Attacking the malware with AI

Where the finest concepts of Data Science & Cybersecurity meet

Introduction

- Master’s in Cybersecurity @ *Georgia Institute of Technology* (USA)
- Security Engineer @ *Trade Republic* (Berlin)
- Cloud Security & DevSecOps
- Artificial Intelligence & Privacy

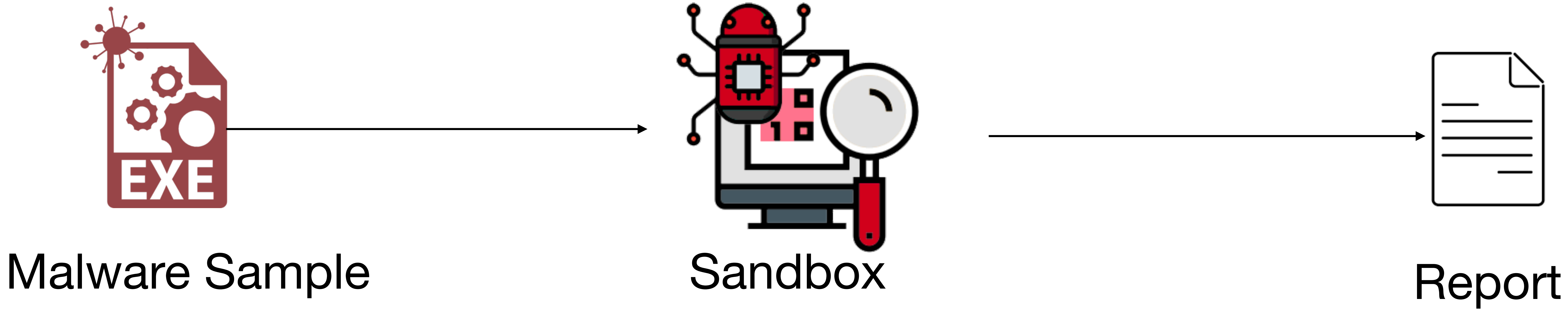
Agenda

1. Malware Analysis: Sandboxing - How does a Sandbox work?
2. Elements of Machine Learning (Classification & Clustering)
3. Malheur Framework - Explained step by step
4. Q/A

What is a sandbox?

And how does it work?

- A sandbox is an **isolated environment** in which malware can be safely executed, in order to **study & monitor** its behaviour

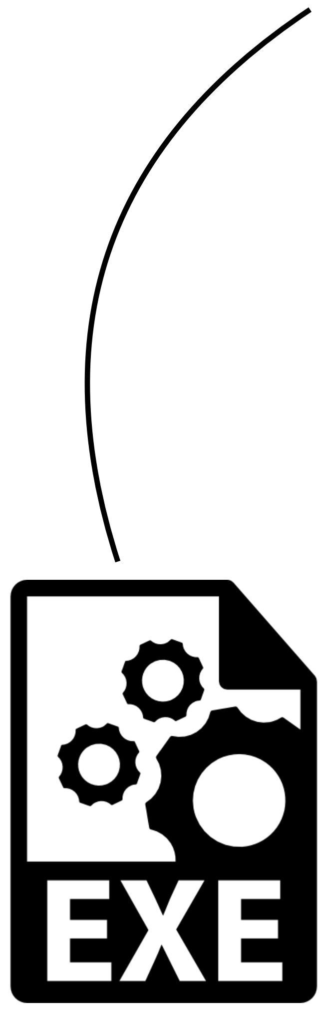


DDoS

What is classification?

Supervised Learning

Classification is the task of identifying the **category** (a.k.a the class) on which an **observation** falls into

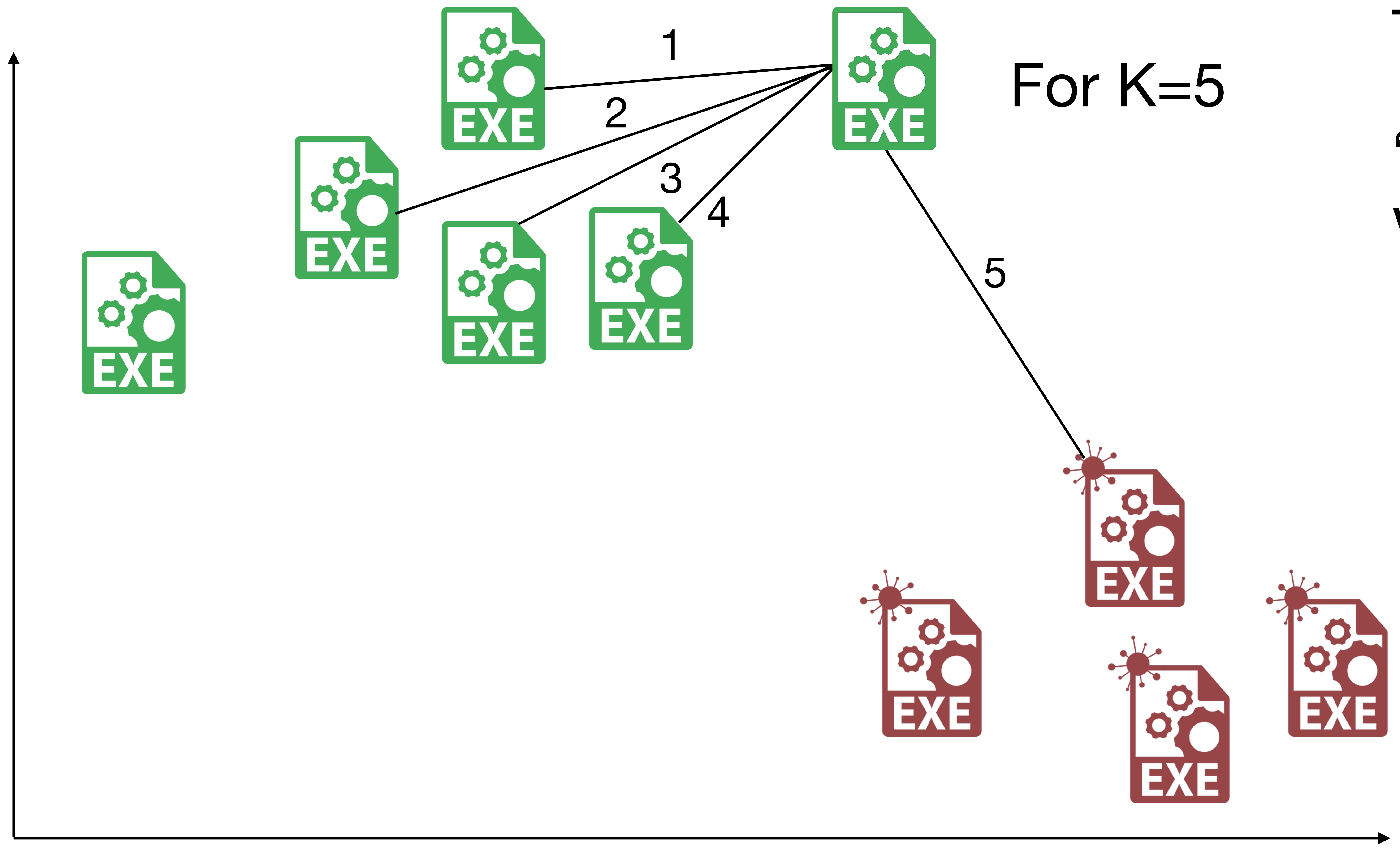


Benign

Malware

What is classification?

