to Symantec's Annual Security Report in 2014, malicious attacks with greater than 10 million confidential data exposed, increased by 700% from 2012 to 2013. The total number of such attacks was 8 in 2013. Further analysis reports that there were 253 total breaches in 2013 (62% more than 2012) and 552 million total exposed identities (493% more than 2012). Most of these attacks were generally delivered via a phishing email to a set of employees within an organization, with an ultimate intention of developing a backdoor to the organization's data [20].

The year of 2014 has seen major attacks of magnitudes higher than prior years. Hackers have breached security of major corporations like Target, Home Depot and

Sony Corporation, to name some. These attacks have been estimated to incur a cumulative loss of more than 210 million dollars and close to 125 million customer data was exposed [31]. In most case, attacks were attributed to point-of-sales malware (BlackPOS). This malware is designed to extract data from the core payment processing system. It integrates with the system and tracks the data until it is decrypted and stored in RAM. Every payment processing has to be decrypted and this is exploited by the POS malware [24].

This emphasizes the fact that malware attacks are growing bigger and the incurred losses are increasing exponentially. Hence, it is the need of the hour to be able to quickly contain an attack. One of the ways is by categorization of the malware. Categorization of malware will help respond to a threat quickly, since the removal technique of same is known due to past history of similar malware. If the malware cannot be categorized in the existing categories, then it would not fit in known removal techniques and a new one would need to be worked upon.

Clustering is one of the approaches that may help to us to categorize the malware. Clustering is a widely used classification mechanism that categorizes observations or data elements. It is one of the popular data mining and statistical data analysis techniques [1, 21]. Cluster Analysis can be done using different algorithms which vary in the methods for determining what belongs to a particular cluster and how proficiently it can find those clusters [36].

In this project, we will be implementing different clustering techniques. Some previous work related to K-means clustering technique for malware detection has been included in [4]. We will be using well-known K-means and EM clustering algorithms for classification of the malware files [13, 15]. These techniques will be tested on approximately 8000 malware samples. The results obtained from all these clus-

tering techniques will be then compared using internal and external validations for measuring cluster quality like silhouette co-efficient [36]. The final goal is to develop an automated malware detection mechanism which will categorize any new malware under its appropriate family without the need to retrain for new malware families.

In Chapter 2, we consider different types of malware and the different detection techniques. Chapter 3 gives an overview of the clustering algorithms. Chapter 4 gives an overview of the previous work on the malware classification. Chapter 5 includes details regarding the implementation of EM clustering. Chapter 6 includes the experiments and the results obtained. Chapter 7 is the last chapter that concludes the paper and provides scope for future work.

CHAPTER 2

Malware Types and Detection

Once installed, malware can seriously affect the privacy and security of the system. While some of them are just for annoyance, others can pose a serious threat to the system. Each different type of malware, offers their own unique threat. Hence, it is important to have a clear understanding of malware classification because knowing how various types of malware spread, is vital in removing it. Malware can consist of viruses, worms, Trojan horses, adware, rootkits and many other nasty infections.

2.1 Malware Types

2.1.1 Virus

Virus is a form of malware that requires a medium to propagate. It spreads by attaching to various programs and then executes the code when the user launches one of these infected programs. The host program functions even after it has been infected by a virus [33]. Only in some cases, the virus overwrites host programs with copies of themselves, thereby destroying the host program completely. Viruses can also propagate through script files, documents and cross-site scripting vulnerabilities in applications. Some viruses can also steal sensitive information, damage data and can cause denial-of-service attacks [6].

2.1.2 Worms

Another most common type of malware is the computer worm. Worms are standalone software programs that do not need any host program to propagate its infection. Worms normally spread by exploiting vulnerability on a target system [6].

The worm spreads it's infection by first entering into a system through some vulnerability and then taking the benefit of information and file transferring features which allows it to propagate independently. As compared to viruses, whose target is very local and limited to one system, worms tend to propagate and can potentially disrupt the entire network [36].

2.1.3 Trojan horse

A Trojan horse is a type of malware that masks itself as a normal file. It appears to be completely harmless at first glance. Once activated, it can attain a large number of attacks on the host, from annoying the user by popping up windows to damaging the system by deleting files, stealing data, etc [40]. Trojans can also generate back doors through which malicious users can access the system. Once malicious user gets access to the system, then he can perform malicious activities like stealing data, installing malware, monitoring user activity, using the computer in botnets, etc. Hence, a Trojan may be capable of almost any type or level of harm [37].

2.1.4 Spyware

A spyware is a type of malware that monitors user's activities without his knowledge. It then, transmits the gathered information to third parties. The gathered information can include the websites visited, browser and system information, etc. Spyware does not have any infection mechanisms and is usually released on the victim's system by Trojans. Spyware can also take control over the system in a manner similar to that of a Trojan horse. Spyware also propagates by misusing software vulnerabilities, wrapping itself with legitimate software, or in Trojans [9].

2.1.5 Rootkit

A rootkit is a type of malware that gains remote access or control of the system without being detected. After the installation of rootkit, malicious users can remotely execute the files or steal sensitive information, modify software, etc. Rootkits are generally hard to detect. Since rootkits repeatedly conceals its presence, most of the security products fail to detect it and remove it from the system. As a result, their detection requires manual methods like monitoring computer behavior for anomalous activities or storage dump analysis. Software security products can use different detection techniques like signature based detection or a heuristic based detection technique for detecting the rootkits. Since rootkits continuously hide their presence, several efficient detection methods need to be combined [28].

2.2 Malware Detection Techniques

2.2.1 Signature based detection

The most common method used in malware detection software is signature-based detection. A signature is a string of bits that uniquely identifies a specific virus. Signature-based detection scans through the computer files and cross-check their contents against a dictionary of virus signatures [19]. On identifying the virus signature, antivirus software can work towards removing the virus. It can quarantine, repair or delete the infected file and resolve the virus issue. However, this technique can identify a small subset of emerging threats. It is not capable of identifying new viruses.

2.2.2 Anomaly Detection

Anomaly Detection is responsible for observing any unusual or possibly malicious activity or behavior. Anomaly detection can be described as an alarm for strange system behavior. In [14], the author describes building an "activity profile" of normal usage over an interval of time. Once in place, the profile is compared against real time events. Anything that deviates from the baseline, or the norm, is logged as anomalous.

This process can be done in two phases—training and detection phase. In first phase, the software is trained to learn the normal behavior of the host. The detection phase then uses the trained model to recognize any unexpected or unusual behavior that happens on the system. One of the advantages of anomaly detection is that we can anticipate to detect previous unknown malware [36].

2.2.3 Change Detection/Integrity Checking

The main purpose of integrity checking is to detect any changes in the files that could have been caused due to malware. Hence, to achieve this, hashes of all the files are computed on the system and then these hashes are periodically monitored. If a file has been changed, then the computed hash will not match with the previously computed hash [36].

2.2.4 Hidden Markov Model for Malware Detection

As per [29], an HMM is a doubly stochastic process with an under-lying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols. As mentioned in [29], following are the notations used in hidden Markov models:

T = length of the observations sequence

N = number of state in the model

M = number of distinct observation symbols

Q = distinct states of the Markov model

V = set of possible observations

A = state transition probability matrix

B = observation probability matrix

π = initial state distribution

$\mathcal{O}$ = observation sequence

A hidden Markov model is denoted as $\lambda = (A, B, \pi)$. Figure 1 represents a generic view of a hidden Markov model.

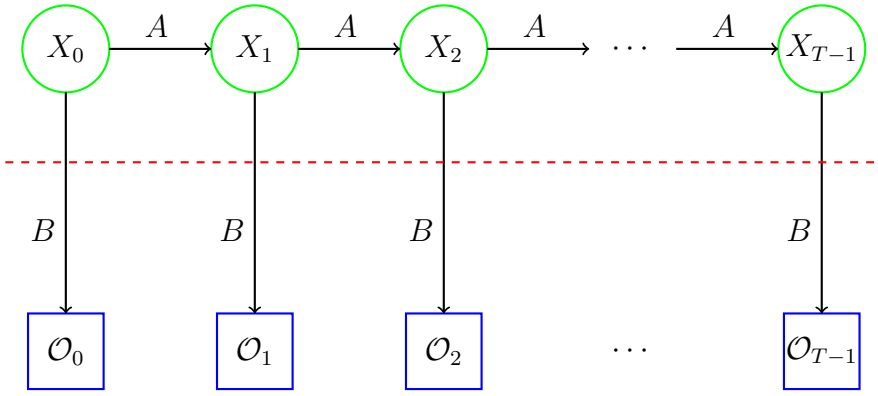

Figure 1: Hidden Markov Model

HMM can be used for solving the below three problems efficiently [35].

Problem 1: Given a model $\lambda = (A, B, \pi)$ and an observation sequence $\mathcal{O}$, we compute $P(\mathcal{O}|\lambda)$. This technique can be used to score a given observation sequence on

how closely it fits a given model.

Problem 2: Given a model $\lambda = (A, B, \pi)$ and an observation sequence $\mathcal{O}$, we find the most likely hidden state sequence that satisfies the given model.

Problem 3: Given observation sequence $\mathcal{O}$, number of states N and number of distinct observation symbols M, we find the model $\lambda = (A, B, \pi)$ that maximizes the probability of given observation sequence.

In this project, we will be using HMM for training and scoring the malware files. We train the HMM on a set of particular family of malware files. As a result, this is similar to Problem 3. Also, we will be using the model generated from the training phase to score the remaining malware files of same and different families. Hence, in this case, we are in similar situation as Problem 1. As a result, we will be using algorithms that would be solving Problem 1 and Problem 3. The above three problems can be solved efficiently by the solutions such as forward algorithm, backward algorithm or Baum-Welch algorithm.

2.2.4.1 Forward Algorithm

The forward algorithm can be used as a solution to Problem 1. Given the model as $\lambda = (A, B, \pi)$ and the observation sequence as $\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, ...\mathcal{O}_{T-1})$, we need to find $P(\mathcal{O}|\lambda)$.

To find $P(\mathcal{O}|\lambda)$, the forward algorithm is used. For $t = 0, 1, ..., T-1$ and $i = 0, 1, ..., N-1$, define

$$\alpha_t(i) = P(\mathcal{O}_0, \mathcal{O}_1, ..., \mathcal{O}_t, x_t = q_i|\lambda)$$

$\alpha_t(i)$ is the probability of the partial observation sequence up to time t, where the Markov process in the state q_i at time t.

In above equation, $\alpha_t(i)$ can be computed recursively as follows:

1. Let $\alpha_0(i) = \pi_i b_i(\mathcal{O}_0)$, for $i = 0, 1, ..., N-1$

2. For $t = 1, 2, ..., T-1$ and $i = 0, 1, ..., N-1$, compute

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

3. Hence, finally we get

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

2.2.4.2 Backward Algorithm

The backward algorithm can be used to solve Problem 2. It is very similar to forward algorithm with the only difference that its starts from back way towards the beginning. Given the model $\lambda = (A, B, \pi)$ and a sequence of observations $\mathcal{O}$, aim is to find the most likely state sequence. For $t = 0, 1, ..., T-1$ and $i = 0, 1, ..., N-1$, define

$$\beta_t(i) = P(\mathcal{O}_{t+1}, \mathcal{O}_{t+2}, ..., \mathcal{O}_{T-1}|x_t = q_i, \lambda)$$

To compute $\beta_t(i)$ recursively, following are the steps:

- Let $\beta_{T-1}(i) = 1$, for $i = 0, 1, ..., N-1$.

