A Feature Selection and Evaluation Scheme for Computer Virus Detection

Olivier Henchiri
olivier@alumni.uottawa.ca
Nathalie Japkowicz
nat@site.uottawa.ca
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Abstract

Anti-virus systems traditionally use signatures to detect malicious executables, but signatures are over-fitted features that are of little use in machine learning. Other more heuristic methods seek to utilize more general features, with some degree of success. In this paper, we present a data mining approach that conducts an exhaustive feature search on a set of computer viruses and strives to obviate over-fitting. We also evaluate the predictive power of a classifier by taking into account dependence relationships that exist between viruses, and we show that our classifier yields high detection rates and can be expected to perform as well in real-world conditions.

1. Introduction

Traditional anti-virus systems are signatures-based, in that they use case-specific features extracted from viruses in order to detect those same instances in the future. While this method yields excellent detection rates for existing and previously encountered viruses, it lacks the capacity to efficiently detect new unseen instances or variants [9]. As new viruses appear virtually every day, a detection method that is responsive, rather than proactive, is bound to be broken.

Modern viruses employ advanced strategies such as polymorphism and metamorphism [7], through which parts of the virus or its very structure change in a sometimes random and unpredictable way each time it replicates. Virus writers, who have access to commercially available anti-virus software, can direct their efforts toward outwitting the scanners, by writing or modifying their code so that it passes undetected. This can be accomplished by strategically modifying the virus such that the characteristics that comprise the virus signature be changed.

Heuristic scanners attempt to compensate for this lacuna by using more general features from viral code, such as structural or behavioral patterns [3]. However, this process still requires human intervention and the resulting models fall short of yielding both good detection rates for new unseen viruses and low false positive rates.

In this paper, we are interested in applying machine learning methods to virus detection, and in particular to the problem of feature selection. We propose a method by which an exhaustive search is conducted on a dataset of viruses, yielding a large number of short generic features. Then we elect those that are most representative of viral properties. We also introduce a cross-validation scheme that tests our classifier using an evaluation method simulating real-world conditions of new virus outbreaks, and we offer evidence that our classifier has high predictive power.

2. Background

Current research applying data mining to virus detection strives to automate the search for features used in classification. This process has been tackled from two different angles: extracting optimal signatures from a dataset of viruses, and discovering more general features for use in a complex classification scheme.

Extracting virus signatures is not a new problem. Kephart et al. [4] developed a popular extraction method for virus signatures, by infecting a large number of files with a given virus and then harvesting for constant regions of 12 to 36 bytes. Then, from the considerable number of signatures collected, the ones with lowest predicted false positive rates were selected. While this method make it possible to extract signatures quickly and without the help of an expert, the authors concede that the algorithm fails for viruses that are moderately polymorphic.

Some detection methods utilize a variety of features, such as Win32 dll file calls, ASCII strings and byte sequences contained in the binary files. In an early heuristic approach [5], features such as duplicated UNIX system calls and files targeted by the program for writing purposes were used to detect malicious executables. In a machine learning method developed by Matthew Schultz et al. [6], ASCII strings and bytes sequences yielded good results. However, despite the byte sequences having a fixed length of 16, the feature space was very large, such that their dataset had to be split into partitions and different classifiers trained separately on each of them.

Proceedings of the Sixth International Conference on Data Mining (ICDM'06)
0-7695-2701-9/06 $20.00 © 2006 IEEE

Research in non signature-based heuristics has shown that sequences as short as 4 bytes can be used to detect unseen virus instances successfully [1]. However, as was found in [8], the list of candidate features extracted from a small dataset can contain tens of thousands of sequences.

Finally, many viruses are considered to belong to common virus families, based on the similarities in structure, code or method of infection that they share [2]. This classification is crucial to properly evaluating the effectiveness of a virus detection system. The first occurrence of a new kind of virus is typically the most devastating, as virus scanners are often incapable of detecting it. Then a host of variants typically emerge soon after the initial outbreak, albeit with less damaging consequences. Our method uses a priori knowledge of virus families, and evaluates the ability of our classifier to detect instances of a family without having been trained on any other instance from that same family.

