CS 4035 LAB3 - STATE MACHINE AND STREAMS GROUP 29

Yuhang Tian
5219728

Mingyu Gao
5216281

June 13, 2021

1 Part 1: Discretization

1.1 Discretization Implementation

In the discretization task, we combine the protocol and duration into one feature. As shown in figure1 and 2, their performances are quite different in a botnet and benign data. The histogram shows that the protocol of botnet cases is mainly ICMP while the protocol in benign cases is always TCP and UDP. The box plot proves that the botnet cases always have a shorter duration.

To better combine two features into one, we implemented the PCA that we learned in Lab2. To select the best bins, we plot the relationship between the number of bins and the spearman correlation coefficient. As the number of bins also relates to the following tasks, the larger bins, the more 3-gram combination. We choose the smallest bins that can reach the highest spearman in its neighbors. For example, for dataset 10, we choose 10 bins.

1.2 Useful Observation

From figure4, We can thus clearly see the effect on discretization, which can easily and quite fast be implemented to be used for infection detection. In the combined feature, the majority of botnet cases are 0 while the benign cases mainly are 1. It proves the validity of our discretization, which can distinguish these two types of flows.

2 Part 2: Frequent Task

According to the figures above, we can see that:

For all data including botnet and benign cases, the result in figure1 shows that:

1) The amount of the same feature found by the Space Saving exceeds the actual amount, resulting in an accuracy larger than 1.

2) With the number of counters increment, the number of the same feature and the accuracy are getting better. In other words, the more counters, the better the result.

3) The turning points of the number of the same features and accuracy are different. When the number of counters reaches 16, Space Saving could find out all top 10 features as the actual result. But the average accuracy keeps getting closer to the perfect value(1).

4) By separating the benign and botnet data in the dataset, we can see that performance of the non-infected data set and the infected data are slightly different. From both perspectives of the number of the same features in the top 10 frequent elements and accuracy - the amount of the same feature found by Space Saving, non-infected data has a worse result. The reason could be that this data set is more varied. These results could be concluded that the SpaceSaving algorithm is a good method to approximate the most frequent elements.

5) The approximation errors could be explained using the mathematical theory as mentioned in the slides - "Any item x whose true count > m/k is stored" where m is the sum of the counters and k the number of counters. Approximation errors could occur when, for example, the last item in the stream only occurs once. In this case, the last n-gram replaces the counter with the lowest minimum value. However, this n-gram only occurred one time while the value with the minimum counter value could be way higher. The approximation is upper bounded by m/k, making Space Saving an excellent approximating distribution of data with limited memory.

3 Part 3: Sketching Task

Count-Min Sketch is memory-efficient for the continuous data stream and provides probabilistic data structure and upper bound for the model, as it approximates the actual data using sub-linear space. Thus, the method using approximation brings errors when storing the data. The error is generated by 1) the width ($1/\epsilon$) and the depth ($1/\delta$) and 2) the in-dependency of the pair-wise hash function.

According to the results, when pair-wise hash functions are carefully designed - the fewer occurrences of collision, the counting results can be more precise as each counting is deduced from the minimum accumulations of different hashing functions. With the increase of width and depth, the space of the matrix becomes more adequate to store all counting values correctly. In our experiments, the error is mainly caused by feature sequence slice *'000'* which both botnet and benign contain.

In conclusion, Count-Min Sketch is an efficient scheme whenever space requirements could be problematic and exact results are not needed which is trading off between accuracy and memory space.

4 Part 4: Min-wise LSH Task

We implement the Min-wise LSH task on all data including botnet and benign cases after the discretization on dataset 10.

The implementation of the Min-wise LSH task can be divided into the following steps. First, the minhash matrix is created where the rows represent the pairs of source and destination addresses and the columns are the 3-gram-permutations of features. The feature is a combination of protocol and duration after the first discretization task. The cell value is set to 1 if the 3-gram exists once in the stream data of the IP-addresses pairs. Second, we transpose the minhash matrix for further hash signature. Third, we design a function to calculate the Jaccard distance. We run the Jaccard distance calculation for minhash matrix to prepare for the comparison. Fourth, we set different bins to obtain a hash signature from 2 to 20 with a step size of 2. In this step, we also get the Jaccard distance and computation time for each bin. Last, we compare the runtime and estimate error for the LSH algorithm.

The time-comparison result shows that the hash signature generated by the LSH algorithm improves the speed of calculating Jaccard distance by approximately 98 times. This is a huge performance increase, especially if you are dealing with much more data.

The sub-task error estimation plots the error rate compared with the original minhash matrix with different numbers of bins(hash functions). Although the error rate varies a lot in each run, the average error rate is lower than 0.1, which proves the guarantee of precision of the LSH algorithm. We can see from the last figure that the best bin of the Min-wise LSH algorithm is around 10 with the smallest error rate.

5 Part 5: Random Hyperplane LSH Task

Random Hyperplane LSH splits signatures into bands and hashes them, and only the identical hash values fall into the same bin. Thus, Random Hyperplane LSH is a useful algorithm to generate fingerprints for the data.

Similar to the Min-hash task, the LSH algorithm has an advantage in run-time (the time for distance calculating), since it calculates the distance between data points in a lower-dimensional space. However, it needs extra time for mapping the data into a sub-space. In addition, it reduces memory storage when storing the data points.

6 Part 6: Profiling Task

From the four confusion matrices, we can see that the profiling method does not raise any false positives in the three data sets. It only raises one false positive for the data set 11. As for the false negative, it also detects some newly infected hosts in the data sets 9,10, and 12. These results prove the effectiveness of profiling and the exception of the data set11 performance could be the problem of the data set itself. As this method is highly dependent on the selection of threshold, the more threshold, the more false positives. This is a trade-off someone should be willing to make (or not).

7 Part 7: Fingerprinting

The fingerprinting task relies on the fact that some 3-grams do not occur for benign hosts and do occur for infected hosts rather than the distances between a selected host and other hosts in the profiling task to distinguish the hosts.

7.1 Implementation

First, we discretize all four data sets and then extract the benign ones.

Second, we do the 3-gram counting to the benign data set and all data set. This step could gather how many times one feature appearing in the data set. The former serves as the train set to find and store the occurred 3-gram features. The latter contains both infected and non-infected data that is used to be compared with the predicted result. In this step, we modify the method to define the occurred and un-occurred features with a probability of 1/27. In other words, if a 3-gram feature appears more than 1/27 of the length of the feature list, we identify it as occurred.

Third, the occurred featured found in step2 is applied in the 3-gram discretized all-data set to predict whether the host with different features is infected or not. The predicted result is appended in the column 'Pred' of the data frame.

Last, by comparing the 'Label' and 'Pred' columns, we get the four confusion matrices above.

7.2 Result Analysis

As for the results, the fingerprinting task does not raise false positives except for data set 11. But it detects some newly infected hosts for all four data sets. If we compare the confusion matrices to those of the botnet profiling task, we see that the results of fingerprinting are slightly worse. The reason behind their slight difference is that profiling measures the distance between profiles while fingerprinting merely builds upon the fact that some 3-grams only appear in botnet cases.

8 Bonus Task

Either individual net flows analysis or grouped by the host can solve the problem for this lab experiment. However, the packet level detection is more suitable for the scenario where the number of IP addresses is huge or the botnet IPs are mercurial, whereas the host level detection is more suitable for the scenario where the number of IP addresses is limited or the botnet IPs are fixed. In addition, the former method can both be online and offline detection and the time and memory costs are comparably larger, in contrast, the latter one is more useful for online botnet detection and it uses less memory space and reacts faster.