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

Faculty of Information Technology
Czech Technical University in Prague

December 1, 2020

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.
- **Malware category detection** - Malware can be categorized according to their behaviours and objectives. AV vendors have not still agreed upon a standardized taxonomy of malware categories, effectively recognising the categories of a sample 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 the available knowledge base the families that a given malicious sample likely belongs to.
  - **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

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

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:

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

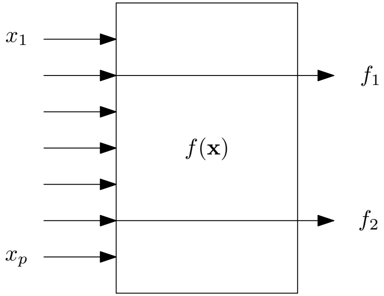
  if the population mean and population standard deviation are known.
- feature selection
- instance selection

Feature Selection and Extraction

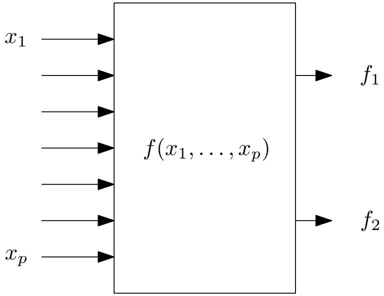
Reasons for doing this may be:

- 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

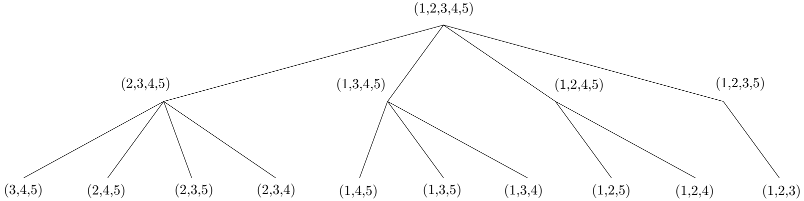


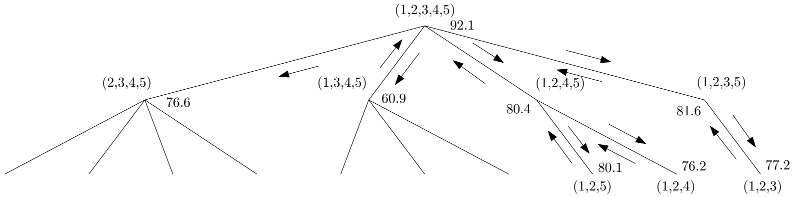
(b) feature extractor

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





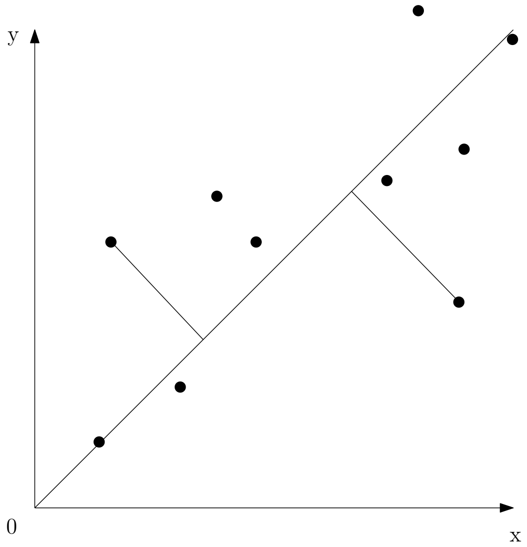
Branch and Bound Procedure - pseudocode

- 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 learn 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 Classifier

- let $T = \{(x_1, c_1), \ldots, (x_m, c_m)\}$ be the training set
- given a query point x_q, its unknown class c_q is determined
- select the set $T' = \{(x_1, c_1), \ldots, (x_k, c_k)\}$ of k nearest neighbors to the query point x_q
- $$c_q = \arg\max_c \sum_{(x_i, c_i) \in T'} w_i \cdot \delta(c, c_i),$$
- $$w_i = \begin{cases} \frac{d_k - d_i}{d_k - d_1} & \text{if } d_k \neq d_1 \\ 1 & \text{otherwise} \end{cases}$$

Naive Bayes

- classification rule is stated as

$$\text{If } P(c_i|x) < P(c_j|x), i \neq j \text{ assign } x \text{ to class } c_j$$

