Malware Program Classification Using Machine Learning and Principal Component Analysis

Nathan Kueterman
University of Dayton
Dayton, OH
nkueterman1@udayton.edu

***Abstract*—The field of cyber security involves processing and analyzing massive amounts of data for potential malware. The malicious programs continue to evolve and grow, which means the counter efforts need to be robust and efficient. In this project, the byte files from thousands of different malware programs were converted to images and then classified using machine learning algorithms. To remove unwanted features in the dataset, Principal Component Analysis is utilized leaving the statistically relevant features for classification. The performance of Artificial Neural Network (ANN) algorithms along with K-Nearest Neighbors (*k*NN) and Support Vector Machine (SVM) are compared in terms of malware classification accuracy. K-fold validation is employed to ensure that each approach is robust and effective. The study uses the Microsoft Malware Classification Challenge (BIG 2015), which is publicly available on the Kaggle database.**

Keywords—Malware Detection, Principle Component Analysis, Support Vector Machine, K-Nearest Neighbors, Neural Network

I. Introduction

The classification of malware data is an essential problem at the forefront of cyber security. Having the capability to find and identify malicious software is increasingly becoming more important as the internet and device interconnectivity rapidly expand in today's society.

The approaches used in this study are scaled back in terms of data usage. The BIG 2015 dataset includes both an assembler source (.asm) file as well as the binary (.bytes) file but only the binary files were used in the classification process. Leveraging the assembly code is much more demanding as it must be decrypted and analyzed via program outputs. Doing this to thousands of files would take immense processing times. If these dynamic code approaches are employed in real-time the malware might have enough time to deploy some or all of the malicious content.

In this study, the approaches are identical to that in [1]. Essentially this is a replication of that study except only a back propagation, single hidden layer neural network was tested instead of the other ANNs that were used in [1].

II. Approach

A. Binary File Processing

As mentioned in the introduction, only the byte files were utilized for classification. The general format of the binary files were the same besides the variations of dimensionality. Each row of the file began with an 8 digit, decimal number and then was followed by 16 pairs of hexadecimal numbers separated by spaces. This only changed on the last line of the file where the number of hexadecimal pairs varied. There was one special case that was common among files where the hexadecimal pair was '??', which was mitigated by replacing it with -1. This was done in order to add priors to the PCA dimensionality reduction which decreases the variance and statistical relevance of this special case.

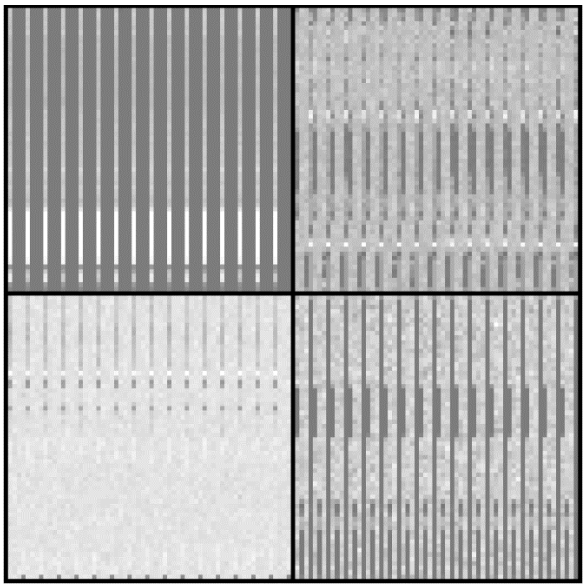

Fig. 1. 4 random samples of binary files converted into 256x16 images.

The procedure for generating the images was to remove the decimal row headers, convert each hexadecimal pair into a decimal number, concatenate them into an array, scale them from 0 to 1, replace the special cases with -1, and reshape the array using bicubic interpolation to create a grayscale image. Four samples of the malware byte images, which were scaled to 256x16, can be seen in Figure 1.

B. PCA Transformation

Once the binary files are converted to images, PCA transformation is applied in order to reduce data dimensionality. PCA transformation maps an N-dimensional vector into an M-dimensional ($N > M$) one without redundancy.

C. kNN Classifier

The k-nearest neighbor, or kNN, classifier utilizes distance in the feature space to determine the class of each sample. This

method has very few parameters but they consist of the weights of the distances, number of neighbors, and distance method. The default parameters used in this experiment for PCA performance analysis were squared inverse, 5 neighbors, and Euclidean distance.

D. SVM

SVM draws decision boundaries in the feature space using optimal hyperplanes. This hyperplane is generated using the training data and is used to classify testing data. There are more parameters and methods to calculate the hyperplane as well as decision metrics compared to the kNN method. The hyperplane was a polynomial of order 2, as well as one k-fold verification with order 1. The other kernel functions, or hyperplane methods, are Gaussian and linear which are better suited for one/two class learning.

E. ANN

The artificial neural network approach focused purely on a single layer network with a logistic, mean-squared error loss function. The number of hidden layers, iterations, and learning rate were varied for performance analysis. The initial weights were initialized randomly in the range [-0.12, 0.12] on each run of the k-fold validation. The class with the highest output from the sigmoid activation function is assigned to the respective class. All pixels were considered during the training function so varying number of PCA did not affect ANN performance.

III. Results

A. PCA Transformation

Experimentation with different amounts of features using PCA reveals a statistical hierarchy of the data. This information can be essential in optimizing classification in terms of speed. In this study the amount of features used varied from 10 to 25 in intervals of 5. After approximately 10 features is where the kNN classifier plateaued in performance. Approximately 25 features is where the SVM classifier started to plateau. The classifier accuracies can be seen in Figure 2.

