Practical 2: Malware Classification

Ethan Alley, Grigory Khimulya, Walter Martin
alley@college.harvard.edu, khimulya@college.harvard.edu, wmartin@college.harvard.edu

March 10, 2017

1 Technical Approach

We explored the problem of malware classification from several different angles:

- Feature engineering
- Running through a variety of models quickly to test which show the most immediate promise
- More focused tuning of hyperparameters for neural nets

Feature engineering process (Grigory and Ethan)

With several different feature sets to test on, we tested out classification with a random forest, a support vector machine, a standard neural net (sklearn's MLPClassifier), and an LSTM net.

Tuning standard neural net parameters was an accessory step, more with the intention of learning about overfitting rather than improving performance. The parameters tuned were:

- Number of hidden layers (1 or 2)
- Size of hidden layers (between 800 and 50)
- Activation function (tanh, relu, or logistic)
- Maximum number of optimization iterations

Model	Acc.
RF, $n = 100$, ON TFIDF	0.791
DEEP NET ON TFIDF	0.785
SVM ON 4G BOW	0.772
SVM ON GBOW	0.735
SVM ON TOPIC MODELED FV	0.342
PASSIVE AGGRESSIVE ON GBOW	0.150

Table 1: Model accuracy on private tests.

How did you tackle the problem? Credit will be given for:

- Diving deeply into a method (rather than just trying off-the-shelf tools with default settings). This can mean providing mathematical descriptions or pseudo-code.
- Making tuning and configuration decisions using thoughtful experimentation. This can mean carefully describing features added or hyperparameters tuned.
- Exploring several methods. This can contrasting two approaches or perhaps going beyond those we discussed in class.

Thoughtfully iterating on approaches is key. If you used existing packages or referred to papers or blogs for ideas, you should cite these in your report.

2 Results

We had a wide variety of performance across the methods and feature sets we tested. For the models that we submitted to Kaggle, our best result was approximately .79 accuracy on the private tests, produced by a random forest classifier.

In this table, GBOW is "garbage bag of words," FV is "feature vector," 4G is "4-gram," and TFIDF is [Ethan help!]

We also did some tuning of neural net parameters. These cross-validation scores were generated using sklearn's "cross val score" function; we took the average of 5-fold CV results. We were using the "tanh" activation unless otherwise specified. The numbers associated with the model is the dimension of each hidden layer. This data came from training on our length 10000 feature vectorization.

Model	Acc.
400	0.867
400, LOGISTIC	0.867
400, 50, LOGISTIC	0.867
400, 100	0.866
800	0.866
400, 50	0.863
400, 200	0.862
200	0.862
200, 100	0.860
100	0.858
400, RELU	0.855

Table 2: Neural network 5-fold CV accuracy.

This section should report on the following questions:

- Did you create and submit a set of predictions?
- Did your methods give reasonable performance?

You must have *at least one plot or table* that details the performances of different methods tried. Credit will be given for quantitatively reporting (with clearly labeled and captioned figures and/or tables) on the performance of the methods you tried compared to your baselines.

3 Discussion

End your report by discussing the thought process behind your analysis. This section does not need to be as technical as the others but should summarize why you took the approach that your did. Credit will be given for:

- Explaining the your reasoning for why you seqentially chose to try the approaches you did (i.e. what was it about your initial approach that made you try the next change?).
- Explaining the results. Did the adaptations you tried improve the results? Why or why not? Did you do additional tests to determine if your reasoning was correct?