- For $t = T-2, T-3, ..., 0$ and $i = 0, 1, ..., N-1$ compute

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

For $t = 0, 1, ..., T-2$ and $i = 0, 1, ..., N-1$, define

$$\gamma_t(i) = P(x_t = q_i|\mathcal{O}, \lambda).$$

Since $\alpha_t(i)$ measures the relevant probability up to time t and $\beta_t(i)$ measures the relevant probability after time t,

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(\mathcal{O}|\lambda)}$$

2.2.4.3 Baum-Welch Algorithm

The most amazing aspect of HMM is to efficiently re-estimate the model itself [4]. The Baum-Welch algorithm helps to achieve this and can be used as solution to problem 3. For $t = 0, 1, .., T-2$ and $i, j \in 0, 1, ..., N-1$, define "di-gammas" as

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

Re-estimation is am iterative process. It begins by initialization of the model $\lambda = (A, B, \pi)$ with a best guess where we start by randomly choosing values such that $\pi_i \approx 1/N$ and $a_{ij} \approx 1/N$ and $b_j(k) \approx 1/M$. The solution to Problem 3 can be summarized as below:

- Initialize $\lambda = (A, B, \pi)$.
- Compute $\alpha_t(i), \beta_t(i), \gamma_t(i, j)$ and $\gamma_t(i)$.
- Re-estimate the model $\lambda = (A, B, \pi)$
- If $P(\mathcal{O}|\lambda)$ increases, goto 2.

When a HMM is trained, it has proved that it can distinguish between the malware and benign files. A lot of previous work has been done where HMM has been used for malware detection [5, 43]. This project also uses HMM for training and scoring the malware files. Based on the scores obtained from the HMM trained

model, we will be using clustering techniques like K-means and EM for classifying the malware files.

CHAPTER 3

Clustering Techniques

Clustering is a widely used classification mechanism that categorizes observations or data elements. It is one of the popular data mining and statistical data analysis techniques where the objective is to learn something from the data [7]. Cluster Analysis can be done using different algorithms which vary in the methods for determining what belongs to a particular cluster and how proficiently it can find those clusters. For example, some algorithms performing clustering based on the distance among the cluster members whereas some use statistical distributions of the cluster to determine the clusters [1].

A key element in cluster analysis is the grouping, or classification of measurements based on either (i) goodness-of-fit to a postulated model, or (ii) natural groupings (clustering) revealed through analysis [21]. It is expected that elements belonging to a valid cluster are more related to each other than they are to a elements belonging to a different cluster.

There are different ways in which the clustering techniques can be performed.

- Intrinsic approach: In this approach, raw data has been used, that is, the class labels for the data are missing and we don't have any information about the data. This can be considered as unsupervised learning.
- Extrinsic approach: In this approach, the data has class labels. Hence, this technique can be considered as supervised learning.

Also, clustering can be agglomerate or divisive. In an agglomerate approach,

we start with many cluster and then converge to less larger clusters. Hence, this can be considered as a bottom up approach. In a divisive approach, we start with less number of large clusters and then slowly divide them into more cluster. Hence, can be equated as top down strategy [30].

In this paper, we are mainly focusing on K-means and EM clustering technique. Our goal is to analyze these well-known algorithms for malware classification.

3.1 K-Means

K-means algorithm is one of the most popular clustering techniques. When given a set of x data points, the algorithm partitions data into k mutually exclusive clusters [2]. The main goal is to assign a cluster to each of the data point. This method aims to find the means of the clusters such that the distance between the data point to the cluster is minimized. Hence, the problem that we want to solve can be stated as

$$\text{Given: } K \text{ and data points } x_1, x_2, ..., x_m$$

$$\text{Minimize: } \text{distortion}_K = \sum_{i=1}^{m} d(x_i, \text{centroid}(x_i)).$$

The above problem does not guarantee exact solution. However, we can definitely expect an efficient approximate solution.

There are different ways in which the distance between the data points and cluster means can be computed. Below are some of the techniques that can be used for distance calculation.

- Euclidean distance: The Euclidean distance between two points namely $x =$

$(x_1, x_2, x_3, ..., x_m)$ and $y = (y_1, y_2, y_3, ..., y_m)$ can be given as follows

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$$

- Manhattan distance: The Manhattan distance, also known as taxicab distance, is the shortest distance allowing movements parallel to the axes.

$$d(x, y) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|$$

- Mahalanobis distance: This is the statistical measure that accounts for mean and covariances.

Euclidean method is mostly widely used and in our experiments as well, we will be using this technique for distance calculation in K-means algorithm.

K-means algorithm is an iterative process that alternates between two important steps.The two basic steps in this algorithm are as follows:

Step 1: Assign each data point to its nearest cluster.

Step 2: Compute the centroids/means of the cluster based on the points belonging to the cluster.

The above two alternating steps form an iterative process that generate a series of solutions that improve as the algorithm proceeds through the iterations [26]. As a result, k-means can be viewed as hill climb approach. Below is the algorithm for K-means:

1. Given are the data points $x_1, x_2, ..., x_m$.
2. Randomly initialize the means for k cluster.
3. For each data point

- Calculate the distance between that point and mean of the cluster.
- Assign the data point to the cluster with minimum distance.

4. Re-compute the mean of the cluster based on the data points in that cluster.
5. Repeat Step 3, until there is no or very negligible change in cluster means.

3.2 Expectation-Maximization Clustering

Expectation-Maximization (EM) clustering is an unsupervised learning technique. EM clustering technique uses Gaussian mixture models and mixture likelihood approach to find a structure in datasets. It follows an iterative approach, which tries to find the parameters of the probability distribution that has the maximum likelihood of its attributes. It consists of two steps: Expectation step (E-step) and Maximization step (M-step). For each of the iteration, the algorithm first executes the E-step, which computes the probability of each data point belonging to a particular cluster. This step is followed by the M-step that re-estimates the cluster parameters for each cluster. The algorithm terminates when the parameters converges or the algorithm reaches the maximum number of iterations.

EM clustering is a soft clustering technique, in which each data point has a probability of being associated with every cluster. That is, the points split between the clusters based on probability. However, when the algorithm terminates, the data point is assigned to the cluster for which it has the maximum relative probability.

Steps for EM Clustering [25]:

1. Initialize the cluster parameters randomly.
2. E-Step: Re-estimate the expected values of data elements under current esti-

mate model. As a result, this step calculates the probability of a data element belonging to a particular cluster.

$$P^{\text{new}}(c_l|x_i) = P(c_l)P(x_i|c_l)$$

3. M-Step: Re-estimate the model parameters based on the maximum likelihood estimate of the complete data.

   Cluster mean:

   $$\mu_l^{\text{new}} = \frac{\sum_{i=1}^{N} x_i P^{\text{new}}(c_l|x_i)}{\sum_{i=1}^{N} P^{\text{new}}(c_l|x_i)}$$

   Cluster covariances:

   $$\mu_l^{\text{new}} = \frac{\sum_{i=1}^{N} (x_i - \mu_l^{\text{new}})(x_i - \mu_l^{\text{new}})' P^{\text{new}}(c_l|x_i)}{\sum_{i=1}^{N} P^{\text{new}}(c_l|x_i)}$$

   Cluster densities:

   $$P^{\text{new}}(c_l) = \frac{1}{N} \sum_{i=1}^{N} P^{\text{new}}(c_l|x_i)$$

4. Termination: Until convergence.

For this project, we implemented EM clustering algorithm using R language. One of the challenges faced during implementation was initialization of the clusters. There are different ways in which the initialization can be done. For this project, we performed experiments with 3 different initialization techniques. The below section gives an overview about these techniques.

3.2.1 Initialization techniques

The first step in the EM clustering involves initialization. Initialization can be done in different ways [12, 17]. In this project, three different initialization techniques have been explored. The final results obtained as result of clustering are then analyzed

to determine which initialization technique gives better results. The initialization techniques used are:

1. Initialization of cluster parameters.
2. Initialization of data elements.
3. Euclidean distance approach.

3.2.1.1 Initialization of cluster parameters

In this approach, the initial values for clusters parameters are randomly selected. That is, we start with initialization of mean, covariances and densities of the clusters randomly. Below are the steps used to randomly initialize the cluster parameters:

- *Step 1:* Initialize mean—In this step, first the minimum and maximum value for each of the columns of the datasets is calculated. The mean is randomly selected between the minimum and maximum values for each of the cluster. Table 1 is the dataset of old faithful data. Each data point has two co-ordinates.

Table 1: Old Faithful Data

Case	Duration	Wait
1	3.60	79
2	1.80	54
3	3.33	74
4	2.28	62
5	4.53	85
6	2.88	55
7	4.70	88
8	3.60	85
9	1.95	51
10	4.35	85

From Table 1, we find that the minimum and maximum value for duration is 1.8 and 4.7 respectively. Similarly, the minimum and maximum value for wait is 51 and 88 respectively. Hence, for initialization of mean, we would randomly select values between 1.8 and 4.7 for first co-ordinate and between 51 and 88 for the second co-ordinate. As a result, if we have two clusters, then the initial values of means may look like Cluster 1 mean = [2.6, 53] and Cluster 2 mean = [3.6, 79]

- *Step 2:* Initialize covariances—Since old faithful data is bivariate Gaussian dataset, the covariances matrix will be a 2×2 matrix. The covariances matrix S for bivariate Gaussian distribution can be given as

$$\begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}$$

  For initialization of covariances matrix, s_{11} and s_{22} are randomly selected. We randomly select between -1 and 1 and then the below formula is used to compute s_{12}. However, this technique is very specific to bivariate datasets and cannot be employed if the dimensions of the datasets change.

- *Step 3:* Initialize cluster densities—This is done by randomly assigning probabilities to each of the cluster such that sum of the cluster probabilities must be equal to 1. For example: If we have 3 clusters, then the cluster probabilities may be as shown below:

  Cluster 1 probability = 0.37

  Cluster 2 probability = 0.43

  Cluster 3 probability = 0.20

Once the cluster parameters have been initialized, we start with the E - step where we calculate the probability of a data element belonging to a particular cluster. The algorithm then iterates through E-step and M-step, until the clusters converge.

3.2.1.2 Initialization of data points

In this approach, we initiate by assigning data points to the cluster based on random probabilities. Since, EM uses soft clustering technique which means that each data point will be split between the clusters. For example, let us consider the old faithful data. If we consider 10 data points in this dataset and the number of clusters to be 2, then we would randomly select a relative probability of the data point with respect to Cluster 1 (say x). Then, the probability of this data point with respect to Cluster 2 would be $1 - x$. This continues for all the data points. An example of this technique can be seen in Table 2.

Table 2: Initial Assignment of Probabilities under Clusters

			Probability	
Case	Duration	Wait	Cluster 1	Cluster 2
1	3.60	79	0.83	0.17
2	1.80	54	0.34	0.66
3	3.33	74	0.42	0.58
4	2.28	62	0.67	0.33
5	4.53	85	0.56	0.44
6	2.88	55	0.76	0.24
7	4.70	88	0.20	0.80
8	3.60	85	0.01	0.99
9	1.95	51	0.23	0.77
10	4.35	85	0.14	0.86

Once this is done, we proceed with the M step and then the iterations continue, until the clusters converge.