Supervised Learning



There's a folk saying…

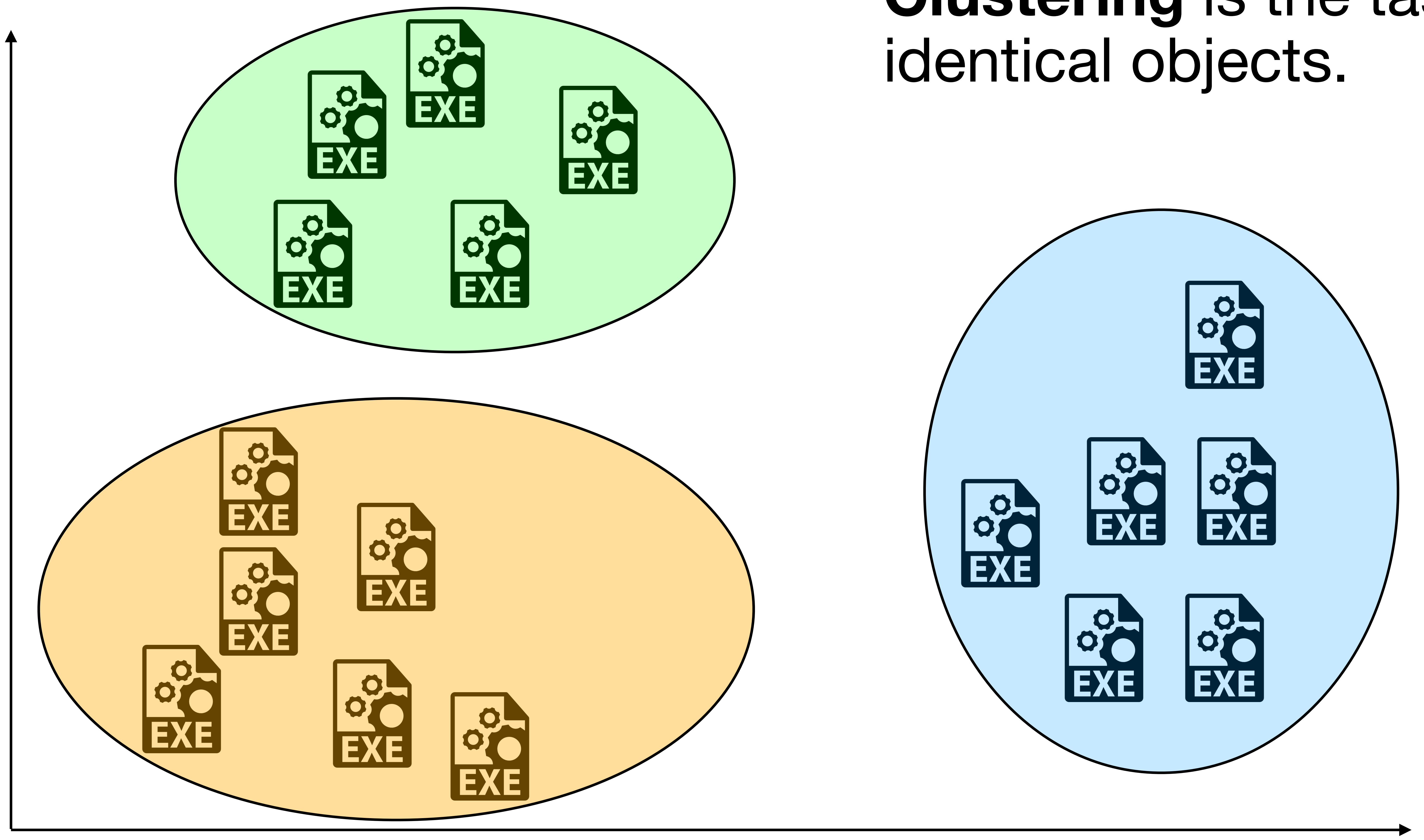
"Show me your friends and I will tell you who you are"

There is also an algorithm based on this folk wisdom:

K-Nearest ~~Friends~~ **Neighbours**

What is clustering?

Unsupervised learning



Clustering is the task of forming groups of identical objects.

Malheur Framework (Rieck et al., 2011)

"More than 450,000 new malicious software and potentially unwanted applications (PUA) are registered every day"

AV-Test Institute, 2022

Malheur is an open-source framework for Malware Analysis with Machine Learning.

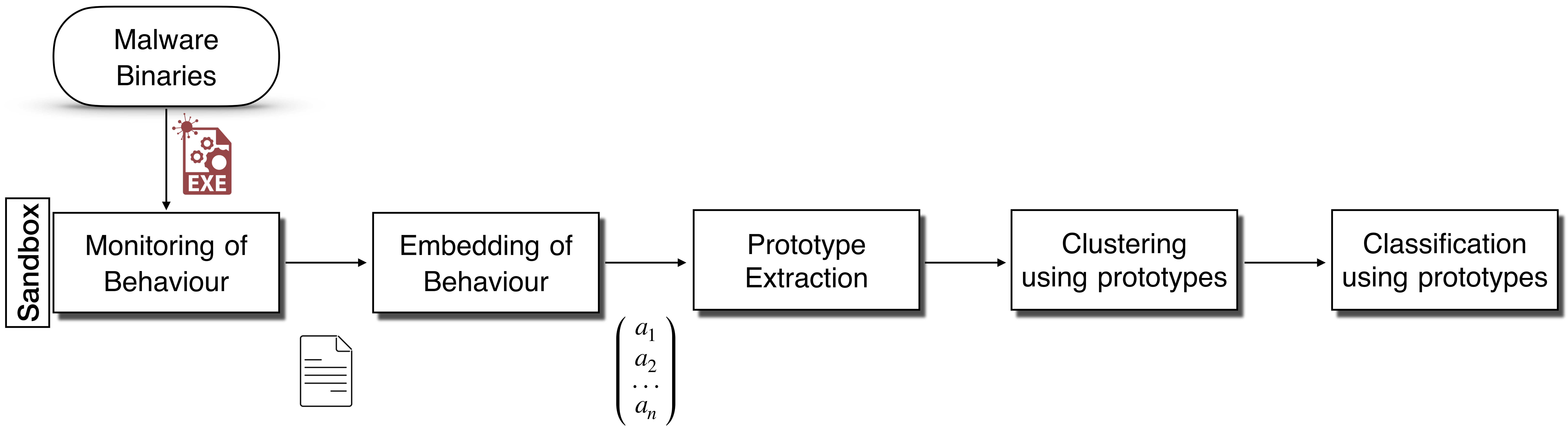
It is used to:

1. Automatically discover novel classes of malware

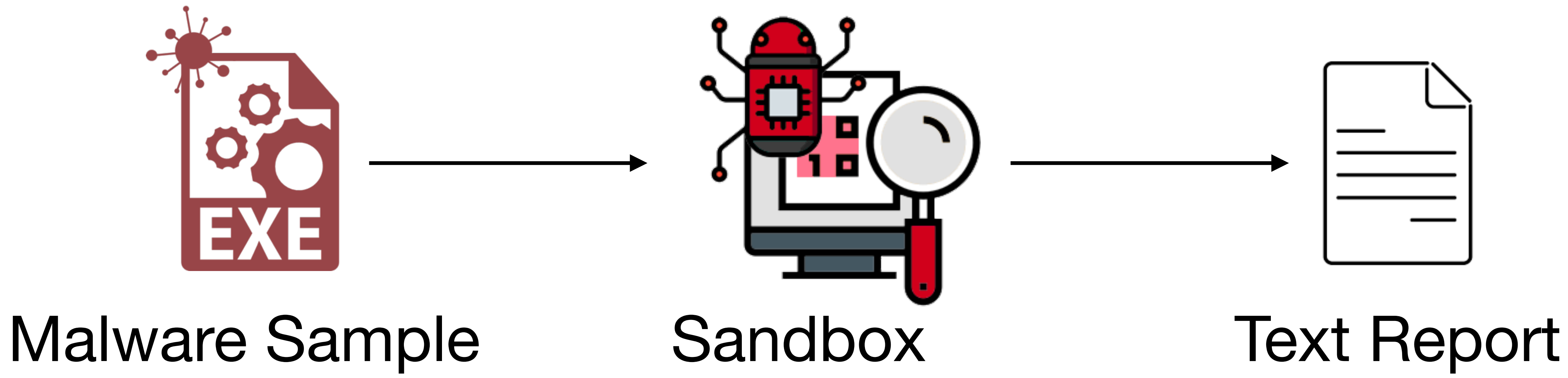
2. Classify unknown malware to known classes

Malheur Framework Simplified

Overview



Step 1: Monitoring of the behaviour




```
<load_dll filename="C:\WINDOWS\system32\kernel32.dll"
<get_system_time apifunction=“GetLocalTime”/>
…
…
<copy_file filetype="file" srcfile="c:\\malware.exe" dstfile="C:
\system32\csrss.exe"/>
<set_value key="HKEY_LOCAL_MACHINE\CurrentVersion\Run" value="UpDaTer" data=“C:
\system32\csrss.exe”/>
…
…
<check_for_debugger apifunction=“IsDebuggerPresent”/>
<load_dll filename="C:\WINDOWS\system32\UxTheme.dll"/>
…
…
```

