Project 2: Extra Credit

Malware Analysis and Machine Learning

Summer 2020

The goals of this assignment:

- This project is to get you familiar with machine learning concepts applied to malware analysis.
- You will build on your experience using Cuckoo on the main Project 2 in order to learn machine learning strategies and tools.
- You will get familiar with malware behavior clustering, and how to use it to perform malware classification.
- You will build a machine learning based malware classification tool, your very own Intelligent AV software!

Additional information:

- This is an extra-credit assignment worth 05/100 extra points on your overall grade.
- This project is not mandatory, but we strongly recommend completing it.
- You will have 21 days to complete the project, starting from its release date.
- This semester, we also have a Frequently Asked Questions (F.A.Q.) thread in Pizza (post @548) at:
  https://piazza.com/class/k92zc5a2dmi2fj?cid=548

- The F.A.Q thread will be constantly updated. Therefore, before asking anything, take a look at it, and, if your question is not covered already, feel free to post it as a reply in the same thread.

Background (0 points):

Throughout the past few decades, Machine Learning has spread to nearly every corner of computer science. Computer Security is no exception. In this project you will learn how to apply Machine Learning concepts to malware classification. You'll be given a dataset of malware samples (similar to those analyzed in Project 2). Using Malheur, you'll run an unsupervised learning clustering algorithm on them, in order to classify them by behavior.

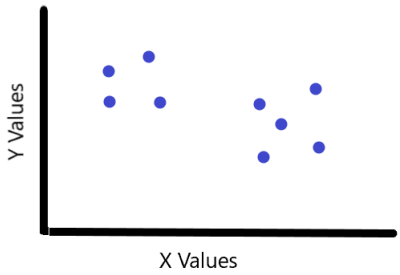
Understanding Unsupervised Learning and Clustering (0 points)

1. Unsupervised Learning is a type of Machine Learning used to extract information from a dataset. It's considered unsupervised, because no prior information on the structure of the dataset is provided to the algorithm. Instead, the unsupervised learning algorithm attempts to extract this information. One of the most common types of unsupervised learning is called clustering. Clustering looks for clusters of data-points in the dataset, and assigns those to their own group (called a cluster).

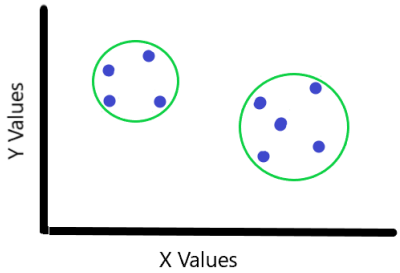
2. There are many kinds of clustering used in data analysis. Malheur, the software used for clustering malware in this project, uses a hierarchical clustering technique. Hierarchical clustering begins by setting each entry in the dataset to be its own cluster. Then, it combines the closest two clusters into one new cluster. It repeats this until either a target number of clusters is reached, or the distance between each cluster exceeds a target amount.

3. To further help you understand hierarchical clustering, we provide an example below. This example uses 2-d data to help you visualize what clustering is actually doing. The process of clustering malware samples is similar, but the data has many more features.

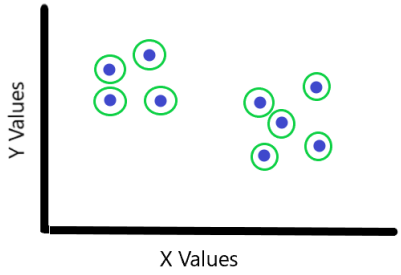
4. Let's say we're given the following data-set. It consists of x-y coordinate data that, when plotted, looks like:



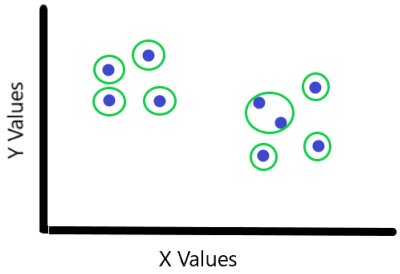
We want to determine if there are clusters of points in the dataset. We can see from looking at it, that there are two well defined clusters of points.



