


Heterogeneous Opcode Space for Metamorphic Malware Detection

Jithu Raphel¹ · P. Vinod¹ 

Received: 9 September 2015 / Accepted: 27 June 2016 / Published online: 15 July 2016
© King Fahd University of Petroleum & Minerals 2016

Abstract Metamorphic viruses are equipped with morphing engine responsible for transforming the structure of the code in subsequent generations, thereby retaining the malicious behavior. Thus, commercial anti-virus software based on signature approach is unable to identify the unknown or zero-day malware. Each metamorphic malware has its own unique pattern since its internal structure changes from generation to generation. Hence, detection of these viruses is a challenge for researchers working on computer security. The degree of metamorphism in the dataset is estimated by aligning the locations of common opcodes using Smith–Waterman sequence alignment method suggesting that a generic pattern representing malware or benign classes cannot be extracted, thus demonstrating the failure of signature-based approach. The proposed statistical non-signature-based detector creates two different meta feature spaces each comprising 25 attributes for their detection. Three categories of opcode features are extracted from each sample: (a) branch opcodes, (b) unigrams and (c) bigrams. Insignificant features are initially eliminated using the Naïve Bayes approach; obtained feature space is further reduced using two feature reduction techniques: (1) Discriminant Feature Variance-based Approach (*DFVA*) and (2) Markov Blanket. Learning models are created using the prominent attributes obtained from each dimensionality reduction methods. The models which provided the highest accu-

racy at minimum feature length were retained, and unseen instances are classified using these optimal models. Later, two meta feature spaces were generated by ensembling the prominent branch, unigram and bigram opcodes obtained from *DFVA* and Markov Blanket. Both feature reduction techniques were found to be equally efficient in detecting the metamorphic malware samples. The proposed system detected Metamorphic Worm and Next Generation Virus Construction Kit viruses with 100% accuracy, Precision 1.0, Recall 1.0 and a promising F1-score of 1.0 is achieved. The results demonstrate the efficiency of the proposed metamorphic malware detector, and we thus recommend that this approach can be used to assist commercial AV scanners.

Keywords Metamorphic malware · *DFVA* · Markov blanket · Feature reduction · Sequence alignment · Meta feature space

1 Introduction

Computer virus is a piece of malicious code that replicates and injects itself into legitimate programs [1]. Metamorphic viruses employ diverse techniques to obfuscate their code while maintaining the same functionality in the later generation. Traditional signature-based approach cannot help in the detection of zero-day malware [2] due to structural diversification. Hence, it also fails to detect the metamorphic malware samples [3].

In the case of a metamorphic virus, malware mutates the entire virus body; hence, no constant sequence is preserved in the new variants [4]. Hidden Markov Model (HMM)-based method [3] was used to detect the metamorphic malware samples in the earlier times. The exper-

✉ P. Vinod
pvinod21@gmail.com

Jithu Raphel
jithuraphelvd347@gmail.com

¹ Department of Computer Science and Engineering, SCMS
School of Engineering and Technology, Ernakulam, Kerala,
India

iments demonstrated that viruses created using the Next Generation Virus Construction Kit (NGVCK) exhibited a higher degree of metamorphism. Profile Hidden Markov Model was proposed in [5] for metamorphic malware detection. This detector recognized the variants created by PS-MPC and VCL-32 but failed to identify NGVCK viruses.

However, it was experimentally proved that Metamorphic Worm (MWORM) developed by authors in [6] can thwart the HMM-based detection. This was achieved by padding the MWORM with blocks of subroutine instructions from benign files. Padding is done in a way that the metamorphism of the worms is retained and new malware files thus generated appear structurally similar to legitimate programs. Later, structural entropy [7]-based method was proposed for identifying the metamorphic variants. Entropy score measures the statistical variation in bytes within the files. This method detected the MWORM with better accuracy but resulted in few false positive errors.

The contributions of this study are as follows:

1. Development of an efficient statistical detector for identifying metamorphic viruses using *DFVA* and Markov blanket method. These methods generate robust models for identifying future instances.
2. Generation of two meta opcode space each of 25 features from training set that detected the malware and benign unseen instances in the test set.
3. Evaluation of the performance of malware and benign learning models at varying feature length.
4. Validation of unseen instances was carried out using the individual optimal models as well as the meta feature space.
5. Investigation of the applicability of discriminant malware opcodes in classification.
6. Validation of degree of metamorphism in the dataset used to conduct experiment.
7. Investigation of the applicability of non-signature approach that resulted in the pruned opcodes obtained by feature selection methods to obtain better outcomes.

The structure of this paper is as follows: Sect. 2 discusses related works done in this area. Section 3 describes the proposed methodology. The experimental setup used for carrying out the study is explained in Sect. 4. Results of the experiment are described in Sects. 5 and 8. Comparative analysis with prior works is done in Sect. 6. Finally in Sect. 7, we discuss the inferences and concluding remarks are covered in Sect. 9.

2 Related Works

Daniel Bilar [8] reported that opcode (operational code) frequency distributions can be used to classify malware instances. A total of 67 malware executables were disassembled, and the frequency distribution of opcodes was compared with the histogram of opcodes in normal programs. It was showed that the malicious opcode frequency distributions vary significantly from that of benign samples and rare opcodes were a stronger predictor for classification. Igor Santos et al. [9,10] designed a novel system to identify the malware variants of familiar families using the distribution and appearance of opcode sequences. They reported that certain opcodes occurrence varied in benign and malicious executables. Hence, similarity score calculated based on opcode frequency might fail in differentiating the samples. To avoid such situations, the relevance of each opcode was determined by multiplying the term frequency of each opcode sequence by its calculated weight. Robert Moskovitch et al. [11] introduced a technique where the benign and malware executables were represented by opcode expressions by streamlining an executable into a sequence of opcodes. The authors treated *n-grams* of the opcodes as features for classifying the unknown instances. Their method achieved an accuracy greater than 99% on a training set with malicious file contents lower than 15%. This method was found to be superior to the byte sequential *n-gram* representation technique. In [12], Asaf Shabtai et al. also used opcode *n-gram* patterns for classification. Authors performed an extensive analysis of this method with the help of a test set constituting more than 30,000 files. The 2-gram opcodes outperformed compared to other *n-gram* pattern.

Sushant Priyadarshi and Mark Stamp in [13] developed an emulator to identify the dead code inserted in malware instances. If the virus files were completely unmorphed, the HMM tool was able to detect them. However, generating a precise program normalizer is a complex task. In [6], the authors designed a prototype of metamorphic worm which was capable of evading the HMM-based scanner. The worm carried a morphing engine that was morphed on each replication. This metamorphic worm employed two morphing techniques: (a) equivalent instruction substitution and (b) insertion of dead code. The worm also uses blocks of dead code from benign executable files making the malicious files structurally similar to benign executable files. Since the patterns of malware files were more similar to benign population, HMM-based scanner raised higher false alarms. Shanmugam et al. [14] designed and implemented an opcode-based similarity technique for metamorphic malware detection. The detection system was trained based on opcodes of a specific metamorphic family. An unknown file was scored using the trained model to

determine if the file belongs to same virus family. Authors in [15] developed a new static method for the classification of malware files that employed common segment analysis. With the help of common segment analysis, the system discards the code segments that originated from benign code. A new feature termed ‘meta-feature’ is used by the system to capture the features of analyzed segments. Vinod et al. [16] addressed the problem of detection of metamorphic malware using multiple sequence alignment. The system identifies the unseen instances using the signatures extracted for a malware family. These signatures created using their proposed technique were superior to that generated by the commercial AV engines. Neha Runwal et al. [17] implemented a similarity score-based approach for malware detection. A weighted opcode graph was constructed where opcodes represented the nodes in the graph. Edges represented the probability of co-occurrence of two opcodes.

In [7], the authors utilized the structural entropy theory to study the dissimilarities of data in an instance. The detection system works in two stages namely file segmentation and sequence comparison. File segmentation phase uses entropy scores and wavelet analysis for segmenting a file. During the second phase, the edit distance between the sequence of segments is computed and it was employed for measuring the similarity between files. Sayali Deshpande et al. [18] proposed an approach for identifying metamorphic viruses based on a well-known facial recognition technique which uses eigenvalue analysis. Using the opcode sequences which were derived from the set of well-known viruses, eigenvectors were computed. An executable was scored using these eigenvectors.

In [19], HMMs were trained for several compilers and malware generators. The malware instances were scored using each model, and the samples were clustered based on their scores. Clustering was performed using k -means clustering algorithm. Each cluster represented certain features of malware. Authors in [20] developed a non-signature-based scanner that can detect the Metamorphic worm (MWORM) and viruses created using Next Generation Virus Construction Kit (NGVCK) with 100% accuracy and precision. Bigram opcodes were ranked using several feature ranking techniques, and the significance of these feature ranking approaches at varying feature length was investigated. Our previous work [21] employs a two-phase feature reduction technique for detecting the metamorphic malware instances. In the first phase, we extract features having a higher feature to class correlation score. Subsequently, redundant features are eliminated from the earlier feature set during the next phase.

3 Proposed Methodology

The architecture of the proposed system is illustrated in Fig. 1. The proposed system consists of the phases such as: (1) Preprocessing, (2) Rare feature elimination using Naïve Bayes approach, (3) Dimensionality reduction with Discriminant Feature Variance-based Approach (DFVA) and Markov Blanket, (4) Model generation, (5) Prediction and (6) Validation of location of opcode sequence to evaluate if the dataset consisted of enough metamorphism.

3.1 Preprocessing

The metamorphic malware samples were collected from the authors in [22], and the benign samples were downloaded from The Internet. Initially, all samples are unpacked and disassembled using IDA PRO disassembler [23]. The assembly files obtained from IDA PRO are randomly divided into a train and test sets in 50:50 ratio. The assembly files from both train and test set are preprocessed, and features such as (a) branch instructions, (b) unigrams and (c) bigrams are extracted from each assembly files.

3.1.1 Branch Instructions

Control flow obfuscation can be introduced into malicious files by the metamorphic engine by adding conditional or unconditional branch instructions. It is assumed that the frequency distribution of such branch opcodes might significantly differ in malware and benign classes.

3.1.2 Unigram

From the assembly files, unigram opcodes are extracted. Branch instructions are not considered here since they are already extracted in the previous set.

3.1.3 Bigram

Bigram opcodes are generated by taking two consecutive unigram opcodes in a sliding window fashion as shown in Table 1. In prior works [20,24], it was experimentally demonstrated that bigram features resulted in improved classification than unigram opcodes.

