Algorithms of Information Security: Machine learning-based malware detection techniques I

Faculty of Information Technology
Czech Technical University in Prague

November 23, 2022

Problem Statement

- **Malware detection** - detect whether a given sample is malicious. This objective is the most important and depending on what machine learning (ML) technique is used, the generated output can be provided with a confidence value. Modification of this problem is to detect as much malware as possible while keeping a low false positive rate.
- **Malware category detection** - Malware can be categorized according to its behaviors and objectives. AV vendors have not still agreed upon a standardized taxonomy of malware categories, effectively recognizing the categories of samples can add valuable information for the analysis.

- **Malware similarity analysis** - this objective has three slightly different versions:
    - **variants detection** - recognizing that a sample is actually a variant of a known malware.
    - **families detection** - selecting from known malware families to which a given malicious sample likely belongs.
    - **similarities detection** - discovering what parts and aspects of a sample are similar to something that has been already examined in the past and also identifying what is different from everything else already observed in the past results worthwhile.

Workflow of ML-based detection

The workflow of ML-based malware detection is an iteration process and consists of the following five steps:

1. data extraction (using static and/or dynamic analysis)
2. data preprocessing and normalization
3. feature selection
4. machine learning algorithms (classification or clustering)
5. evaluation of performance

Workflow of ML-based detection

- **Data extraction** - The process of feature extraction is performed through either static and dynamic analysis, and it depends on the types of features.
- **Data preprocessing** - Features extracted from the previous step can have various representations. Depending on the data representation, features can be normalized, encoded to a more appropriate representation, missing values can be filled by average values, and noisy values can be removed.
- **Feature selection** - After preprocessing step, a number of features can be reduced to improve performance. A subset of the most relevant features can be selected, or original data can be transformed into a space with a lower dimension.

Workflow of ML-based detection

- **Machine learning algorithms** - Malware detection problems are usually defined as classification or clustering problems. The choice of ML algorithms depends on whether training data is labeled or not, and this choice depends on data representation as well.
- **Evaluation of performance** - Various performance metrics are used to evaluate the success of ML algorithms. Evaluation results indicate how good results were achieved and which ML algorithm performed better with respect to a given dataset.

Data Pre-processing

- cleaning (fill in missing values, smooth noisy data,...), transformation (normalization, reduction)
- Min-max feature scaling:

$$x_i \to \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

  maps all values into the range [0,1].
- Standard score:

$$x_i \to \frac{x_i - \mu}{\sigma}$$

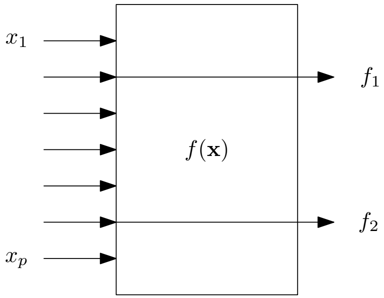
  if the population mean and population standard deviation are known.
- feature selection
- instance selection

Feature Selection and Extraction

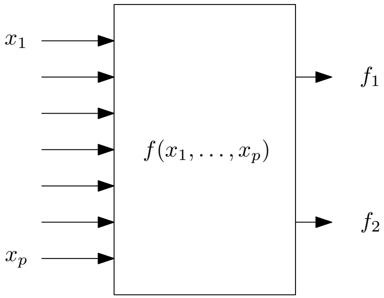
Reasons for doing this may be:

- easier and faster subsequent analysis
- improved classification performance through a more stable representation
- removal of redundant or irrelevant information

Feature Selection and Extraction



(a) feature selector



(b) feature extractor

Feature Selection and Extraction

- Both of these approaches require the optimisation of some criterion function, J
- feature selection:

$$J(\widetilde{X}_d) = \max_{X \in \mathcal{X}_d} J(X)$$

where $\mathcal{X}_d$ is a set of all possible subsets of size d.

- feature extraction:

$$J(\widetilde{A}) = \max_{A \in \mathcal{A}} J(A)$$

where $\mathcal{A}$ is a set of all possible transformations of the features.

Feature Selection and Extraction

- **Solution:** Evaluate the optimality criterion for all possible combinations of d variables selected from p and select that combination for which this criterion is a maximum

- The difficulty arises because the number of possible subsets is

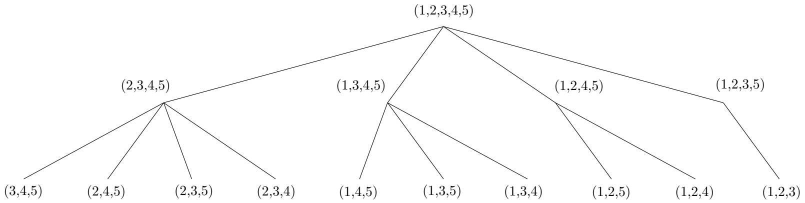
$$\binom{p}{d}$$

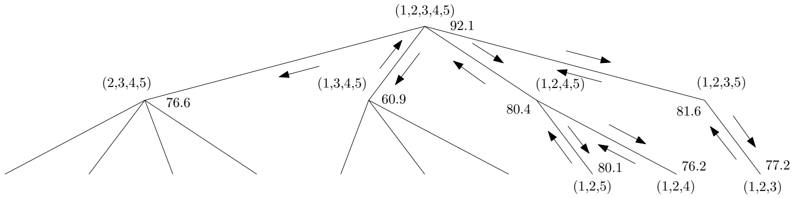
which can be very large even for moderate values of p and d. E.g. for $p = 25$ and $d = 10$ we get 3,268,760 combinations.