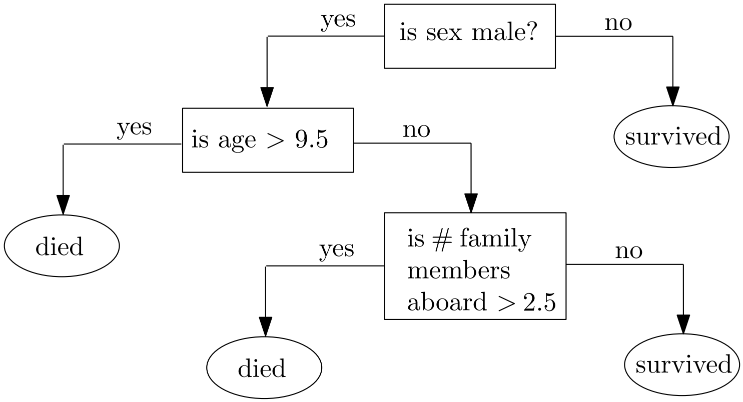
- *a posteriori* probabilities $P(c|x)$ may be expressed in terms of the *a priori* probabilities and the $P(x|c)$ probabilities using Bayes' theorem as

$$P(c|x) = \frac{P(x|c)\ P(c)}{P(x)}$$

- Assuming that the values of the features are conditionally independent on one another, the equation may be expressed as

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

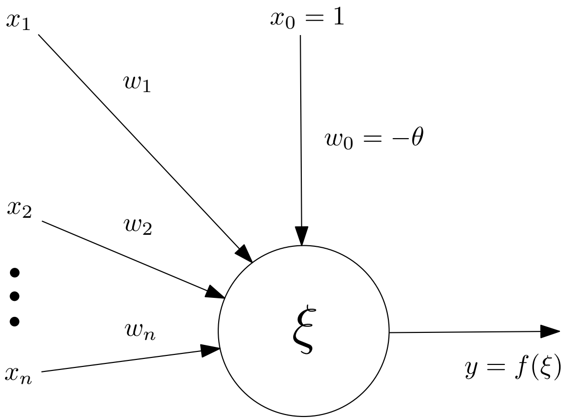
Decision Tree



Decision Tree - Basics

- goal: achieve perfect classification with minimal number of decisions
- each level splits the data according to different attributes
- 1. select attribute or value along dimension that gives "the best" split
  2. create child nodes based on the split
  3. recurse on each child using child data until a stopping criterion is reached
- finding optimal tree for arbitrary data is NP-hard

Single Neuron (Perceptron)

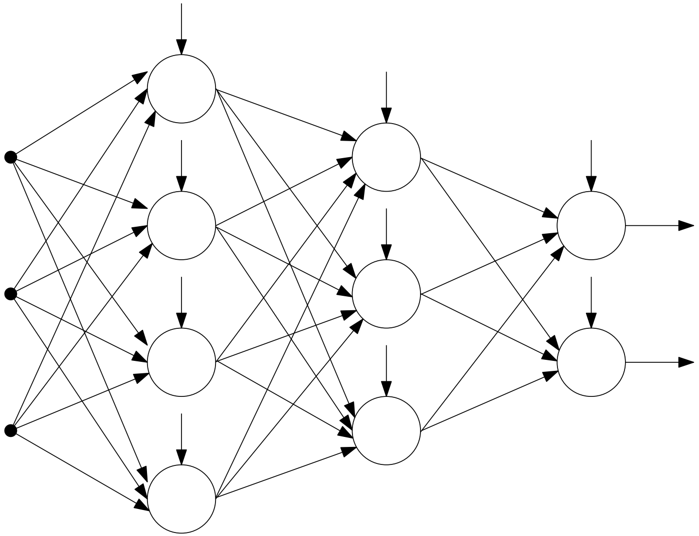


Single Neuron - Details

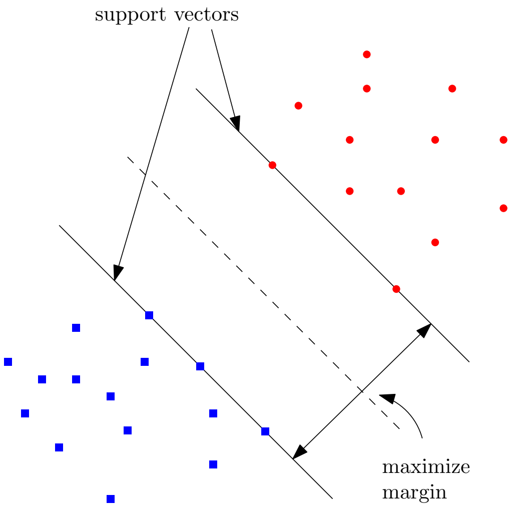
$$\xi = \sum_{i=1}^{n} w_i x_i - \theta = \sum_{i=0}^{n} w_i x_i$$ action potential