3. Non-Specific Feature Search

We propose an exhaustive search for n-grams, short sequences of *n* bytes, where n-grams beginning at every byte of the files' machine code are recorded. The search process is comparable to a scanning window moving across the binary code and examining all sequences of a specified length. Our feature selection, shown in Figure 1, involves a selection step followed by an elimination step. In the fist step, we scan sequences, record their frequencies within each virus family, and construct lists of all the features that meet a given support threshold within each family. In the second step, we consolidate these lists and retain the features that meet a given support threshold among the feature lists.

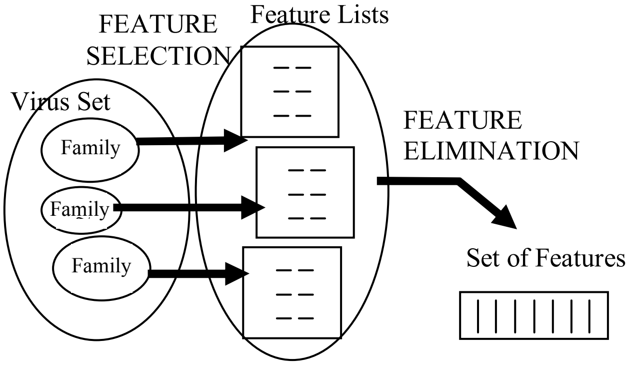

Figure 1: Hierarchical feature selection process.

The FEATURE_SEARCH algorithm, shown below, sets how general the features are, and how rigorous the feature elimination, by adjusting the following parameters:

Sequence Length: The sequence length can be specified. The shorter the length, the more likely the feature is to have general relevance in the dataset. But a short length will yield a larger number of features in each virus family. The length of the sequence is specified as a number of bytes.

Intra-Family Support: In the feature selection step, the intra-family support threshold of features within each virus family can be specified. Sequences occurring at or above a specified frequency are retained as candidate features. This puts a constraint on the number of sequences obtained for each virus family. Because there are a variable number of instances in each virus family, the intra-family support is specified as a percentage. In addition, a maximum number of features per virus family can be specified. If the feature selection yields an amount of features that exceeds this limit, then the intra-family support is increased by one instance, and the feature list is rebuilt.

Inter-Family Support: In the feature elimination step, the inter-family support threshold of features within the general corpus of viruses can be set to a desired value. This ensures that only those features that appear with a high enough inter-family support, as specified, are retained. This step discards unwanted features that occur frequently in one virus family but that are exclusive to that family. This parameter is specified as a number of virus families.

```
---------------------------------------------------------------------
FEATURE_SEARCH (S, len, intraSup, intraLim, interSup)
---------------------------------------------------------------------
Input: A non-empty set of viruses, S, classified in virus
       families.
Input: A non-zero sequence length, len.
Input: The intra-family support, intraSup, as a percentage.
Input: The intra-family limit, intraLim, as a number of
       features.
Input: The inter-family support, interSup, as a number of
       virus families.
Output: A set features, F, representing common
        characteristics of the viruses in S.
 1: for (each virus family $S_i$ in S) do
 2:     for (each virus $V_{ij}$ in family $S_i$) do
 3:             record all sequences of length len found in
                $V_{ij}$ (without repeats)
 4:     compute minimum incidence $intraMin_i = intraSup$ x
        number of viruses in $S_i$
 5:     build the list $L_i$ of $M_i$ sequences with support of at
        least $intraMin_i$ in $S_i$
 6:     if ($M_i > intraLim$)
 7:             increment $intraMin_i$ and go to 4:
 8: build the set of features F of sequences with support of
    at least interSup in S
 9: return F
---------------------------------------------------------------------
```

This method conducts the feature selection in a hierarchical fashion and is scalable to large datasets. Sequence scanning is done only once, and the first feature selection is conducted on small family subsets, while a second selection is conducted on shorter feature lists. The method also ensures that all retained features represent viral properties that are common to many types of viruses, as opposed to idiosyncrasies specific to one family.