3.1.4 Unique List Creation

Next, we construct a unique list of branch opcodes, unigrams and bigrams for samples of the train set. In total, 17 and 19 unique branch opcodes were observed in malware and benign

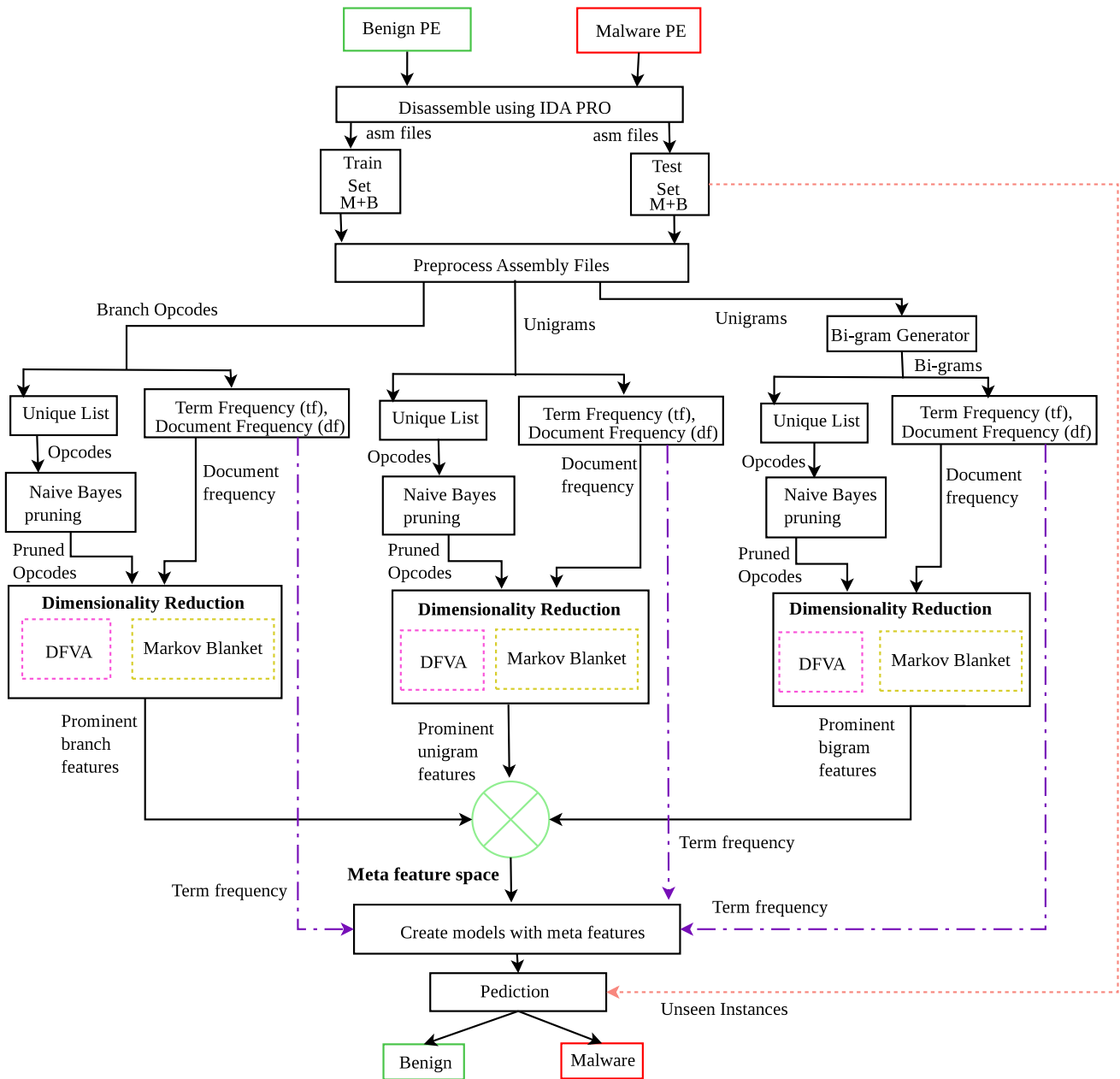


Fig. 1 Architecture of proposed metamorphic malware detector

Table 1 Bigram generation

N-gram	Opcode n-gram
Unigram	add, mov, push, call, xor
Bigram	addmov, movpush, pushcall, callxor

Table 2 Term frequency and document frequency of unique bigram opcodes

Bigram	tf_B	df_B	tf_M	df_M
adcpop	40	18	6	5
addjmp	12376	471	12367	434
jmpjmp	12960	405	0	0
movadc	0	0	62	62
addsub	3085	349	9698	433

training set, respectively. A unique list of unigram opcodes consisted of 130 malware and 271 benign features. The legitimate software contains 5906 unique bigram opcodes, whereas there existed 2276 unique opcodes in malware samples. Term frequency (tf) and document frequency (df) of each opcode are later calculated (refer to Table 2). In Table 2,

tf_B represents the term frequency of opcode in benign files, and df_B depicts the document frequency of an opcode in benign files. Similarly, tf_M represents the term frequency of

Table 3 Naïve Bayes Score of each unique bigram opcode

Bigram	NB_B	NB_M
adcpop	0.783	0.217
addjmp	0.520	0.479
jmpjmp	1.0	0.0
movadc	0.0	1.0
addsub	0.446	0.554

opcode in malware files, and df_M refers to the document frequency of an opcode in malware files.

3.2 Rare Feature Elimination Using Naïve Bayes Method

Naïve Bayes (NB) can predict class membership probabilities, i.e. the probability that a given tuple belongs to a particular target class. The Naïve Bayes score [25] of each unique opcode is calculated using Eq. 1. In the equation, C denotes the target classes (Malware, Benign) and X represents the opcode. Table 3 shows the malware and benign NB scores of the bigram. For example, if $X = \text{adcpop}$, then

$$P(C_i|X) = \frac{P(X|C_i).P(C_i)}{P(X)} \tag{1}$$

$P(\text{adcpop}|B) = df_B/\text{No}$: of benign files in train set

$P(\text{adcpop}|M) = df_M/\text{No}$: of malware files in train set

$P(B) = \text{No}$: of benign files in train set/
(Total training files)

$P(M) = \text{No}$: of malware files in train set/
(Total training files)

$P(\text{adcpop}) = (df_B + df_M)/(\text{Total training files})$

Opcodes are sorted in decreasing order of NB_M and NB_B scores (refer Tables 4, 5). The opcode space is initially pruned by selecting the prominent 60 % opcodes from both the sorted lists. Naïve Bayes pruning is performed to select the features that have a higher possibility of identifying the target class. The same process is repeated on the unique list of unigrams and branch opcodes. Out of the 27 unique branch instructions in the train set, 20 branch opcodes were selected from each sorted list. In the case of unigram, 195 opcodes were extracted out of 325 unique features in the train set. And for bigrams, 4854 opcodes were picked out of 6923 features.

3.3 Dimensionality Reduction of Opcode Space

Dimensionality reduction plays a significant role in various domains such as pattern recognition, data compression and machine learning. Dimension of data space may be huge in many applications, whereas the meaningful part of the rep-

Table 4 Opcodes sorted based on NB_B score

Bigram	NB_B
jmpjmp	1.0
adcpop	0.783
addjmp	0.520
addsub	0.446
movadc	0.0

Table 5 Opcodes sorted based on NB_M score

Bigram	NB_M
movadc	1.0
addsub	0.554
addjmp	0.479
adcpop	0.217
jmpjmp	0.0

resentation of data lies at a lower-dimensional space. Thus, we need to derive only optimal dimensions that can effectively reflect the variability of data. Dimensionality reduction achieves the following: (a) provides accurate representation of high-dimensional data in low-dimensional space, (b) helps in better data visualization, (c) eliminates noise and outliers during learning phase and (d) improves classification accuracy also reduces time for model generation.

The optimal feature space should have a set of attributes qualifying following conditions:

- Case 1: An attribute having higher frequency in a class compared to the opponent class is considered useful.
- Case 2: An attribute present in majority of files in a class but absent in another class is significant (also known as discriminant attribute).
- Case 3: Attributes having same frequency (larger or fewer) in both classes are termed as insignificant and thus not considered for modelling.

In our feature selection approach, we try to determine the attributes qualifying case 1 and 2. Feature set obtained after Naïve Baye pruning is further reduced using proposed *Discriminant Feature Variance-based Approach (DFVA)* and *Markov Blanket*.

3.4 Discriminant Feature Variance-Based Approach (DFVA)

Opcodes pruned with NB_M score are treated as malware features, whereas opcodes selected based on NB_B are considered as benign. For investigating the efficiency of benign and malware features individually in classification, *DFVA* approach was applied separately on benign and malware fea-

Table 6 Optimal Feature Vector obtained with *DFVA* Feature Selection

Features	Total unique features	Features after Naïve Bayes pruning		Prominent features after applying <i>DFVA</i>	
		Benign	Malware	Benign	Malware
Branch opcodes	27	20	20	17	17
Unigram	325	195	195	110	112
Bigram	8090	4854	4854	2038	2916

tures. Thus, individual benign and malware training models were generated for *DFVA* and used for predictions.

For each opcode, we compute the $DFVA_{score}$ as per the Algorithm 1. This method initially considers two hypothesis, null hypothesis and alternate hypothesis. Null hypothesis (H_0) assumes that the frequency distribution of an opcode is same in both target classes. Alternate hypothesis (H_A) states that frequency distribution of an attribute is different in target classes.

$$H_0 : F_{opcode}^M = F_{opcode}^B$$

$$H_A : F_{opcode}^M \neq F_{opcode}^B$$

Here, F_{opcode}^M and F_{opcode}^B represent the frequency distribution of an opcode in malware and benign class, respectively. The confidence interval α is set to 0.05 level. If the calculated $DFVA_{score}$ is not in the range of threshold δ (-1.96 & $+1.96$) [26], then null hypothesis is rejected and we accept the alternate hypothesis which states that the frequency distribution of that particular opcode differs significantly in target classes. Hence, this feature is preserved and considered in optimal feature space. Thus, the reduced feature space of bigrams has 2916 prominent malware and 2038 bigram benign opcodes. There are 17 branch opcodes each in the prominent set of both the classes. And the optimal set of unigrams consists of 112 malware and 110 benign features. The optimal feature space obtained with *DFVA* is shown in Table 6.

3.5 Markov Blanket

A two-phase feature reduction method using Markov Blanket [21, 27] was proposed to determine the prominent features that are relevant but have minimum redundancy. Two types of correlation measures are thus estimated: (a) Feature–Class correlation denoted as *FCR* and (b) Feature–Feature correlation represented as *FFR*. In the first phase of feature reduction, the relevant features that are highly correlated to the target class are determined. In the second step, we try to remove the redundant features among the relevant features extracted in the previous step. The redundant features are eliminated by applying Markov Blanket where approximate Markov Blankets for relevant features are determined. A feature having larger feature–class correlation score car-

Algorithm 1 Proposed *DFVA* Method

Input:

- $F \leftarrow \{F_1, F_2, \dots, F_D\}$; F is the pruned set obtained after Naïve Bayes method, where F_i represents each feature in Dimension D
- $C \leftarrow \{M, B\}$; target classes where M is Malware and B is Benign
- $\theta \leftarrow 1.96$ at $\alpha \leftarrow 0.05$; where θ is the threshold value and α denotes the level of significance.

Output

- S : A set containing optimal features where $|S| = k$ such that $1 \leq k < D$

```

1:  $S \leftarrow \{\phi\}$                                 ▷ Optimal feature set initialised to null
2: for  $i \leftarrow 1$  to  $|F|$  do                       ▷ For each feature  $f_i$  in set F
3:    $p_B \leftarrow df_{f_i,B}/|B|$                     ▷  $df_{f_i,B}$  denotes the document
      frequency                                     of feature  $f_i$  in benign
      files,  $|B|$  denotes                             the number of
      benign files in train set
4:    $p_M \leftarrow df_{f_i,M}/|M|$                     ▷  $df_{f_i,M}$  denotes document frequency of
      feature  $f_i$  in malware files,  $|M|$ 
      denotes                                         the number of malware
      files in train set
5:    $q_B \leftarrow 1 - p_B$ 
6:    $q_M \leftarrow 1 - p_M$ 
7:    $p \leftarrow (|B|.p_B + |M|.p_M)/(|B| + |M|)$ 
8:    $q \leftarrow 1 - p$ 
9:    $SE \leftarrow \sqrt{p * q * [(1/|B|) + (1/|M|)]}$ 
10:   $DFVA_{score} \leftarrow (p_B - p_M)/SE$            ▷ Calculate  $DFVA_{score}$ 
      for each fea-
      ture
11:  if  $-\theta < DFVA_{score} < \theta$  then
12:    discard feature  $f_i$ 
13:  else
14:     $S = S \cup \{f_i\}$                              ▷ Add feature  $f_i$  to optimal set
15:  end if
16: end for
17: Sort ( $S$ )                                       ▷ sort  $DFVA_{score}$  in increasing order
18: Return  $S$ 

```

ries more information about the target class than a feature with lesser feature–class correlation value.