Optimal and Suboptimal Methods

- There are two basic strategies for feature subset selection:
  - Optimal methods: exhaustive search methods, accelerated search (branch and bound), Monte Carlo methods (simulated annealing, genetic algorithms) can lead to a globally optimal solution, but are computationally expensive
  - Suboptimal methods: the optimality of the above strategies is traded for computational efficiency

Branch and Bound Procedure





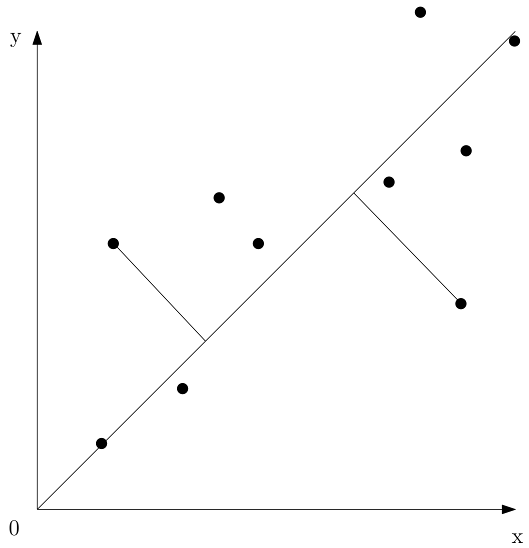
Branch and Bound Procedure - pseudocode

- Start at the top level and proceed down the rightmost branch, evaluating the cost function J at each node.
- If the value of J is less than the current threshold then abandon the search down that particular branch and backtrack to the previous branching node.
- Continue the search down the next rightmost branch.
- If, on the search of any branch the bottom level is reached, then if the value of J for this level is larger than the current threshold, the threshold is updated and backtracking begins.

Principal Components Analysis - Introduction

- Unsupervised feature extraction technique.
- The purpose of PCA is to derive new variables (in decreasing order of importance) that are linear combinations of the original variables and are uncorrelated
- Geometrically, PCA can be thought of as a rotation of the axes of the original coordinate system to a new set of orthogonal axes

PCA - Geometrical Derivation



ML algorithms

- **Supervised learning** - is where we have input variables (feature vector) x_i and an output variable c_j and you use an algorithm to learn the mapping function from the input to the output.
- **Unsupervised learning** - does not rely on any training phase and learns directly from unlabeled data.
- **Semi-supervised learning** - combines both labeled and unlabeled data for feeding statistical models to acquire knowledge (LLGC, Co-training).

Supervised Classification Techniques

- k-nearest Neighbors
- Naive Bayes
- Decision Tree
- Support Vector Machines
- Neural Networks
- and many others...

k-nearest Neighbors (KNN) Classifier

- let $T = \{(x_1, c_1), \ldots, (x_m, c_m)\}$ be the training set
- for a given a query point x_q, its unknown class c_q is to be determined
- select the set $T' = \{(x_1, c_1), \ldots, (x_k, c_k)\}$ of k nearest neighbors to the query point x_q
- $$c_q = \arg\max_c \sum_{(x_i, c_i) \in T'} w_i \cdot \delta(c, c_i),$$

  where $\delta(c, c_i)$ takes a value of one if $c = c_i$ and zero otherwise.
- Weighted k-nearest neighbors classifier takes into account distances between the query point and k-nearest neighbors:

$$w_i = \begin{cases} \frac{d_k - d_i}{d_k - d_1} & \text{if } d_k \neq d_1 \\ 1 & \text{otherwise} \end{cases}$$

KNN - Distance Function in Metric Space

- Numerical Data: Euclidean distance

$$\mathcal{D}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Nominal Data: Value Difference Metric (VDM)

$$\mathrm{VDM}_a(x, y) = \sum_{c=1}^{C} \left| \frac{n_{a,x,c}}{n_{a,x}} - \frac{n_{a,y,c}}{n_{a,y}} \right|^q$$

where

- C is the number of classes,
- $n_{a,x,c}$ is the number of instances in the training set $\mathcal{T}$ which have value x for attribute a and the instance belongs to class c,
- $n_{a,x}$ is the number of instances in $\mathcal{T}$ that have value x for attribute a.