$$f(\xi) = \frac{1}{1+e^{-\lambda\xi}}$$ activation function

Neural Network



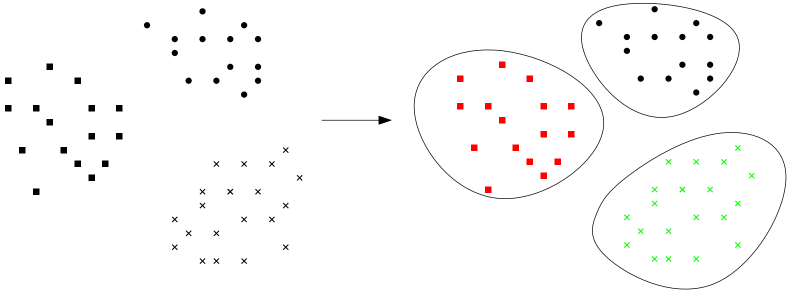
Support Vector Machines



Support Vector Machines - Basics

- training set $\{\mathbf{x_i}, i = 1, \ldots, n\}$, with corresponding labels $y = \pm 1$.
- optimal hyperplane $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ for linearly separable patterns
- decision rule
$$y_i(\mathbf{w}^T\mathbf{x} + w_0) > 0 \text{ for all } i$$
- SVMs maximize the margin around the separating hyperplane

Clustering Algorithms



K-means pseudocode

```
1 initialize cluster centroids $\mu_1, \ldots, \mu_k \in \mathrm{R}^n$ randomly
2 repeat until convergence {
      for every $i$, set
          $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2$$
      for every $j$, set
          $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}$$
  }
```

Model Evaluation

- Model evaluation consists of two steps:
  1. Model selection
     - **Training** - estimate the parameters of the model.
     - **Validation** - select the model with the best results.
  2. **Testing** - evaluate the selected model in order to assess the real performance of the model on new unseen data
- We split data into training set and a testing set. The training set is used to estimate the parameters of the model (training) and also for model selection (validation).
- The testing set is then used to assess the performance of the model.

Evaluation metrics

Evaluation metrics are used to measure the performance of the classification models. In a binary classification problem, the following classical quantities are employed:

- **True Positive** (TP) represents the number of malicious samples classified as malware
- **True Negative** (TN) represents the number of benign samples classified as benign
- **False Positive** (FP) represents the number of benign samples classified as malware
- **False Negative** (FN) represents the number of malicious samples classified as benign

Evaluation metrics

- The most intuitive and commonly used evaluation measure in Machine Learning is the Accuracy (ACC):

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{1}$$

- It is defined on a given test set as the percentage of correctly classified instances.
- Alternative for accuracy is an error rate, it is defined as

$$\mathrm{ERR} = 1 - \mathrm{ACC}. \tag{2}$$

Evaluation metrics

- True Positive Rate (TPR) (or recall), is defined as:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3}$$

- TPR is the percentage of truly malicious samples that were classified as malware.
- False Positive Rate (FPR) and is defined as follow:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} \tag{4}$$

- FPR is the percentage of benign samples that were wrongly classified as malware.

Evaluation metrics

- Another metric is precision, and it is defined as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

- Precision is the percentage of samples classified as malware that are truly malware.
- F_1-measure is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

Overfitting

- A training set is not a representative set of a population and it usually contains some noise elements.
- A model that captures structure in the data together with the noise is "over-fitted".
- Classifiers have then high performance on training set, but low generalization ability on testing set.

Challenges (for AV vendors or attackers)

- **Minimizing of reaction time** - The main problem with anti-virus software is that a new malware may spread quickly and infect thousands of computers worldwide before any AV software makers can isolate and analyze it and issue an update, and certainly before most users can download, install, and run this update.
- **Interpretability of the models** - Most of the models used at the present time are treated as a black box. This black box is given an input X and it produces an output Y through a sequence of operations hardly understandable to a human. This could pose a problem in cybersecurity applications when a false alarm occurs as analysts would like to understand why it happened. The interpretability of the model determines how easily the analysts can manage and assess the quality and correct the operation of a given model.

Challenges (for AV vendors or attackers)

- **Operation set** - Opcodes, instructions, APIs and system calls (referred as operations) are the most used and powerful features employed for malware analysis, as they allow to directly and accurately model sample behaviour. To reduce complexity and required computational power, only a subset of all the available operations is considered.
- **Adversarial learning** - is a technique employed to attempt to fool machine learning by automatically crafting adversarial examples. That is, samples with small, intentional feature perturbations that cause a machine learning model to make an incorrect prediction.

Challenges (for AV vendors or attackers)

- **Anti-analysis techniques** - Malware developers want to avoid their samples to be analysed, so they devise and refine several anti-analysis techniques that are effective in hindering the reverse engineering of executables. Static analysis is commonly prevented by rendering sample binary and resources unreadable through obfuscation, packing or encryption. Such a kind of anti-analysis techniques can be overcome by using dynamic analysis to make the sample unveil hidden information and load them in memory, where they can then be extracted by creating a dump.