Practical 2: Hidden Markov Models for Malware Analysis

Abdul Saleh, Dean Hathout, Brendan O'Leary
{abdelrhman_saleh, dhathout}@college.harvard.edu, boleary@g.harvard.edu
abdulsaleh, dhathout, boleary134

March 9, 2019

1 Technical Approach

Malware detection is an active area of research due to the rapid increase in the number of new malware variants in the past few years. Recent studies have experimented with a variety of machine learning approaches to detect malicious programs, ranging from applying CNNs on image representations of malware binaries (Kalash et al., 2018) to applying random forest classifiers on memory access patterns (Banin and Dyrkolbotn, 2018).

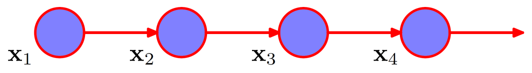
Many malware detection techniques rely on analyzing behavioral patterns of malicious programs during execution. For example, multiple studies treat system calls as a sequence of words and apply bag-of-words document classification techniques to identify malware classes (Canzanese et al., 2015; Bacci et al., 2018; Lin et al., 2015). The issue with bag-of-words approaches is that they do not model the sequential nature of the system calls and assume i.i.d. observations. To address this limitation, we use **hidden Markov models** (HMM) for malware classification instead. We hypothesize that Markov models are better suited for this task due to their ability to model sequential data. *Hypothesis*

For simplicity, we start by considering only one sequence or data point. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a sequence of n system calls associated with a malware label y. We assume $\mathbf{x}$ is a one-hot categorical variable that can take one of $D = |\{\text{system calls}\}|$ values. We could model this sequence as a first-order Markov chain where the probability of each observation only depends on the previous one. In this case the joint distribution of the sequence is:

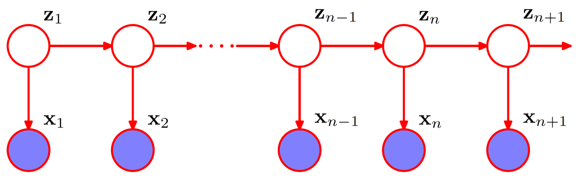
$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n | \mathbf{x}_{n-1}) \tag{1}$$

Although analytically tractable, first-order Markov chains are limited when it comes to modeling long sequences since each observation is only influenced by the previous one. We can extend the transition probabilities to depend on the past M observations but the number of parameters grows exponentially in M rendering this approach impractical.

Hidden Markov models are an alternative formulation that model long term sequences with a reasonable number of parameters. HMMs introduce a latent variable, $\mathbf{z}_i$, for each system call, $\mathbf{x}_i$,



(a) A first order Markov chain of observations $\{\mathbf{x}_n\}$.



(b) A hidden Markov model with a sequence of observations $\{\mathbf{x}_n\}$ conditioned a Markov chain of latent variables $\{\mathbf{z}_n\}$.

forming a sequence $\mathbf{z}_1, \ldots, \mathbf{z}_n$ of latent variables. These latent variables are assumed to follow a first-order Markov chain giving the following joint sequence probability:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_1, \ldots, \mathbf{z}_n) = p(\mathbf{z}_1) \prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}) \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n) \tag{2}$$

Under this formulation, $p(\mathbf{x}_{n+1}|\mathbf{x}_n, .., \mathbf{x}_1)$ no longer exhibits independence properties as any two observations are connected through a sequence of latent variables.

Since $\mathbf{x}$ is a categorical variable we model each observation as a D dimensional multinomial distribution of 1 trial. We define $\mathbf{z}$ to be a one-hot variable that can take one of K possible states, and each state corresponds to a mean vector, $\boldsymbol{\mu} \in \mathbb{R}^D$, of a multinomial distribution. Thus our model has a total of $D \times K$ learned "emission" probabilities in addition to the Markov chain transition probabilities. The probability of an observation is thus given by:

$$p(\mathbf{x}_n|\mathbf{z}_n) = \prod_{i=1}^{D} \prod_{k=1}^{K} \mu_{ik}^{x_{ni}\, z_{nk}} \tag{3}$$

The likelihood of a sequence can be calculated by marginalizing the latent variables:

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta}) \tag{4}$$

The parameters are learned using an expectation-maximization algorithm (Dempster et al., 1977) to maximize this likelihood function. The details of this iterative algorithm are beyond the scope of this practical but we refer the reader to the original paper for more information.

Implementation

For our classifier, we split the data by malware class to get 15 disjoint sets of system call sequences. We then train a different HMM on each set of sequences during the training phase. During inference, we calculate the log-likelihood of a given test sequence using each of the 15 HMMs. The test sequence is assigned the label corresponding to the HMM producing the highest log-likelihood. In other words, we associate each sequence with the HMM most likely to generate that sequence. We use the `hmmlearn`[1] library for training and scoring.

[1] https://github.com/hmmlearn/hmmlearn

Model	Features	Accuracy
LR	Count 1-gram	62.5
	↳+ 2-grams	64.3
	↳+ 3-grams	69.8
LR	TFIDF (1,2,3-grams)	82.2
RF		80.5
GBC		80.6
MLP		**83.0**
HMM	One-hot	80.5

Table 1: Test set accuracy of all trained models. Best result in bold.