Relevance and redundancy analysis is performed as per the Algorithm in 2. Feature–class correlation score is calculated using Eqs. 2, 3 and 4. Entropy score [28] of any event ‘ x' ’ can be computed by the Eq. 4 where $p(k)$ is the probability of the k^{th} unit of information in event x' ’s sequence of N symbols. If $FCR(F_i, C) > \delta$, then feature F_i is considered as relevant, where δ is the experimentally determined relevance threshold. The relevance threshold δ selected for the

Table 7 Optimal Feature Space obtained with Markov blanket Approach

Features	Total unique features	Features after Naïve Bayes pruning	Relevant features after evaluating condition $SU_{i,c} > \delta$	Prominent features after evaluating condition $SU_{i,j} \geq SU_{i,c}$
Branch	27	22	16	7
Unigram	325	303	77	33
Bigram	8090	6925	919	406

experiment is 0.005. If F_i and F_j are any two features such that $FCR(F_j, C) \geq FCR(F_i, C)$, then we try to see whether feature F_j can behave as an markov blanket for feature F_i . Feature–Feature score is determined using Eqs. 5, 6 and 4. If $FFR(F_i, F_j) \geq FCR(F_i, C)$, then F_j forms a Markov Blanket for feature F_i . Hence, feature F_i is considered to be a redundant feature compared to F_j . We eliminate F_i from relevant feature set and keep its Markov blanket attribute F_j in the relevant feature set. The reduction in the feature space during successive stages of Markov Blanket is reported in Table 7.

$$FCR(F_i, C) = 2 * \frac{InfGain(F_i|C)}{H(F_i) + H(C)} \tag{2}$$

$$InfGain(F_i|C) = H(F_i) - H(F_i|C) \tag{3}$$

$$H(x) = - \sum_{k=1}^N \begin{cases} p(k) * \log_2 p(k), & p(k) \neq 0 \\ 0 & p(k) = 0 \end{cases} \tag{4}$$

$$FFR(F_i, F_j) = 2 * \frac{InfGain(F_i|F_j)}{H(F_i) + H(F_j)} \tag{5}$$

$$InfGain(F_i|F_j) = H(F_i) - H(F_i|F_j) \tag{6}$$

3.6 Model Generation

Initially, training models were created using the prominent opcodes obtained from individual feature reduction techniques. The pruned benign and malware features obtained after applying the DFVA method are arranged in the ascending order of their $DFVA_{score}$. Training models (vector space models) are created using the sorted features at different lengths. The classifiers used are Rotation Forest [29] and Random Forest [30], and the models are trained in WEKA [31–33]. We selected the model that provided the maximum value of evaluation metric at a minimum feature length. Likewise, training models with prominent Markov Blanket features were also constructed.

3.7 Prediction

Finally, the unseen samples were predicted with individual learning models obtained at optimal feature length. Two different meta feature spaces were created by ensembling the optimal set of unigram, bigram and branch features obtained

Algorithm 2 Proposed Markov Blanket Method

Input:

- $F \leftarrow \{F_1, F_2, \dots, F_D\}$; F is the pruned set obtained after Naïve Bayes method, where F_i represents each feature in Dimension D
- $C \leftarrow \{M, B\}$; target classes where M is Malware and B is Benign
- $\delta \leftarrow 0.005$; where δ is the relevance threshold

Output

- S_P : selected prominent feature subset where $|S_P| = k$ such that $1 \leq k < D$

```

1:  $S_{rel} \leftarrow \{\phi\}$  ▷ relevant feature set initialised to empty set
2: for  $i \leftarrow 1$  to  $|F|$  do ▷ For each feature  $f_i$  in set  $F$ 
3:    $FCR(F_i, C) = 2 * \frac{InfGain(F_i|C)}{H(F_i)+H(C)}$  ▷ Calculate feature-class score for each feature  $F_i$ 
4:   if  $FCR(F_i, C) > \delta$  then ▷ if feature-class score is greater than relevance threshold
       score
5:     Append  $F_i$  to  $S_{rel}$  list
6:   end if
7: end for
8: order relevant feature set  $S_{rel}$  in descending  $FCR(F_i, C)$  value
9:  $F_j = acquire\_First\_Feature(S_{rel})$ 
10: repeat
11:    $F_i = acquire\_Next\_Feature(S_{rel}, F_j)$ ;
12:   if  $(F_i \neq NULL)$  then
13:     repeat
14:        $FFR(F_i, F_j) = 2 * \frac{InfGain(F_i|F_j)}{H(F_i)+H(F_j)}$  ▷ Calculate feature-
       feature score
15:       if  $FFR(F_i, F_j) \geq FCR(F_i, c)$  then
16:         remove  $F_i$  from  $S_{rel}$  ;
17:       end if
18:        $F_i = acquire\_Next\_Feature(S_{rel}, F_j)$ ;
19:     until  $F_i = NULL$ 
20:   end if
21:    $F_j = acquire\_Next\_Feature(S_{rel}, F_j)$ ;
22: until  $F_j = NULL$ 
23:  $S_P = S_{rel}$ ;
24: Return prominent feature set  $S_P$  ;

```

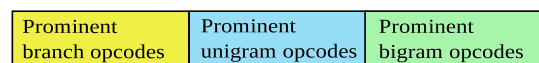


Fig. 2 Meta feature space

from individual feature reduction method (refer Fig. 2). Finally, test instances were also predicted using the meta feature space.

4 Experimental Setup

The experiments were performed on an Intel Pentium Core 3 Duo 2.3 GHz processor, 4GB RAM with Linux 14.04 operating system. Following are the types of investigations:

- Impact of feature length on classification.
- The capability of each category of a feature in classifying the metamorphic malware samples.
- Evaluation of malware/benign attribute for modelling.
- Empirical evaluation of the classifier which can ascertain better detection rate with minimum false alarms.

4.1 Dataset

The benign dataset consists of 1218 executables including files from `System32` directory of Windows XP operating system and files downloaded from other internet sources. The files were initially scanned in various commercial detectors in order to ensure that all the samples are legitimate. Malware dataset [22] consists of 868 files created using the NGVCK and MWORM engine. The malware samples were prepared by adding ‘dead codes’ by copying blocks of subroutines and instructions from the benign programs. Authors used three different code obfuscation techniques to create the virus samples : (a) equivalent instruction substitution, (b) dead code insertion and (c) instruction transposition. A dynamic scoring algorithm [22] was developed for comparing the similarity of resulting malicious code against the benign program after each morphing operation. The result of a code obfuscation operation is applied only if it makes the new virus structurally more similar to benign programs. Hence, this may cause viruses to bypass signature-based AV. The parameters used for evaluating the experimental results are discussed in the subsequent subsection.

4.2 Evaluation Parameters

Information retrieval parameters such as precision and recall are computed as follows,

$$\text{Precision} = TP / (TP + FP) \quad (7)$$

$$\text{Recall} = TP / (TP + FN) \quad (8)$$

F1 score is determined using the Eq. 9,

$$\text{F1 score} = (2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (9)$$

Accuracy is the ratio of total number of files correctly classified to the total number of files in the dataset. It is calculated using Eq. 10,

$$\text{Accuracy} = (TP + TN) / (TP + FN + TN + FP) \quad (10)$$

Here, TP represents the files correctly classified as malware, and FN denotes the files incorrectly classified as benign. TN stands for the files correctly classified as benign and FP means the files misclassified as malware.

5 Results and Discussions

In this section, the experimental results are discussed. Mainly, two experiments were performed here: (a) Exploration of metamorphism in the dataset and (b) Detection of metamorphic malware samples using the proposed feature selection methods.

5.1 Estimation of Degree of Metamorphism in the Dataset Using Sequence Alignment Method

Sequence alignment [34] is a fundamental approach used in biological study for comparing two or more protein or DNA sequences. This strategy attempts to discover similar regions as a whole or part to infer the evolutionary association among sequences. Metamorphic malware samples used in dataset [22] contain fragments of code binded with subroutines extracted from benign programs to defeat the detection from the scanners. This implies similar code regions will exist in the malware and benign instances. Hence, a threshold cannot be defined to differentiate the instances, and therefore, signature-based detection may fail. As the length of opcode sequence differs from one file to other, local alignment technique is used. Locations of non-discriminant opcodes in malware and benign instances are aligned using Smith–Waterman alignment method [35]. Following are the basic steps involved in aligning opcode locations in malware and benign files using this method:

- *Initialization* An alignment and a traceback matrix of dimension $(P + 1) \times (Q + 1)$ are created where P and Q are the length of two location sequences of a opcode in any two instances. Let the score and direction matrices be $AS(P + 1, Q + 1)$ and $TB(P + 1, Q + 1)$, respectively. First column and first row of these matrices are initialized as 0.
- *Populate score matrix and direction matrix* Alignment score of cell $AS(i, j)$ in the score matrix is computed with the scores of its neighboring three cells (left, diagonal and top). Its corresponding cell $TB(i, j)$ in direction matrix illustrates the direction of cell having maximum value that contributes to the score of new cell $AS(i, j)$. Hence, each cell in the direction matrix can have the values L (left), D (diagonal) or U (up). Score $AS(i, j)$ (refer Eq. 11) for two sequence X and Y is computed as:



$$AS(i, j) = \max \begin{cases} 0 \\ AS(i - 1, j - 1) + \sigma_{ij} \\ AS(i - 1, j) + \gamma \\ AS(i, j - 1) + \gamma \end{cases} \quad (11)$$

where σ_{ij} indicates match/mismatch score while aligning locations of an opcode and γ is gap penalty. We used a match score of +1, gap and mismatch score of -1 in our experimentation.

- **Direction:** Trace back step will find the optimal alignment from the direction matrix. Traceback starts at bottom-right cell $TB(M + 1, N + 1)$ until first column or row is reached.

For validation process, the locations of non-discriminant opcodes (branch instructions, unigrams and bigrams) in each malware file are aligned with the opcode locations in the every other malware files. Average value (denoted by avg) of alignment scores of an opcode in each malware file is calculated. After obtaining the average alignment scores for all the malware files, the minimum (\min_{avg}), maximum score (\max_{avg}) and average of average alignment scores (denoted by A_{avg}) for each opcode among all malware files are computed. Then, the number of instances that lie in the range $[\min_{avg} : A_{avg}]$ denoted by $\#\min_{avg}-A_{avg}$. Likewise, number of instances having value in the range $[A_{avg} : \max_{avg}]$ are also ascertained; this is represented by $\#A_{avg}-\max_{avg}$. Similarly, opcode locations in benign files are also aligned employing the above mentioned procedure. Using the alignment scores, we investigate whether appropriate threshold can be determined that can differentiate malware and benign files.

Validation results of few non-discriminant unigram, branch and bigram opcodes are shown in Tables 8, 9 and 10, respectively. Fig. 3 shows the possible cases that may occur while aligning the opcode locations in malware and benign files. In cases 1 and 2, we may be able to define a margin for discriminating the malicious files from normal programs; thus, this margin can be used for detection. In case 3, as the alignment scores of malware and benign training instances overlap, a threshold cannot be set to differentiate the unseen specimens. From Table 8, it can be observed that in case of opcode *bt*, more number of malware instances (327 out of 435 instances) fall in the range $[\min_{avg} : A_{avg}]$. As depicted in case 3, we can notice an overlap in the range of alignment scores $[\min_{avg} : A_{avg}]$ of malware ($[0: 0.000685]$) and benign $[0 : 0.000094]$. Similar trend as in case 3 was observed for all unigrams branch and bigram non-discriminant opcodes.

