# Analysis of Prioritizing Malware

Zhen Tang[1], Jeff Schneider[2]

[1] Department of Physics, Carnegie Mellon University, zhent@andrew.cmu.edu
[2] The Robotics Institute, Carnegie Mellon University, Jeff.Schneider@cs.cmu.edu

**Abstract.** Millions of malware samples are reported everyday, but cannot be carefully reviewed one by one due to the time limitation of human analysts. A ranking list to recommend the most important samples for our human analysts is thus desirable. In this thesis, we propose two different criteria to evaluate the importance of each sample. For each criterion, several popular algorithms are applied to choose the best method by comparing results using hold-out data. In the end, we combine the scores from the two criteria to generate the final ranking list.

## 1 Introduction

Malicious software (malware) has been a serious threat on the Internet for many years. They can infect personal computers and attack web servers by detecting other software's vulnerabilities. And severe consequences can be caused, such as the leaking of personal bank information or the huge amount of registered user materials from public web. For example, the recent attack to Target by a specially designed malware has affected about 40 million customers. About $\$3.2$ billion dollars are lost per year due to malware for consumers in the US.

It is urgent to provide early warnings to take proactive measures against these highly malicious malware. However, we have to face a big challenge, which is that thousands of new malware samples emerge everyday. It is impossible to check all of them carefully due to the limited availability of human analysts.

One important reason that there are so many malware samples is the variants. People who make these malware samples can easily produce many variants by slightly modifying the original malware's code. We can make use of this characteristic by selecting only one representative from samples who have similar behaviors for detailed study. By doing this we can reduce unnecessary efforts and achieve better performance at the same time.

To achieve such a goal, criteria need to be proposed to generate a ranking list, where the most malicious malware sample that has not been analyzed before will be put on the top.

The schematic process is shown in $Fig.\ 1$. The dashed line on the left side of the plot shows the traditional way of analyzing malware, where all the reported samples will be directly handed to human analysts, which is very ineffective. Instead, we propose to analyze these samples as shown on the right side of the plot. The reported samples will be automatically analyzed in a closed system. Features will be extracted by executing these samples. We will get a group of numbers which represent the activities of the sample on corresponding features. Then a ranking list can be generated after applying the methods we will introduce in this paper, which will be sent to human analysts. Instead of analyzing all the samples, human analysts now only need to pick up the first several samples in the ranking list to do futher examination.

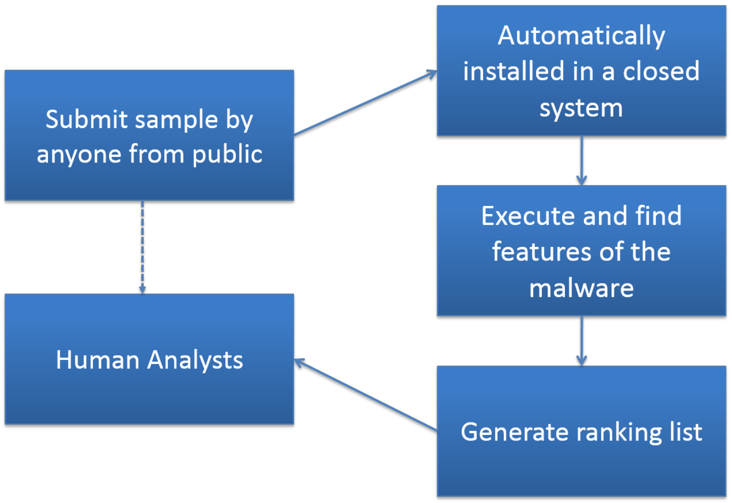
Submit sample by anyone from public
Automatically installed in a closed system
Execute and find features of the malware
Generate ranking list
Human Analysts

**Fig. 1.** The process of generating malware ranking lists for human analysts

In the following sections of this report, we will firstly introduce the data and feature selection methods before we use them (section 2). Two criteria will be proposed to generate the ranking lists for human analysts (section 3). We will also review some of the related work and compare their methods to ours (section 4). Future work and conclusions will be shown in the last two sections.

## 2 Data description

We received the dataset for this report from Jose Morales at SEI.

Our dataset is composed of two parts, the labeled samples and the unlabeled samples. For the labeled samples, we use positive labels to describe the malware samples, and negative labels to describe those safe ones.

There are 1041 samples in the labeled set, where 356 of them are positive, and 658 of them are negative. We also have 13061 samples in the unlabeled set, from which we will generate ranking lists for futher review by human analysts.

All the samples have 38 features recommended by domain experts, which describe their activities in the isolated system when being executed. The list of features includes openFiles, findFiles, createFiles, getFileAttrs, deleteSelf, moveFiles, moveSelf, openCryptoDll, copyFiles, createdRegKeys, openRegKeys, setRegVal, delRegVal, enumRegVal, delValsFromMCVR, disableAVReg, startSelfAtReboot, setValInCreatedKey, deleteInternetSettings, createAVRegKeys, createProc, killProc, openProc, startProcFromFile, openThreadWriteProc, openSvc, createSvc, sleeps, enumMods, kDbgChck, createThreads, connAttempts, maxConnAttempts, uniqueIPs, createObjs, DNSRequests, createHiddenWindow, verifiedSig.

The dataset we have is a matrix. Each row represents one sample and each column represents one feature.