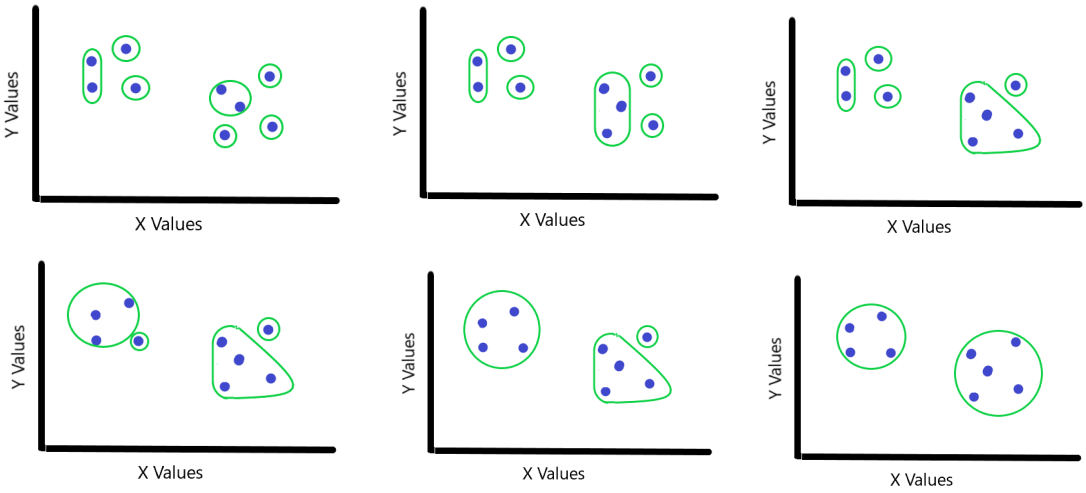
In order to find these clusters computationally, we're going to use hierarchical clustering. We start by setting each point as a cluster.



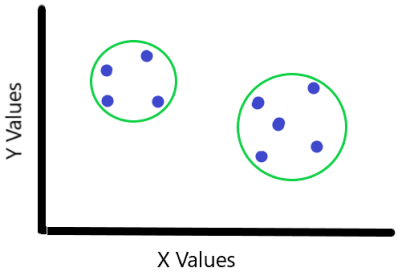
Then, we combine the closest two clusters.



And repeat the process….



We do this until each of the closest two clusters are fairly far away from one another. The exact minimum distance between the final clusters, can usually be set as a parameter to the clustering algorithm. This gives us the clusters we were expecting:



5. This is a general example of hierarchical clustering. In order to cluster malware, malhuer extends this method, as you will see in the next section. In order to complete this assignment, you will have to develop a deeper understanding of what Malheur is and how it performs this clustering. We encourage you to further research Machine Learning and clustering methods, beyond this section, in order to better understand the next section.

Understanding malheur (0 points)

1. Malheur uses a hierarchical clustering method to cluster malware samples. However, due to the number of features the malware has (there are many dimensions to the malware data), just performing a hierarchical clustering will take too long. In order to complete this assignment, and become accustomed to how Malheur works, you should read the following sources about malheur:
    - malheur original paper: http://www.mlsec.org/malheur/docs/malheur-jcs.pdf
    - malheur manual: http://www.mlsec.org/malheur/manual.html

2. Pay especial attention to malheur configuration parameters. You are given a malheur sample configuration file to serve as starting point. You will be asked to work with these parameters (not necessarily all of them) during this assignment. Nevertheless, some of these parameters should remain unchanged, since changing them will be detrimental to the assignment's objectives. This way, unless you know exactly what you are doing, you should not change values for:
    - input_format = "text";
    - event_delim = ";";
    - state_dir = "./malheur_state";
    - hash_seed1 = 514084890;
    - hash_seed2 = 1978915395;

Understanding the dataset (0 points)

1. In machine learning, a data set is a collection of data points for one of more features that represent the object of study (malware behavior, in our case).

2. Our dataset is based on information extracted from Cuckoo malware behavior reports.

3. Cuckoo malware behavior reports have a section called 'behavior'. In this section of the report, you can find all API Calls that were part of the malware behavior during Cuckoo's analysis.

4. The way we chose to encode malware behavior was by simply listing all API Calls, in the order they happened, as indicated by the report.

5. More specifically, we build a list of API call names by extracting the contents of Cuckoo JSON report fields such as 'behavior->process->[X]->calls->[Y]->api', where X is the process index number and Y is the process API call index number.

6. In the end, every feature file will be a semicolon separated list of strings, where each string is an API call name, and those API call names are displayed in the order they have actually been called by the malware during Cuckoo analysis.

7. Each sample in our dataset is labeled as to indicate which malware family it belongs to. To gather labels and assign them to our samples, we have used a malware labeling tool called AVClass (https://github.com/malicialab/avclass) along with data from Virus Total website (https://www.virustotal.com/). Because of the way malheur recognizes labels, we have placed them in each dataset sample's file name as the file extension, while the main part of the file name is the SHA256 hash code for the original malware binary. For example, take the feature file named as:

    *0bc19b9304d5c409b9f480a9121c8c8abcef2f3a595ed6b2758daeb2d679b74a.**dinwod***

    (Since the file extension for this malware sample is 'dinwod', it belongs to malware family Dinwod.)

8. You can use the hash code that represents the file name to research extra information about each individual dataset sample. Several Malware research related websites provide complete malware reports indexed by Hash codes. Since we have used data from Virus Total for our labels, it's probably a good place to start. You can use the URL https://www.virustotal.com/gui/file/<HASH_CODE> to directly open a malware detection report in Virus Total for the hash code <HASH_CODE>.