When the initialization was done using the first technique (Initialization of cluster parameters), the algorithm delivered accurate results most of the times. However, there were some few times when it failed to deliver correct results. The number of iterations taken by the algorithm for convergence of clusters was also less in the first technique case as compared to second technique. However, the main challenge in the first technique is the initialization of covariances matrix. The covariance matrix initialization can be simple for 2-dimensional dataset. However, it becomes trickier with increase in data dimensions.

The second technique of initialization (Initialization of data points) also delivered results similar to first approach. However, the number of iterations required in this case was much more compared to the first.

Since, the above results were not satisfactory, an efficient method for cluster initialization was implemented. This method has been explained in next section.

3.2.1.3 Euclidean method for initialization

In this approach, we first randomly select mean for each of the cluster. Then, we find the relative probability of the data point to the cluster based on Euclidean distance i.e., we calculate the Euclidean distance between the data points and the means of the cluster. Lesser the distance, greater are the chances of the data point belonging to the particular cluster. Below are the steps on how the initialization is done.

- *Step 1:* Initialize mean—In this step, first the minimum and maximum value for each of the columns of the datasets is calculated. The mean is randomly selected between the minimum and maximum values for each of the cluster.

- *Step 2:* Calculate Euclidean distance—In this step, we calculate the Euclidean distance between the data points and means of each cluster. Based on this distance, we assign relative probability to the data point with respect to particular cluster. The Euclidean distance between two points p = (p_1, p_2,..., p_n) and q = (q_1, q_2,..., q_n) is calculated by the formula

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{N} (q_i - p_i)^2}$$

For example:

Let us consider following randomly selected means for Old Faithful data.

Cluster 1 mean = [2.6, 53]

Cluster 2 mean = [3.3, 80]

Now, for the first data point $x = (3.6, 79)$, Euclidean distance with mean of cluster 1 would be

$$D(x, \text{mean of Cluster1}) = \sqrt{(2.6 - 3.6)^2 + (53 - 79)^2} = 26.02$$

Similarly, the Euclidean distance of data point (3.6, 79) with mean of cluster 2 would be

$$D(x, \text{mean of Cluster2}) = \sqrt{(3.3 - 3.6)^2 + (80 - 79)^2} = 1.04$$

Hence, we can conclude that the data point is closer to cluster 2 and the probability of it belonging to cluster 2 is very high. We now calculate the probability of data point under both the clusters using the below formula.

$$P(x|\text{Cluster A}) = \frac{\text{Sum of all the distance} - D(x, \text{mean of Cluster A})}{\text{Sum of all the distance}}$$

Hence,

$$P(x|\text{Cluster 1}) = \frac{27.06 - 26.02}{27.06} = 0.038$$

$$P(x|\text{Cluster 2}) = \frac{27.06 - 1.04}{27.06} = 0.961$$

Similarly, we determine the initial probabilities of data points under each cluster and then proceed to the M step. The iterations then continue, until the clusters converge.

The results from the above approach were much better than the other two approaches. The results provided by this approach were more accurate and the number of iterations required for cluster convergence was much less compared to other two approaches. Hence, this approach was mostly used to perform the experiments.

3.3 Cluster Quality

Cluster quality can be measured using internal and external validation techniques. External validation is based on the information that we have about the data used while internal Validation is based on the characteristics of the clusters. In this project, we will be using silhouette coefficient for measuring the cluster quality. A silhouette coefficient calculation is illustrated in Figure 2.

3.3.1 Silhouette coefficient

Silhouette coefficient is an internal validation technique used to measure the quality of the cluster. Based on cohesion and separation, it computes the silhouette coefficient of the data point. Ideally, each of the cluster must be cohesive, that is the points within the cluster must be closely together. Also, the different clusters must reveal separation [3].

Suppose we have a set of data points $X = (x_1, x_2, ..., x_m)$ that have been divided into n clusters. Let these clusters be $C_1, C_2, ..., C_n$. Let M_i be the number of elements

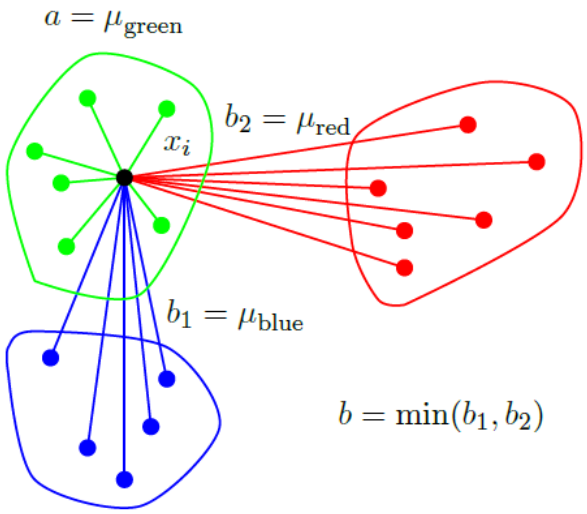

Figure 2: Silhouette Coefficient Example.

in cluster C_i. Hence, we have $M_1 + M_2 + M_3 + ... + M_n = m$.

Now, let us take a data point $x_i \in C_j$. Let a_j be average distance between x_i to all other points in that cluster.

$$a = \frac{1}{M_j - 1} \sum_{y \in C_j,\, y \neq x_i} d(x_i, y)$$

Let b_k be average distance from x_i to all the points in cluster C_k.

$$b_k = \frac{1}{M_k} \sum_{y \in C_k} d(x_i, y)$$

Let b be the smallest of all the b_k

$$b = \min_{k \in 1,2,...,n, k \neq j} b_k$$

The silhouette coefficient of X_i as

$$S(x_i) = \frac{b - a}{max(a, b)}$$

Ideally,if x_i is much closer to all the points in its cluster and well separated with the points of other clusters, then a will be very small compared to b. Hence, in such cases, $S(x_i) \approx 1$.

The average silhouette coefficient

$$s = \frac{1}{n} \sum_{i=1}^{m} S(x_i)$$

If the average silhouette coefficient is closer to 1, then a strong structure can be found. The average silhouette coefficient very well combines both cohesion and separation in one single number. We will be using silhouette coefficient for measuring clustering score. This will be detailed in Chapter 5.

CHAPTER 4

Related Work

A lot of research has been done on detection and classification of malware. This chapter includes some work done in the area of malware classification. The first section gives an overview of different data mining techniques used for categorization. The second section includes previous project work on clustering techniques.

4.1 Malware classification using Data Mining Techniques

The paper [34] was one of the first papers that introduced data mining techniques for malware classification. The three different static features for malware classification used by authors are: Portable Executable (PE), strings and byte sequences. Different features based on a variety of parameters were extracted from each Portable Executable (PE). These features were then fed into the algorithms that produced detection models. GNU strings program was used to extract successive printable characters from each file. The byte sequence is one of the most critical features as it signifies machine code. For transforming binary files into hexadecimal files, the authors used hexdump tool. RIPPER, a rule based induction algorithm was applied to find patterns in DLL data. Naive Bayes approach was used for finding patterns in string and byte sequences. Based on the results, it was observed that Naive Bayes approach gives the highest classification accuracy taking strings as input data. The authors claimed that the rate of detection of malware using data mining method is twice as compared to signature based method.

In this paper [22], the authors used different classifiers including Naive-Bayes, Support Vector Machine, Decision Tree and their boosted versions for malware classi-

fication. In this paper, the author used n-gram that were produced by combining each four-byte sequence into a single term. Using the n-grams from all of the executable, techniques from text classification were applied. They concluded that boosted decision tree gives the best classification results.

In [23], the authors present a generic framework that exploits the rich structural information inherent in malware programs for automated malware classification. The function call graph of a malware program is used to initiate the process of feature extraction. After extracting the features for each malware sample, the similarity is evaluated for two malware samples by (i) applying discriminate distance metric learning and (ii) pairwise graph matching. The authors then apply ensemble of classifiers namely (SVM) or the k-nearest neighbor classifier (kNN) which clusters the malware samples belonging to same family while keeping the different clusters separate by a marginal distance.

In paper [32], the author has introduced a framework for automatic analysis of malware behavior using machine learning. The framework was responsible for collecting large number of malware samples and monitoring their behavior in a sandbox environment. The observed behaviors were then embedded in a vector space and the learning algorithms, such as clustering and classification were applied for the analysis of malware behavior. The proposed analysis framework enables automatic and efficient analysis of malware behavior, which provides the basis for timely defense against malware development.

In paper [8], the authors have proposed a unique approach towards the automation of malware classification and analysis. In the experiments, authors implemented a dynamic approach that executes the malware in a virtualized environment and captures the behavioral fingerprints of the malware's activity. These fingerprints basically

capture all the malicious actions performed by the malware like files modifications, creation of processes, etc. The fingerprints were compared by applying single-linkage hierarchical clustering using normalized compress distance as a distance metric. This distinctive technique provided a unique way of understanding the relationship between the malware. Also, it was successful in understanding and connecting existing malware classifications.

4.2 Hidden Markov Model for Malware Classification

In the paper [4], the author proposed and evaluated K-means clustering algorithm for classifying over 9000 malware samples. These malware samples were scored using HMM from variety of compilers and malware generators (GCC, MINGW, CLANG, TURBOC, TASM and MWOR). These scores were consolidated to form a 7-dimensional matrix and given as input to the k-means clustering algorithm. The experiments were performed for values of k ranging from 2 to 15. The three dominant malware families present in this dataset were Winwebsec, Zbot and ZeroAccess. The ROC curves and AUC were computed for these three dominant families for all the different values of the k. It was observed that the AUC improved with the increase in the values of k. This suggested that with the increase in values of k, more clusters were formed that contributed to better classification of malware.

Our project is the extension of this work. However, in our project, we trained the HMMs on a four malware family (Winwebsec, Zbot, ZeroAccess and NCGVK). Initially, we combined the scores of malware samples obtained from Winwebsec and Zbot HMM to form a 2-dimensional matrix. We performed K-means clustering on these scores for values of k ranging from 2 to 10. We then included Zbot scores and performed clustering and compared these results with 2-dimensional results. Finally,

we added the scores obtained from NCGVK family as well into our experiments that results in the formation of 4-dimensional model. We evaluated this model by performing both K-means and EM clustering.

CHAPTER 5

Implementation

This chapter includes information related to the implementation of the project. The first section comprises of the information related to the malware families used for the experiments. The second section includes details about training the HMMs on these malware families. The third section gives details about testing and scoring the malware files against the generated HMM models. The fourth section is about using K-means and EM algorithm for clustering the malware files.

5.1 Setup

In this project, we have used a virtual machine for disassembling the files. Also, regular snapshots of the system were taken for backup and recovery purpose. The virtualization software used in this project is Oracle VM virtual box. Following are the specifications of the host and the guest machine:

Host:

Model: Lenevo G580

Processor: Intel Core i3-3120M CPU @ 2.50GHz

RAM: 6.00GB

System type: 64-bit OS

Operating System: Windows 8

Guest:

Software: Oracle Virtual Box 4.2.18 VMs

Base memory: 1000MB

System type: 64bit OS

Operating System: Windows 7

5.2 Malware Dataset

The dataset used for the project was requested from Malicia project website. Previous work related to Malware classification has been done using this dataset. The dataset consisted about 11000+ malware binaries. However, for the purpose of experiments our main focus is on the three dominant families Zbot, ZeroAccess and Winwebsec [27].