The results imply that common fragments of code exist in both malware and benign instances, and hence, we cannot define a threshold to differentiate them and thus signature-based detection fails. Tables 8, 9 and 10 ✗ represent that threshold cannot be defined. Thus, this experiment also suggests that the dataset used in here has enough metamorphism.

Table 8 Sequence alignment results of unigram opcodes

opcode	Malware files				Benign files				If threshold can be set?
	\min_{avg}	A_{avg}	\max_{avg}	$\#\min_{avg}-A_{avg}$	\min_{avg}	A_{avg}	\max_{avg}	$\#A_{avg}-\max_{avg}$	
bt	0	0.000685	0.009032	327	0	0.000094	0.003786	582	✗
clc	0	0.000015	0.001628	430	0	0.000023	0.003284	604	✗
cmovbe	0	0.000801	0.013569	373	0	0.000052	0.002456	584	✗
cmovns	0	0.000995	0.014534	366	0	0.000038	0.001561	590	✗
cwde	0	0.000416	0.010718	400	0	0.000001	0.000821	608	✗
enter	0	0.000865	0.016687	404	0	0.00009	0.006604	599	✗
pusha	0	0.000178	0.003793	410	0	0.000511	0.014209	576	✗
setl	0	0.000203	0.007597	419	0	0.000033	0.002243	594	✗
setle	0	0.000688	0.013621	411	0	0.000396	0.00986	579	✗
test	0	0.001317	0.002183	200	0	0.001987	0.013298	466	✗
xor	0	0.001384	0.003127	200	0	0.000614	0.008448	466	✗

Table 9 Alignment scores of branch opcodes

Branch opcodes opcode	Benign files										If threshold can be set?
	Malware files					Benign files					
	min _{avg}	A _{avg}	max _{avg}	#min _{avg} -A _{avg}	#A _{avg} -max _{avg}	min _{avg}	A _{avg}	max _{avg}	#min _{avg} -A _{avg}	#A _{avg} -max _{avg}	
ja	0	0.001291	0.002047	239	196	0	0.001226	0.008225	430	179	×
jb	0	0.001325	0.002114	230	205	0	0.001109	0.008192	439	170	×
jbe	0	0.001248	0.001892	236	199	0	0.001099	0.008191	427	182	×
jg	0	0.001239	0.001743	230	205	0	0.000885	0.008376	391	218	×
jge	0	0.000912	0.001684	129	306	0	0.000899	0.008214	402	207	×
jl	0	0.000902	0.001625	129	306	0	0.000936	0.008385	410	199	×
jle	0	0.001302	0.001926	231	201	0	0.001009	0.008444	388	221	×
jmp	0	0.001297	0.00196	248	187	0	0.001229	0.008538	401	208	×
jns	0	0.010256	0.041004	246	189	0	0.006115	0.034307	417	192	×
js	0	0.024382	0.071283	276	159	0	0.008288	0.047315	462	147	×

Moreover, the malicious files were structurally similar to legitimate files suggesting that in future hackers might use similar obfuscation strategy to bypass scanners and to fail opcode-based pattern matching approach.

5.2 Results of Proposed Methods

We generated learning models with the prominent features at variable feature length obtained using both the feature selection methods.

5.2.1 DFVA–Model Generation and Prediction Using Optimal Models Developed with Independent Feature Category

Branch Opcodes

(A) Benign Branch Opcodes

The prominent benign branch feature set consists of 17 opcodes. Three training models were created at varying feature lengths of 5, 10 and 17. The performance of the models is shown in Fig. 4. Random Forest (Random.F) classifier provided highest F1 score of 1.0 at feature lengths of 10 and 17, respectively. But out of these two models, preference is given to model with smaller feature length (ie. *FL* = 10) and hence utilized in prediction phase. In case of Rotation Forest (Rotation.F) classifier, highest F1 score of 1.0 was gained at a feature length of 17.

Unseen instances not used for modelling are classified using the optimal models constructed with Random and Rotation Forest classifiers. Table 11 shows the results of benign branch opcodes, and it can be observed that both classifiers provided an F1 score of 1.0 using their respective optimal learning models.

(B) Malware Branch Opcodes

Seventeen pruned opcodes were obtained from the malware dataset. From Fig. 5, it can be noticed that both the classifiers gave a F1 score of 1.0 in all feature lengths (ie. 5, 10 and 17). Hence, the model with minimum feature length (ie. *FL* = 5) is preferred during the predictions.

Table 12 shows the prediction results using the prominent malware branch features. It can be seen that both Random Forest and Rotation Forest classifiers provided the same F-measure of 1.0 at minimum feature length of 5. But Random Forest took less time (0.11 s) for predicting sample compared to Rotation Forest (0.18 s). Figure 6 shows the ROC curves plotted for optimal malware and benign branch features.

Table 13 shows the significant branch opcodes in malware and benign classes. We can observe that seven opcodes are common in both target classes. The Naïve Bayes score of these non-discriminant branch opcodes is illustrated in Fig. 7. Difference in the Naïve Bayes scores of branch opcodes in malware and benign files clearly demonstrate that these

Table 10 Alignment scores with bigram opcodes

opcode	Malware files					Benign files					If threshold can be set?
	min _{avg}	A _{avg}	max _{avg}	#min _{avg} -A _{avg}	#A _{avg} -max _{avg}	min _{avg}	A _{avg}	max _{avg}	#min _{avg} -A _{avg}	#A _{avg} -max _{avg}	
addadd	0.00113	0.003986	0.002175	243	192	0	0.014779	0.001669	490	119	×
addand	0	0.003841	0.002152	206	229	0	0.008311	0.000683	326	283	×
addimul	0	0.002857	0.000483	298	137	0	0.00826	0.00046	486	123	×
addjmp	0	0.002949	0.000486	297	138	0	0.00826	0.000455	486	123	×
addlea	0	0.004287	0.002346	215	220	0	0.008437	0.000987	377	232	×
addleave	0	0.003345	0.000531	334	101	0	0.008187	0.0003	492	117	×
addmov	0	0.003407	0.000539	333	102	0	0.008186	0.0003	491	118	×
addmovzx	0	0.004813	0.002646	203	232	0	0.008367	0.000536	379	230	×
addor	0	0.002981	0.000621	277	158	0	0.008286	0.000361	466	143	×
addpop	0	0.002979	0.000647	267	168	0	0.008331	0.000595	381	228	×
addpush	0	0.004273	0.001147	241	194	0	0.008182	0.000915	404	205	×
addsar	0	0.004278	0.00046	339	96	0	0.00553	0.000339	476	133	×
addshl	0	0.005042	0.001295	230	205	0	0.00529	0.000282	510	99	×
addshr	0	0.004565	0.000897	258	177	0	0.008284	0.000334	503	106	×
addsub	0	0.004266	0.000891	256	179	0	0.008361	0.000396	474	135	×
addtest	0	0.005288	0.002799	205	230	0	0.008264	0.000904	395	214	×
addxor	0	0.005405	0.002978	207	228	0	0.009016	0.001296	399	210	×
andadd	0	0.005909	0.003186	202	233	0	0.008821	0.001085	426	183	×
andand	0	0.006739	0.003518	199	236	0	0.008091	0.000795	342	267	×
andcall	0	0.006981	0.00358	206	229	0	0.00867	0.000552	412	197	×
andemp	0	0.007073	0.003852	201	234	0	0.008897	0.000661	393	216	×
andjmp	0	0.008016	0.004168	199	236	0	0.010784	0.001259	423	186	×
andlea	0	0.008689	0.004391	204	231	0	0.0097	0.00119	421	188	×
andmov	0	0.008353	0.004455	196	239	0	0.028061	0.003864	415	194	×

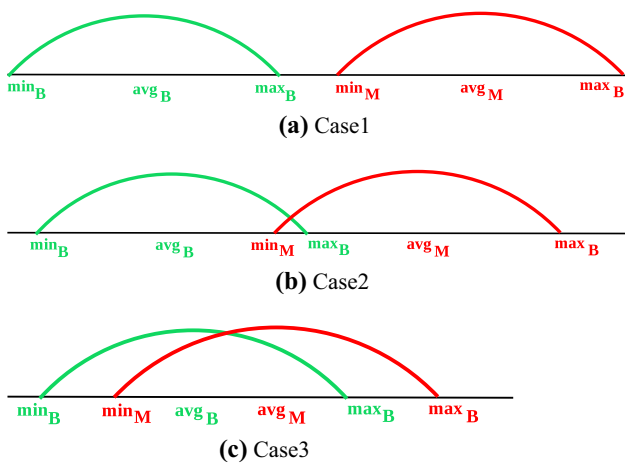


Fig. 3 Alignment score range

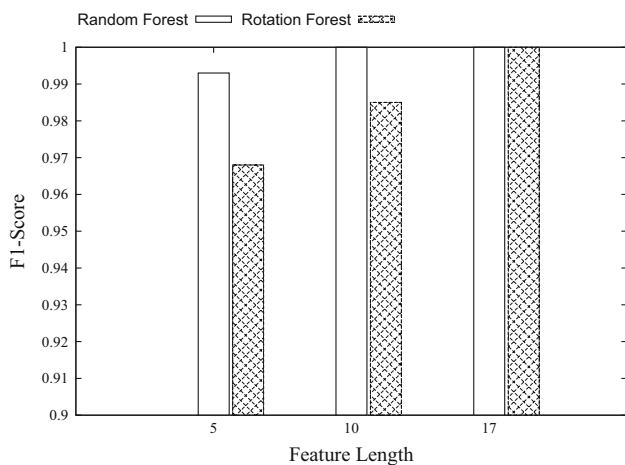


Fig. 4 Model created for branch opcodes (benign training set)

instructions can be used for classification of metamorphic malware samples.

Unigram The performance with unigram opcodes is discussed next for both benign and malware set.

(A) Benign Unigram Opcodes

The pruned set of unigram contains 110 opcodes. Maximum F-measure of 0.993 was gained with Random Forest at full feature space (ie. 110 opcodes). With Rotation Forest, F1 score of 0.997 was obtained again at full feature space. From Fig. 8, it can be observed that the performance of both classifiers degrades with decrease in the feature length since at smaller feature space, there is lack of useful information required for discriminating malware v/s benign.

Table 11 Evaluation Metrics of Benign Branch opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random.F	10	100	1	1	1	0.06
Rotation.F	17	100	1	1	1	1.28

Bold font indicates the best performance obtained with different experimental setting

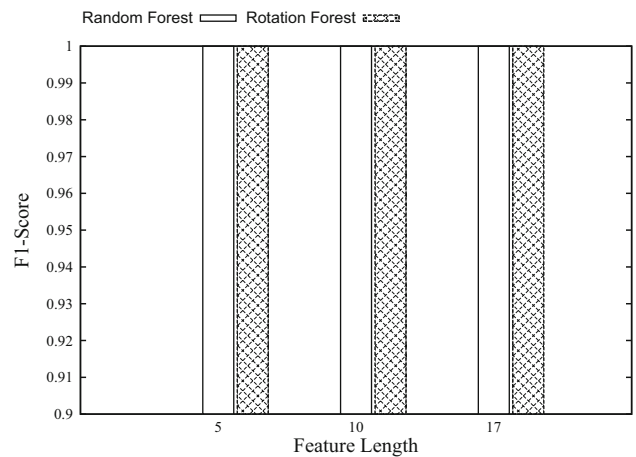


Fig. 5 Model created with branch opcodes (malware training instances)

The prediction results obtained with benign unigram opcodes are shown in Table 14. Random Forest provides the highest F1 score of 0.997 at its optimal feature length of 110. Rotation Forest could only produce a maximum F-measure of 0.993 at the same feature length.