    - For example, to gather extra information about a dataset sample with the following file name:

    0bc19b9304d5c409b9f480a9121c8c8abcef2f3a595ed6b2758daeb2d679b74a*.dinwod*

    - All you have to do is to remove the label (dinwood) from the file name and use the hash to craft and visit the following URL:

https://www.virustotal.com/gui/file/0bc19b9304d5c409b9f480a9121c8c8abcef2f3a595ed6b2758daeb2d679b74a

9. All dataset files are located in the "dataset" folder and they are distributed in two groups: "training" and "testing".

    - The "training" folder contains files with features extracted from various malware samples based on the strategy we discussed previously. You will use those files to "train" your machine learning model using clustering techniques.
    - The "testing" folder is much smaller and contains the files you will use to test the model you will eventually create.
    - We will discuss the training and testing phases of your assignment in more details ahead.

Assignment Preparation (0 points):

Setup (0 points)

1. Initially, you will have to install all necessary files and software binaries on the same Virtual Machine (VM) you have used for Project 2. To do that, log into your Project 2 VM, open a terminal window, make sure you have internet connection, and run the following command inside the terminal:

```
$ wget --quiet "http://bit.ly/cs6035p2" -O - | /bin/bash
```

2. After finishing running the installation script, enter the assignment directory by issuing the following command:

```
$ cd /home/debian/Desktop/avml/
```

Using malheur (0 points)

1. Now, it's time to get familiar with ***malheur***. As an initial warm up exercise, let's learn same sample commands:

    - **Cluster samples (training phase):**

```
$ malheur -c config.mlw -o training.txt -vv cluster dataset/training/; head training.txt
```

        - *This command tells malheur to analyze the dataset contained in dataset/training/ (11,000+ samples) in order to cluster all its samples into groups of similar malwares. Since malheur is using the dataset to learn how to group malware families, we are using a special subset that is called training dataset.*

        - *In this phase, malheur will output quality assessment information. More specifically, it will show you three main measurements: precision, recall, and f-score. Take this opportunity to research a little bit about them. Those measurements are all related to how good our clustering has been. The way malheur does it is by looking at each sample's label. If you take a look at the contents of directory 'dataset/training/', you will see lots of files, each of which with a different file extension. Those files extensions are the labels for each sample, and malheur extracts them to see if each cluster actually groups similar samples (similar labels) and how many samples that should have been part of the group were left out. This analysis is eventually represented in numbers by precision, recall, and f-score.*

    - **Verify clustering (testing phase):**

```
$ malheur -c config.mlw -o testing.txt -vv classify dataset/testing/; head testing.txt
```

        - *This command tells malheur to use the clusters it has built in the training phase (using samples in dataset/training/) to classify all samples inside dataset/testing. Our objective is to decide whether the model generated in the training phase is good or bad. That is one of the reasons all samples in the testing dataset used to be part of the training*

dataset but were randomly selected and put apart. They are selected this way, so testing data represents reality while minimizing bias towards the training data. Think about It as it were a student preparing for an exam: the end goal is succeeding in life, but the exam will give us a rough idea if that will actually happen. The same way, malheur "studies" (analysis) and "learns" (builds a model) from the training dataset (training phase) so it can succeed in real life (classify malware samples in the wild). Nevertheless, malheur will have to pass the exam (testing phase) to make sure it's ready. Would you put the exact same questions from study material in the exam? I hope not! Exam questions should be unique and, at the same time, representative of the real-life skills students are supposed to learn. Welcome to the testing phase!

- *In this phase, malheur will also output precision, recall, and f-score. You should think of those measurements as malheur's final grade. During this phase, malheur isn't being tested against the same exact data used to train it. It now uses a different dataset where all testing instances were statically chosen to adequately represent all training data, even though it is not the same data used in training phase.*

Assignment Tasks (5 points):

First Task: Cluster and Classify (2 points)

1. Work with malheur parameters in its configuration file so you can build a model to classify malware samples with at least 70% f-score during the testing phase. For that, you should do through successive runs of the training phase command followed by the testing phase command. Before every run, you should adjust malheur's configuration file parameters the way you see fit, based on the knowledge you have acquired during assignment preparation (if you have skipped it, it is probably a good idea to take a step back now and read it). You can choose to do it blindly or to make judgement calls before each run. You just have to remember two things:
    - You have to achieve at least 70% f-score in the testing phase;
    - You will have to explain the strategy you used to achieve your results, and how the value for each parameter relates to those results.