	10	15	20	25
kNN	96.50%	96.40%	96.40%	96.30%
SVM	94.40%	94.90%	95.10%	95.30%

Fig. 2. Number of PCA features used versus overall 10-fold validation accuracy for each classifier method.

B. kNN Classifier

The kNN classifier had the highest overall 10-fold validation accuracy based on the different parameters and methods that are comparatively included in this study. The parameters that gave the highest performance were using squared inverse, 5 neighbors, Euclidean distance, and 15 PCA features. The confusion matrix output from this model can be seen in Figure 3.

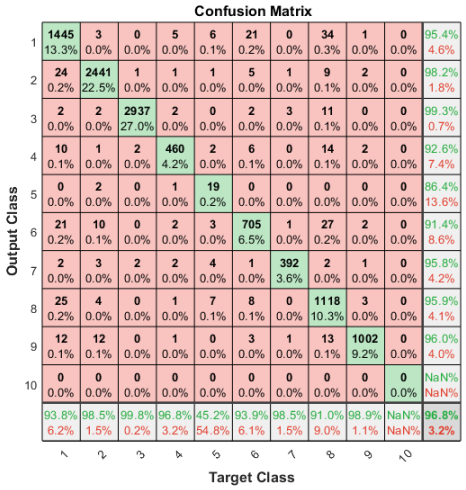

Fig. 3. Confusion matrix for highest performing kNN malware classifier.

Some parameter experimentation was performed to determine what methods were optimal for classification in this dataset. Number of neighbors was the largest impact and therfefore was closely studied. The highest accuracy resulted from the lowest neighbors, but not below 5. The results of this study can be seen in Figure 4.

C. SVM

The SVM algorithm did not seem too affected from changing the polynomial order. The performance went down approximately 12% when reducing the order from 2 to 1 and did not have any substantial increase when changing the order to 3. However, the number of PCA features had a large impact on performance which is prevalent in Figure 2. The highest performing SVM model, order 2 polynomial with 25 PCA features is presented in Figure 4.

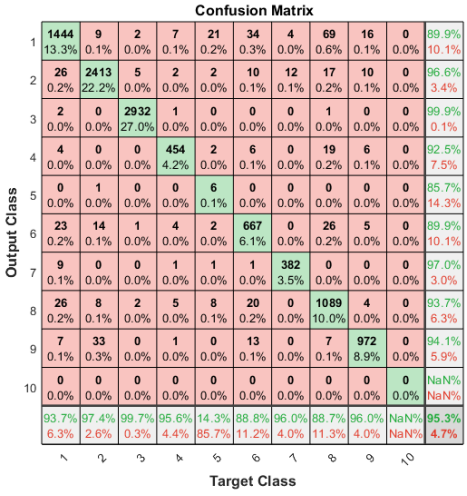

Fig. 4. Confusion matrix for the highest performing SVM malware classifier.

D. ANN

The ANN approach produced the lowest accuracy across the board but this is most likely because of the shallowness of the network. One aspect that is very important for ensuring proper training for the ANN was to normalize the training data. The accuracy of the same network went from 51.3% to 91.2% just from normalizing the data. Learning rate was varied from 0.05 to 0.2 and 0.1 was consistently performing the highest. Also, increasing iteration up to approximately 10000 increased accuracy but plateaued after that and was not worth the extra computing times. The model parameters that provided the highest accuracy was using normalized features, 25 hidden layers, 10000 iterations, and a learning rate of 0.1 which produced an accuracy of 91.2% which can be seen in Figure 5 below.

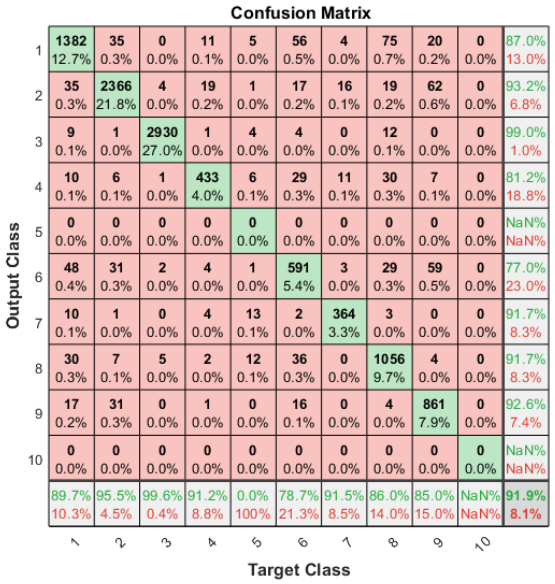

Fig. 5. Confusion matrix for the highest performing ANN malware classifier.

IV. Conclusion

Overall it seemed that most simple solution, kNN, was the most robust in this situation. There are many other algorithms out there that could outperform it in terms of accuracy but it is among the fastest of classifiers which is an aspect to consider in this realm. The one glaring issue in this study is the very low classification accuracy on class 5. This class only had 42 training samples so it was heavily imbalanced. Some interesting future studies or improvements could include trying different types of interpolation on the images, using different neural networks especially deeper ones, experimenting with class imbalance and possibly augmenting more of the minority class.

References

[1] B. N. Narayanan, O. Djaneye-Boundjou, and T. M. Kebede, "Performance analysis of machine learning and pattern recognition algorithms for Malware classification," *2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*, 2016.