## 2.1 Data preprocessing

The 38 features are chosen by domain experts, which are believed to be significantly important in determining if the samples are malware or not. However, for some of the features, we do not find any activities from both positive samples and negative ones. Hence, these features will not be useful in our studies and we will simply remove the corresponding columns in the dataset. These features are getFileAttrs, openRegKeys, enumRegVal, disableAVReg, startSelfAtReboot, openThreadWriteProc.

## 2.2 Data normalization

We perform data normalization before we apply any training algorithms on them. Otherwise the different scales of data in the feature space will bias the training results.

We calculate the mean and standard deviation for each feature column, and then the data normalization can be done as follows,

$$x_i = (x_i - mean_k)/std_k, \quad i = 1, 2, ...N, \quad k = 1, 2, ...K \tag{1}$$

where N is the size of data points, and K is the total number of features.

Now each feature column has mean at zero and standard deviation at one.

## 2.3 Feature selection

After initial data preprocessing, we still have 32 features left for each sample, which is still too many considering the very limited number of labeled samples we have. To avoid "curse of dimensionality" for methods we apply later, feature selection is necessary.

Lasso is a widely used method in both statistical and machine learning fields for feature selection. Logistic regression with lasso is used here to do the feature selection, and the algorithm is described as follows, 1, The labeled samples are randomly split into 10 folds with approximately equal size.

2, For the $i^{th}$ loop (i = 1,2,...10) we use the $i^{th}$ fold as the test set, and the other nine folds as the training set. We then apply logistic regression with lasso to do the feature selection and record the selected features.

3, Features which are selected at least three times are kept.

$Fig.$ 2 describes the number of times being selected for each feature, and 14 features are selected after feature selection, which are,

deleteSelf, moveSelf, copyFiles, setValInCreatedKey, deleteInternetSettings, createProc, killProc, createSvc, enumMods, kDbgChck, createThreads, maxConnAttempts, createHiddenWindow, verifiedSig.

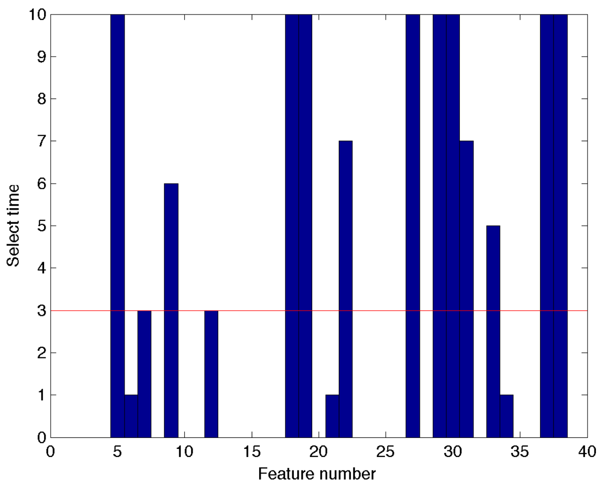

**Fig. 2.** Feature selection by logistic regression with lasso

In the following section, we will use "original data" to describe the data after data preprocessing but before feature selection, where each sample has 32 features. We use "data after feature selection" to describe the data after feature selection by logistic regression with lasso, which only have 14 features for each sample.

## 3 Selection criteria

We propose two criteria to rank the unlabeled samples. Both criteria can be used to assign scores between 0 and 1 to the unlabeled samples. We will try different combinations of scores from the two criteria to generate the final ranking list for human analysts.

**Criterion 1**: How strongly the events are classified as positives.

It is natural for us to assign higher scores to samples who have higher probabilities of being classified as positive samples. The classification methods can provide us probabilities of samples to be classified as positives. Since the probabilities are already within [0, 1], we can directly assign these probabilities to the samples as the scores from criterion 1.

**Criterion 2**: How anomalous the events are relative to the labeled samples.

As we have described before, because of the limited number of human analysts, we cannot examine all the submitted samples. Those samples whose variants have not been analyzed before should be given higher priority according to criterion 2. Similarities between the unlabeled samples and labeled samples will be calculated first. If an unlabeled sample is very close to some labeled samples, we can conclude that a variant of this sample has already been analyzed, and this unlabeled sample's characteristics can be obtained by reviewing its analyzed variant, instead of handing it to human analysts. Therefore, we will assign such samples lower priorities while the samples with smaller similarities with labeled samples are put on top of the ranking list.

## 3.1 Criterion 1

According to the definition of Criterion 1, we need to assign the probabiliites of being classified as positive samples for all the unlabeled samples. Several classification methods which are popular in both academia and industry will be applied and compared in this section, such as logistic regression and support vector machine. The winner of the comparisons will be chosen as the classification method for criterion 1.

One common metric for the comparison is classification accuracy. For each method, we will apply the same steps to calculate the accuracies as follows,

1, Split the labeled samples into 10 folds.

2, For the $i^{th}(i = 1, 2, ...10)$ loop, choose the $i^{th}$ fold as the testing set, and the remaining nine folds as the training set. Use different methods to do classification.

3, Record the number of mis-classified events for each iteration as $mc_i, i = 1, 2, ...10$, and assume the number of all labeled events is $tc$, then the accuracy of this classification method is $\sum_i mc_i/tc.$

However, we would like to know more information from the classification besides the accuracy according to our objective. For example, the mis-classified events for positive and negative samples may result in different affection. Therefore, we cannot compare these classification methods only by the accuracies. More details need to be reviewed.