Class	LR (1,2,3-grams)	HMM
Agent	20.0	30.8
AutoRun	83.3	62.5
FraudLoad	57.1	40.0
FraudPack	66.7	66.7
Hupigon	66.7	22.2
Krap	28.6	50.0
Lipler	92.3	85.7
Magania	00.0	71.4
None	84.6	87.5
Poison	00.0	22.2
Swizzor	92.7	98.6
Tdss	57.1	46.2
VB	33.3	84.6
Virut	22.2	36.4
Zbot	88.9	88.9

Table 2: Test set F1 scores for each class comparing non-sequential (LR) and sequential (HMM) models.

2 Results

In addition to our HMM-based classifier, we experiment with logistic regression (LR), random forest classifiers (RF), gradient boosted classifiers (GBC), and multilayer perceptrons (MLP) for reference. We also experiment with count and TFIDF features in addition to the one-hot sequence features. We split the data into a training and a test set. Model hyperparameters are tuned via cross-validation on the training set, and we report results on our own test split in this section.

Table 1 shows test set accuracy for all of our tested models. Overall, bag-of-words and sequence features achieve comparable accuracy. Models trained on TFIDF features provided significantly more predictive power than those trained on count features – LR achieved 82.2% accuracy when trained on TFIDF word unigram, bigram and trigram features, but only 69.8% accuracy when trained on unweighted count features. HMMs, trained on one-hot sequences, achieved 80.5% accuracy, outperforming LR trained with counts, but matching the lowest accuracy achieved with TFIDF features. The highest test set accuracy was achieved by an MLP trained on TFIDF features, at 83.0%.

Table 2 shows test set F1 scores where $\mathrm{F1} = 2 \cdot \frac{\mathrm{prec} \cdot \mathrm{recall}}{\mathrm{prec} + \mathrm{recall}}$ for each class of malware, comparing the performance of LR on unigram, bigram, and trigram counts and HMM on one-hot sequences. We show F1 scores since they can be unambiguously defined at the class level, unlike accuracy. For most classes, HMM F1 scores are either similar to or significantly greater than LR F1 scores (e.g. Magania, Poison, VB), with the exception of AutoRun, FraudLoad, and Hupigon.

Thus, using sequence features produces higher overall accuracy and F1 scores by class than unweighted count features, but lower accuracy than TFIDF features.

3 Discussion

In our initial exploration of this task, we consider consider system call counts as features for logistic regression, a linear method of classification. More specifically, we use various forms of n-grams (unigrams, bigrams and trigrams). As seen in Table 1, the classification accuracy increases as higher order n-grams features are included. Bigrams and trigrams can be thought of as crude forms of measuring sequences within system calls. *Given the significance of order that the simple n-gram classification suggests, hidden markov models are worthy of consideration to further model the relevance of system call sequencing in classifying malware.*

As described previously, 15 individual HMM models are trained to classify executables based on the log likelihood observed across the HMM models. Table 1 further confirms the importance of order as the HMM classification methodology produces a test set accuracy of 80.5%, higher than the simple unigram count linear classification. However, computational complexity should also be considered with classification accuracy as the HMM classification requires 15 models to be trained, each of notable computational significance.

It is also worth noting that the number of states the latent variable $\mathbf{z}$ can take is a crucial hyper-parameter for HMMs as it controls model complexity by limiting the number of possible emission probabilities, $\boldsymbol{\mu}$. We carefully tune this parameter to prevent **overfitting** where a smaller number of states produced a simpler model.

Given the computational requirements of HMM, we extend our analysis to other classification models as well as improved feature engineering. The frequency of each n-gram varies within the dataset of executable. It seems likely that certain calls or sequence of calls are standard across executables and will be uninformative in distinguishing malware families. We adopt Term Frequency-Inverse Document Frequency (TFIDF), a simple but powerful weighting scheme native to natural language processing call. TFIDF decreases the weight of n-grams that occur frequently in an executable or frequently throughout the entire dataset of executables. This enhancement improves the classification accuracy compared to the simple count version and performs well in linear and nonlinear models.

Table 2 provides classification accuracy by malware class. The HMM performs better than the (1,2,3)-gram logistic regression for None, Swizzor and VB. These are the most prevalent classes within the dataset at 52.14%, 17.56% and 12.18% respectively. For other classes the performance is mixed. It is probable that HMM models of smaller classes require more data to fully distinguish the sequential ordering of its system calls. Nevertheless, there is strong evidence that HMMs are good candidate models for malware classification. Although the overall accuracy of bag-of-words features is higher, Table 2 shows that HMMs are better at detecting certain malware classes. This suggests that sequence features and HMMs should be used along with bag-of-words features to achieve the best performance.

References

Alessandro Bacci, Alberto Bartoli, Fabio Martinelli, Eric Medvet, Francesco Mercaldo, and Corrado Aaron Visaggio. 2018. Impact of code obfuscation on android malware detection based on static and dynamic analysis. In *ICISSP*, pages 379–385.

Sergii Banin and Geir Olav Dyrkolbotn. 2018. Multinomial malware classification via low-level features. *Digital Investigation*, 26:S107–S117.

Raymond Canzanese, Spiros Mancoridis, and Moshe Kam. 2015. System call-based detection of malicious processes. In *2015 IEEE International Conference on Software Quality, Reliability and Security*, pages 119–124. IEEE.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Mahmoud Kalash, Mrigank Rochan, Noman Mohammed, Neil DB Bruce, Yang Wang, and Farkhund Iqbal. 2018. Malware classification with deep convolutional neural networks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE.

Chih-Ta Lin, Nai-Jian Wang, Han Xiao, and Claudia Eckert. 2015. Feature selection and extraction for malware classification. *J. Inf. Sci. Eng.*, 31(3):965–992.