4. Evaluation Method

The evaluation of heuristics is often biased when variants of known viruses or members of the same family are used in testing. To evaluate how well our classifier could classify new unseen viruses, we must use test cases from a family group that was neither used in feature extraction nor in training. For this purpose, we propose a cross-validation scheme that takes into account the correlation between viruses belonging to the same family.

Given a set of viruses categorized into *N* families, we partition the dataset into *k* sets of virus families, each consisting of *N / k* families. These sets of families are labeled *S1 , S2 , S3 , … , Sk*.

Prior to conducting the experiments, virus families and benign programs were selected at random and added to each family set. Experiments are then carried out on the dataset using a k-fold cross-validation scheme in the following way:

1. For each sets *Si* (*i* = 1 to *k*) scan all viruses and record to a separate file all byte sequences occurring with a frequency that meets a given threshold.

2. For (*j* = 1 to *k*) do:

a. Consolidate feature lists for set *Strain* = { *Si* | *i≠j* } keeping only those meeting a given inter-family support threshold

b. Train classifier on set *Strain* = { *Si* | *i≠j* }

c. Validate classifier on test set *Stest* = *Sj*

This cross-validation scheme simulates an environment where a virus detection system is faced with the outbreak of a new unseen type of virus. This test evaluates its performance in more stringent conditions, and therefore offers a better measure of its predictive power.

Figure 2 gives an overview of our system using a 5-fold cross-validation scheme. Each fold uses family-independent sets, on which feature selection can be performed as described in Section 3. Feature selection is repeated on the set of viruses in the training subsamples for each cross-validation fold.

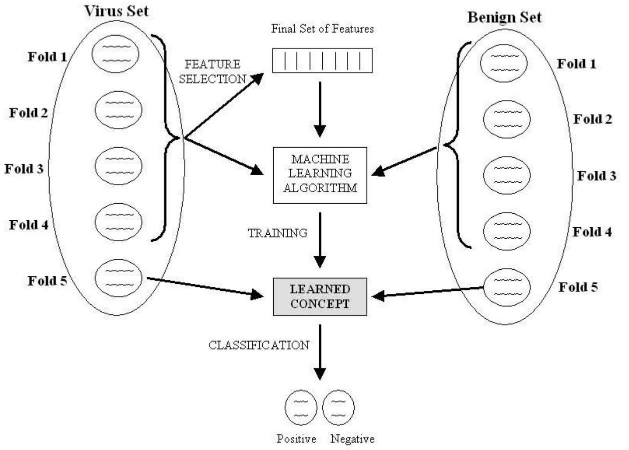

Figure 2: Feature selection, classifier training and evaluation using 5-fold cross-validation (using fold 5 as test set).

Critics of traditional testing methods argue that "only completely new worms cause global outbreaks" [10], and stress that standard random selection of test sets does not provide a reliable assessment of the proactive abilities of virus scanners. Our testing methodology evaluates the performance of our classifier in conditions similar to a real world environment. By separating all virus families, it provides a more accurate measure of its proactive abilities and, in particular, its capacity to detect completely new instances. In the next section, we report the results of a number of experiments intended to demonstrate the advantages of our feature selection model and the validity of our testing methodology.

5. Experimental Results

Our experiments were carried out on a dataset of 3000 examples consisting of 1512 previously labeled viruses and 1488 small benign executables, collected from desktop computers using various versions of the Windows operating system. The viruses were taken from the collection used in previous research [6] and were classified into 110 families, based on their filenames.

5.1. Comparison Against a Benchmark Model

We compared our method with a model used in previous research [6]. We examined each consecutive 16-byte sequence in every virus in the training set, and retained those appearing with a support of at least 1%. However, cross-validation was done in the same way as in our hierarchical selection scheme, where the classifier was evaluated on test sets of new unseen families.