2. After you have reached the 70% f-score goal, use the same model you have trained malheur with to classify each of your project 2 original malware samples. For that, we have provided you feature files for all of Project 2 malware samples containing features extracted from their Cuckoo's JSON. Those feature files are located inside the directory "subjects" in the main assignment path (/home/debian/Desktop/avml/subjects).

3. In order to use your newly created malheur model to classify Project 2 malware samples, make sure you are inside the assignment main directory, and run the following command:

```
$ malheur -c config.mlw -o classify.txt -vv classify subjects/; cat classify.txt
```

    - Your goal is to have all Project 2 samples properly classified (none should be labeled 'rejected') using your model with maximum distance value of 1. The maximum distance for classification can be set inside malheur configuration file by altering the following property:

```
classify = {
        max_dist        = 1.0000;
};
```

    - If you can't achieve those classification results for Project 2 malware samples, try revisiting your training and testing phases with different parameters until you can comply with everything.

- Also, you can take a look at Virus Total reports for the prototypes that malheur has assigned to your subjects. Seeing the full malware behavior might give you a better idea about the direction you should take.

4. By achieving all Task's goals, you will have built a tool that is able to classify unknown executable binary samples (no prior knowledge needed) into potential malware families. It is your very own AV software!

5. **GOALS AND PARTIAL CREDIT**: You have two main goals for this task. Partial credit of 0.5 points will be given in case you deliver GOAL 1 without GOAL 2. In case you do not achieve GOAL 1, you will be given 0 credits for the entire task, even if you achieve GOAL 2.

   - GOAL 1: the first goal is to achieve 70% f-score during the testing phase only (you should NOT consider f-score results for other phases for the first goal, just the testing phase).

   - GOAL 2: the second goal is to classify all Project 2 malware samples with maximum distance of 1 (none should be labeled 'rejected') as indicated above.

6. **DELIVERABLE**: Your deliverable for this part of the assignment will be your final malheur configuration file (config.mlw), the one you have used to achieve all goals (70% f-score during testing phase and Project 2 malware samples classification with maximum distance of 1). There should be only one configuration file for both objectives.

Second Task: Explain your Results - (3 points)

1. Answer the following questions about malheur and machine learning in general: **(1 point)**

   a) Explain the meaning of precision, recall, and f-score in the context of the model you have just created using malheur. **(0.5 point)**

   b) What are 'prototypes' with regards to malheur? Your answer must include an explanation about what malheur uses them for, what is the strategy malheur uses to find them, and their relationship to malheur clustering and classification. **(0.25 point)**

   c) What is the strategy malheur uses to calculate distances between samples and why it is important to have API call names (our features) in the order calls were actually made in the malware execution? What would happen if this order is lost? **(0.25 point)**

2. Explain your results in Task 1 by providing the following:**(2 points)**

   a) Write a short paragraph explaining the meaning of the following malheur configuration parameter, how you have achieved your proposed value for it, and how that value relates to your results (precision, recall, f-scores): **(0.25 point)**

```
features = {
      ngram_len = XXX;
}
```

   b) Write a short paragraph explaining the meaning of the following malheur configuration parameter, how you have achieved your proposed value for it, and how that value relates to your results (precision, recall, f-scores): **(0.25 point)**

```
features = {
    vect_embed        = "XXX";
}
```

c) Write a short paragraph explaining the meaning of the following malheur configuration parameter, how you have achieved your proposed value for it, and how that value relates to your results (precision, recall, f-scores): **(0.25 point)**

```
prototypes = {
    max_dist = XXX;
}
```

d) Write a short paragraph explaining the meaning of the following malheur configuration parameter, how you have achieved your proposed value for it, and how that value relates to your results (precision, recall, f-scores): **(0.25 point)**

```
prototypes = {
    max_num = XXX;
}
```

e) Write a short paragraph explaining the meaning of the following malheur configuration parameter, how you have achieved your proposed value for it, and how that value relates to your results (precision, recall, f-scores): **(0.25 point)**

```
classify = {
    max_dist = XXX;
}
```

f) Write a short paragraph explaining the meaning of the following malheur configuration parameter, how you have achieved your proposed value for it, and how that value relates to your results (precision, recall, f-scores): **(0.25 point)**

```
cluster = {
    link_mode = “XXX”;
}
```

g) Write a short paragraph explaining the meaning of the following malheur configuration parameter, how you have achieved your proposed value for it, and how that value relates to your results (precision, recall, f-scores): **(0.25 point)**

```
cluster = {
    min_dist = XXX;
}
```

h) Write a short paragraph explaining the meaning of the following malheur configuration parameter, how you have achieved your proposed value for it, and how that value relates to your results (precision, recall, f-scores): **(0.25 point)**

```
cluster = {
    reject_num = XXX;
}
```

3. **DELIVERABLE**: You should compile all your answers into a single PDF document labeled with your Georgia Tech user ID. Upload this document as deliverable for this part of the assignment.