Several corresponding concepts will be used in the comparisons. They are,

**True positive**: Positive event which is classified as positive.

**False positive**: Negative event which is classified as positive.

**False negative**: Positive event who is classified as negative.

**True negative**: Negative event who is classified as negative.

False negative is the counterpart of true positive and true negative is the counterpart of false positive. We can derive all of the above numbers if we know true positive and false positive, as well as the accuracy if we know the total number of samples.

Receiver operating characteristic (ROC) curve can be used to provide this information. The ROC curve describes the relationship between the true positive rate and the false positive rate at various discrimination thresholds for one classification method. The accuracy of the classification method only shows one cutpoint in the ROC curve. A classifier with higher accuracy does not necessarily mean better performance. Since the high accuracy may come because of a biased threshold we choose. With a different threshold, the result may be twisted. Hence a more comprehensive standard to compare these classification methods needs to be provided. We decide to use the area under the curve of ROC (AUC) as the new standard, which sees increasing usages in machine learning field. Methods with largest AUC will be considered as the winner. To obtain the ROC curve, we only need to modify the previous algorithm slightly as follows,

1, Split the labeled samples into 10 folds.

2, In the $i^{th}$ loop, we choose the $i^{th}$ fold as the testing set, and the remaining nine folds will be used as the training set. In the training, we use different threshold. In the previous algorithm, when the probability of one sample to be classified as positive is greater than 0.5, we classify this sample as a positive sample. However, to make the ROC curve, we choose a group of threshold values from 0 to 1. With each threshold we choose, number of misclassified negative events will be recorded as false positives $fp_i$, and the number of positive events who are classified correctly will be recorded as true positive as $tp_i$.

3, Calculate the true positive rate (tpr) and false positive rate (fpr) for each threshold for 10 runs. Use mtpr to represent the mean, and stpr to represent the standard error of results from 10 runs. Draw ROC curve with the points (fpr, mtpr), upper limit with the points (fpr, mtpr+stpr) and lower limit with the points (ftpr, mtpr-stpr). We also provide the semilog version of ROC curves, where we use natural log of fpr as x coordinate.

The best method we select to do classification after comparisons is SVM with radial kenel. And we use the labeled samples as training set to calculate scores for unlabeled samples, which are shown in $Fig. 3$.

In the remaining part of this section, we will show the details about these comparisons to choose the best classification method for criterion 1.

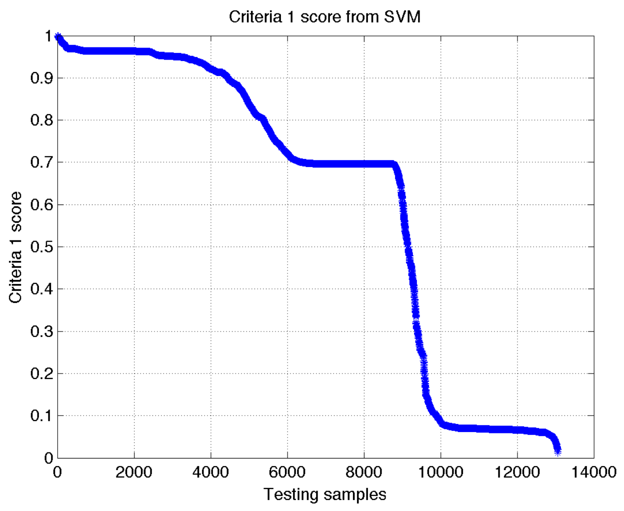

**Fig. 3.** Criterion $1$ score from SVM with radial kernel

### 3.1.1 Logistic regression

Logistic regression is widely used for classification because it is relatively fast compared to other complex methods as well as its reasonable accuracy. We will always use the data after feature selection for the logistic regression method.

we have tried two regularization terms, ridge and lasso. The corresponding ROC curves are shown in $Fig.\ 4$. No obvious difference can be found according to the plots. However, the ridge achieves its maximum accuracy at $89.45\%$ when the threshold is 0.64, while lasso achieves $90.63\%$ at the normal threshold. Therefore we choose logistic regression with lasso for the next round of comparison.

Logistic regression is a relatively simple method and hence tt is naturally for us to apply adaboost algorithm on it. And the result is shown in $Fig.\ 5$. From the plots, we can conclude that adaboost algorithm does not bring us obvious benefits. One of the reasons could be that the logistic regression with lasso is too complex to be boosted. Considering the large increase in time complexity after applying the adaboost, we will stick to the non-boost version of logistic regression.

After the two comparisons, we decide to use the logistic regression with lasso as the representative for logistic regression.

