# CS4035 Cyber Data Analysis: Assignment 3

Yadong Li(4608283) Xuehan Jiang(4578791)

June 30, 2017

# 1 Sampling task

## 1.1 Load data and count

For this task, we loaded the data and we found that there is a record in each line, and the first ip address is in the middle of '$$', so we split and extract the ip address easily.
There are 3414011 records in total, the 10 most frequent ones are listed in table 1.

## 1.2 FREQUENT algorithm

| IP address | True Frequency | size 10 | size 100 | size 1000 |
|---|---|---|---|---|
| 115.177.11.215 | 294690 | 0 | 266465 | 291985 |
| 115.176.182.196 | 259241 | 67257 | 242671 | 257617 |
| 192.3.106.42 | 66572 | 0 | 43831 | 64119 |
| 191.96.249.189 | 51251 | 339 | 37805 | 49627 |
| 104.192.0.20 | 18707 | 0 | 0 | 16002 |
| 198.204.234.26 | 6817 | 0 | 0 | 4216 |
| 198.204.234.27 | 5284 | 0 | 0 | 2877 |
| 198.204.234.28 | 5239 | 0 | 0 | 2832 |
| 185.93.185.10 | 5157 | 0 | 698 | 4428 |
| 198.204.234.30 | 5049 | 0 | 0 | 2643 |

Table 1: Packet level evaluation

Theoretically speaking, FREQUENT algorithm returns all items with frequency more than $m/k$. If $k$=10, it is guaranteed to find items with more than 341401 occurances. That explained why some of the addresses are not found when $k = 10$. If $k$=100, the theoretical bound is 34140, which indicates that 2 most frequent IPs are guaranteed to be found. If $k$=1000, the theoretical bound is 3414, and we can see that all 10 most IPs are found.

# 2 Hashing task

## 2.1 BLOOM filter

For this task, we've built a BLOOM filter and perform the same test using different sizes: 10000, 100000, 1000000. The result is shown in table 2. We compare the difference of theoretical false positive rate and the true test false positive rate, and we can find that all the true false positive rate is bounded by the theoretical ones. Actually, much lower than theoretical ones.

| bit size | 10000 | 100000 | 1000000 |
|---|---|---|---|
| Theoretical $fp$ | 0.99718 | 0.97225 | 0.7547 |
| Test $fp$ | 0.6564 | 0.6384 | 0.2008 |

Table 2: BLOOM filter evaluation

## 2.2 Count-min sketch

For Count-Min sketch algorithm, the result is shown in table 3. Here we use top-ten average error of the true count and the estimated count. If we increase the width, the theoretical bound guarantees that the error will be very low. Thus, we can get better estimation compare to FREQUENT algorithm.

| Width | 200 | 2000 | 20000 |
|---|---|---|---|
| Theoretical top-ten average error | 0.001 | 0.0001 | 0.00001 |
| Test top-ten average error | 0.0040 | 0.00035 | 0.0000026 |

Table 3: CM Sketch evaluation

# 3 Botnet classification task

## 3.1 Data exploration

The netflow dataset contains background, benign and malware packets. Here we just take scenario 9 as an example. First we visualize the amount of each categories, as shown in Figure . We can easily tell that the distribution is skewed and imbalanced. Also, we noticed that there are both categorical variables and numerical variables. If we want to use categorical variables, one-hot encoding is needed.

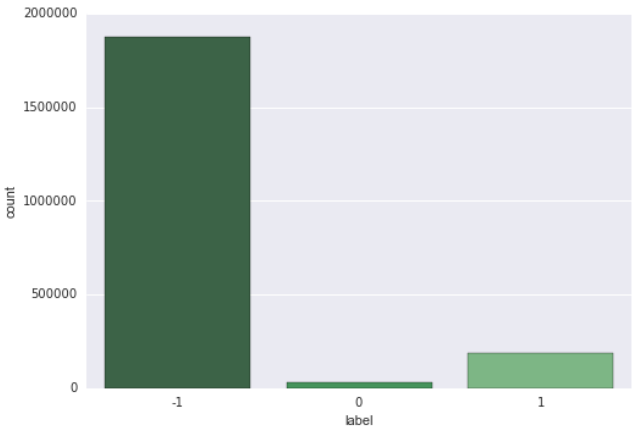


Figure 1: Number of netflows in different categories. 1 represents "malware", 0 represents "benign" and -1 represents others.

## 3.2 Classifier building

For classification task, first we should do data cleaning. Here we can simplify it as a binary classification problem: benign or malware, so we ignore all the background netflow. Also, we just use 4 numerical features: *Duration, Total packets, Total bytes, Source bytes*, and 1 categorical variables: *Protocols*. The classifiers we have tested here are Logistic regression and Random Forest. The experiment setting is 10-fold cross validation, and for the imbalanced issue, down sampling has been applied here.