ZeroAccess is a Trojan horse that makes use of an advanced rootkit for hiding itself. Nevertheless, it possess the capability of creating hidden files, downloading more malware files,etc. It is able to achieve the above functions silently as it infects a system driver that acts as a rootkit hiding all of its components on the computer [39].

Zbot is also a Trojan horse that tries to steal confidential information from the systems that have been compromised. A toolkit that could be generated using the marketplaces for online criminals was used for attacking the computers. This toolkit allowed for high degree of control over the compromised computers [38].

Winwebsec programs belong to the category of Windows malware that cheat the computer users by convincing them to purchase false anti-virus software. They display false error messages indicating that the system has been infected. The malware then offers to remove these treats for some fees [42].

The malware samples are of EXE or DLL types. These files were disassembled using IDA Pro disassembler on Windows virtual machine. The opcodes were extracted

from the disassembled files. The HMMs were trained using 200 files of each dominant family namely Zbot, Winwebsec and ZeroAccess. Along with the binaries, the Malicia project also consisted of MySQL database, that consisted of metadata information related to each of the malware file.

5.3 Training and Scoring using HMM

For the experiments, we used 2135 Zbot files, 1305 ZeroAccess files and 4359 Winwebsec files. The HMMs were trained on 200 files of each family. The remaining files were used for scoring purpose. Before training the HMMs, the opcodes sequences were extracted from the disassembled files. The program first creates a set of unique opcodes from the files in the training set. These opcodes are treated as the observation symbols for the HMM. The number of states used is 2. The HMM is trained for 800 iterations and then the model is generated. This model is then used for scoring the remaining files.

Once again, the opcodes are extracted from the files that have to be scored against the HMM models generated. Ideally, the scores of the same family files should be similar. After training, a model should assign high probabilities to files similar to the training dataset and low probabilities to all other files. The log likelihood of the malware samples was computed. Log likelihood is the sum of log transition probabilities and log observation probabilities. As a result, it is highly dependent on the length of the sequence. If the sequence is long, it will have more transitions and more observations. Thus, it will have greater log likelihood probability. Since the sequences in the dataset may have different lengths, we divided the log likelihood of a sequence by the sequence length (which is the number of opcodes) to obtain the log likelihood per opcode (LLPO), which adjusts for the length difference [43].

5.4 Clustering

After training and scoring the malware files, we obtained 3 scores for each of the files: one from Zbot HMM model, second from ZeroAccess HMM model and third from Winwebsec model. As a result, each malware sample is represented as a 3-tuple, where each dimension represents its score generated by the specific HMM. This 3D matrix is given as input to K-means and EM clustering algorithm. The experiments were performed varying the number of clusters. Based on clustering results, we compute score for each of the malware sample. These scores are used to plot Receiver Operating Characteristic (ROC) curve. The main component of this score is the silhouette coefficient discussed in Section 3.3.1. Suppose we have a set of clusters where we have data point x belonging to cluster C_j . First, we compute silhouette coefficient for the data point. After that, we compute $p_{ij} = m_{ij}/m_j$ where m_{ij} is the number of elements of family i in cluster C_j, and m_j is number of elements in cluster C_j. Hence, we get the scoring function as $\text{score}_i(x) = p_{ij}S(x)$

Once we obtain the score, we use these scores to plot ROC. The ROC curve is constructed by plotting true positive rate against false positive rate as the threshold varies through range of data values. The quality of binary classifier can be measured by the area under the ROC curve (AUC). As a result, if the AUC is 1.0, it means that there exists a threshold that separates for which no detection error occur thereby yielding ideal separation.

However, an AUC of 0.5 denotes that the classifier is no more successful than flipping a coin [10].

Figure 3 is the diagrammatic representation of the whole process.

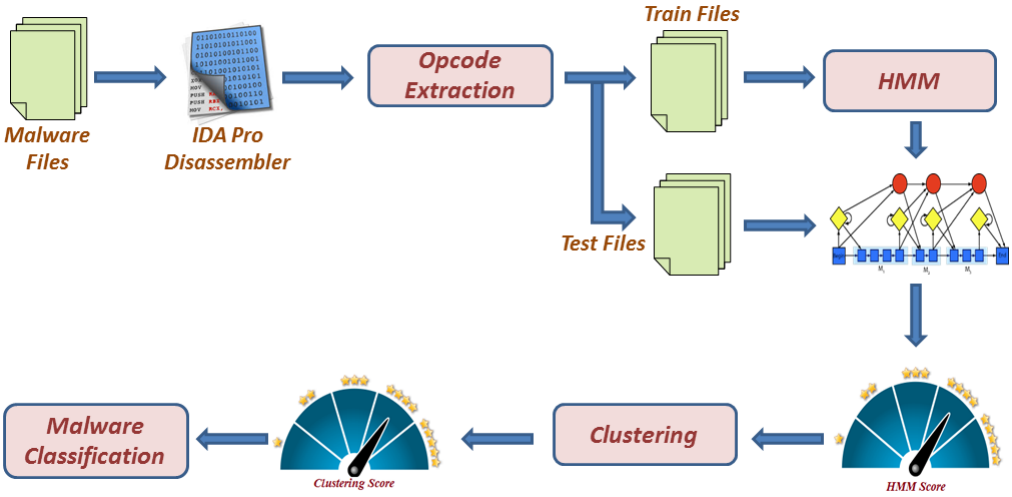

Figure 3: Implementation.

CHAPTER 6

Experiments and Results

This chapter includes the results of the experiments performed in this project. As discussed before, the final goal of this project is to develop an automated malware detection mechanism which could classify the new incoming malware under its appropriate family without the need to retrain the models. Below are the malware families used from Malicia project for experimental purpose.

Table 3: Training Set

Family	Number of Files
Zbot	200
ZeroAccess	200
WinWebSec	200
NCGVK	200
SmartHDD	53

Table 4: Testing Set

Family	Number of Files
Zbot	1936
ZeroAccess	1105
WinWebSec	4158
Benign	213

Since, Zbot, ZeroAccess and Winwebsec were dominant families in the Malicia project, we focused on these families for training the HMM models. Later, we included NGVCK malware family in our experiment.

6.1 Results from 2-Dimensional Clustering

We started our experiments by first combining the scores of Winwebsec and Zbot families. As a result, a 2D matrix was created. We performed K-means clustering on this matrix with k ranging from 2 to 10. Since, there were only four families in the test set, we limited the number of clusters to 10. Based on the clustering results, we scored the malware samples using the equation $\text{score}_i(x) = p_{ij}S(x)$ detailed in Chapter 5.

Using this equation, we obtained three scores for each of the malware sample where

$\text{score}_0(x) =$ Zbot score of file x

$\text{score}_1(x) =$ ZeroAccess score of file x

$\text{score}_2(x) =$ Winwebsec score of file x

Using $\text{score}_0(x)$, we plotted ROC curves for Zbot family. Similarly, we used $\text{score}_1(x)$ and $\text{score}_2(x)$ for plotting ROC for ZeroAccess family and Winwebsec family respectively. Table 5 gives the AUC for each of these families with different values of k.

Table 5: AUC for 2-Dimensional Matrix

Clusters	Zbot	ZeroAccess	Winwebsec
2	0.74471	0.63452	0.77551
3	0.74606	0.81266	0.77682
4	0.75399	0.81842	0.81611
5	0.59052	0.70672	0.89468
6	0.64819	0.71742	0.85159
7	0.68493	0.74515	0.8704
8	0.68444	0.70736	0.85621
9	0.67223	0.70215	0.90576
10	0.74817	0.7427	0.87674

Cluster Analysis for $k = 4$: Figure 4 shows the clustering results obtained when $k = 4$. In cluster 4, we see that there are two dominant clusters, one represented by

red color and other represented by blue color. The Cluster Red consists of majority of the malware samples. However, we notice that Cluster Blue purely consists of ZeroAccess family. Also, we notice that Winwebsec and Zbot family are almost classified into Cluster Red which indicates that statistically the malware samples of these two families might have delivered almost same scores when tested using Winwebsec and Zbot trianed HMM models.

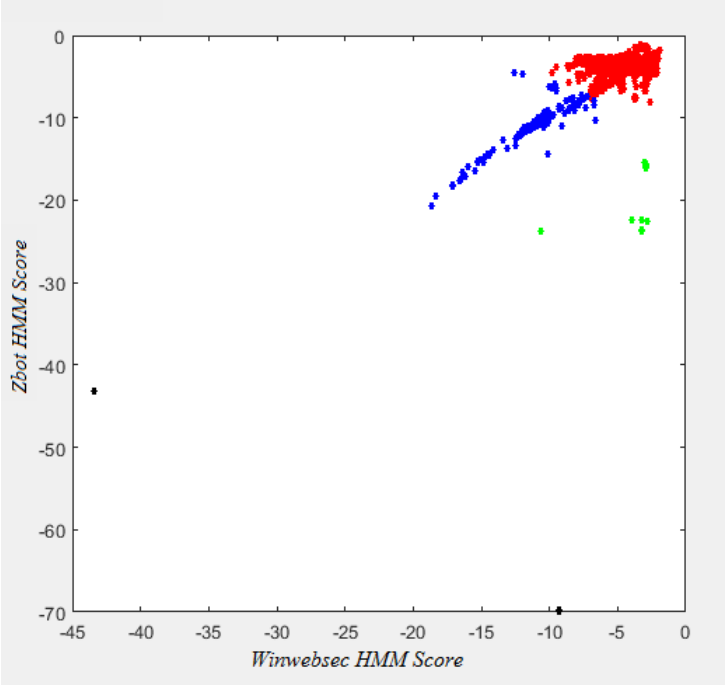

Figure 4: K-means Clustering Results when $k = 4$

Cluster Analysis for $k = 9$: In cluster 9, we notice from Figure 6 that there are four major clusters formed. The remaining clusters consist of very less malware samples (less than 100). Once again, we see a very similar trend to what was observed in Cluster 4. Here as well, the first cluster consists of majority of the malware files primarily Zbot and Winwebsec family files. However, if we compare the AUC for Winwebsec family files when $k = 4$ and $k = 9$, then we notice that the Winwebsec AUC increased when $k = 9$. One of the reason might be Winwebsec malware files

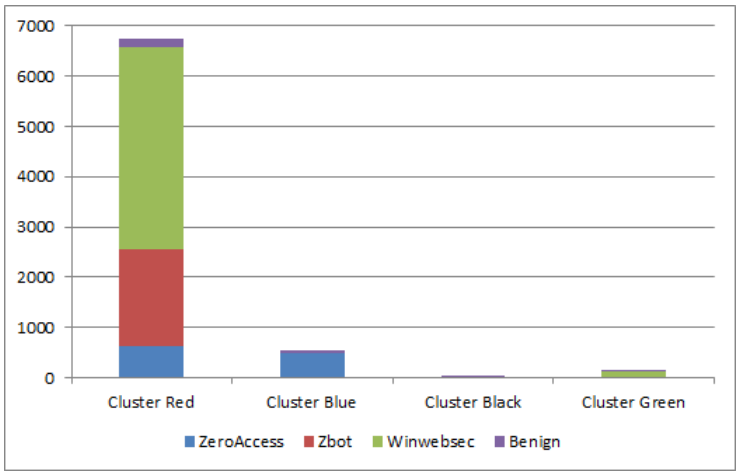

Figure 5: Stacked Column Chart for 2D model using K-means when $k = 4$

constituted 77% of the first cluster when $k = 9$ whereas it constituted just 60% when $k = 4$. And these percentages denote p_{ij} which basically represents fraction of number of elements of same families belonging to a particular cluster to the total elements present in that cluster. As a result, the Winwebsec family files might have been rated higher compared to other family files thereby leading to better separation.