### 3.1.2 Support vector machine (SVM)

SVM is a method with outstanding classification capability[3]. By using kernel tricks on SVM, we can obtain nonlinear decision boundary, which greatly increases the flexibility of this method. In this section, we will try SVM with the radial kernel on the full data and data after feature

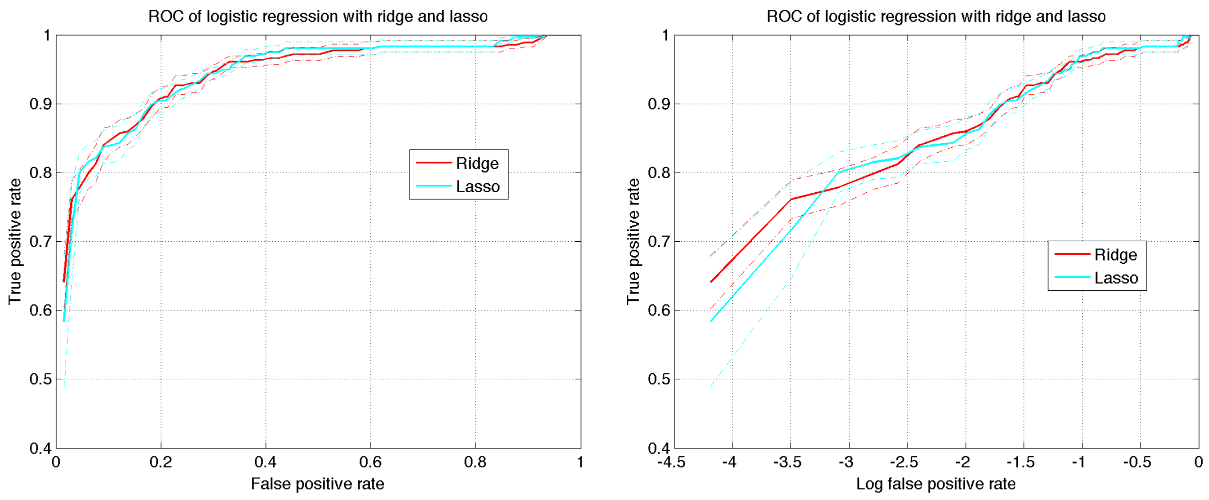

**Fig. 4.** ROC curves of logistic regression with ridge and lasso

selection. The results are shown in $Fig.\ 6$, from which we can say that although the difference is not statistically significant, we do not see the benefit from feature selection and we will stick to use full data for SVM from now on.

### 3.1.3 Comparison of two methods

Now we will compare the classification results from logistic regression with lasso and SVM to see which one is better. The result is shown in $Fig.\ 7$. A student's t-test has been done for the two ROC curves, and $15\%$ of the points have the p-value less than 0.05. We will choose SVM as the method for criterion $1$ since the mean value from SVM is larger than that of logistic regression as shown in $Fig.\ 7$.

### 3.1.4 SVM kernel selection

In SVM, several kernel functions are available. Radial kernel is the best choice in many situations, which is the reason why we choose radial kernel in the previous section. In this section, we run SVM with different kernel functions to see if our assumption is right. And the result shown in $Fig.\ 8$ verifies this assumption.

## 3.2 Criterion 2

The criterion 2 is used to do the anomaly detection. Scores will be assigned to the unlabeled samples. Samples with higher scores means they are more likely to be anomalies. Several methods will be introduced and a heuristic comparison algorithm will be applied to select the best method

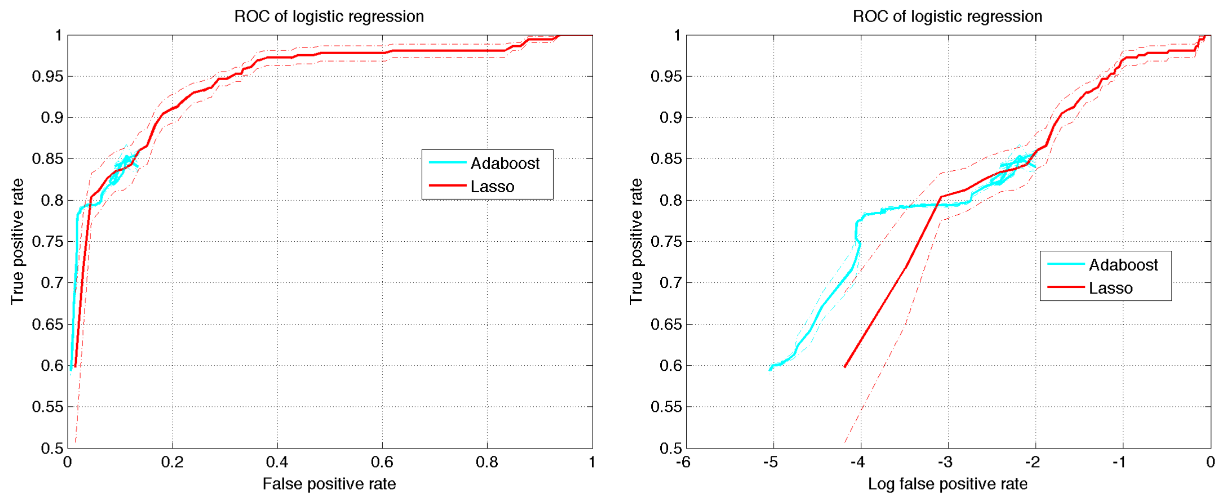

**Fig. 5.** ROC curves of logistic regression with and without adaboost algorithm

for criterion 2. The labeled samples will be used as training set and the unlabeled set will be used as the testing set. All the data we use in this section are data after feature selection.

One problem for criterion 2 is we are lacking of effective testing method. The only way we can rely on is obtaining feedbacks directly from human analysts. But until now we have not received any feedbacks and the heuristic testing algorithm we use to compare the following methods has not been verified to be effective, which actully de-emphasize the importance of criterion 2.