(B) Malware unigram opcodes

There were 112 opcodes in the pruned feature set. Both the classifiers obtained highest F-measure of 1.0 at all feature lengths (10, 20, 30, 40, 50, 60, 70, 80, 90, 100 and 112) refer Fig. 9. Hence, the model with minimal feature length (ie. 10 opcodes) is selected as the optimal model.

Table 15 depicts the results obtained with malware unigram opcodes. Both Random and Rotation Forests result in an F-measure of 1.0 at their optimal feature length (i.e. 10). But Random Forest is efficient since it consumes less time (0.02 s) for identification of new instances. On the other hand, Rotation Forest takes 0.21 s for predicting test instances. Figure 10 shows the ROC curves for optimal malware and benign branch features.

Table 16 shows the top ten unigram opcodes for both benign and malware samples. The top ten unigram malware features were found to be discriminant. A feature is considered discriminant to a class if it is prominent in one class compared to other.

Bigram

(A) Benign bigram opcodes

With Random Forest classifier, an optimal model is obtained with 500 features as shown in Fig. 11. However, with Rotation Forest, highest performance is obtained with 1500 features.

Table 12 Evaluation metrics of Malware Branch opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random.F	5	100	1	1	1	0.11
Rotation.F	5	100	1	1	1	0.18

Bold font indicates the best performance obtained with different experimental setting

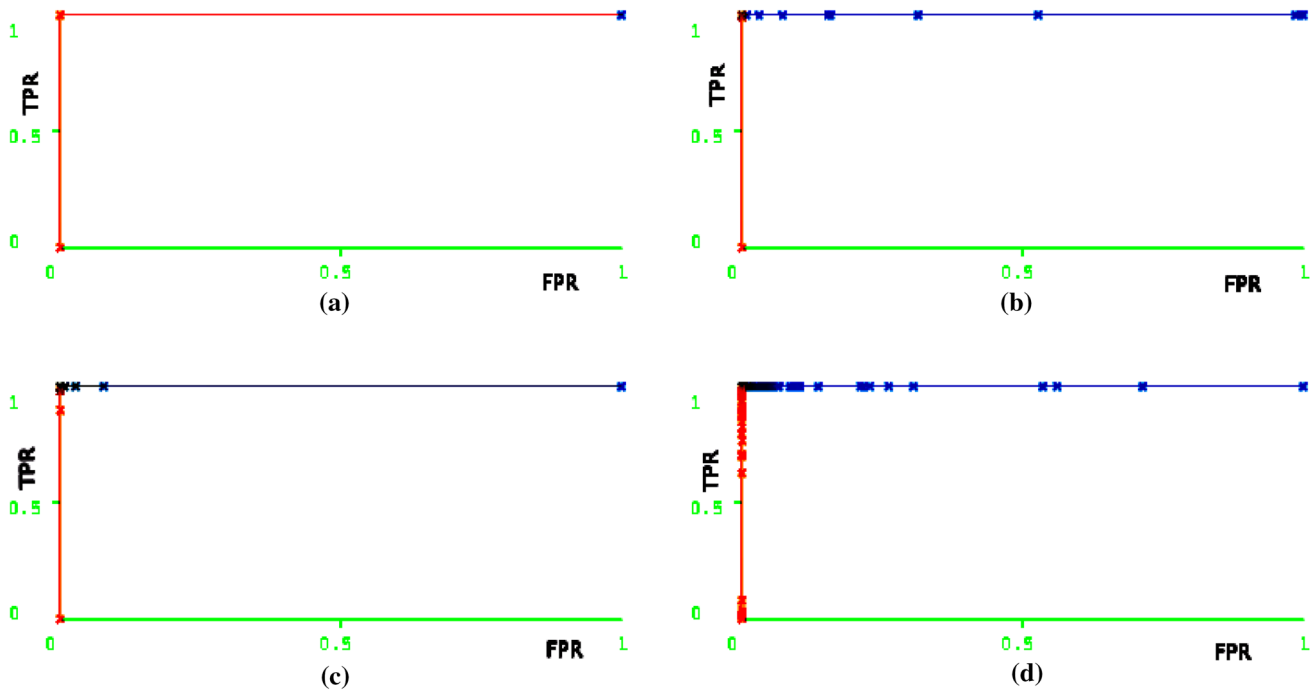


Fig. 6 ROC for Branch opcodes. **a** Malware Branch opcodes—Random Forest. **b** Malware Branch opcodes—Rotation Forest. **c** Benign Branch opcodes—Random Forest. **d** Benign Branch opcodes—Rotation Forest

Table 13 Top ten branch opcodes

S. No	Malware opcodes	Benign opcodes
1	je	jb
2	jne	ja
3	jae	jg
4	jb	jl
5	ja	jbe
6	jg	jle
7	jl	jmp
8	jbe	jno
9	jle	jo
10	jmp	jnp

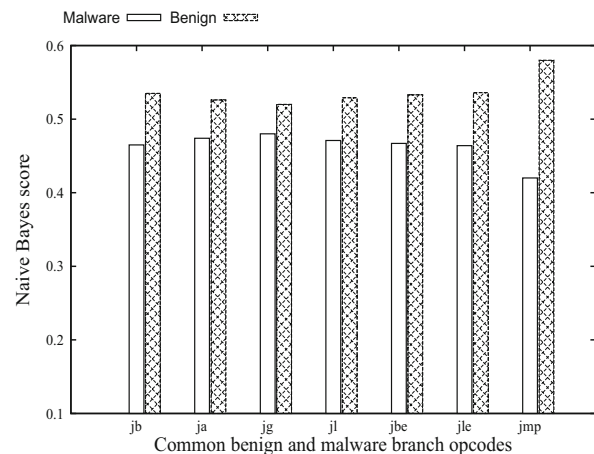


Fig. 7 Naive Bayes score of common branch opcodes in Target Class

Outcome of prediction with bigram benign features is shown in Table 17. Rotation Forest achieves F1 score of 1.0 at feature length of 1500, whereas Random Forest provides F-measure of 0.995 with a feature space of 500.

(B) Malware bigram opcodes

From Fig. 12, it can be seen that similar F1 scores are obtained with malware opcodes at varied feature length. Hence, the

model at minimal feature length (ie. 10) is used for predicting the unseen instances. From Table 18, it can be observed that F1 score of 1.0 was obtained by both classifier at a feature length of 10. But Random Forest classifier is considered superior since it identifies test samples faster compared to

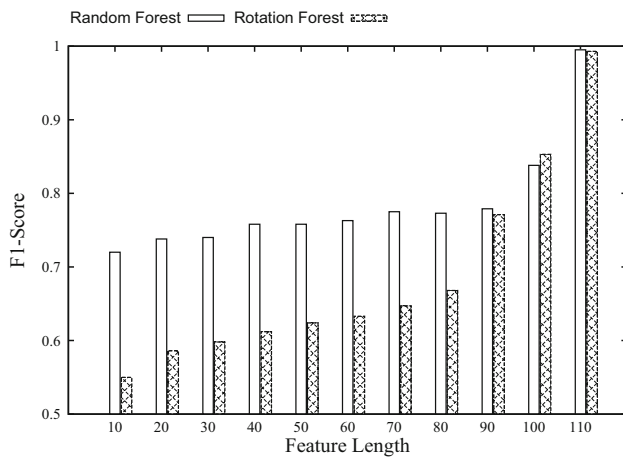


Fig. 8 Model created for unigram benign opcodes

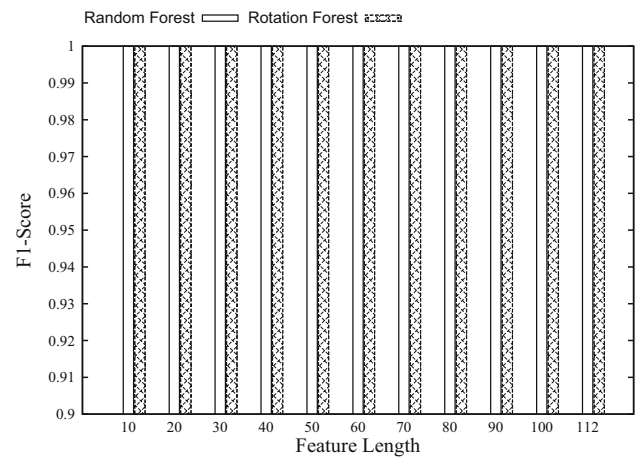


Fig. 9 Model created for unigram malware opcodes

Rotation Forest. ROC curves for prominent bigram features are depicted in Fig. 13. Significant ten bigrams for both set are shown in Table 19.

While comparing the results, it can be observed that benign models require more number of features for precisely predicting the new samples. On the other hand, malware models can be preferred since they need only less number of features for classification. The reason behind this difference is that benign files are distinct. So, the common opcodes are limited due to their diversification. But malware samples are either developed using metamorphic engines or created by authors in low-level languages. Hence, there is a high probability for common opcodes to appear in variants of malicious base files. So, these opcodes will be utilized by the metamorphic engine to retain the maliciousness. Also, it is too difficult to mutate the complete x86 assembly codes with equivalent opcodes. Hence, a malware model may be developed, but it is hard to generalize a benign model.

Prediction Using Meta Feature Space Created by DFVA

It was observed that malware features are have established more efficacy in classification than benign features. Hence, the unseen samples were predicted using a meta feature space formed by ensembling the prominent malware features (known as meta features) obtained from the previous prediction results. It consists of:

1. Five Malware branch opcodes
2. Ten Malware unigram opcodes
3. Ten Malware bigram opcodes

From Table 20, we could visualize that the proposed system yielded an accuracy of 100% and F1 score of 1.0, using the ensemble of malware features. Similar trends were obtained with Random Forest classifier compared to Rotation Forest.

5.2.2 Markov Blanket–Model Generation and Prediction Using Individual Models

Branch Opcodes

The prominent set of attributes contains seven branch instructions. As depicted in Fig. 14, we could see that two models were generated with branch opcodes, one with a feature length of seven (full feature space) and another consisting of five opcodes. Random Forest yielded a F1 score of 1.0 for both the learning models. Rotation Forest classifier resulted in an F-measure of 1.0 for the model with feature space 7; moreover, F1-value degraded to 0.969 with feature length 5 branch instruction.

Hence, the optimal feature length in case of Random Forest and Rotation Forest is 5 and 7 instructions, respectively. Now, the class of unlabelled samples is predicted using optimal models. From Table 21, we can observe that F1 score of 1.0 is obtained with both learning models. Again, model developed with Random Forest is considered as it uses fewer feature and consumes minimum time for classification. ROC curves plotted for prominent branch opcodes are shown in Fig. 15.