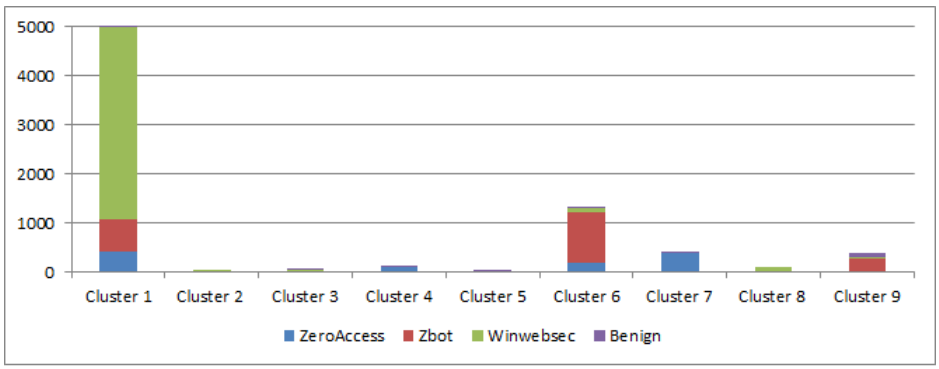

Figure 6: Stacked Column Chart for 2D model using K-means when $k = 9$

Conclusion from 2-Dimensional Model: In this model, we just considered the HMM scores from Winwebsec and Zbot trained model. The 2-dimensional plot that was obtained from this model can be seen in Figure 4. We observe that majority of the malware families especially the Winwebsec and Zbot families files are coupled near

the area of Cluster Red. An ideal model must be able to separate these families very well. Hence, we concluded that using this model, we were not able to get promising results especially for Zbot and ZeroAccess family. Hence, we took the experiments to the next level by including the HMM scores for the malware sample files obtained from ZeroAccess trained HMM.

6.2 Results from 3-Dimensional Clustering

To the 2-dimensional model, we now added the scores of the malware samples obtained by ZeroAccess trained model, thereby creating 3-dimensional model. We kept our test set same as before i.e. no new malware samples were added to the test set. We once again performed K-means clustering on this scores with values of k ranging from 2 to 10. Table 6 gives the AUC for each of these families with k ranging from 2 to 10.

Table 6: AUC for 3-Dimensional Model

Clusters	Zbot	ZeroAccess	Winwebsec
2	0.74257	0.50826	0.80535
3	0.75294	0.73854	0.82617
4	0.75343	0.79548	0.83981
5	0.65712	0.65759	0.82701
6	0.65759	0.69673	0.87544
7	0.57469	0.71307	0.86031
8	0.66476	0.71108	0.84725
9	0.65194	0.69194	0.87508
10	0.67225	0.77547	0.88622

Here as well, we see the same trend as 2-dimensional case. We see that when $k = 3$ or 4, we get better AUC results compared to other clusters. Figure 7 shows the K-means clustering results obtained when $k = 4$.

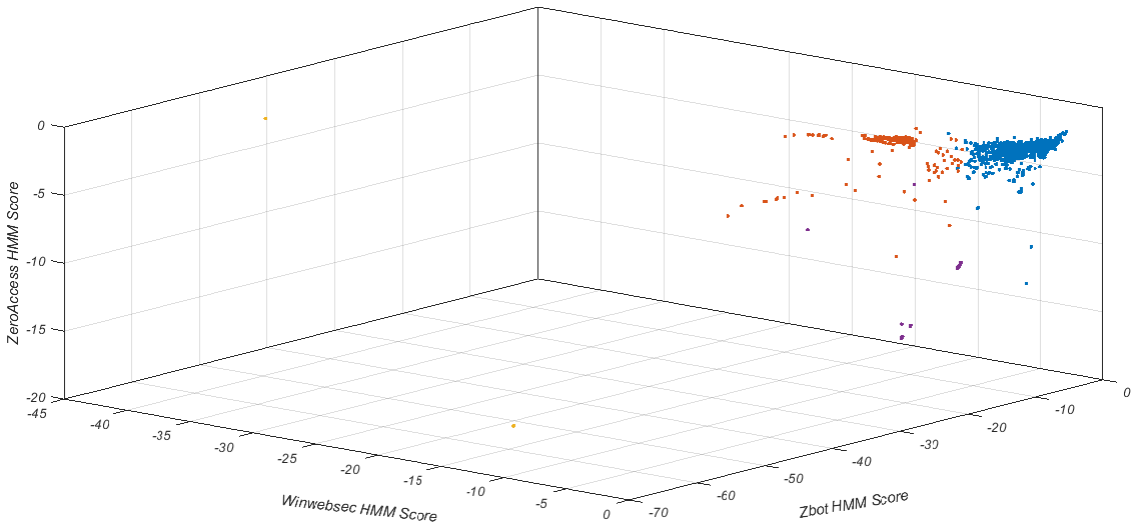

Figure 7: K-means Clustering Results when $k = 4$

Cluster Analysis for $k = 4$: In this clustering results, we once again noticed that there two dominant clusters, one represented by blue color and other represented by brown color. One surprising observation was the clustering results obtained from 3D model when $k = 4$ was almost same as what we obtained for 2D model when $k = 4$. The Figure 8 shows the stacked column chart for 3D model, which is almost same as stacked column chart for 2D model when $k = 4$.

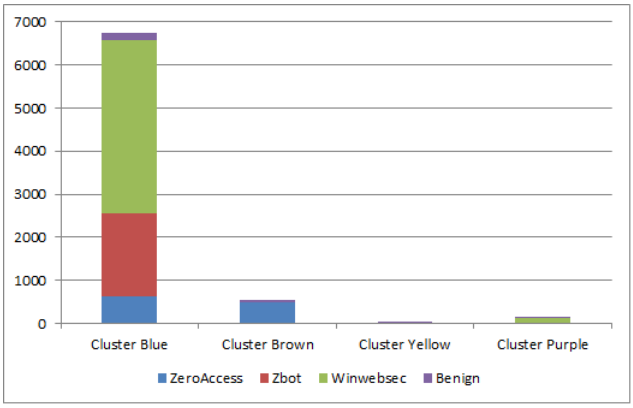

Figure 8: Stacked Column Chart for 3D model using K-means when $k = 4$

Also, on comparing the AUC for these two cases, we observe that there are very slight variation in the values. This is because of the differences in silhouette coefficient of the malware sample files, which is one of the factor for scoring the malware files.

Conclusion from 3-Dimensional Model: On observing the results, we notice that adding scores obtained from ZeroAccess trained model, did not result is much difference in the clustering results. The results were still not promising and there were no significant improvements in the AUC. Hence, we thought of adding HMM scores for the malware files obtained from NGVCK trained model and observe how the results look like.

6.3 Results from 4-Dimensional Clustering

To the 3-dimensional model, we now added the scores of the malware samples obtained by NGVCK trained model, thereby creating 4-dimensional model. We kept our test set same as before i.e. no new malware samples were added to the test set. We again performed K-means clustering on this scores with values of k ranging from 2 to 10. Table 7 gives the AUC for each of these families with k ranging from 2 to 10.

Table 7: AUC for 4-Dimensional Matrix

Clusters	Zbot	ZeroAccess	Winwebsec
2	0.63404	0.65365	0.82852
3	0.67827	0.66399	0.81664
4	0.70752	0.69939	0.80821
5	0.77389	0.76926	0.75341
6	0.78642	0.76528	0.77594
7	0.8382	0.77068	0.77122
8	0.85271	0.76867	0.79047
9	0.83175	0.77303	0.77317
10	0.82056	0.78967	0.85885

Cluster Analysis for $k = 10$: In this case, the AUC for Zbot and zeroAccess family is better when compared to 2-dimensional and 3-dimensional model for $k = 10$. On observing figure 9, we see that Winwebsec family is dominating cluster 3, cluster 9 and cluster 10. As a result, the AUC for Winwebsec is highest. Zbot dominates cluster 2 and cluster 7. However, ZeroAccess is dominating only cluster 5 and is mis-classified in cluster 1 and cluster 8 and hence, the AUC is comparatively low.

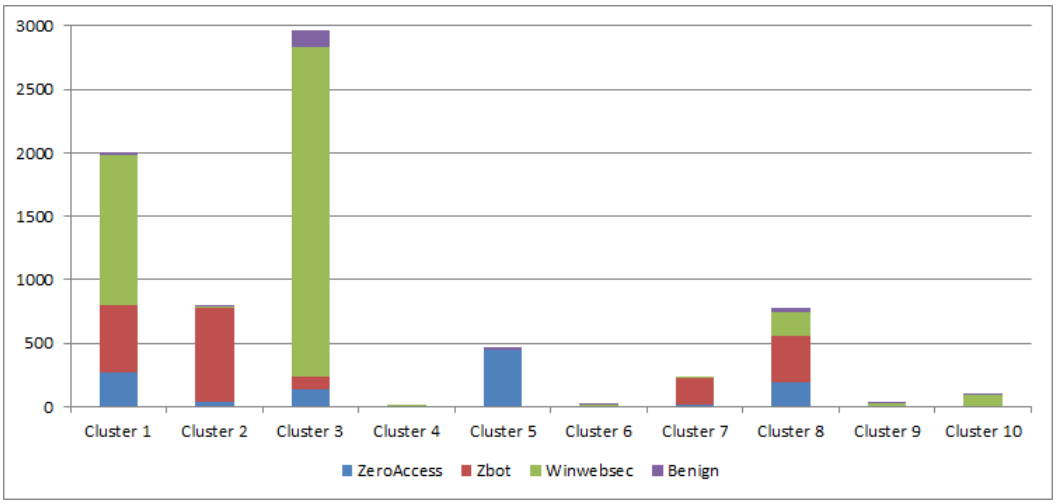

Figure 9: Stacked Column Chart for 4D using K-means when $k = 10$

Cluster Analysis for $k = 11$: Since, with $k = 10$, the AUC seemed better, we performed clustering with $k = 11$. And the AUC was better compared to what we got for $k = 10$. Table 8 shows the AUC when $k = 11$.

Table 8: AUC for 4-Dimensional Matrix for $k = 11$

Cluster	Zbot	ZeroAccess	Winwebsec
11	0.82970	0.84588	0.89423

Comparing the figure 9 and figure 10, we see that more number of clusters of noticeable clusters were formed when $k = 11$. For example, in case when $k = 10$, we just had 6 noticeable clusters, namely, cluster 1, cluster 2, cluster 3, cluster 5, cluster 7 and

cluster 8. However, in case when $k = 11$, we just had 8 noticeable clusters excluding cluster 3 and cluster 4 which have very fewer items. This could be one of the reason for increased AUC values. Figure 11, 12 and 13 shows the ROC curves for Winwebsec, Zbot and ZeroAccess family when $k = 11$ for 4D model.