### 3.2.1 T statistics method

In the section, we will describe a simple statistical method using t scores. Since we have mean at zero and standard deviation at one for each feature column after data normalization, the evaluation expression can be shown as follows,

$$\frac{(x - mean)^2}{std^2} = x^2 \tag{2}$$

The larger this value is, the more anomalous this sample will be. We can then assign scores by calculating the summation of $Equ.$ 2 over all features and then normalize it to [0, 1]. This method actually calculates the Euclidean distance between each testing sample and the center of the training samples.

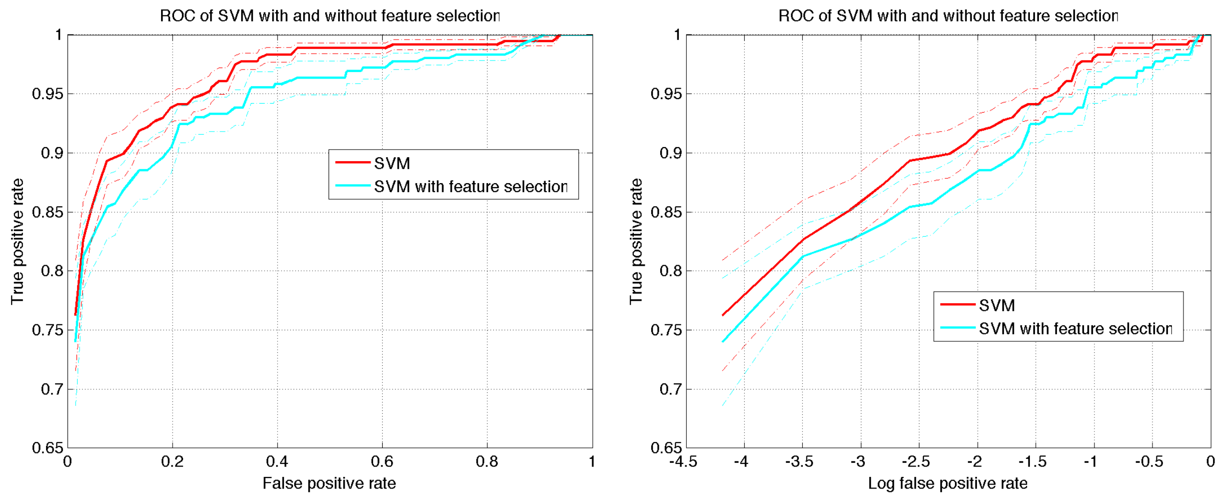

**Fig. 6.** ROC curves of SVM with radial kernel apply on the data with and without feature selection

### 3.2.2 Nearest neighbor

Nearest neighbor has been widely used as a nonparametric classification method. The unlabeled sample will be assigned a label based on ite neighbors' labels in the training set. However, in this section, we will not use nearest neighbor to do classification. Instead, we will use the intermediate result, the distances between query sample and its nearest neighbors to do anomaly detection. Scores will be computed according to the distances between samples in the unlabeled set and their nearest neighbors in the labeled set.

The nearest neighbor algorithm is shown as follows,

1, For each sample in the testing set, calculate the distance between this sample and its $1^{st}$, $2^{nd}$ and $3^{rd}$ nearest neighbors in the training set according to Euclidean distance. And take the mean of the three distances.

2, Normalize the calculated mean distance into [0, 1], and assign it to the corresponding sample as its score from criterion 2.

However, this straightforward algorithm has some drawbacks. For example, if the first two samples in the ranking list are very similar, and after the first sample examined, we do not want to waste human analysts' time in checking the second one. An updated version of nearest neighbor is hence proposed as follows,

1, Generate the ranking list according to normal nearest neighbor (without normalizing the scores).

2, Transfer the first sample in the ranking list from the testing set to the training set.

3, Repeat step 1 and step 2, until the testing set is empty. And normalize the distances to [0, 1].

