CS185C:
Final Project
Malware Classification

Jordan Conragan, Brett Dispoto

May 10, 2020

Contents

1 Preprocessing

Most of the machine learning techniques used in this report use variations of the same preprocessing steps. Here are the preprocessing steps taken for the methods described in this report. The preprocessing was done in python3, and the relevant files can be found in the `preprocessing` directory of our submission.

1. Download the dataset.
2. Split the dataset into directories based upon their family label. (Already completed by the dataset provider.)
3. For each malware family, the following steps were then taken:
   (a) Read through all of the files, count the occurance of each uniqe opcode across all files.
   (b) Take the n (turning parameter) most common Opcodes, and convert them to an ASCII symbol for observation sybmols for our HMMs. The Opcodes which are not within the n most common will be converted to an "other" symbol. This will reduce noise in our model.
   (c) Once each opcode is assigned a symbol, we again read through the files and convert the opcodes to symbols.
      i. If bagging is being used, make copies of **each** converted malware file, which will later be split up accordingly during training.
      ii. Otherwise, if boosting or stacking is being used, we can simply dump the converted opcodes (symbols) for the entire family into one huge file. This file will be our observation sequence.

2 Bagging Procedure

The following steps were performed in order to use bagging as an ensemble method:

1. Split assembly instructions into their appropape family, and translate the instrutins to HMM symbols,
2. Within these family folders, use a shell script (included, `makeTesting.sh`) in order to split 10 percent of the samples into a test set.
3. Train x HMMs on each family,
4. Once each HMM for a given family is trained, score the observation sequence which just used to train the model.
5. Write down this score.
6. Once all HMMs of a given family are trained, use whichever ensembler aggegrate function to take all the generated scores and aggregate them into one score.
7. Once training is complete for all families, for each family we now have this "aggregate score" written down in a log in the directory where the training samples are located.
8. Finally, to test all of these trained HMMs, we go into all of the test sets, score the test sample using the **each of the three "ensemblers"** (each made up of x HMMs.). Once we have the score from each ensembler, we can then go back to the logs where the original score was written down for the training samples.
9. Now, our sample has three scores for each ensembler. We classify this sample as whichever has the minimum of $abs(S(x_{\text{test}}) - score_{\text{family}})$

   "written down" score is closest to the score the ensembler