Cyber Data Analytics Assignment 4

INTRODUCTION

Traditional machine learning algorithms are tested on a separate but fixed testing dataset. Possible modifications in data characteristics are not tested. This creates issues for cyber security applications, where an attacker will likely modify his or her attacks if they are detected. A smart attacker might even have access to the defensive infrastructure and created targeted attacks that avoid detection altogether.

The core of this issue is that machine learning algorithms are not designed to be robust against such adversarial modification of testing data. The field of adversarial machine learning studies how to make such robust designs. In this exercise, you will build both attacks and defences for the malware domain.

For a long time, there has been hope that machine learning could improve the detection of malware by learning what exactly makes a file malicious. However, it turns out that many models (e.g., neural networks) are extremely sensitive to small changes in the data. This means that malware authors can easily evade detection by adding useless imports to their code for example.

In this assignment you will make steps towards training robust neural networks for malware detection by protecting against evasion by useless imports.

LEARNING OUTCOMES

After completing this assignment, you will be able to:

1. Attack classifiers to avoid detection.
2. Defend against such attacks by making classifiers more robust.

INSTRUCTIONS

In this assignment we will use the AdvML malware dataset from Brightspace, both files X.npy and y.npy can be loaded with NumPy. The X matrix contains thousands of columns with zeroes and ones that represent whether a windows executable contained an import. The y vector represents whether a file was benign (0) or malware (1).
For the model, we will use Scikit-learn's SGDClassifier (Stochastic Gradient Descent Classifier).
This is a linear model that starts with randomly initialized coefficients (coef) that we can easily update with gradient descent using partial_fit(X_batch, y_batch).

Important: Do not forget to set the loss function to 'log_loss' for logistic regression.

Attacking linear models with useless imports (5 points)

- Split the dataset into a train and test set and train a logistic regression model
- Implement a function evade(sample, coef) that takes a data row and linear model coefficients and returns a new data row that the model will hopefully predict as benign.
- Compute the accuracy of your model on the test set before and after applying your attacks to the test set's malware rows. How does the performance change? Is your model robust? Explain why.

Improving robustness by learning from attacks (5 points)

- One way to improve the robustness of your model is to continuously learn from attacks on your train set. Train a new linear model by first applying evasion attacks to your train batches before giving them to the model.
- Compute the accuracy of your model on the test set before and after applying your attacks to the test set's malware rows. How does the performance change? Is your model more robust? Explain why.

Non-negative linear models (5 points)

- While hardening might help improve robustness it could take much more data before arriving at a good model. Since we are now only interested in useless import attacks, we can be immune by forcing coefficients to only take non-negative values. Train a new linear model by clipping coefficients to non-negative values after every batch.
- Compute the accuracy of your model on the test set before and after applying your attacks to the test set's malware rows. How does the performance change? Is your model more robust that the previous two models?
- Vary the number of training iterations (number of partial_fit() calls) and plot the test accuracy with evasion attacks over those iterations. How do hardening and nonnegative models compare in terms of the number of iterations before getting good scores? Explain why.

Bonus!: Competition (5 points)

- The same recipe for training non-negative linear models using stochastic gradient descent can be used for training neural networks. Use any method you like (non-negative neural networks are one option) to beat the baseline on Kaggle.
- Explain why you chose this method and what performance you expect.

RESOURCES

Slides from Lecture 7

Paper describing the competition and GRAMS method available on Brightspace.

Links on Brightspace to online tutorials. Code samples available on Brightspace.

PRODUCTS

A zip containing:

- A Jupyter Python notebook for all parts of the assignment (including individual tasks). The word count should not exceed 1000 words (see first cell). Include libraries used to run the code other than numpy, scipy, pandas, and scikit-learn.

The notebooks will be assessed using the below criteria.

ASSESSMENT CRITERIA

Knockout criteria (will not be evaluated if unsatisfied):

Your code needs to execute successfully on computers/laptops of your fellow students (who will assess your work). You may assume the availability of 4GB RAM. Please test your code before submitting. In addition, the flow from data to prediction has to be highlighted, e.g., using inline comments.

Your report needs to satisfy the page limit requirements for the different parts. Submissions submitted after the deadline will not be graded.

The report/code will be assessed using these criteria:

Criteria	*Description*	*Evaluation*
Attacking	*Attacks work, reasoning is sound.*	*0-5 points*
Improving	*Defense works, reasoning is sound.*	*0-5 points*
Non-negative	*Second defense works, evaluation and analysis are sound.*	*0-5 points*
Bonus	*Performance is OK, reasoning for used technique is sound and reason for expected performance or insightful.*	*0-5 points*
Report and code	*The data-detection flow is clearly described, including preprocessing and post-processing steps.*	*0-5 points*

Your total score will be determined by summing up the points assigned to the individual criteria. Your report and code will be graded by the teacher and assistants, and the peer reviews are used as guidance.

In total 130 points (including bonus) can be obtained in the 4 lab assignments, of which 30 are individual. The total number of obtained points will be divided by 110 to determine the final course grade.

SUPERVISION AND HELP

We use Mattermost for this assignment. Under channel Lab1, you may ask questions to the teacher, TAs, and fellow students. It is wise to ask for help when encountering start-up problems related to loading the data or getting a machine learning platform to execute. Experience teaches that students typically answer within an hour, TAs within a day, and the teacher the next working day. When asking a question to a TA or teacher, your questions may be forwarded to the channel to get answers from fellow students. Important questions and issues may lead to discussions in class.

SUBMISSION AND FEEDBACK

Submit your work in Brightspace, under assignments.