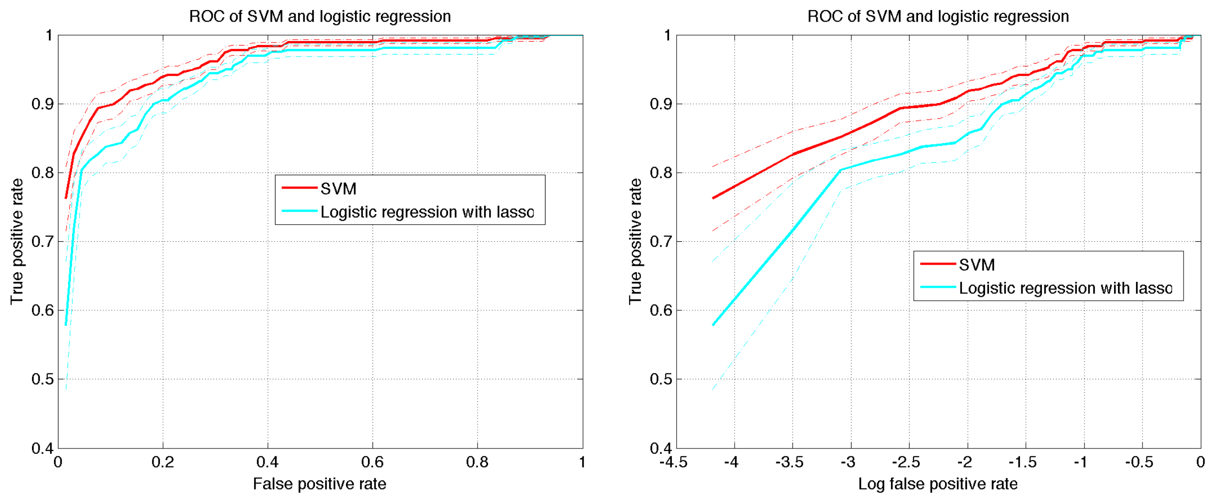

**Fig. 7.** ROC curves of SVM and logistic regression with lasso

For the update version of nearest neighbor, we can make use of the previous step's result to avoid recalculating distances between samples in the testing set and samples in the previous step's training set, which makes this algorithm time efficient and feasible under our limited computing resources.

### 3.2.3 Kernel density estimation (KDE)

KDE is another frequently used method to estimate probability density with the help of kernel functions. Gaussian kernel function is selected for our case. To simplify the training, we set up a strong assumption that all the band widths in different features are the same. Then we can use one-dimensional likelihood to search for the best band width. After training, we can calculate the probability density for each sample in the testing set and then generate the ranking list by the normalization of these probability densities. The small density means very few training samples are around this position, which tells us that this sample is likely to be anomalous.

The KDE method can suffer from "curse of dimensionality". We only have about 1000 training samples, while each sample has 30 features before feature selection. Therefore only data after feature selection will be used in this method, which has 14 features for each sample.

Meanwhile, an updated version of KDE is considered. However, not like the nearest neighbor case, we need to re-train everything after we change the training set for KDE, which means we need to do about 13000 times KDE training. This is too time-consuming to be done for our DAP project due to the limited computing resources we have.

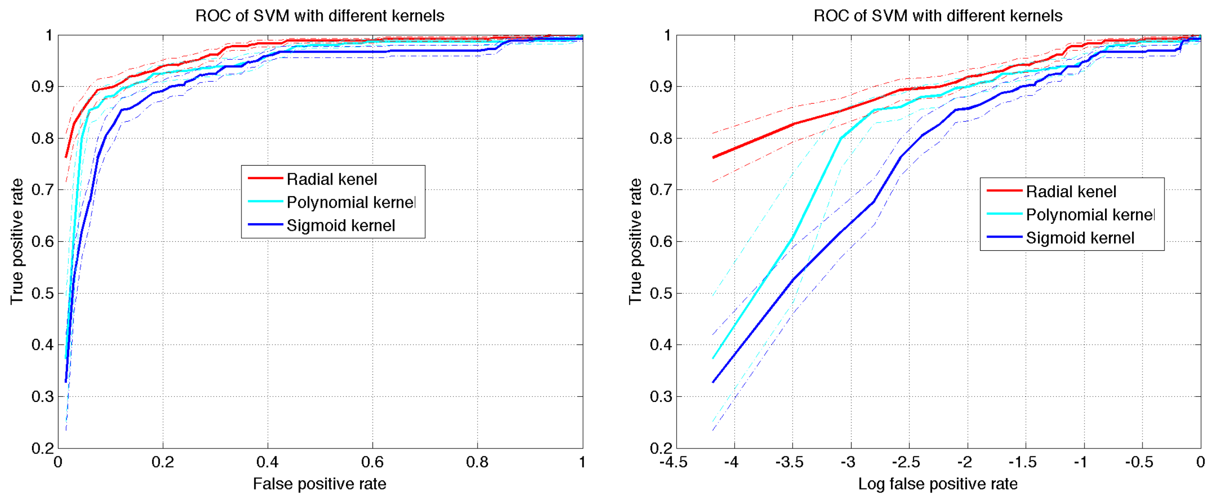

**Fig. 8.** ROC curves of SVM with radial kernel, polynomial kernel and sigmoid kernel

### 3.2.4 Selection of the three methods

The best way to evaluate these methods for criterion 2 is to generate ranking lists based on all the methods we just introduced, and then get feedbacks from human analysts to judge which method is the most appropriate. However, such process could take a very long time and is not feasible at this time. We here propose an alternative algorithm as follows, where only labeled samples will be used.

1, Split the negative samples randomly into two sets with approximately equal sizes. Use the first set as the training set. And combine the second set of negative samples and all the positive samples into the testing set.

2, Do the training according to the above methods we discussed in this section, and get ranking lists for the testing set.

3, In the generated ranking lists, we treat negative samples as relevant events, and positive samples as irrelevant events since all the training set are composed of negative samples. And use mean average precision (MAP) at size 300 to do the comparison. The method with highest MAP will be the selected method for criterion 2.

However, this evaluation could have some limitations. One assumption is made here that the positive samples and the negative samples are separated with each other, which cannot be verified directly.

The result of the comparison is listed in $Table$ 1, from which we can conclude that KDE will be the best method we choose for criterion 2. And the scores from criterion 2 is shown in $Fig.$ 9.

