Weekly Homework 3

Alex Ring

1 Writing Assignment

1.1 Title and author of paper

Learning to detect Malicious Executables in the Wild
Jeremy Z. Kolter
Marcus A. Maloof

1.2 Summary of paper

The two researches set out to classify malicious software found "in the wild". They point out, that currently, classification of malicious software requires a prior knowledge of that software. Commercial products such as McAFee use a malicious software database to compare against files on a computer. This requires at least one computer to become infected to discover a new type of malicious software. These researchers used text based classification of features of malicious software to attempt to classify software found "in the wild" before it could infect a computer. They found that by using the hexdump utility and converting executable to hexadecimal codes, they were able to form n-grams that then went through many different types of classifiers. These included: Instance-based Learner, TFIDF, Naive Bayes, Support Vector Machines, Decision Trees, and various Boosted Classifiers. Using a ROC curve they measured performance metrics of these classifiers against one another and found that Boosted J48 was the best classifier and most capable of scaling up to larger datasets. In the end the researches came up with a method of classify unknown software found "in the wild" as being malicious or benign.

1.3 Good things about the paper

The researchers are very thorough in their testing methodology, making sure to give many different configurations of n-grams, and many different classifiers a chance. By being so redundant, there is faith in the data set that their methodology produced tangible and repeatable results. The researchers also acknowledge past experiments and their implication on their own, as well as comparing their findings to older findings. In the conclusion the researches also list all shortcomings and constraints they faced in coming up with their results.

1.4 Major comments

It would seem to be that further elaboration is needed in the experimental results section. When going over their pilot studies, the authors note that 500 n-grams of size four produced from single bytes were the best preforming combination. While they note that this was through trial and error selecting different sizes for each of the given metrics, they do not elaborate on what they deemed to be best preforming. What metric are they measuring each of these against? What makes them best preforming?

1.5 Minor comments

I would like to see the various classifiers measured on more than one metric. Right now they are only measured by performance on a ROC curve. I would like to see them also measured in things like the space that they take up and other physical constraints to repeating such an experiment.

1.6 Recommendations

Graphs could be larger, more elaboration is needed on the measurement of performance on selection of n-grams.

2 Written Homework

1. Regarding the paper assigned for Part A, answer the following:

a) What is a relevant feature, and how did this paper determine which features are "relevant?" Do you agree with this? Why? Relevant features are the aspects of the problem domain that would be best at predicting a class for unknown data. That is to say they are the stuff that you train the data on to predict the class of a new piece of data. In the paper they determine relevant features to be n-grams that contributed to the largest information gain (IG). They then chose the top 500 from these. I think that the way they chose the n-grams seems to be a fine way of doing so. You would want n-grams that contribute heavily to information gain to determine which features would be good for classifying future programs, but I don't really like how they chose only 500 of them. It seems like such a small amount of n-grams to train on for the problem domain. They mention later that they went with 500 because they were best preforming, and that their methodology had a large amount of computational overhead. So I understand why they went with only 500, but it seems like such a small sample size.

b) In the assigned paper, which is more important? What would be an application where less than perfect precision or recall are unacceptable? Justify your answer. I would think for the assigned paper that recall would be more important. We are attempting to classify malicious software, and with precision we are measuring true

positives against true positives and false positives. False positives are not that big of a deal here because it would just mean that a benign piece of software is misclassified and no harm is done. But with recall we are measuring the false negative, which would mean that a piece of malicious software could be classified as being benign. This would be terrible for a system that is attempting to protect against malicious software. You would much rather have the system be overzealous in classification of benign software as malicious, then have it misclassify malicious software as benign, causing infection.