We used 5-fold cross validation, where each fold consists of a viral set of 22 virus families and 297 benign executables. For each fold, a separate set of features was generated from the training set.

For our hierarchical model, we chose a sequence length of 8, which represents half that of the traditional method. We set the intra-family support threshold to 40% with an intra-family limit of 500, and the inter-family support threshold to 3 (out of 110 families). The minimum overall support of the features can be estimated to be the product of these two parameters. Table 1 shows that out final set of features is of comparable size and overall support as that of our benchmark test, thus providing a reliable comparison of the usefulness of each set of features for classifying new unseen viruses

Table 1. Number and support of features for the hierarchical and traditional models

Model	Average Number of Features	Minimum Expected Overall Support
Hierarchical	428	0.4 x (3 / 110) = 1.09%
Traditional	377	1%

Once the final feature set was obtained, we represented our positive and negative data in the feature space by using, for each feature, “1” or “0” to indicate whether or not the feature is present a given executable file.

Table 2. Experimental results using the hierarchical and traditional search methods.

Classifier	Overall Accuracy	False Positive Rate	Detection Accuracy
Hierarchical Feature Search			
ID3	93.29%	4.16%	90.56%
J48	93.65%	5.24%	92.56%
Naïve Bayes	69.51%	0.13%	37.17%
SMO	93.39%	5.71%	92.26%
Traditional Feature Search			
ID3	63.52%	14.02%	47.83%
J48	64.69%	13.18%	46.72%
Naïve Bayes	60.69%	0.16%	18.66%
SMO	65.04%	13.36%	47.52%

In our experiments, we used WEKA’s implementation of the ID3 and J48 decision trees, Naïve Bayes and the SMO algorithm, with the default settings. The results, displayed in Table 2, indicate that a virus classifier can be made more accurate by using features representative of general viral properties, as generated by our feature search method. With up to 93.65% overall accuracy, our system outperforms sub-50% rates of traditional detection methods and achieves better results than some of the leading research in the field [6], which only performs at 63.52% when tested under our more stringent evaluation method.

5.2. Optimal Feature Selection Criteria

In this section, we use a wrapper approach to search through a wide range of search parameters in an attempt to optimize our feature selection. We decrease the sequence length from 8 to 3 bytes, while limiting the number of candidate features per virus family to a maximum of 500. To achieve this, our initial intra-family support threshold of 40% is automatically incremented until a reasonably small number of features are generated, as we described in Section 3. We also investigate different inter-family thresholds, as they directly affect the generality and final number of features. We explore a range of threshold values from 3 to 6. We report the results of the ID3 classifier in Table 3. Experiments using the three other classifiers shown in Table 2 yielded consistent results.

Table 3. Virus detection accuracy (top) and false positive rates (bottom), using ID3 (values are %)

Sequence length Inter-family support	**8**	**7**	**6**	**5**	**4**	**3**
3	90.31 4.16	93.92 2.55	94.69 1.68	96.11 0.81	95.85 1.41	99.77 0.34
4	84.28 4.77	92.71 4.50	93.86 3.50	96.44 1.68	94.45 1.14	94.58 1.28
5	80.47 5.98	92.40 6.45	93.54 3.63	93.85 2.22	94.73 1.68	92.26 1.21
6	64.42 4.03	92.28 6.45	94.88 4.62	94.61 2.76	94.51 1.48	95.12 1.41

The results show that the classifier achieves better overall performance with shorter sequences. At length 5, performance reaches a peak, as the results at lengths 3 and 4 remain comparable. The inter-family support generally yields better results when low. This is especially true for longer sequences, where we observe a more rapid drop in accuracy as the support threshold decreases. This deterioration is likely due to the dwindling number of features. Table 4 shows that, at low support thresholds, the searches for sequences of lengths 6, 7 and 8 generate sets of less than 100 features, and as low as 16. In the case of shorter sequences, the performance remains very good, thanks to an abundance of features. However, we notice that performance is hindered when the classifier is working with a set smaller than 200 features. Our classifier performs generally better with a larger set of features, but some sets yield better accuracy than similar sets equal or bigger in size. The sets diagonally adjacent – down and to the right – to feature sets of lengths 6 to 8, are most often significantly smaller than the latter, and yet most often lead to a better performance.