## 3.3 Evaluation: packet level and host level

### 3.3.1 Packet level

For packet level evaluation, we tested both Logistic Regression classifier with L2-Regularization and Random Forest. The result is shown in Table 7. We found that Random Forest is a better choice here.

| Classifier | TP | FP | TN | FN | Precision | Recall |
|---|---|---|---|---|---|---|
| LR | 37028 | 5894 | 32 | 22 | 0.86 | 0.99 |
| RF | 36516 | 579 | 5247 | 534 | 0.98 | 0.99 |

Table 4: Packet level evaluation

### 3.3.2 Host level

For host level evaluation, we simply detect whether all netflows from a host have been detected as benign or malware. If one record of a host is malware, then we say that the host is detected as a malware. Using this strict evaluation metric, We get the following result in table

| Classifier | TP | FP | TN | FN | Precision | Recall |
|---|---|---|---|---|---|---|
| RF | 10 | 3 | 4 | 0 | 0.78 | 1 |

Table 5: Host level evaluation

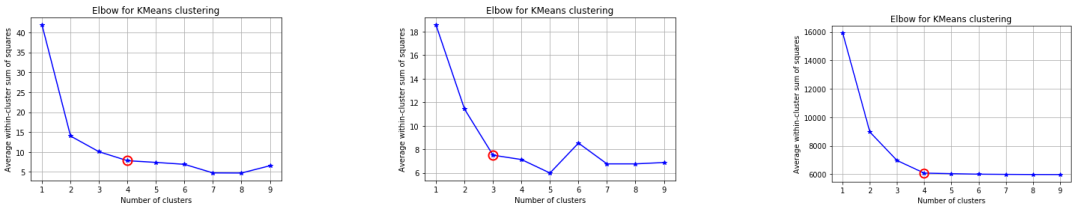


Figure 2: "ELBOW" for three numerical features. The red circle is the optimal number of clusters

# 4 Botnet fingerprinting task

## 4.1 Sequential model: N-grams

According to paper [1], five features was used to map the states for the following sequential model. For the categorical features ('protocal' and 'direction'), we directly assigned non-negative value for them. For the other three numerical features, we used percentiles for clustering. "ELBOW" method was applied to determine the optimal number of clusters. As shown in the Figure 2, the optimal number for 'Duration' ,'TotPkts' and 'TotBytes' is four, three and four, seperately. Then, we use uniform percentiles to discretize these features. For example, 'Duration' has a "break point" at number of cluster-5, i.e. 20th, 40th, 60th, and 80th percentiles.After discretization, we employed the mapping algorithm in [1] to obtained the timed event (i.e. states). The final states is some numbers like 412, 523, etc.

After aggregating Netflows of each hosts, a 40ms sliding window was applied to get the sequence data. We slided the window state by state so that enough number of sequence data can be obtained. Afterwards, we extracted n-grams data from these sequences. In our case, we used top 10 frequent n-grams to model the fingerprint. The fingerprint of each host comprise pairs of n-gram and its occurrence frequency in this host. The formula in page 194 of [2] was used to calculate the similarity distance of different fingerprints. With these distances we can find the nearest neighbor of the host to be detected.

## 4.2 Fingerprint matching

### 4.2.1 Within-Scenario

From [1], we know that the malware in one Scenario are similar. To detect malware within one scenario, We chose Scenario 9 which contains a network infected by Neris. There are 10 infected hosts and 5 normal hosts which are the ground-truth data. (Normal host 6 was discarded because it only has 6 Netflows). Given one infected host in one specific Scenario, we model its fingerprint and treated this fingerprint as the symptom of this malware. Likewise, we did the same for one normal host and used its fingerprint as the benign standard. The other 14 labeled hosts are testing data. If the nearest neighbor of one testing host is the symptom, it will be labeled "infected", vice versa.

We tested with different n and the results are shown below (L=10,40ms sliding window). For example, when n =3, all the 9 infected host were found but 1 benign host was mislabeled. It is obvious that the performance of detection using fingerprint based on n-grams is quite nice, due to the fact that the symptoms in one scenario are similar. We also tested with different length of sliding window and L, these parameters did not influence the results that much.

| n-grams | TP | FP | TN | FN | Precision | Recall |
|---|---|---|---|---|---|---|
| 3 | 9 | 1 | 3 | 0 | 0.9 | 1.0 |
| 5 | 7 | 2 | 2 | 2 | 0.78 | 0.78 |
| 7 | 6 | 2 | 2 | 3 | 0.75 | 0.67 |

Table 6: the results of different numbers of n-grams

### 4.2.2 Between-Scenario

Given the fingerprint we modeled based on the infected and normal hosts of Scenario 9, We detected the hosts in Scenario 1. As expected, the performance of dropped as a result of the big difference between scenarios. It is worth mentioning that the testing data is too small, which only have one infected host and five normal hosts.

| TP | FP | TN | FN | Precision | Recall |
|---|---|---|---|---|---|
| 1 | 1 | 4 | 0 | 0.5 | 1.0 |

Table 7: detect Scenario 1 using fingerprint of Scenario 9

# 5 Reference

[1]Pellegrino, Gaetano, et al. "Learning Behavioral Fingerprints From Netflows Using Timed Automata."
[2]Abou-Assaleh T, Cercone N, Keselj V, et al. Detection of New Malicious Code Using N-grams Signatures[C]//PST. 2004: 193-196.