It would be unacceptable to have less than perfect precision on something that involved life and death. If you had an automated missile defense system that would attempt to shoot down foreign attacks by classifying them as foreign, a misclassification would lead to death and destruction. You would absolutely need perfect precision/recall in that sort of situation.

c) Give an example of a set of n-grams for a word that are used in this work. They used the hexdump utility to convert executables to hexadecimal codes in ASCII. From these they produced n-grams 4 bytes long. So an given n-gram might look like this after the hexdump : ff 00 ab 3e 12 b3. Then after combining them to 4 byte length they would look like : ff00ab3e. The paper gives examples of ff00ab3e, 00ab3e12 and ab3e12b3.

d) Describe the feature space and label space for this problem. The feature space of this problem is equal to the number of n-grams they measured. Because they went with 500 n-grams for the training data, this particular feature/label space would be in 500 dimensions. All of them binary. The n-grams were measured as being present, or not being present.

e) Describe at least two things about the paper that you did not understand. I did not really understand how Support Vector Machines work as a classifier. I found the explanation that they use a kernel function to map training data into a higher dimensioned space, to be confusing. I also did not really understand what exactly boosting naive bayes did for the classifier. They mention in the paper that boosting uses some combination of classification techniques and boosted naive bayes saw and improvement for them in their experimentation, but they did not really elaborate on how they boosted naive bayes. I would have appreciated some more explanation on what they were actually doing to boost.

f) Are you surprised by the results of this paper? Why or why not? I wouldn't say I am surprised. I think that the way they went about classifying malicious software is super cool. I wouldn't have thought that you could use a text based classification to determine if a piece of software was a virus. Their methodology is more surprising to me than the results. I guess I am a little surprised that Naive Bayes preformed so poorly, I was under the impression that it was one of the best ways to do text classification.

g) What is the IG function and its relationship to entropy? The IG function used in this paper is $IG(j) = \sum_{v_j \in \{0,1\}} \sum_{C \in \{C_i\}} P(v_j, C) log \frac{P(v_j,C)}{P(v_j)P(C)}$ Which measure how much of the n-grams are going to be "pure". That is to say which ones provide the most

amount of information about the thing we are trying to classify in a tree. Which of the features can move the software into a more specific child node in a decision tree the best? Entropy is the exact opposite of this. It measure the "impurity" of the information gain. For a binary class it is defined as: E = - P(a) logP(a) - P(b) logP(b). So in short, IG is the difference in entropy at decision points in a tree.

h) What is the prior in the IG function and how is it estimated? The prior in IG is the value of the K-L divergence of the prior distribution. According to the paper it looks at the previous example and measure how well the new n-gram is going to classify the previous piece of data. It is all estimated on the previous run of the classifier.

i) The arg max function in the Naive Bayes formula returns C. What is it? What would the max function return, if it were used instead of arg max? The arg max function returns the class in which the function is maximized. So with naive bayes in this paper the arg max would return what sort of malicious software class would maximize the output of the function. Max on the other hand returns the largest output of the function rather than the input. So it would just return some large probability, rather than the class the features belong to.

j) What reason do the authors give for the inconclusive results of boosted SVMs? They discuss how boosting can adversely affect stable classifiers. They say that stability may also explain why the benefit of boosting SVMs was inconclusive in their study. They do also mention that the SVM seemed to improve performance as the data size was scaled larger.

k) In statistics, features are called "dependent variables." In data mining, features are often called "attributes." (1) What criterion is typically used to create splitting nodes in the C4.5 algorithm mentioned in this paper? (2) What is its relationship to KL-divergence? And (3) what is this metric's relationship to entropy? The information gain, or difference in entropy is typically used to create splitting nodes in the C4.5 algorithm. information gain is essentially synonymous to KL-divergence. This metric is directly related to entropy as information gain tries to minimize the entropy at any given decision point.

2. What is the loss function of a Naive Bayes classifier (in words or as a formula)?

The loss function is a function that indicates the penalty for an incorrect prediction. It measures how wrong the classifier is. The Bayesian Approach to a loss function uses the following: $L_c = -\sum_{k=1}^{P} F_{ck} log\theta_k + S_c \sum_{k=1}^{P} \theta_k$

3.Does logistic regression have the same problem with 0's as Naive Bayes, thus requiring smoothing (such as Laplace)? Why or why not?

Logistic regression handles zero values much better than Naive Bayes. Because the formula for logistic regression is a summation rather than a product, a zero value doesn't affect the end product. That is to say if you have a 0 value you might have something

like, $\beta(10) + \beta(2) + \beta(3) + \beta(0)$ the zero value at the end there doesn't affect the total sum. Adding 0 to something doesn't change that something. Conversely, in Naive Bayes, because of the product multiplying by a zero value can mess everything up, and requires smoothing. Logistic regression has no such failings.