Step 2: Embedding of the behaviour

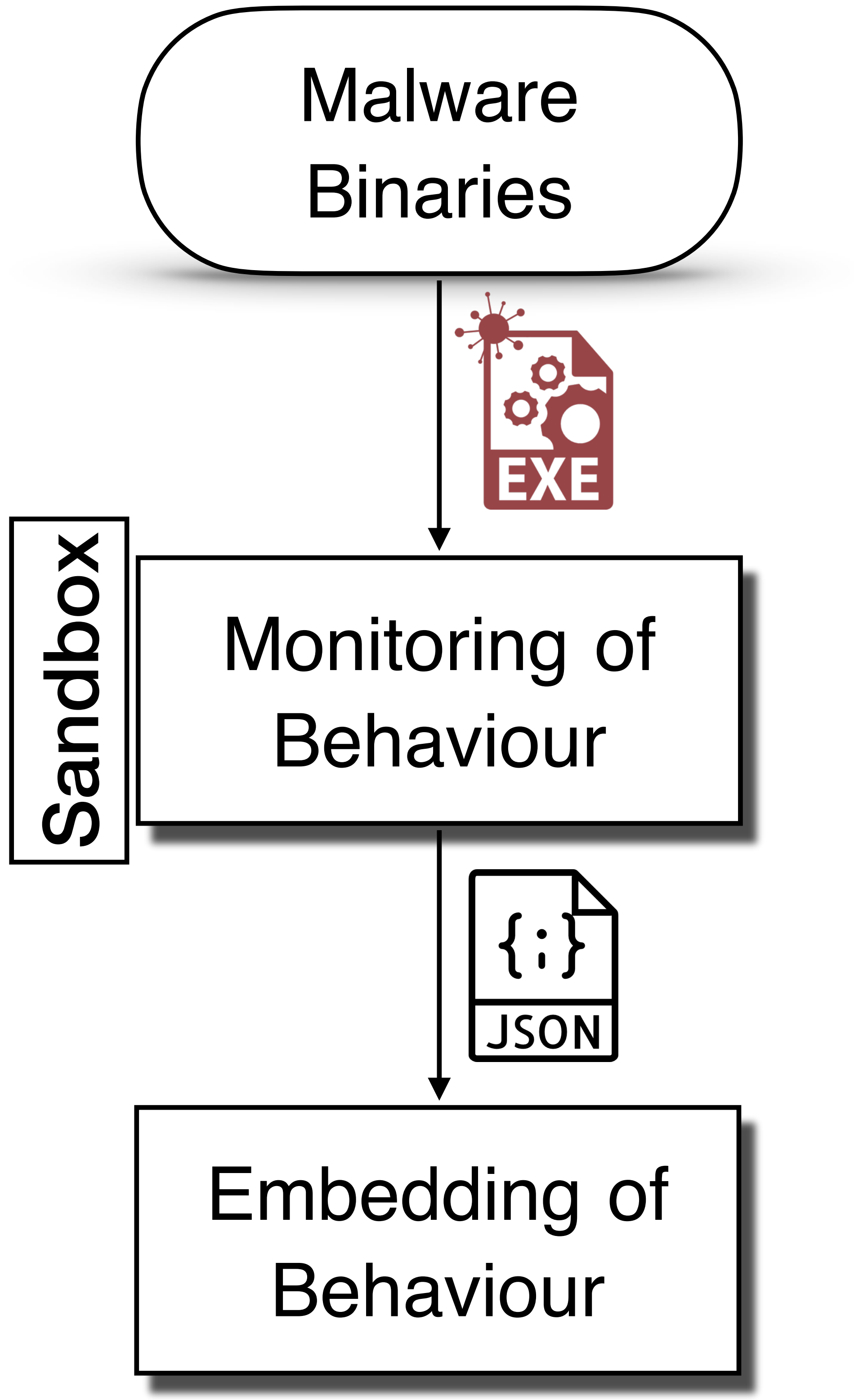
Instruction Q-grams

- Behaviour is often manifested as a **sequence of instructions,** which are formally known as **q-grams**
- For example, a 2-gram:

```
1. <copy_file filetype="file" srcfile="c:\\malware.exe" dstfile="C:
\system32\csrss.exe"/>
2. <set_value key="HKEY_LOCAL_MACHINE\CurrentVersion\Run" value="UpDaTer" data=“C:
\system32\csrss.exe”/>
```

```
1. <check_for_debugger apifunction=“IsDebuggerPresent”/>
2. <load_dll filename="C:\WINDOWS\system32\UxTheme.dll"/>
```

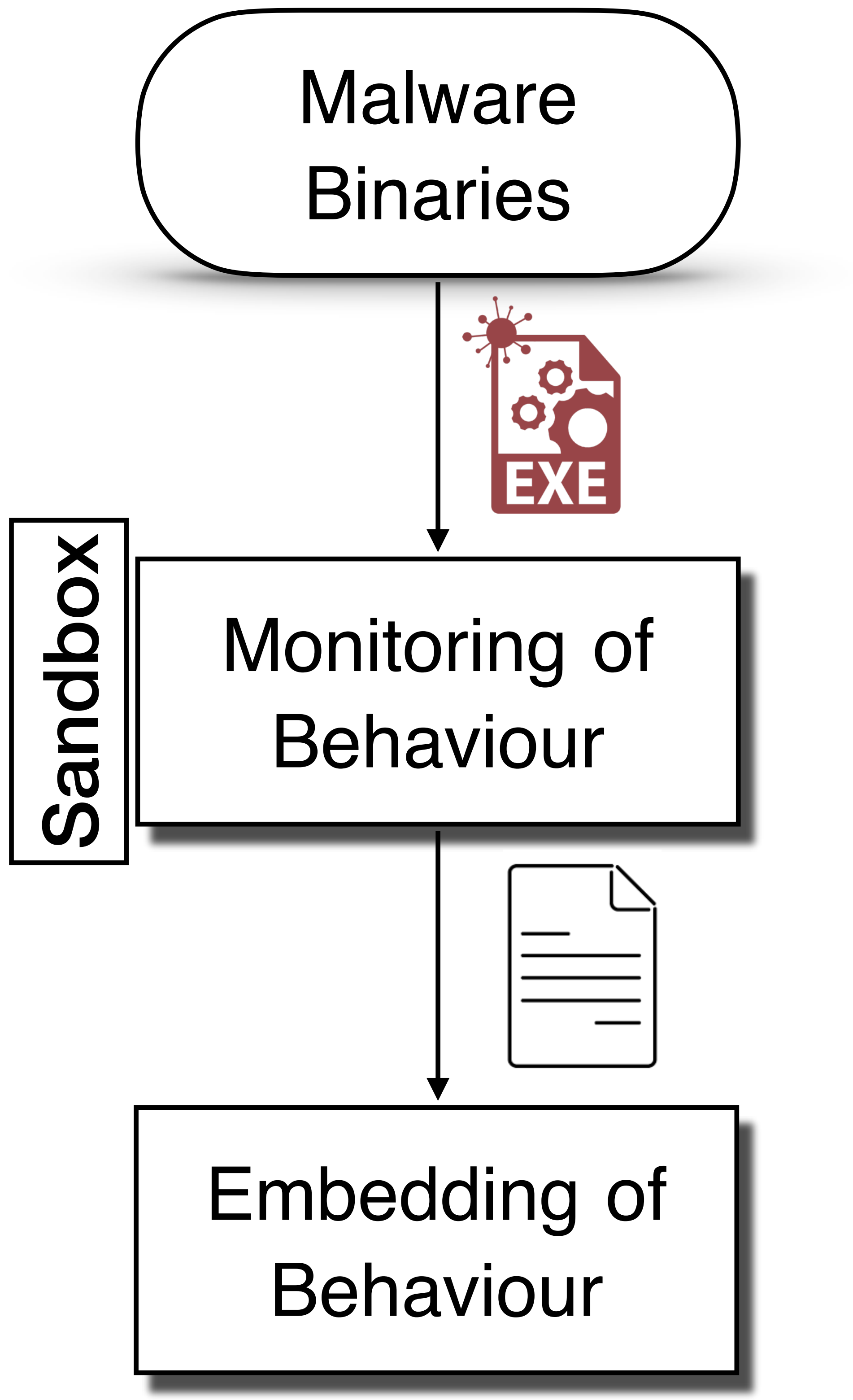
Step 2: Embedding of the behaviour



The embedding of reports in vector spaces enables us to express the similarity of behaviour *geometrically*

We model the text files into mathematical vectors using an ***embedding function***