Table 14 Evaluation metrics of benign unigram opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random.F	110	99.808	0.995	1	0.997	0.16
Rotation.F	110	99.425	0.989	0.998	0.993	2.3

Bold font indicates the best performance obtained with different experimental setting

Table 15 Results of Malware Unigram opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random.F	10	100	1	1	1	0.02
Rotation.F	10	100	1	1	1	0.21

Bold font indicates the best performance obtained with different experimental setting

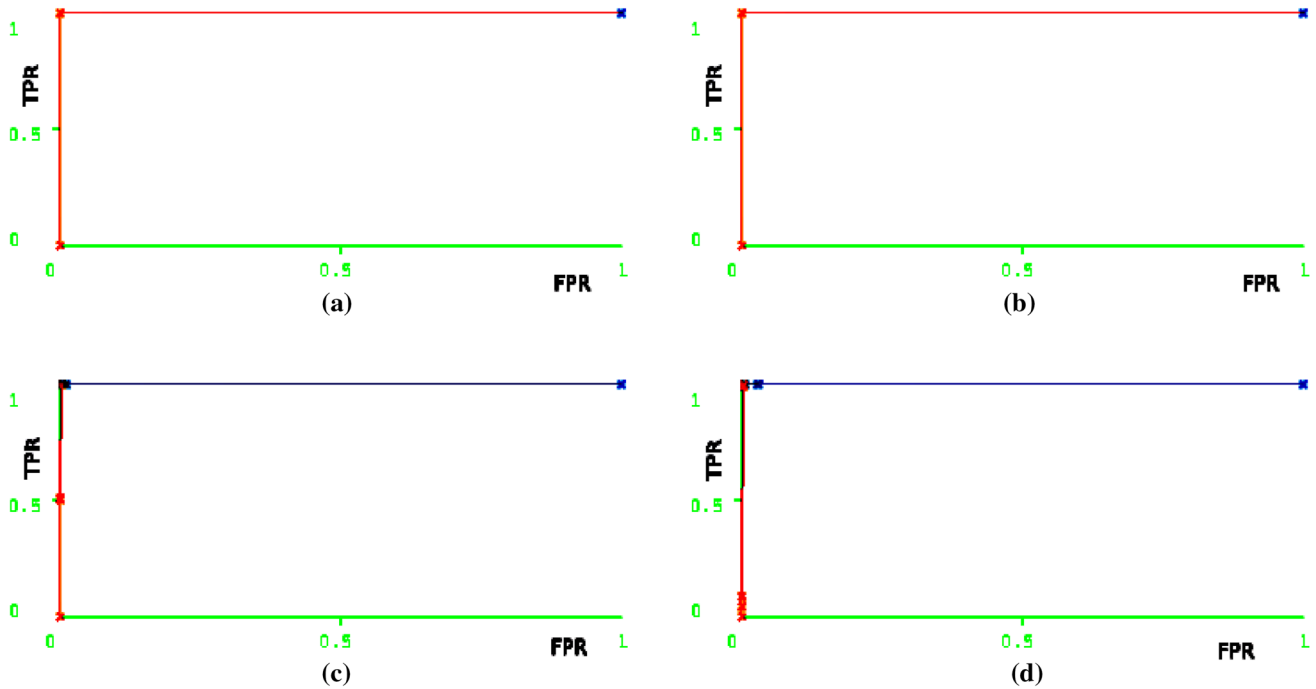


Fig. 10 ROC curves for Unigram opcodes. **a** Malware unigram opcodes—Random Forest. **b** Malware unigram opcodes—Rotation Forest. **c** Benign unigram opcodes—Random Forest. **d** Benign unigram opcodes—Rotation Forest

Table 16 Top ten unigram opcodes

S. No	Malware opcodes	Benign opcodes
1	ret	cli
2	repz	cmpsd
3	seta	das
4	cmove	fucom
5	data32	fucomp
6	cdqe	movq
7	repz	prefetchnta
8	seta	rdtsc
9	sete	scasd
10	movsxd	ffree

Unigram Opcodes

There are 33 opcodes in the prominent unigram set. Four training models are created at feature length of 10, 20, 30 and 33 opcodes. Maximum F-measure of 1.0 was obtained for all the four models with both classification algorithms (refer Fig. 16). Hence, we select the model with minimum feature length (10) for predicting unseen samples not involved in modelling.

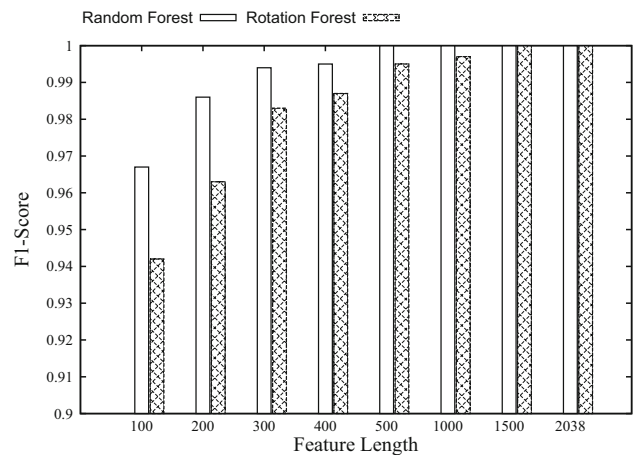


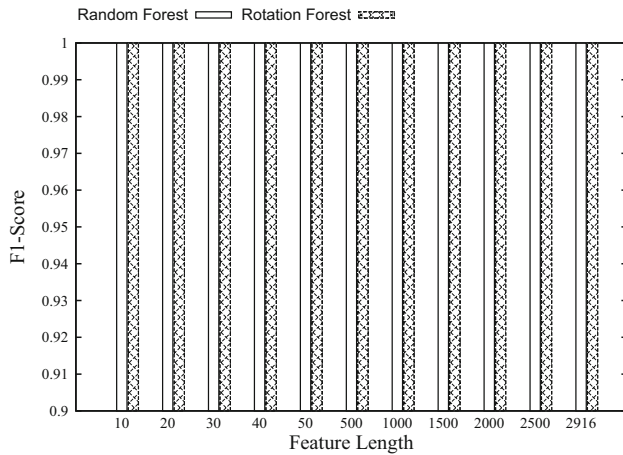
Fig. 11 Optimal benign bigram model

Table 22 shows the results of prediction obtained for unigram. F1 score of 1.0 is obtained with both classifiers using ten significant opcodes. Random Forest takes less time (0.04s) for prediction, whereas Rotation Forest takes 0.22s. We again consider Random Forest model as superior in com-

Table 17 Evaluation metrics of Benign Bigram opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Rotation.F	1500	100	1	1	1	80.81
Random.F	500	99.90	1	0.99	0.995	0.44

Bold font indicates the best performance obtained with different experimental setting

**Fig. 12** Optimal malware bigram model

comparison with Rotation Forest. Fig. 17 illustrates the ROC curves for unigram features.

Bigram opcodes

As shown in Fig. 18, prominent set of bigram consists of 406 features and 13 training models were generated for bigram at varying feature lengths of 406, 300, 200, 100, 90, 80, 70, 60, 50, 40, 30, 20 and 10. Both classifiers yielded a F1 score of 1.0 with all the 13 models generated. Again, the model with smallest feature length (ie. 10) is selected.

From Table 23, it can be observed that both Random Forest and Rotation Forest resulted in an F-measure of 1.0 with bigrams. As in case of unigram, Random Forest predicts faster (0.16 s) than Rotation Forest (0.89 s) and hence considered as precise for modelling. ROC curves plotted for bigrams are shown in Fig. 19.

5.2.3 Prediction Results with Meta Feature Space Generated Using Markov Blanket

Unseen instances are also predicted using a meta feature space model created by ensembling the prominent features obtained from optimum models considering independent categories of attributes. Therefore, ensemble attribute space consisted of

Table 18 Evaluation Metrics of Malware Bigram opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random.F	10	100	1	1	1	0.02
Rotation.F	10	100	1	1	1	0.24

Bold font indicates the best performance obtained with different experimental setting

1. Top 5 branch opcodes
2. Top 10 unigram features
3. Top 10 bi-gram features

The outcome of prediction for features pruned with Markov Blanket is shown in Table 24. Thus, the proposed system classified all the unseen samples with an accuracy of 100 %, precision 1.0, recall 1.0 and F1 score of 1.0 with individual as well as with meta feature attribute set.

6 Comparison with Prior Works

Table 25 shows the performance of proposed scanner and other metamorphic malware detection techniques utilizing the similar dataset (synthetic metamorphic virus samples). From the tabulated results, it can be found that the proposed system is acceptable to other approaches.

7 Inference of Experiments

The conducted experiments revealed following answered questions:

- *What is the influence of feature length on classification performance and accuracy?* During preparation of learning models, there aroused situations wherein maximum performance was obtained at smaller feature lengths. When opcodes were extracted from malware and later used for modelling; then, higher accuracy was gained. However, with benign models, better performance was obtained at a larger feature length compared to malware models. This denotes that generic benign models are difficult to be constructed. Larger feature space consisted of irrelevant attributes that appear as noise and increases the misclassification rate.
- *Which classifier provides best results?* Random Forest [30] was found to be the better classifier since it provides highest value of evaluation metrics with minimum features and consumes less time for prediction compared to Rotation Forest. Random Forest classifier

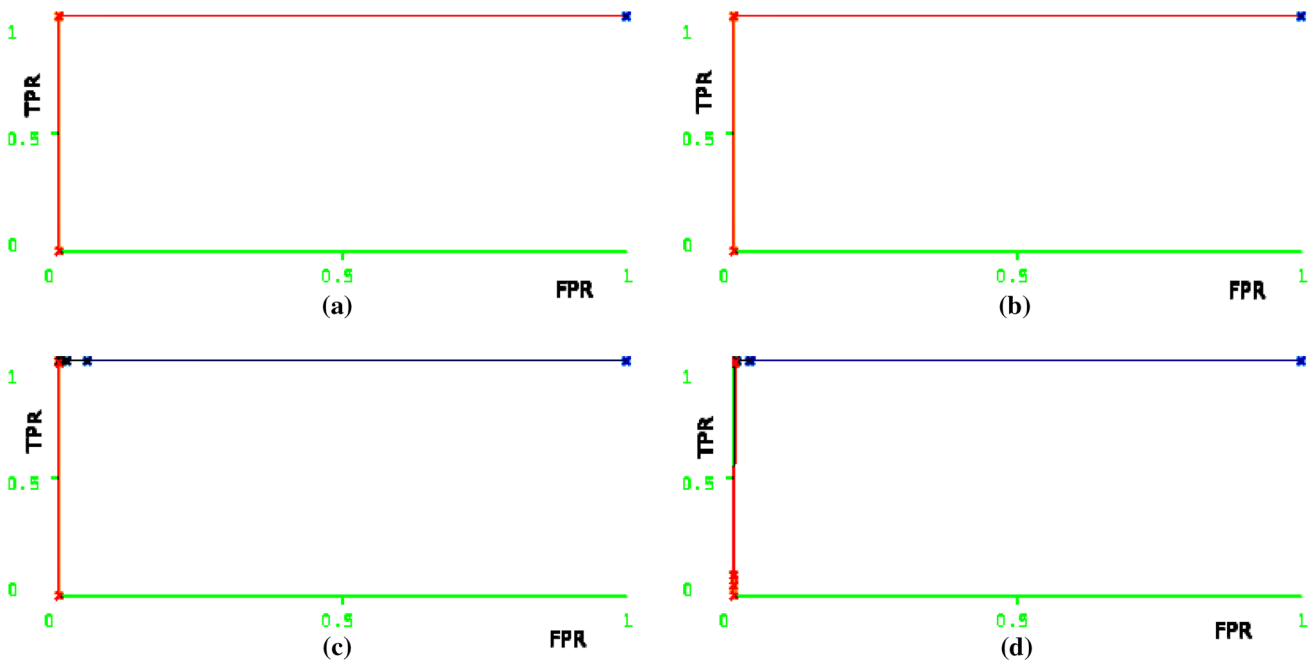


Fig. 13 ROC for Bigram opcodes. **a** Malware bigram opcodes—Random Forest. **b** Malware bigram opcodes—Rotation Forest. **c** Benign bigram opcodes—Random Forest. **d** Benign bigram opcodes—Rotation Forest