Some interesting phenomena can be observed from $Table$ 1. The reason that nearest neighbor with update gets such a bad MAP score is, transferring samples from the testing set to the training

**Table 1.** Method selection for criterion 2

| Method | MAP | Standard Deviation |
|---|---|---|
| Statistical Method | 0.9375 | 0.0130 |
| Nearest Neighbor | 0.8041 | 0.1505 |
| NN with update | 0.5647 | 0.1091 |
| KDE | 0.9377 | 0.0131 |

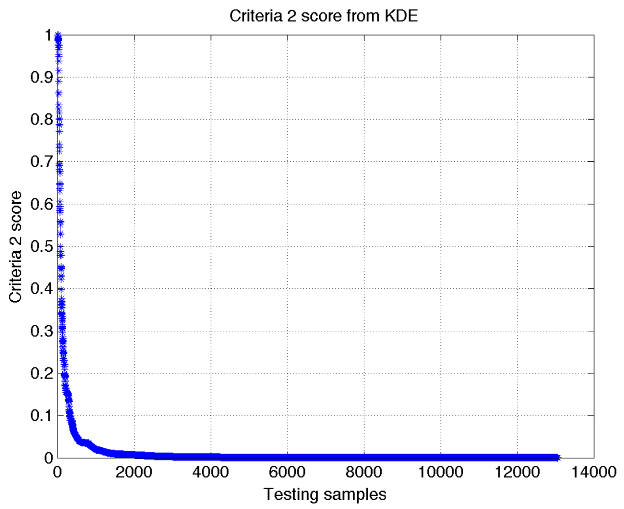

**Fig. 9.** Criterion 2 score from nearest neighbor with update

set will affect the purity of the negativity in the training set. But we do not consider this when setting up the evaluation system. The significant difference between the normal nearest neighbor and its updated version indirectly proves the effectiveness and necessity of the update. And it also confirms the assumption that the distances between negative samples should be smaller than the distances between negative and positive samples in general.

## 3.3 Combination of two criteria

We can generate two group of scores by SVM from criterion 1, and nearest neighbor with update from criterion 2. Four ways can be used to generate the final ranking.

1, Highest ranking according to criterion 1.

2, Highest ranking according to criterion 2.

3, Highest ranking by the summation of scores from criterion 1 and criterion 2.

4, Highest ranking by the product of scores from criterion 1 and criterion 2.

We have already shown the first two ways in $Fig.$ 3 and $Fig.$ 9. The third way is shown is $Fig.$ 10 and the fourth way is shown in $Fig.$ 11.

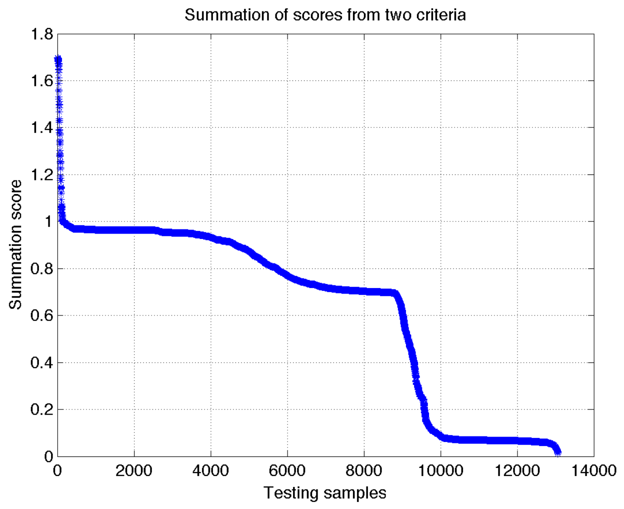

**Fig. 10.** Summation of scores from two criteria

An algorithm is implented as follows to compare the two combination ways, summation and product. All the data we use are from the labeled samples.

1, Split both the positive and negative samples randomly into two sets with approximately equal size, represented by pos1, pos2, neg1 and neg2. Use pos2+neg2 as the testing set.

2, Use pos1+neg1 as the training set, and apply SVM from criterion 1 to assign scores for the testing set.

3, Use neg1 as the training set, and apply KDE from criterion 2 to assign scores for the testing set.

4, Combine the scores from step 2 and step 3 by summation and product. Draw the ROC plots for the two combinations.

The result of this comparison is shown in $Fig.$ 12. We can conclude that the summation is slightly better than the product. But the best way to decide is still through feedbacks from human analysts.

## 4 Related work

The activities of samples used in our paper are from dynamic analysis results. However, the huge amount of malware and their variants can make dynamic analysis resource intensive. Neugschwandtner et al. [5] introduces the FORECAST system to effectively shrink the size of samples in the dynamic analysis process, which can be used as the pre-step for our research.

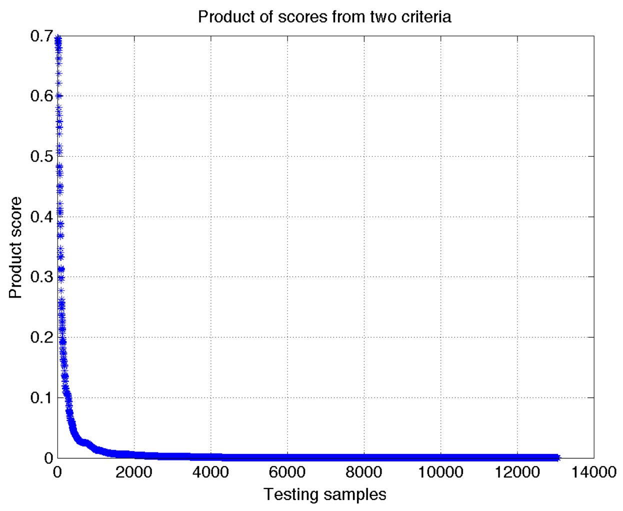

**Fig. 11.** Product of scores from two criteria

Several other works have been done to improve the accuracy when assigning the labels to the malware. An effective approach called automated malware classification has been proposed by Bailey et al. [1].