Step 2: Embedding of the behaviour

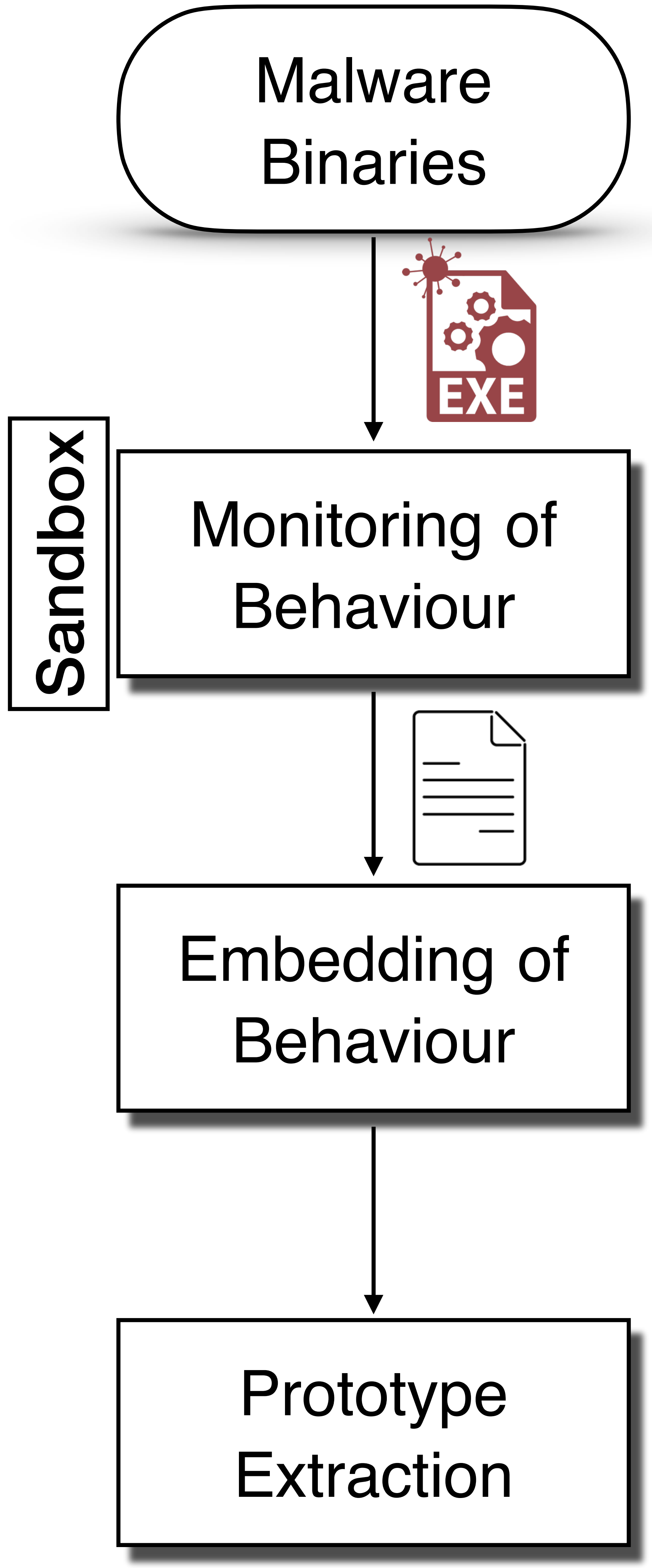


$$\mathcal{S} = \{(a_1, \ldots, a_q) \mid a_i \in \mathcal{A} \text{ with } 1 \leq i \leq q\},$$

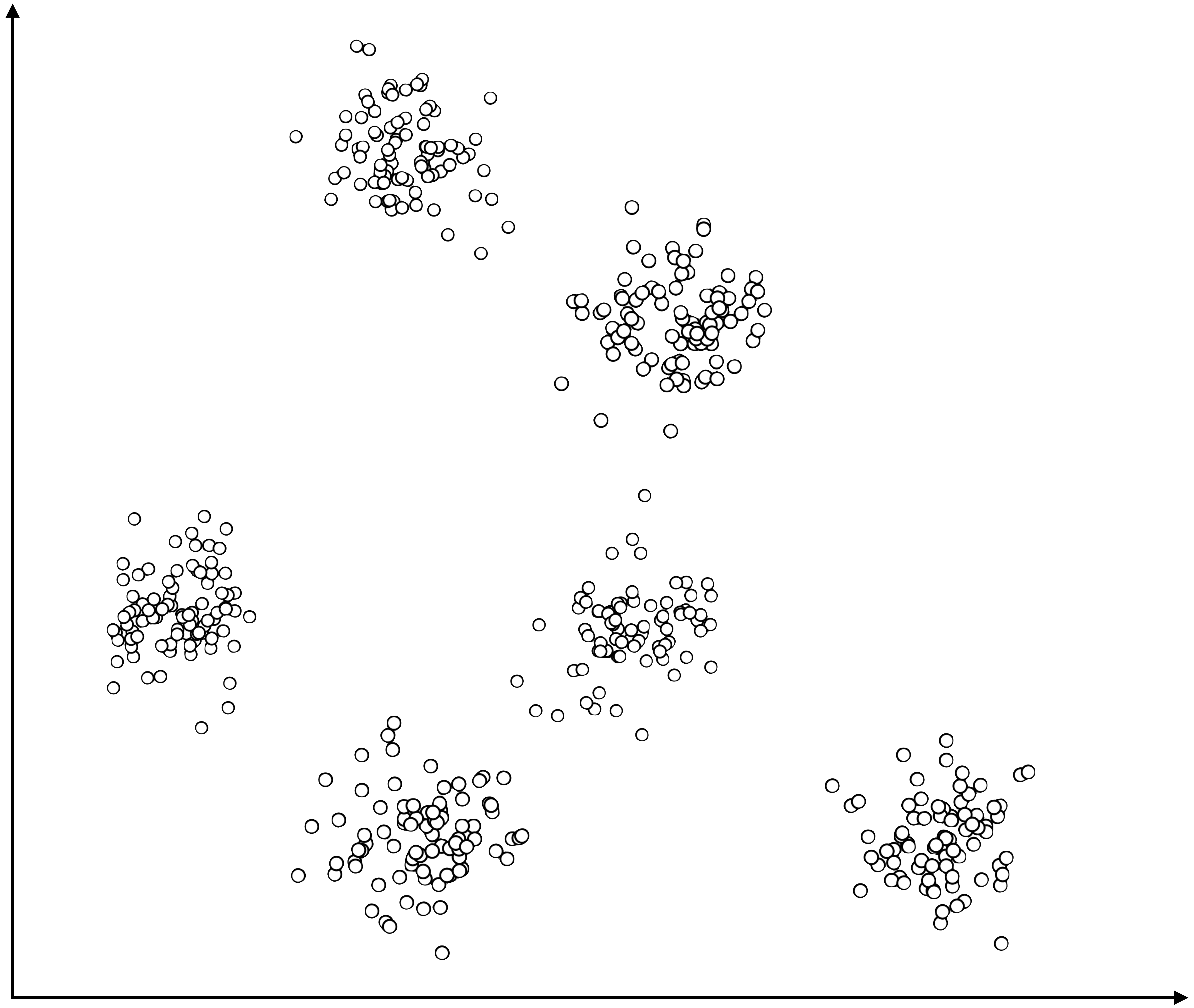
$$\phi(x) = (\phi_s(x))_{s \in S} \quad \text{with} \quad \phi_s(x) = \begin{cases} 1 & \text{if report x contains q-grams S,} \\ 0 & otherwise \end{cases}$$

$$\phi('A_1 \quad A_2 \quad A_1 \quad A_2') \rightarrow \begin{array}{cc} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} & \begin{array}{l} 'A_1 \quad A_1' \\ 'A_1 \quad A_2' \\ 'A_2 \quad A_1' \\ 'A_2 \quad A_2' \end{array} \end{array}$$