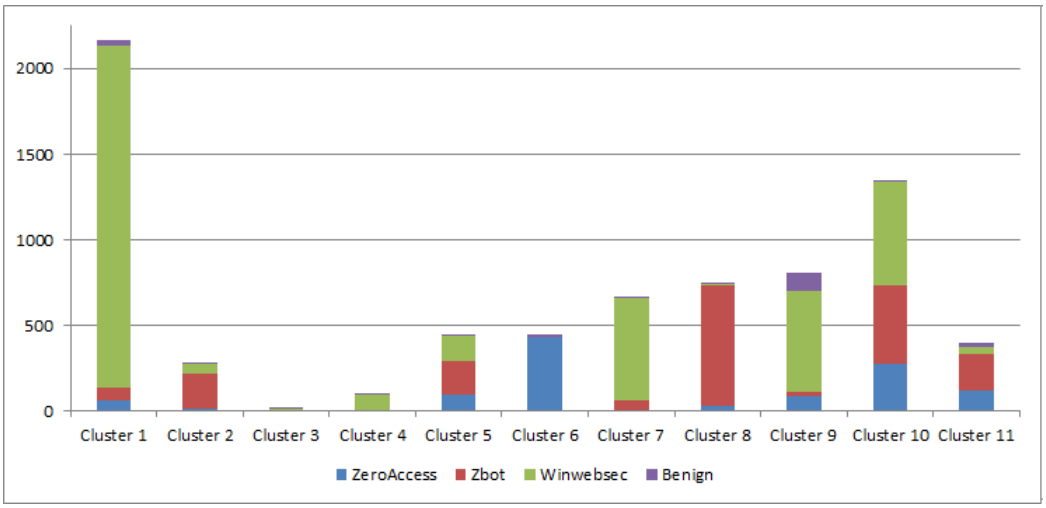

Figure 10: Stacked Column Chart for 4D model using K-means when $k = 11$

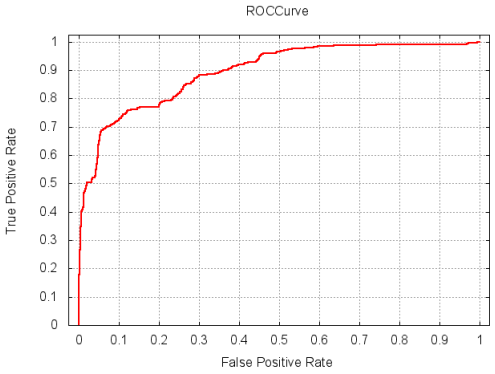

Figure 11: Winwebsec ROC

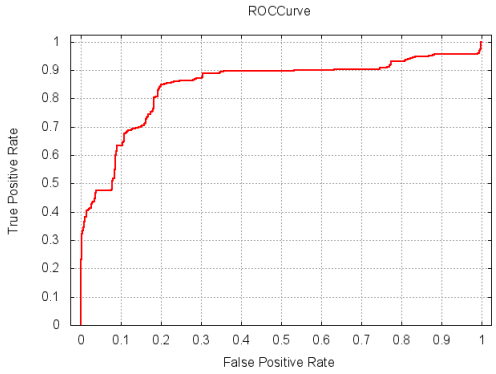

Figure 12: Zbot ROC

Conclusion from 4-Dimensional Model: On observing the results, we notice that adding scores obtained from NGVCK trained model, improved the AUC values with the increase in the values of k. These were better than the ones obtained from 2-dimensional and 3-dimensional. However, the model cannot be considered as ideal

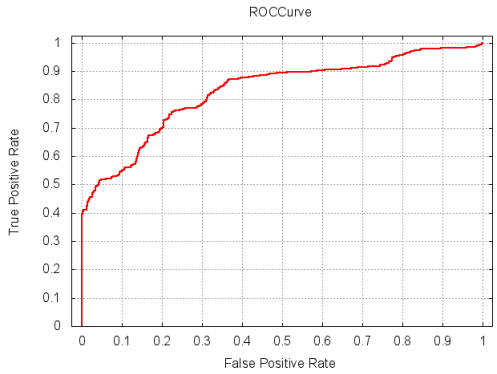

Figure 13: ZeroAccess ROC

one, which can perform perfect classification of the malware samples.

6.4 Results from EM Clustering

Since, we were getting good AUC using 4-dimensional model, we thought of using EM clustering on this 4-dimensional data and see if we can get a better AUC than K-means clustering. Hence, we performed EM clustering with values of k ranging from 2 to 10.

Table 9: AUC using EM clustering

Clusters	Zbot	ZeroAccess	Winwebsec
2	0.69327	0.73114	0.80383
3	0.60726	0.66238	0.79393
4	0.61268	0.85287	0.72414
5	0.64998	0.72575	0.79679
6	0.57069	0.62534	0.79656
7	0.55276	0.77531	0.74921
8	0.5615	0.62208	0.62348
9	0.65946	0.56042	0.54535
10	0.60154	0.55707	0.536

From Table 9, we notice that the AUC obtained using EM clustering is worse than what we got for K-means clustering. Moreover, with the increase in the values

of k, the AUC values start decreasing.

Does that mean K-means is better than EM clustering?

Before answering the above question, let us first have a look at the scatter plot for EM clustering for $k = 10$.

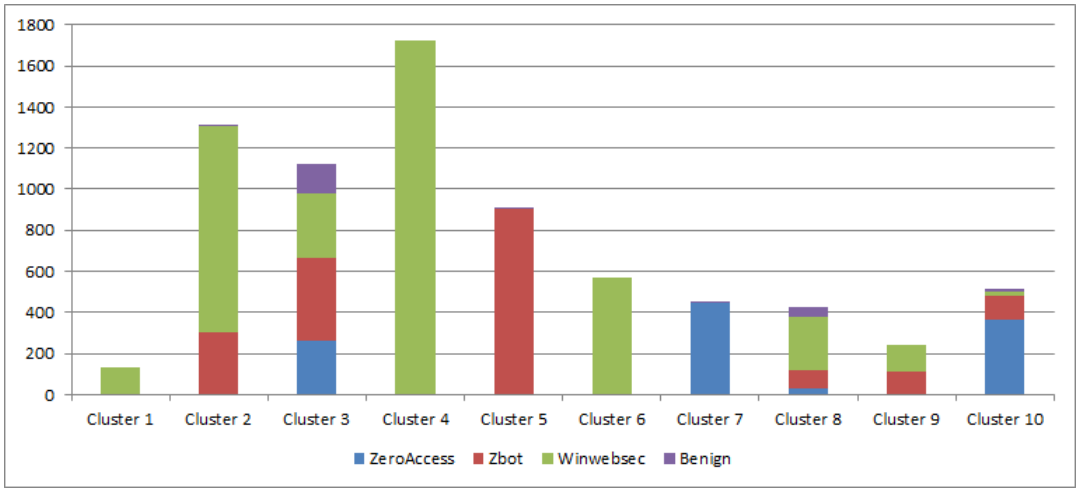

Figure 14: Stacked Column Chart for 4D model using EM when $k = 10$

From figure 14, we observe that the scatter plot for EM clustering looks better compared to that of K-means clustering. Especially, if we notice cluster 1, cluster 4, cluster 5, cluster 6 and cluster 7 are pure clusters consisting only one malware family. Then why are the AUC so less? The reason being silhouette coefficient, which is one of the factors for computing score. The silhouette coefficient for EM clustering decreases with the increase in the number of cluster. This is basically because of the inherent property of the EM clustering that tries to find the structure in the data. Figure 15 shows an example of EM clustering.

Let us consider data points x and y that are a part of blue cluster. Now data point x be is closer to member of its own cluster than member of other cluster. Definitely, silhouette coefficient for x will be good. However, if we consider data point y, it is closer to members of other clusters than members of its own cluster. Hence, the

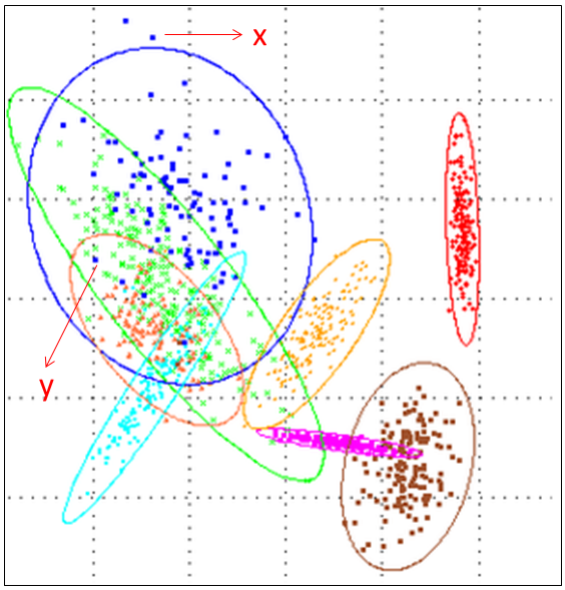

Figure 15: EM Clustering example [16]

silhouette coefficient for y will not be good. Table 10 shows silhouette coefficient of some Zbot samples files belonging to Cluster 5 in figure 14.

Table 10: Silhouette coefficient for Zbot files using EM clustering

Sample Files	Silhouette Coefficient
Zbot1	0.7922
Zbot2	0.8109
Zbot3	-0.8028
Zbot4	0.8142
Zbot5	0.8209
Zbot6	-0.6126
Zbot7	-0.5929
Zbot8	-0.7063
Zbot9	0.8209
Zbot10	-0.3934

Based on Table 10, we notice that in spite of belonging to same family and same cluster, the files vary in silhouette coefficient. As a result, when we use the formula $\text{score}_i(x) = p_{ij}S(x)$, instead of getting similar score, we get very different scores for

these files. Due to this reason, the AUC decreases.

The scoring technique used in this case can be used for comparing the models that are using K-means clustering. However, it cannot be used in case for scoring results from EM clustering.

Table 11 shows the silhouette coefficient for both EM and K-means clustering for 4-dimensional model.

Table 11: Silhouette coefficient for K-means and EM clustering

Clusters	K-means	EM
2	0.8444	0.5860
3	0.8468	0.4369
4	0.8023	0.0309
5	0.7803	0.2107
6	0.7745	0.2354
7	0.7816	0.0182
8	0.7926	0.0513
9	0.7585	-0.1567
10	0.7029	-0.0694

We also performed EM clustering on 2D and 3D model. Figure 16 shows the results of EM Clustering performed for 2D model when $k = 4$. As mentioned before, EM algorithm tries to find structure in the data. Moreover, EM also employs soft clustering technique. As a result, we can see many overlaps in the clusters structure. Figure 17 shows the scatter plot obtained from EM clustering when $k = 4$ for 2D model.