Table 19 Top ten bigram opcodes

S. No	Malware opcodes	Benign opcodes
1	cmpje	adcimul
2	cmpjne	adcor
3	jmov	andnop
4	jeadd	bsrxor
5	retpush	cdqeadd
6	jecmp	cdqlea
7	jnejmp	cdqsar
8	jnemov	cmovbadd
9	testje	cmovbtest
10	testjne	cmovzxor

few features in random that have higher probability in detecting the target class. This property of Random Forest helps in scaling up a model. Also, original feature space is expanded by the addition of diverse category of features found to be minimally correlated. Moreover, Rotation Forest’s [36] prediction is slow since it performs PCA (Principle Component Analysis) on the

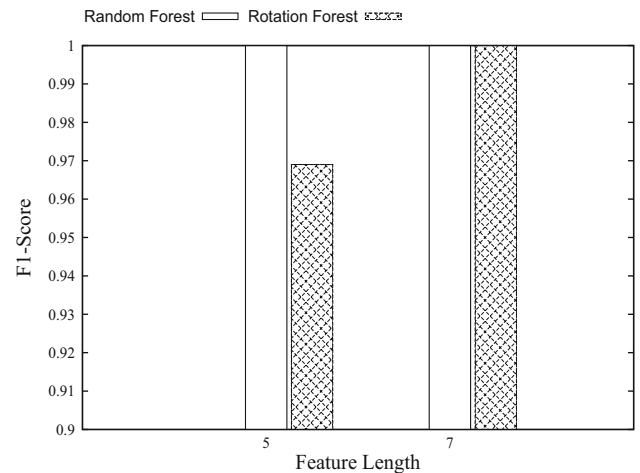


Fig. 14 Model created for branch opcodes

vector space model as a part of dimensionality reduction for each tree in the forest.

- Which is the best feature? Discriminant branch instructions, unigram and bigram opcodes pertaining to malware families are shown to be best in classification. Malware files are coded in low-level language. There-

Table 20 Evaluation metrics of Meta Malware Feature space created using DFVA (branch opcodes (5), unigram opcodes (10), bi-gram opcodes (10))

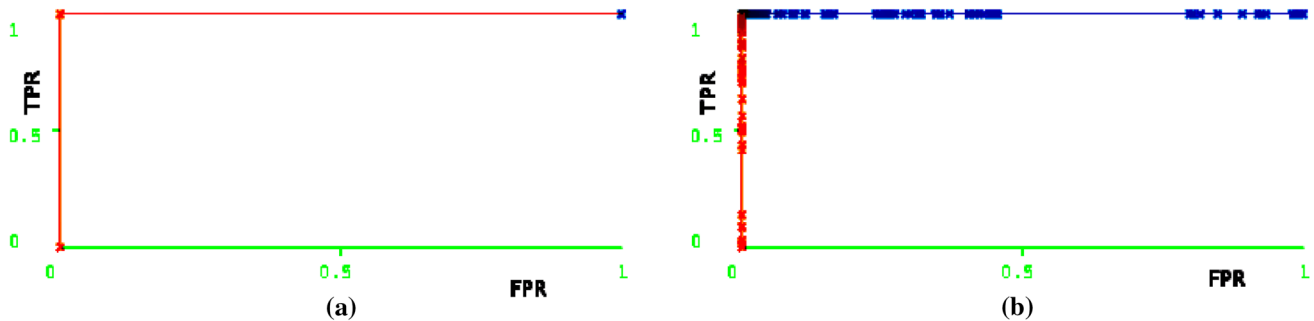
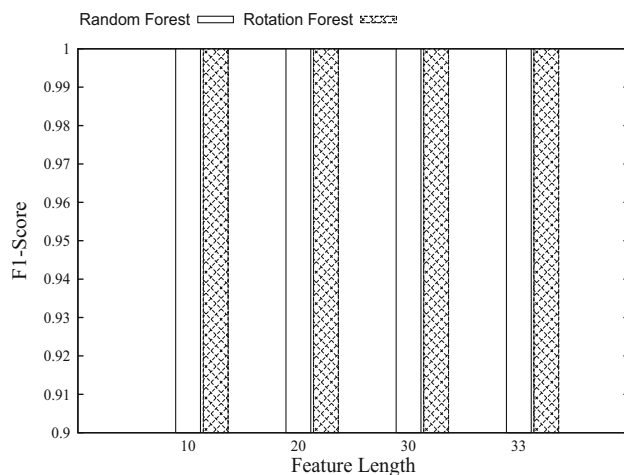
Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random.F	25	100	1	1	1	0.14
Rotation.F	25	100	1	1	1	1.28

Bold font indicates the best performance obtained with different experimental setting

Table 21 Evaluation Metrics for branch opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random.F	5	100	1	1	1	0.02
Rotation.F	7	100	1	1	1	0.22

Bold font indicates the best performance obtained with different experimental setting

**Fig. 15** ROC for Branch opcodes. **a** Branch opcodes—Random Forest. **b** Branch opcodes—Rotation Forest**Fig. 16** Model created for unigram opcodes

fore, few features are possibly preserved in viruses to retain maliciousness in the generations, whereas benign instances are distinct and are usually programmed in high-level language. Hence, it is difficult to generalize the benign models; however, generic malware models may be developed. The role of branch instructions in classification was investigated. Branch instructions were used for obfuscating the malicious payload to complicate analysis of malware files. It was discovered that the prominent five malware branch features classified

the unseen instances with an accuracy of 100% and F1 score of 1.0. Likewise, unigram and bigram malware instructions predicted the test samples with an accuracy of 100% and F-measure of 1.0 at a feature length of 10.

- *Whether the branch opcodes are capable enough in detecting the metamorphic malware instances?* It was discovered that the prominent branch opcodes obtained after applying the *DFVA* and *Markov Blanket* feature reduction methods are efficient enough to identify the samples (F1 score of 1.0 and accuracy 100%). Since the branch opcodes are primarily used to introduce control flow obfuscation in a piece of malware code, also to make malware appear structurally similar to legitimate files.
- *How is the classification performance influenced when individual features (branch, unigram and bigram) and meta feature space are used?* Proposed system classified all the unseen instances accurately with individual features as well as with meta feature space (F1 score of 1.0 and accuracy 100%).

8 Analysis on Real Samples

In order to evaluate the performance on real malware dataset, a total of 5514 Portable Executable (PE) files were collected. The malware samples constituted 2217 executables gathered

Table 22 Evaluation Metrics for unigram opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random.F	10	100	1	1	1	0.04
Rotation.F	10	100	1	1	1	0.22

Bold font indicates the best performance obtained with different experimental setting

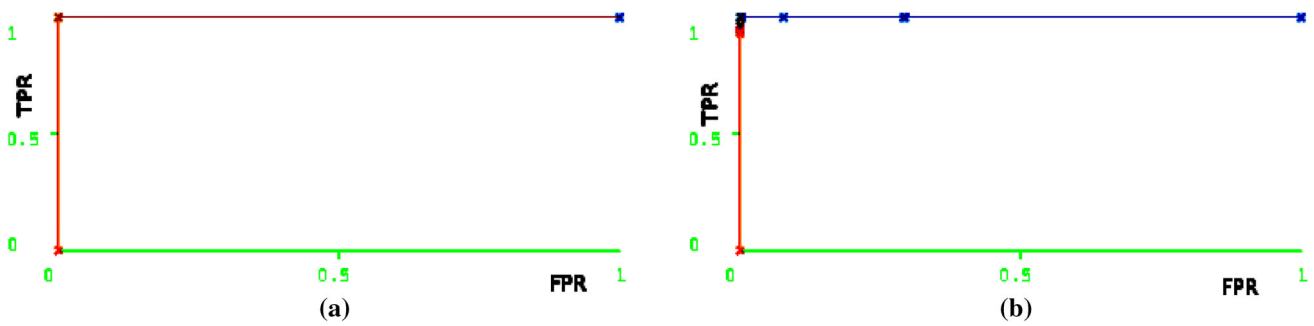


Fig. 17 ROC for Unigram opcodes. a Unigram—Random Forest. b Unigram—Rotation Forest

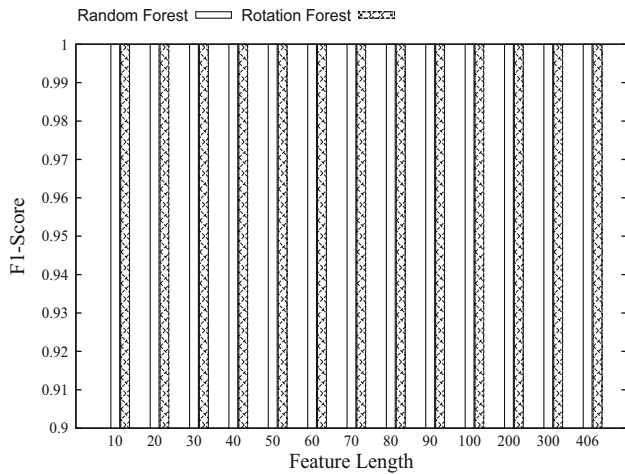


Fig. 18 Model created for bigram opcodes

from sources such as VX Heavens (<http://vx.netlux.org>) and security agencies. Benign dataset (3307 executables) comprised executables obtained from diverse sources such as System32 folder (Windows XP), games, browsers and media

players. Most of the real malware samples were packed to obfuscate the malicious payload to thwart detection. A packer is software compresses the original malware file, thus making the original code and data unreadable. The primary purpose of customized or commercial packers is to introduce an additional layer over malware so as to deter detection.

Malware executable were unpacked using signature-based unpackers such as VMPackers, GUNPacker (<http://www.woodmann.com/collaborative/tools>). Features are independently extracted from malware and legitimate files; later, the classification models are prepared using the algorithms implemented in WEKA with default settings. Models are evaluated using 10-fold cross-validation. The k -fold cross-validation method partitions input data into k equal subsets also known as folds. A model is trained on $k - 1$ folds and validated against the last fold. The entire experiment is repeated k times. The advantage of using cross-validation is to provide an insight into how the classifier perform on variable size input. Also, cross-validation is used to obtain large number of test samples from a limited set of data. Finally, experiments are conducted on prominent features of differ-

Table 23 Evaluation Metrics for bigram opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random.F	10	100	1	1	1	0.16
Rotation.F	10	100	1	1	1	0.89

Bold font indicates the best performance obtained with different experimental setting

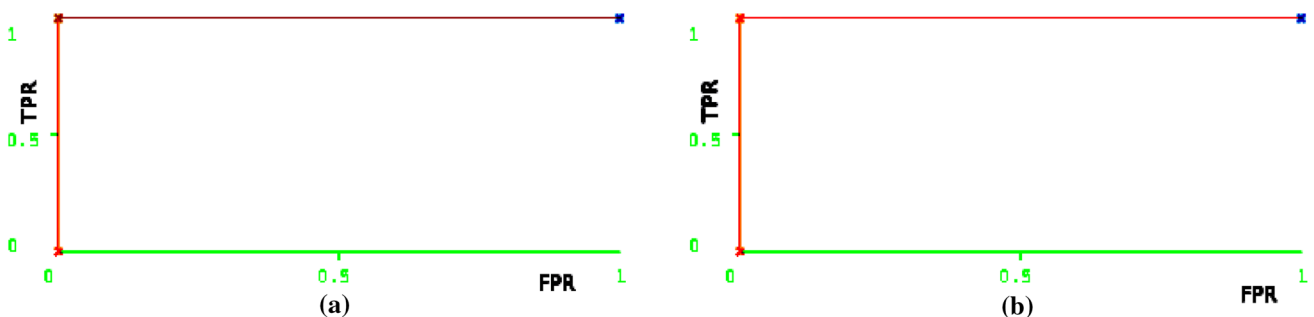


Fig. 19 ROC for Bigram opcodes. a Bigram—Random Forest. b Bigram—Rotation Forest

Table 24 Evaluation metrics of the Meta Feature space created using Markov Blanket. Feature space comprises of five branch instructions, ten unigram opcodes and ten bigram opcodes