Step 2: Embedding of Behaviour

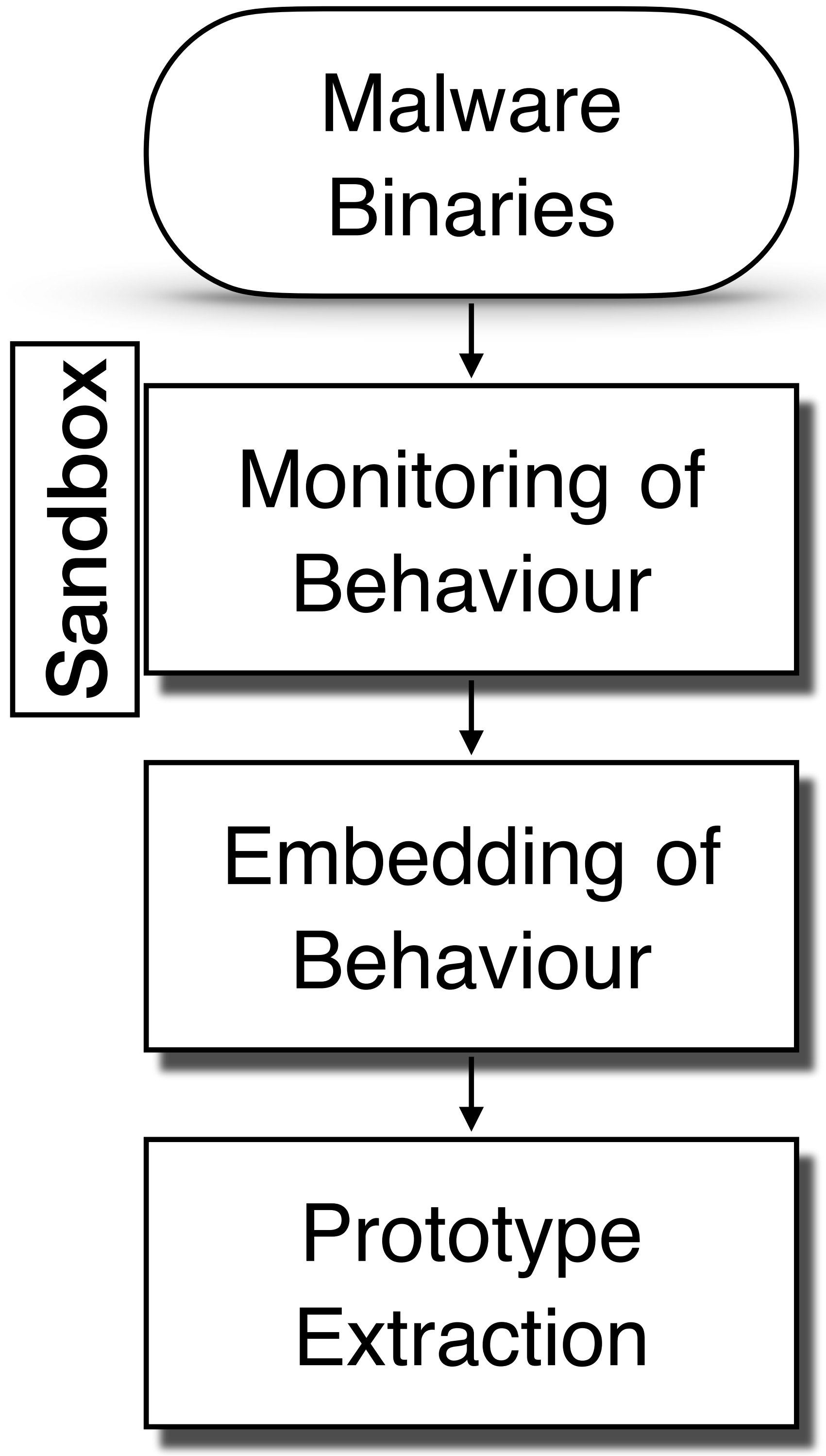


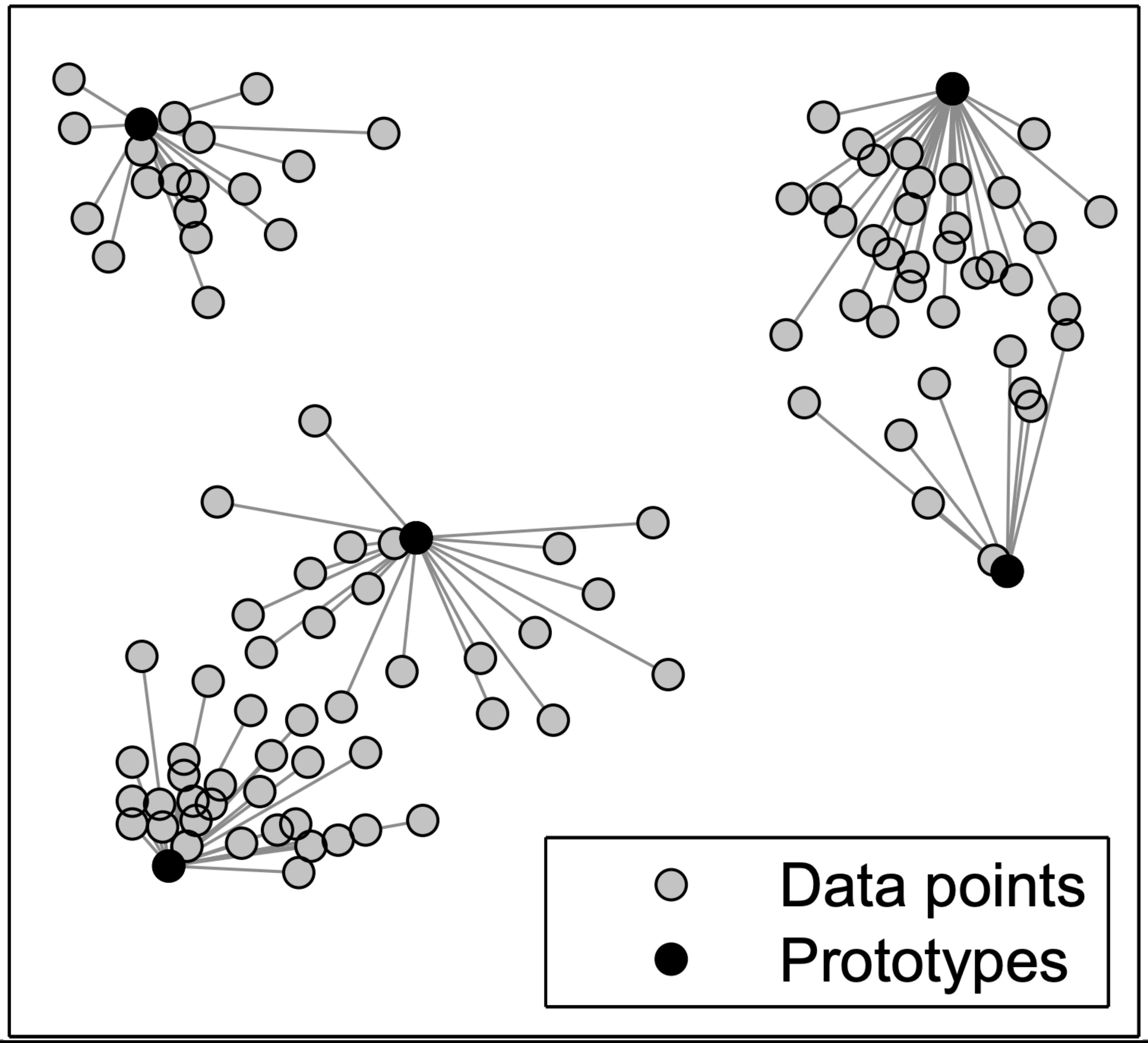
- Identical reports *"form dense clouds in the vector space"*