KNN - Distance Function in Metric Space

$$\mathrm{HVDM}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)}$$

where distance function $d_a(x, y)$ for attribute a is as follows:

$$d_a(x,y) = \begin{cases} 1 & \text{if } x \text{ or } y \text{ is unknown} \\ \mathrm{NORM\_VDM}_a(x,y) & \text{if } a \text{ is nominal} \\ \mathrm{NORM\_DIFF}_a(x,y) & \text{if } a \text{ is linear} \end{cases}$$

Functions $\mathrm{NORM\_VDM}_a(x, y)$ and $\mathrm{NORM\_DIFF}_a(x, y)$ are defined as:

$$\mathrm{NORM\_VDM}_a(x,y) = \sum_{c=1}^{C} \left| \frac{n_{a,x,c}}{n_{a,x}} - \frac{n_{a,y,c}}{n_{a,y}} \right|$$

$$\mathrm{NORM\_DIFF}_a(x,y) = \frac{|x-y|}{4\sigma_a}$$

KNN - Distance Function in Metric Space

- Mahalanobis distance is a generalization of Euclidean distance.
- Mahalanobis distance for two n-dimensional feature vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^{\top} \mathbf{M} (\mathbf{x} - \mathbf{y})}$$

  where $\mathbf{M}$ is a positive semidefinite matrix.
- Positive semidefinite matrix $\mathbf{M}$ can always be decomposed as $\mathbf{M} = \mathbf{L}^{\top}\mathbf{L}$.
- Mahalanobis distance can be expressed as follows:

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{y}) = d_{\mathbf{L}}(\mathbf{x}, \mathbf{y}) = \|\mathbf{L}(\mathbf{x} - \mathbf{y})\|_2$$

KNN - Distance Function in Metric Space

- Mahalanobis distance of two points $\mathbf{x}, \mathbf{y}$ from the original space corresponds to the Euclidean distance between transformed points $\mathbf{x'} = \mathbf{L}\mathbf{x}, \mathbf{y'} = \mathbf{L}\mathbf{y}$ defined as follows:

$$d_{\mathbf{L}}(\mathbf{x}, \mathbf{y}) = \|\mathbf{L}(\mathbf{x} - \mathbf{y})\|_2 = \sqrt{(\mathbf{x'} - \mathbf{y'})^\top (\mathbf{x'} - \mathbf{y'})}$$

- The projection can be used for supervised dimensionality reduction. Considering the matrix $\mathbf{L} \in \mathbb{R}^{d \times n}$ with $d < n$ then the dimension of transformed sample $\mathbf{x}' = \mathbf{L}\mathbf{x}$ is reduced to d.

Naive Bayes

- Naive Bayes classifier for the binary (i.e. 2 classes) classification problem works as follows.
- Naive Bayes classifier is a probability algorithm based on Bayes' theorem, which predicts the class with the highest a posteriori probability.
- Bayes' theorem states that:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \quad (P(B) \neq 0).$$

- Let's have a set of two classes $\{\mathcal{C}, \mathcal{M}\}$, where $\mathcal{C}$ denotes the class of benign samples and $\mathcal{M}$ denotes the malware class.

Naive Bayes

- Let's have a training dataset and a test element (with unknown label) that is represented by a feature vector $x = (x_1, \dots, x_n)$.
- Let $P(\mathcal{M}|x)$ denote the probability that the sample is malware given the feature vector x, which represents the sample. Similarly, let $P(\mathcal{C}|x)$ denote the probability that the sample is benign given the feature vector x. The Naive Bayes classification rule is defined by

$$\begin{aligned} \text{If } P(\mathcal{M}|x) &< P(\mathcal{C}|x), \ x \text{ is classified as benign} \\ \text{If } P(\mathcal{M}|x) &> P(\mathcal{C}|x), \ x \text{ is classified as malware} \end{aligned} \quad (1)$$

Naive Bayes

- a posteriori probabilities $P(C|x)$ can be expressed as a priori probabilities and $P(x|C)$ probabilities using the Bayesian theorem:

$$P(C|x) = \frac{P(x|C)\ P(C)}{P(x)} \tag{2}$$

- Assuming that the values of the features are conditionally independent of each other, the equation (2) can be expressed as

$$P(C|x) = \frac{\prod_{i=1}^{n} P(x_i|C)\ P(C)}{P(x)} \tag{3}$$

Naive Bayes

- The probabilities $P(x_i|C)$ can be estimated from the training set by calculating the feature values for each class.
- More specifically, the probability $P(x_i = h|C)$ is represented as the number of elements of class C in the training set with the value h for the feature x_i divided by the number of elements of class C in the training set.
- The output of the classifier is the class C' with the highest probability:

$$C' = \arg\max_{C} \left(P(C) \prod_{i=1}^{n} P(x_i|C) \right) \quad (4)$$