Classifier	Feature length	ACC (%)	Precision	Recall	F1	Time (s)
Random Forest	25	100	1	1	1	0.17
Rotation Forest	25	100	1	1	1	1.62

Bold font indicates the best performance obtained with different experimental setting

Table 25 Comparison of proposed system with existing work using similar dataset

References	Proposed approach	Outcome
Priyadarshi et al. [13]	Emulator based approach	Detection rate—100% at morphing levels 15, 35% Rate decrease at levels 55 and 75%
Shanmugam et al. [14]	Opcode-based software similarity technique	Accuracy—100% when tested against MWOR, with padding ratios 1.0 and below. Misclassification increases For padding ratios beyond 1.0
Tahan et al. [15], Vinod et al. [16]	Common segment analysis Used multiple sequence alignment method	Accuracy—0.986, FPR—0.006 For group signature, detection rate—72.2%, FPR = 0.01
Donabelle et.al. [7]	Utilized structural entropy	The 8 KB NGVCK files resulted in 100% detection rate Without any false positive
Deshpande et al. [18]	Eigenvalue analysis	The technique gave more than 94% accurate detection of virus files
Zhao et al. [38]	Employed control flow graph of disassembled code	Accuracy—97%, 3.2% false rate
Kuriakose et al. [20]	Used several feature ranking methods	Detected MWORM and NGVCK virus with accuracy of 100%
Proposed system	Created meta feature spaces using DFVA and Markov Blanket	Accuracy—100%, Precision—1.0, Recall—1.0, F1 score—1.0

ent length and the effect of feature length on classification accuracy is investigated. The results are tabulated for the outcomes obtained with Random Forest classifier as it depicted the similar trend with previous experiments. As with all earlier experiments, Random Forest obtained elevated values of performance metric compared to other classification algorithms. Table 26 depicts an accuracy of 96.2% (low FPR value 0.038) and 96.4% (low FPR value 0.036) obtained with fewer benign and malware meta opcodes compared to attribute consisting of more number of opcodes. This experiment also justifies the importance of eliminating insignificant attributes to attain higher classifier performance. We anticipate that the results can be further improved if the unpacking of malware samples is performed with dynamic unpacker such as ether [37]. However, the pivotal limitation is how long a sample is required to be executed in the sandbox is a real challenge. An alternative is to design a random input generator for desktop malware detectors which could simulate the system and UI interactions satisfying complete code coverage.

Table 26 Performance on real samples

Classifiers	50	100	150	200	250	300
RF–Benign	96.3	96.1	96.2	95.7	96.1	96.2
RF–Malware	96.4	96.4	96	96.3	95.6	96

9 Conclusions

Two meta feature spaces each consisting of twenty-five prominent features were used by the proposed system for metamorphic malware detection. Three sets of features were extracted from the files: (a) branch instructions, (b) unigrams and (c) bigrams. The features were initially pruned using Naïve Bayes approach, and then, the dimension of feature space was further reduced using two different methods: (a) DFVA and (b) Markov Blanket. Training models were created at different feature lengths, and the optimal model at synthesized feature space was employed for the prediction of unseen instances. Thus, the impact of

variable feature length on detection accuracy was investigated in this study. The proposed system utilized the relevant/prominent discriminant malware features for classification and detected MWORM and NGVCK viruses with an accuracy of 100 % and F1 score of 1.0. From the results of the experiments, it can be claimed that the degree of metamorphism present in NGVCK and MWORM viruses is strong to defeat signature-based detection. But the non-signature-based approach effectively removes unwanted instruction added as dead code. In future, the work can be expanded on a huge dataset and real metamorphic instances. Also, other robust feature dimensionality reduction methods can be explored. Another form of obfuscation that primarily will be looked on is code packing. Also, malicious samples are encoded using strong encoders such as XOR or Shikata-ganai would be studied. To address this problem, we would initially learn the scanning behavior of the commercial AV to identify whether anti-virus scan the samples using fixed order (i.e. top, bottom) or perform random search. Once this is ascertained the entropy analysis of a fragment of the samples can be investigated to flag an unknown file as malware.

Acknowledgements We would like to convey our sincere gratitude to Professor **Mark Stamp** of **SJSU California** for providing the metamorphic malware samples for carrying out this study.

References

- Skoudis, E.; Zeltser, L.: *Malware: Fighting Malicious Code*. Prentice Hall Professional, Upper Saddle River (2004)
- Bilge, L.; Dumitras, T.: Before we knew it: an empirical study of zero-day attacks in the real world. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pp. 833–844. ACM (2012)
- Wong, W.; Stamp, M.: Hunting for metamorphic engines. *J. Comput. Virol.* **2**(3), 211–229 (2006)
- Stamp, M.: *Information Security: Principles and Practice*. Wiley, Hoboken (2011)
- Attaluri, S.; McGhee, S.; Stamp, M.: Profile hidden markov models and metamorphic virus detection. *J. Comput. Virol.* **5**(2), 151–169 (2009)
- Sridhara, S.M.; Stamp, M.: Metamorphic worm that carries its own morphing engine. *J. Comput. Virol. Hacking Tech.* **9**(2), 49–58 (2013)
- Donabelle, B.; Low Richard, M.; Mark, S.: Structural entropy and metamorphic malware. *J. Comput. Virol. Hacking Tech.* Springer, pp. 1–14 (2013)
- Bilar, D.: Opcodes as predictor for malware. *IJESDF* **1**(2), 156–168 (2007)
- Santos, I.; Brezo, F.; Nieves, J.; Playa, Y.K.; Sanz, B.; Laorden, C.; Bringas, P.G.: Idea: opcode-sequence-based malware detection. In: *Engineering Secure Software and Systems*, pp. 35–43. Springer, Berlin (2010)
- Santos, I.; Brezo, F.; Sanz, B.; Laorden, C.; Bringas, P.G.: Using opcode sequences in single-class learning to detect unknown malware. *IET Inf. Secur.* **5**(4), 220–227 (2011)
- Moskovitch, R.; Feher, C.; Tzachar, N.; Berger, E.; Gitelman, M.; Dolev, S.; Elovici, Y.: Unknown malware detection using opcode representation. In: *Intelligence and Security Informatics*, pp. 204–215. Springer, Berlin (2008)
- Shabtai, A.; Moskovitch, R.; Feher, C.; Dolev, S.; Elovici, Y.: Detecting unknown malicious code by applying classification techniques on opcode patterns. *Secur. Inf.* **1**(1), 1–22 (2012)
- Priyadarshi, S.; Stamp, M.: *Metamorphic detection via emulation*. Masters Report (2011)
- Shanmugam, G.; Low, R.M.; Stamp, M.: Simple substitution distance and metamorphic detection. *J. Comput. Virol. Hacking Tech.* **9**(3), 159–170 (2013)
- Tahan, G.; Rokach, L.; Shahar, Y.: Mal-id: automatic malware detection using common segment analysis and meta-features. *J. mach. learn. res.* **13**, 949–979 (2012)
- Vinod, P.; Laxmi, V.; Gaur, M.; Chauhan, G.: Momentum: metamorphic malware exploration techniques using msa signatures. In: *2012 International Conference on Innovations in Information Technology (IIT)*, pp. 232–237. IEEE (2012)
- Runwal, N.; Low, R.M.; Stamp, M.: Opcode graph similarity and metamorphic detection. *J. Comput. Virol.* **8**(1–2), 37–52 (2012)
- Deshpande, S.; Park, Y.; Stamp, M.: Eigenvalue analysis for metamorphic detection. *J. Comput. Virol. Hacking Tech.* **10**(1), 53–65 (2014)
- Annachhatre, C.; Austin, T.H.; Stamp, M.: Hidden markov models for malware classification. *J. Comput. Virol. Hacking Tech.* 1–15 (2014)
- Kuriakose, J.; Vinod, P.: Unknown metamorphic malware detection: Modelling with fewer relevant features and robust feature selection techniques. *IAENG Int. J. Comput. Sci.* **42**(2), (2015)
- Raphel, J.; Vinod, P.: Pruned feature space for metamorphic malware detection. In: *Proceedings of 8th IEEE International Conference on Contemporary Computing (IC3-2015)*, Jaypee Institute of Information Technology & University of Florida, August 20–22 (2015) [To appear]
- Lin, D.; Stamp, M.: Hunting for undetectable metamorphic viruses. *J. Comput. Virol.* **7**(3), 201–214 (2011)
- IDA Pro disassembler. <http://www.hex-rays.com/products/ida/>. Accessed 24 June 2015
- Nair, V.P.; Laxmi, V.; Gaur, M.S.; Kumar, G.V.S.S.P.; Chundawat, Y.S.: Static cfg analyzer for metamorphic malware code. In: Eli, A., Makarevich, O.B., Orgun, M.A., Chefranov, A.G., Pieprzyk, J., Bryukhomitsky, Y.A., rs, S.B. (eds) *SIN*, pp. 225–228. ACM (2009)
- Lewis, D.: Naive (bayes) at forty: the independence assumption in information retrieval. In: Ndellec, C., Rouveirol, C. (eds.) *Machine Learning: ECML-98. Lecture Notes in Computer Science*, vol. 1398, pp. 4–15. Springer, Berlin (1998)
- Panneerselvam, R.: *Research Methodology*. PHI Learning Pvt. Ltd., New Delhi (2004)
- Yu, L.; Liu, H.: Efficient feature selection via analysis of relevance and redundancy. *J. Mach. Learn. Res.* **5**, 1205–1224 (2004)
- Lyda, R.; Hamrock, J.: Using entropy analysis to find encrypted and packed malware. *IEEE Secur. Priv.* **5**(2), 40–45 (2007)
- Rodriguez, J.J.; Kuncheva, L.I.; Alonso, C.J.: Rotation forest: a new classifier ensemble method. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(10), 1619–1630 (2006)
- Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
- Weka, open source machine learning software. <http://www.cs.waikato.ac.nz/ml/weka/>. Accessed 12 August 2015
- Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, Ian H.: *The WEKA data mining software: an update*. *SIGKDD Explor.* **11**(1) (2009)
- Witten, I.H.; Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Burlington (2005)



34. Gotoh, Osamu: Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Comput. Appl. Biosci.* **9**(3), 361–370 (1993)
35. Smith, T.; Waterman, M.: Identification of common molecular subsequences. *J. Mol. Biol.* **147**(1), 195–197 (1981)
36. Stiglic, G.; Rodriguez, J.J.; Kokol, P.: Finding optimal classifiers for small feature sets in genomics and proteomics. *Neurocomputing* **73**(13), 2346–2352 (2010)
37. Dinaburg, A.; Royal, P.; Sharif, M.; Lee, W.: Ether malware analysis via hardware virtualization extensions In: *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS08)*, pp.51–62 (2008)
38. Zhao, Z.; Wang, J.; Bai, J.: Malware detection method based on the control-flow construct feature of software. *IET Inf. Secur.* **8**(1), 18–24 (2014)