Computational Complexity

Step 3: Prototype Extraction





Algorithm 1 Prototype extraction

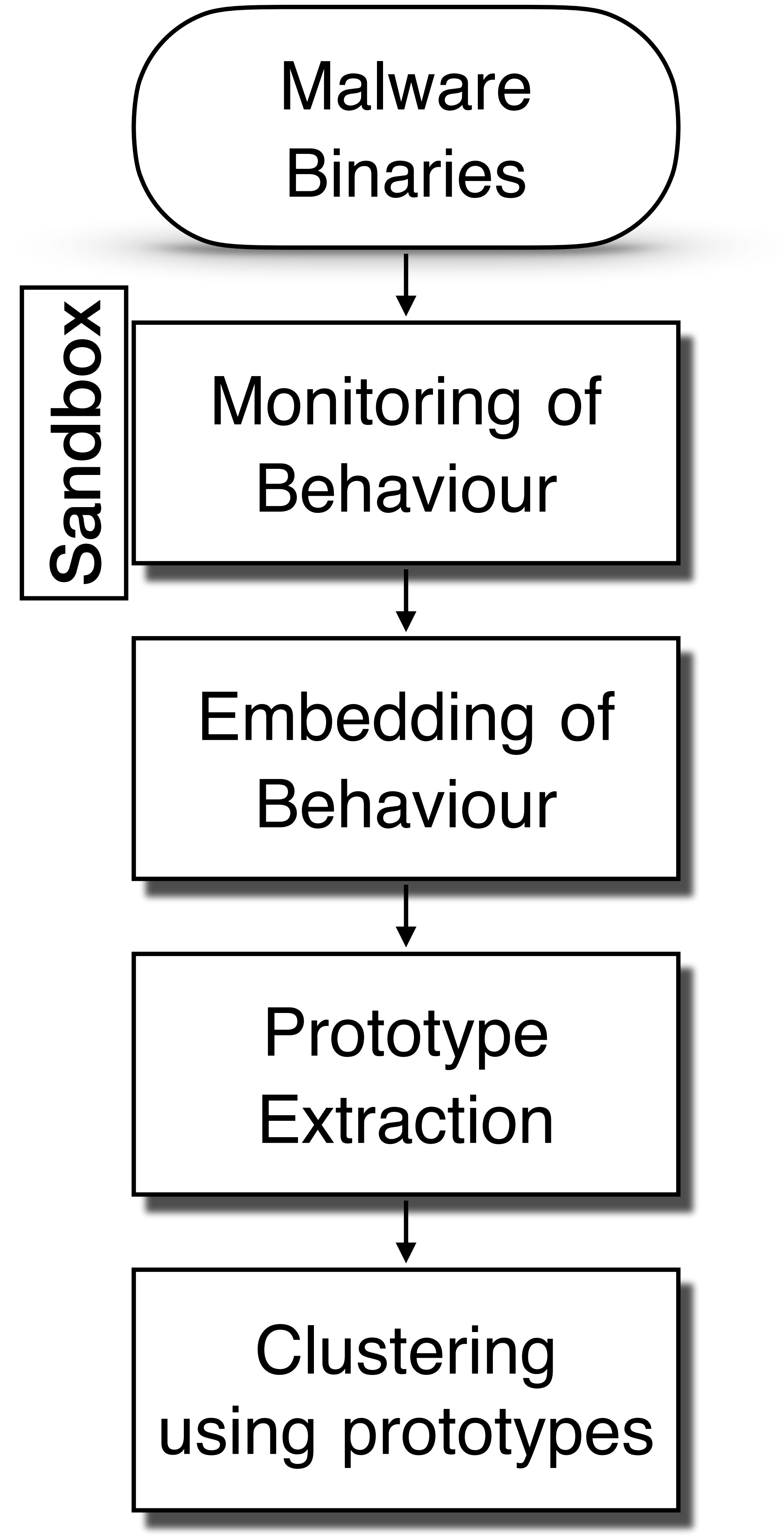
$\mathit{prototypes} \leftarrow \emptyset$
$\mathit{distance}[x] \leftarrow \infty$ for all $x \in \mathit{reports}$
while $\max(\mathit{distance}) > d_p$ **do**
  choose z such that $\mathit{distance}[z] = \max(\mathit{distance})$
  for $x \in \mathit{reports}$ and $x \neq z$ **do**
    if $\mathit{distance}[x] > ||\hat{\varphi}(x) - \hat{\varphi}(z)||$ **then**
      $\mathit{distance}[x] \leftarrow ||\hat{\varphi}(x) - \hat{\varphi}(z)||$
  add z to $\mathit{prototypes}$