The K-Means algorithm finds clusters by choosing data points at random as initial cluster centers. Each data point is then assigned to the cluster with center that is closest to that point. This process is iterated until no data point is reassigned to a different cluster. However, EM finds clusters by determining a mixture of Gaussian

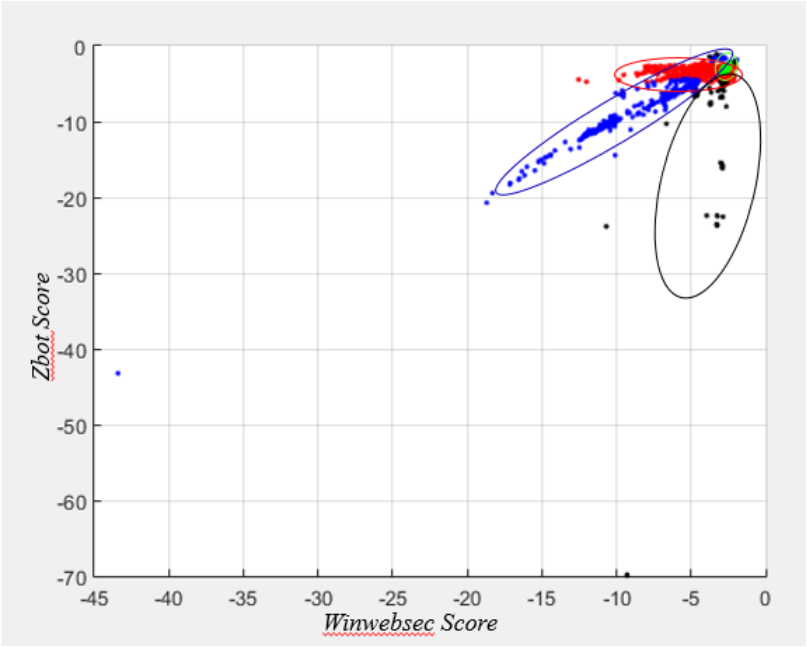

Figure 16: EM Clustering for 2D model when k=4

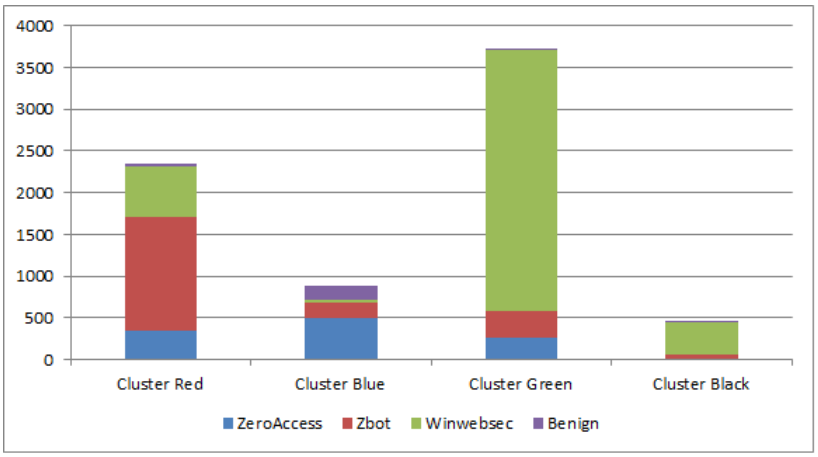

Figure 17: Scatter plot 2D model when k=4 using EM Clustering

that fit a given data set. Each Gaussian has an associated mean and covariance matrix. The algorithm converges on a locally optimal solution by iteratively updating values for means and variances.

From the scatter plots, it can be seen that both the algorithms were able to classify the malware families to a great extent. However, we see that EM clustering

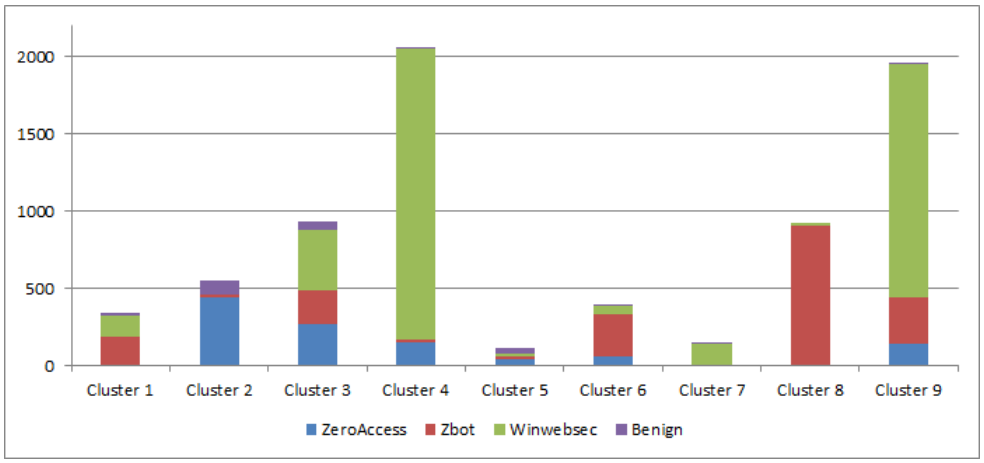

Figure 18: Scatter plot 2D model when k=9 using EM Clustering

formed more noticeable clusters with an increase in value of k. Further, on comparison, when $k = 9$ for 2D model and 4D model, we see that EM formed better clusters than K-means.

6.5 Comparison of K-means and EM based on Purity

When we compared the purity of K-means and EM for 4D model, we noticed that the purity of EM was better than that of K-means. Table 12 shows the purity results for both K-means and EM on 4D model. Figure 19 shows the line chart for the same.

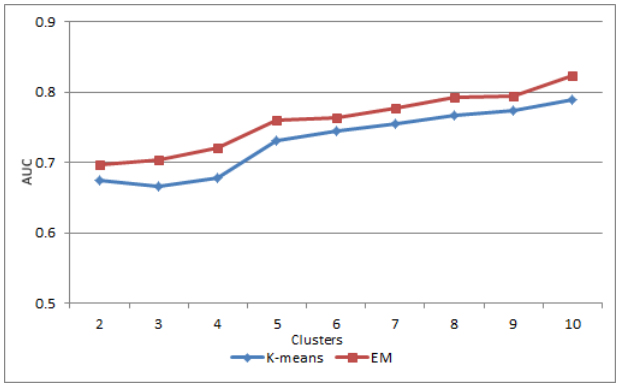

Figure 19: Purity of EM v/s K-means for 4D model

Table 12: Purity of EM clustering v/s K-means for 4D model

Clusters	K-means	EM
2	0.6743	0.6963
3	0.6664	0.7045
4	0.6774	0.6478
5	0.7316	0.7599
6	0.7452	0.7643
7	0.7545	0.7765
8	0.7678	0.7926
9	0.7536	0.7947
10	0.7679	0.8024

We now included the scores from SmartHDD HMM to create a 5D model. We then computed the purity for K-means and EM for values of k ranging from 2 to 10. We observed a similar trend as we noticed for 4D model. Even in this case, it was observed that the purity of EM was always better than that of K-means. Figure 20 shows the purity results obtained from using EM and K-means clustering on 5D model.

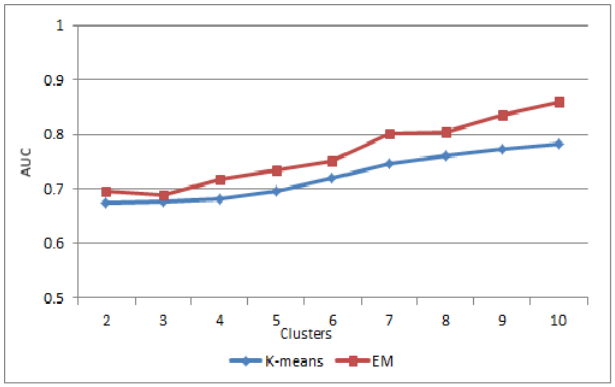

Figure 20: Purity of EM v/s K-means for 5D model

So, we continued our experiments to see whether this similar trend was observed for 2D and 3D model. But surprisingly, we observed that for lower dimensions and lower values of k, K-means was giving better purity than EM. Figure 21 and figure 22

shows line charts for purity results on 2D and 3D model respectively.

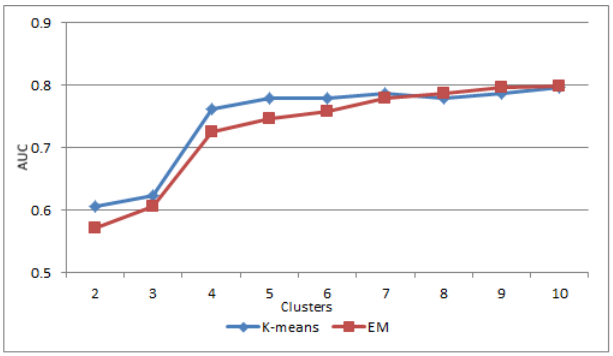

Figure 21: Purity of EM v/s K-means for 2D model

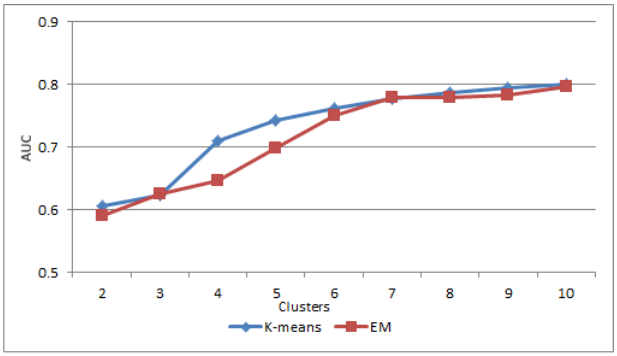

Figure 22: Purity of EM v/s K-means for 3D model

Figure 23 gives a complete picture of purity results for different number of clusters and different dimensions for EM clustering. It is observed that purity increases as the value of k and number of dimensions increases. As a result, we observed highest value of purity of 0.86 for K=10 for 5D model.

On the other hand, when we plotted similar graph for K-means as represented in figure 24, we observed that purity starts decreasing with the increase in number of dimension. For example, we noticed that the purity when K=10 for 5D model was less compared to 4D model and for 4D model was less than 3D model. The reason for this might be, with the increase in the dimension, more structures might be formed

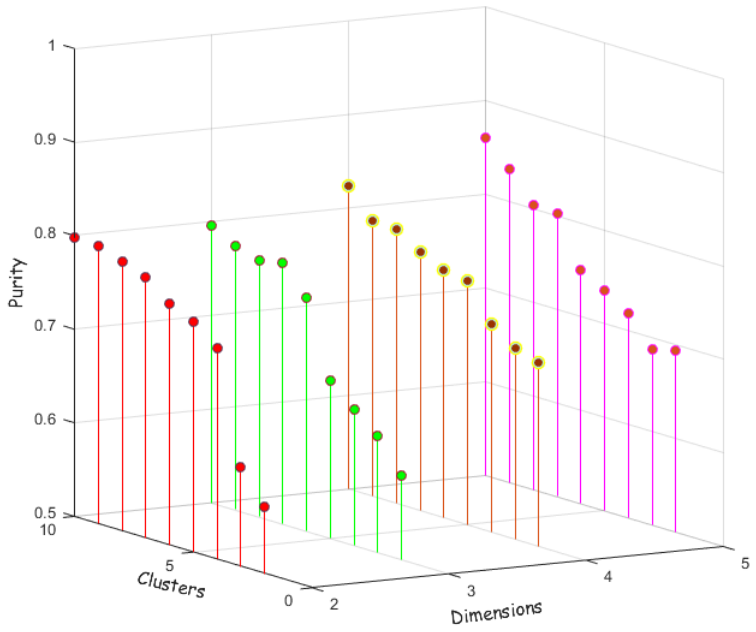

Figure 23: Purity with respect to k and dimensions for EM

in the data. As EM clustering in good with finding structures in data, it gave better purity results than that of K-means.