Table 4. Number of features, averaged over the 5 cross-validation folds, in relation to different feature selection parameters

Sequence length Inter-family support	8	7	6	5	4	3
3	427	531	690.4	879	1150	1633.6
4	108	166	247.4	365.8	531.6	956.4
5	35.2	65.8	111	186.6	333.6	654
6	16.6	33	59.2	111.6	223.6	479

For short sequences of length 7 and 8, as the inter-family support and the number of features decrease, the classifier's performance drops to the lowest recorded levels, while longer features of lengths 3 to 6 generally maintain a performance in the mid-90's. This demonstrates that these feature sets, while decreasing in size, contain general enough features that are capable of covering the majority of training and test examples.

6. Conclusion and Future Work

We presented a feature search method that focuses on selecting generic features that are applicable to different families of viruses. This ensured that our classifier is genuinely heuristic and does not rely on signatures. In experiments testing our method against that of leading research, our method achieved better performance. In both models the features selected and used by the classifier had comparable overall support within the dataset. This indicates that our feature search method produced features that were more useful in detecting new unseen viruses.

We also introduced an evaluation method for virus classifiers that tests more convincingly its ability to detect new viruses. Our method does not allow classifiers to use examples in training that are variants of viruses present in the test set. This denies them an unfair advantage that they would not have in real world conditions. Our results show that our system, which uses family non-specific features, performs very well, while existing techniques for detecting previously unseen viruses perform significantly more poorly under our evaluation method.

In future work we propose focusing on reducing the false positive rate, by using a larger number of benign files, or by training our classifier using a cost matrix and setting a higher cost to misclassifying negative examples. We would also like to explore retrospective testing. Retrospective testing is a recent evaluation methodology for virus detection systems [11]. This would involve using a set of older viruses in the training set and a set of more recent ones in the test set.

7. Acknowledgments

Our thanks to Schultz et al. [6] for graciously sharing their virus dataset with us.

8. References

[1] Arnold, W. and Tesauro, G. Automatically Generated Win32 Heuristic Virus Detection. Proceedings of the 2000 International Virus Bulletin Conference, 2000.

[2] Bontchev, B. Analysis and Maintenance of a Clean Virus Library, in 3rd Int. Virus Bull. Conf, 1993.

[3] Gryaznov, D. Scanners of the Year 2000: Heuristics. Proceedings of the 5th International Virus Bulletin, 1999.

[4] Kephart, J.O. and Arnold, W. C. Automatic Extraction of Computer Virus Signatures. 4th Virus Bulletin International Conference, pp. 178-184, 1994

[5] Kerchen, P., Lo, R., Crossley, J., Elkinbard, G., and Olsson, R. Static Analysis Virus Detection Tools for Unix Systems. 13th National Computer Security Conference, 1990.

[6] Schultz, M. G., Eskin, E., Zadok, E., and Stolfo, S. J. Data Mining Methods for Detection of New Malicious Executables. IEEE Symposium on Security and Privacy 2001.

[7] Ször, P., and Ferrie, P. Hunting for Metamorphic. Virus Bulletin Conference September 2001, pp. 123-144.

[8] Tesauro, G., Kephart, J. O., and Sorkin, G. B. Neural Networks for Computer Virus Recognition. IEEE Expert, 11(4):5–6. IEEE Computer Society, August, 1996.

[9] White, S. R. Open Problems in Computer Virus Research. Virus Bulletin Conference, 1998.

[10] Muttik, I. Comparing the Comparatives, Virus Bulletin Conference, September 2001, pp. 45-56.

[11] Marx, A. Retrospective testing – how good heuristics really work, Proceedings of the 2002 Virus Bulletin Conference, New Orleans, LA, USA, Sept. 2002.