Meanwhile, many related papers have focused on how to cluster malware samples according to their characteristics. Park et al. [6] uses the directed graphs generated from the system call traces to measure the similarities of malware samples. Kinable et al. [4] adapts an approximate algorithm to cluster the malware sample by the structures of their call graph representations. And Bayer et al. [2] provides a fast algorithm to find the similarities of malware samples with respect to the subset of their characteristics such as network activities and system calls for large datasets.

However, none of the above papers have tried to combine the classification and anomaly detection together to generate a ranking list, which is very reasonable and necessary since both criteria can improve the ranking quality.

## 5 Future work

There are several studies we want to do in the future.

First, we have simplified the KDE method in criterion 2 by making band widths for all features the same. We still have 14 features after feature selection. If we want to treat the band widths of all the features differently, say 5 trials for each feature, we will have to train the KDE model for $5^{14} = 30517578125$ times, which exceeds our limited computing power. In the future, we hope that some of the features can be clustered together. And then we can have different band widths for different clusters of features.

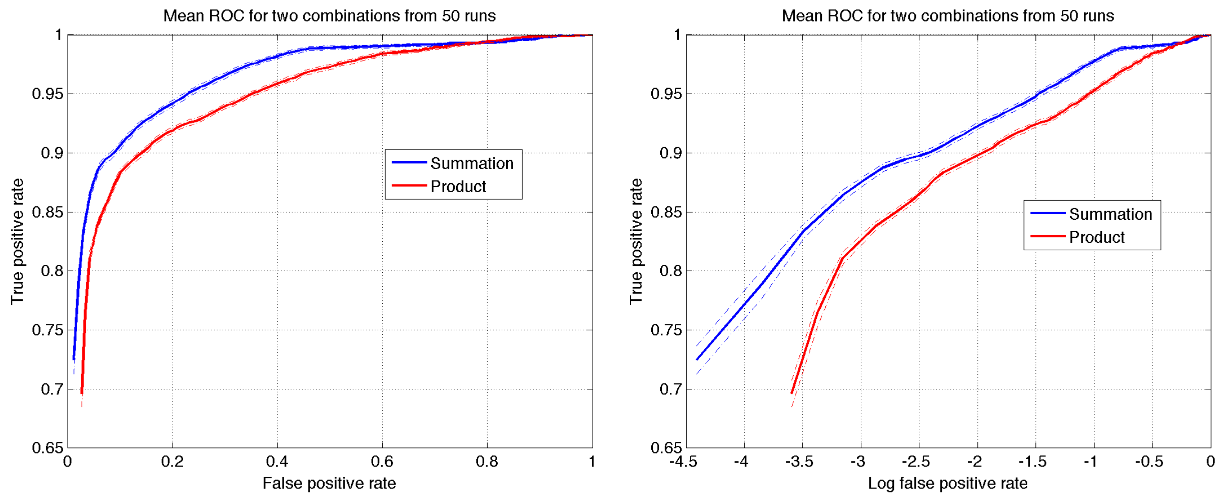

**Fig. 12.** Comparison of the two combination ways

Second, we have applied two straightforward ways to combine scores from the two criteria. In the future, we plan to add tuning parameters in the combination. To achieve this, we need to engage with the human analysts. We can also draw scatter plot of the two scores to see if there is any pattern.

Third, we will try to obtain the time stamps of the dataset to see if there any time evolution in the malware samples.

## 6 Conclusion

In this work, we have proposed two criteria to generate a ranking list of unlabeled samples, where the most malicious and anomalous samples will be put on top of the list. Criterion 1 is used to select the most malicious samples, where logistic regression and SVM are applied and compared, and SVM with radial kernel is chosen. Criterion 2 is used to do anomaly detection, and KDE is selected for this criterion. Both two criteria will assign a score for each sample. Two combination ways, summation and product are compared. And we select the summation to generate the final scores for unlabled samples, and generate a ranking list based on these scores. However, the algorithm we use to select the best method in criterion 2 as well as the testing algorithm for combinations are very heuristic. We will have to improve our results based on the feedbacks from human analysts in the future.

## References

1. Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J.: Automated classification and analysis of internet malware. In: Recent Advances in Intrusion Detection. pp. 178–197. Springer (2007)

2. Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable, behavior-based malware clustering. In: NDSS. vol. 9, pp. 8–11. Citeseer (2009)
3. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning 20(3), 273–297 (1995)
4. Kinable, J., Kostakis, O.: Malware classification based on call graph clustering. Journal in computer virology 7(4), 233–245 (2011)
5. Neugschwandtner, M., Comparetti, P.M., Jacob, G., Kruegel, C.: Forecast: skimming off the malware cream. In: Proceedings of the 27th Annual Computer Security Applications Conference. pp. 11–20. ACM (2011)
6. Park, Y., Reeves, D., Mulukutla, V., Sundaravel, B.: Fast malware classification by automated behavioral graph matching. In: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research. p. 45. ACM (2010)