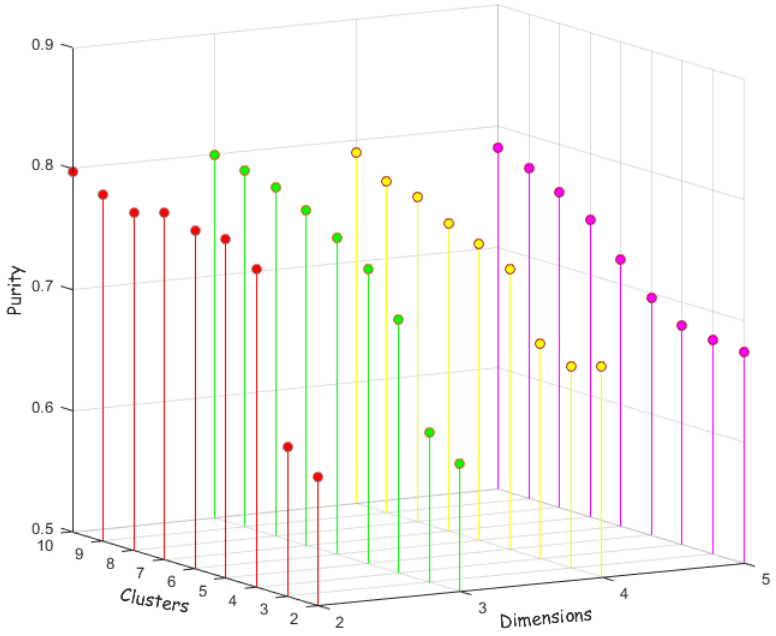

Figure 24: Purity with respect to k and dimensions for K-means

CHAPTER 7

Conclusion and Future Work

This project was executed using both K-means and EM clustering for classifying malware families. HMM was trained on Winwebsec, Zbot, ZeroAccess and NGVCK families for a subset of total files. We scored remaining files from Winwebsec, Zbot and ZeroAccess against the HMMs that were trained. Few benign files were also included in the experiments.

First set of experiment was conducted with a 2-dimensional model, where scores of malware samples from Winwebsec and Zbot HMM Models were combined. K-means clustering was performed on this model for k ranging from 2 through 10.

Further a 3-dimensional model was generated by adding malware samples obtained from ZeroAccess trained HMM model. Once again, K-means clustering was performed on this new model for k ranging from 2 through 10.

Finally, a 4-dimensional model was developed by adding scores of malware samples obtained from NGVCK HMM models. K-means clustering was performed on this 4-dimensional model for k ranging from 2 through 10. In addition, EM clustering was also performed on this model.

After clustering, we used the formula $\text{score}_i(x) = p_{ij}S(x)$ to compute the scores for the malware samples.

To understand the results, ROC curves were plotted for the dominant families ZeroAccess, Zbot and Winwebsec. Based on the AUC values, it was observed that adding scores for the malware samples obtained from ZeroAccess trained HMM to

the 2-dimensional model did not improve the AUC. However, when we added scores obtained from NGVCK trained HMM to the 3-dimensional model, then AUCs improved with an increase in the values of k. This indicates that NGVCK trained HMM was better in classifying the malware as compared to ZeroAccess trained HMM. To validate this finding, a comparison of the HMM results was performed. Table 13 shows the results of ZeroAccess and NGVCK trained HMM Models.

Table 13: AUC from ZeroAccess and NGVCK HMM

Family	ZeroAccess HMM	NGVCK HMM
Zbot	0.65699	0.71352
ZeroAccess	0.74522	0.58681
Winwebsec	0.5097	0.72632

Based on the AUC, we see that both NGVCK model and ZeroAccess model were really not that effective in classifying the malware families individually. However, the combination of the HMM scores obtained from the four models were effective to certain extent in classifying the malware families and improving the AUCs. Given this fact, it can be concluded that an increase in the dimensions can help in getting better results.

Also, it appears that values of k may be proportional to dimension size, to derive a better classification i.e. value of k may need to be increased with an increase in dimension. This is to factor in the possibility of a family having further sub-families within it. For example, let us consider if there are two sub-families within Zbot family that perform different operations. Then statistically these files may differ. There might be some HMMs that will be able to differentiate between these two sub-families and some that might not be able to. So even if one of the HMM is able to distinguish these sub families, then we can expect different clusters for these families.

We also tried using EM clustering on 4-dimensional model. Based on the scatterplot, EM was forming more pure clusters i.e. clusters consisting of just one family. However, the scoring technique used for K-means clustering in this case cannot be used for EM clustering. This is because of the silhouette coefficient decreases drastically with the increase in number of clusters.

During the course of experiments with 2-dimensional, 3-dimensional and 4-dimensional models, trend suggested that there was considerable increase in the AUC values. Hence, it may be worthwhile in future to perform similar experiments with higher number dimensions and observe the clustering results.

We also used purity for comparing the results of EM and K-means. We noticed that for higher dimensions, EM was giving better purity results than K-means. Also, it was observed that EM gives better purity with the increase in the dimensions. However, K-means, it was seen that with higher dimensions and higher values of k, the purity of K-means decreases.

In the current scope of work, cluster results obtained from K-means and EM were not comparable, since the inherent properties used by these algorithms for clustering were very different. In future, it is recommended that better scores can be used so that K-Means and EM cluster results can be compared.

While manually disassembling some of the files for above mentioned families, it was observed that most of the files within the same family followed a similar function call graph structure. It would be interesting to explore the option of using this call graphs to analyze the datasets and compare its results with clustering.

LIST OF REFERENCES

[1] V. Aggarwal, A. K. Ahlawat, and B. N. Panday, A Review of Data Clustering Approaches, *International Journal of Engineering and Innovative Technology (IJEIT)*, 1(4):310–314, 2012

[2] K. Alsabti, S. Ranka and V. Singh, An Efficient K-Means Clustering Algorithm, 1998
`https://www.cs.utexas.edu/~kuipers/readings/Alsabti-hpdm-98.pdf`

[3] M. B. Al-Zoubi and M. A. Rawi, An Efficient Approach for Computing Silhouette Coefficients, *Journal of Computer Science*, 4(3):252–255, 2008

[4] C. Annachhatre, T. H. Austin, and M. Stamp, Hidden Markov Model for Malware Classification, *Journal of Computer Virology and Hacking Techniques*, 2014

[5] T. Austin, E. Filiol, S. Josse, and M. Stamp, Exploring Hidden Markov Models for Virus Analysis: A Semantic Approach, *46th Hawaii International Conference on System Sciences (HICSS 2013)*, 5039–5048, 2013

[6] J. Aycock, *Computer Viruses and Malware*, Springer, 2006

[7] A. R. Babu, M. Markandeyulu and B. V R R Nagarjuna, Pattern Clustering with Similarity Measures, *Int.J.Computer Technology & Applications*, 3(1):365–369, 2012

[8] M. Bailey, J. Oberheide, J. Andersen, Z. Morley Mao, F. Jahanian, and J. Nazario, Automated Classification and Analysis of Internet Malware, *RAID'07 Proceedings of the 10th international conference on Recent advances in intrusion detection*, 1:178–197, 2007

[9] V. Beal, Webopedia, Sypware,
`http://www.webopedia.com/TERM/S/spyware.html`

[10] A. P. Bradley,The use of the area under the ROC curve in the evaluation of machine learning algorithms, *Pattern Recognition*, 30(7):1145-âĂŞ1159, 1997

[11] BullGuard Security Centre,
`http://www.bullguard.com/bullguard-security-center/pc-security/computer-threats/malware-definition,-history-and-classification.aspx`

[12] J. M. Chambers, R Language Definition,
`http://cran.r-project.org/doc/manuals/R-lang.html`

[13] Data Mining Algorithms In R/Clustering/Expectation Maximization (EM), http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/Expectation_Maximization_(EM)

[14] D. E. Denning, An Intrusion-Detection Model, *IEEE Transactions on Software Engineering - Special issue on computer security and privacy*, 13(2):222–232, 1987.

[15] C. B. Do and S. Batzoglou, What is the Expectation Maximization Algorithm?, *Nature Biotechnology*, 26(8):897–899, 2008

[16] EM Clustering Algorithm, http://jormungand.net/projects/misc/em/

[17] C. Fraley and A. E. Raftery, MCLUST Version 3 for R: Normal Mixture Modeling and Model-based Clustering, Technical Report 504, University of Washington, Department of Statistics, 2006

[18] History of malware, https://www.gdatasoftware.com/securitylabs/information/history-of-malware

[19] N. Idika and A. P. Mathur, A Survey of Malware Detection Techniques, 2007, http://cyberunited.com/wp-content/uploads/2013/03/A-Survey-of-Malware-Detection-Techniques.pdf

[20] Internet Security Threat Report, 2014, http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf

[21] A. K. Jain, M. N. Murty, and P. J. Flynn, Data Clustering: Overview, *ACM Computing Surveys*, 31(3):264–323, 1999

[22] J. Kolter and M. Maloof, Learning to Detect Malicious Executables in the Wild, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 470–478, 2004

[23] D. Kong and G. Yan, Discriminant Malware Distance Learning on Structural Information for Automated Malware Classification, *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1357–1365, 2013

[24] M. Lennon, Target Confirms Point-of-Sale Malware Was Used in Attack, 2014 http://www.securityweek.com/target-confirms-point-sale-malware-was-used-attack

[25] C. Lippert, Data Mining in Bioinformatics, http://agbs.kyb.tuebingen.mpg.de/wikis/bg/tutorial_GMM.pdf

[26] J. MacQueen, Some Methods for Classication and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967

[27] Malicia Project - Driving in the Cloud Dataset,
`http://malicia-project.com/dataset.html`

[28] Microsoft Malware Protection Center, Rootkit,
`http://www.microsoft.com/security/portal/mmpc/threat/rootkits.aspx`

[29] L. R. Rabiner, A Tutorial on Hidden Markov Models and selected applications in Speech Recognition, *Proceedings of IEEE*, 7(2):257–286, 1989

[30] P. Rai, Data Clustering: K-means and Hierarchical Clustering, 2011,
`http://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf`

[31] Reuters,
`http://www.reuters.com/article/2014/09/18/`
`us-home-depot-dataprotection-idUSKBN0HD2J420140918`

[32] K. Rieck, P. Trinius, C. Willems and T. Holz, Automatic Analysis of Malware Behavior Using Machine Learning,*Journal of Computer Security*, 19(4):639–668, 2011

[33] Safety & Security Center,
`http://www.microsoft.com/security/pc-security/virus-whatis.aspx`

[34] M. Schultz, E. Eskin, F. Zadok and S. Stolfo, Data Mining Methods for Detection of New Malicious Executables, *Proceedings of 2001 IEEE Symposium on Security and Privacy*, 38–49, 2001

[35] M. Stamp, A Revealing Introduction to Hidden Markov Models, 2012,
`http://www.cs.sjsu.edu/faculty/stamp/RUA/HMM.pdf`

[36] M. Stamp, *Information Security: Principles and Practice*, second edition, Wiley, 2011

[37] Symantec, Trojan Horse, 2004,
`http://www.symantec.com/security_response/writeup.jsp?docid=`
`2004-021914-2822-99`

[38] Symantec, Trojan.Zbot., 2010,
`http://www.symantec.com/security_853response/writeup.jsp?docid=`
`2010-011016-3514-99`

[39] Symantec Security Response, Trojan.Zeroaccess, 2011, http://www.symantec.com/security_response/writeup.jsp?docid=2011-071314-0410-99

[40] Symantec, What is the difference between viruses, worms, and Trojans?, 2009, http://www.symantec.com/business/support/index?page=content&id=TECH98539

[41] J. V. Neumann and A. Burks, *The Theory of Self-reproducing Automata*, University of Illinois Press Campaign, 1966

[42] WinWebSec, http://www.enigmasoftware.com/winwebsec-removal/

[43] W. Wong and M. Stamp, Hunting for Metamorphic Engines, *Journal in Computer Virology*, 2(3):211–229, 2006