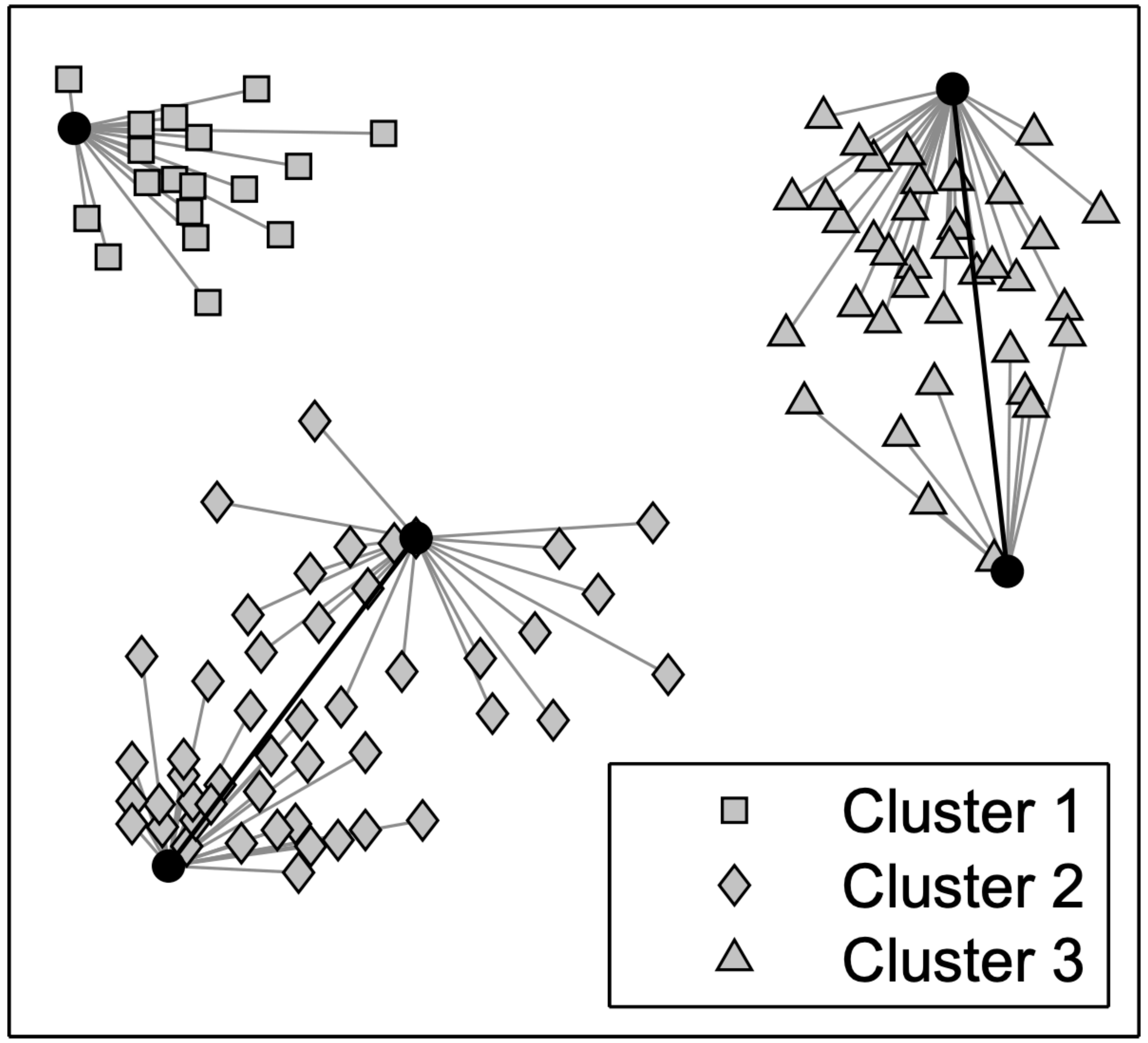
$O(k \cdot n)$

```
k: # of prototypes
n: # of reports
```

(Rieck et al., 2011)

Step 4: Clustering using Prototypes





"Once a clustering has been determined on the prototypes, it is propagated to the original reports"

Algorithm 2 Clustering using prototypes $O(k^2 \cdot log(k) + n)$

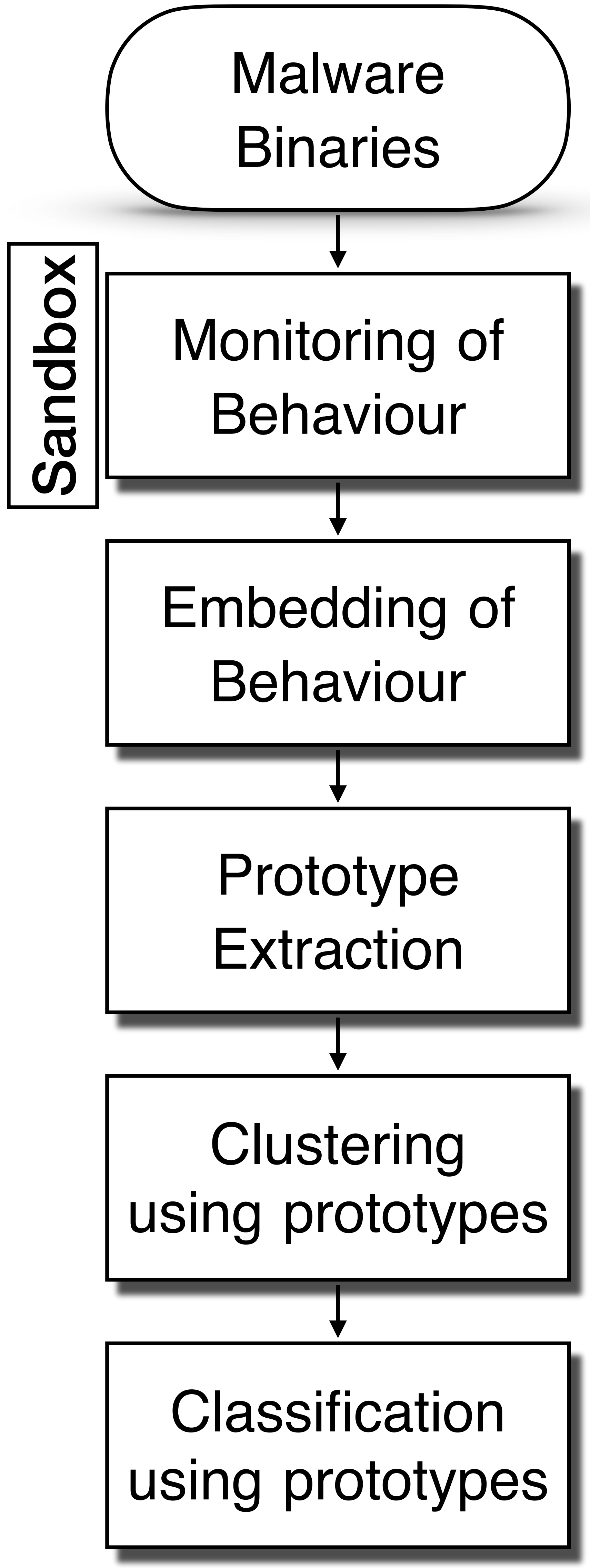
for $z, z' \in$ *prototypes* **do**
  $distance[z, z'] \leftarrow ||\hat{\varphi}(z) - \hat{\varphi}(z')||$
while $\min(distance) < d_c$ **do**
  merge clusters z, z' with minimum $distance[z, z']$
  update *distance* using complete linkage
for $x \in$ *reports* **do**
  $z \leftarrow$ nearest prototype to x
  assign x to cluster of z
reject clusters with less than m members

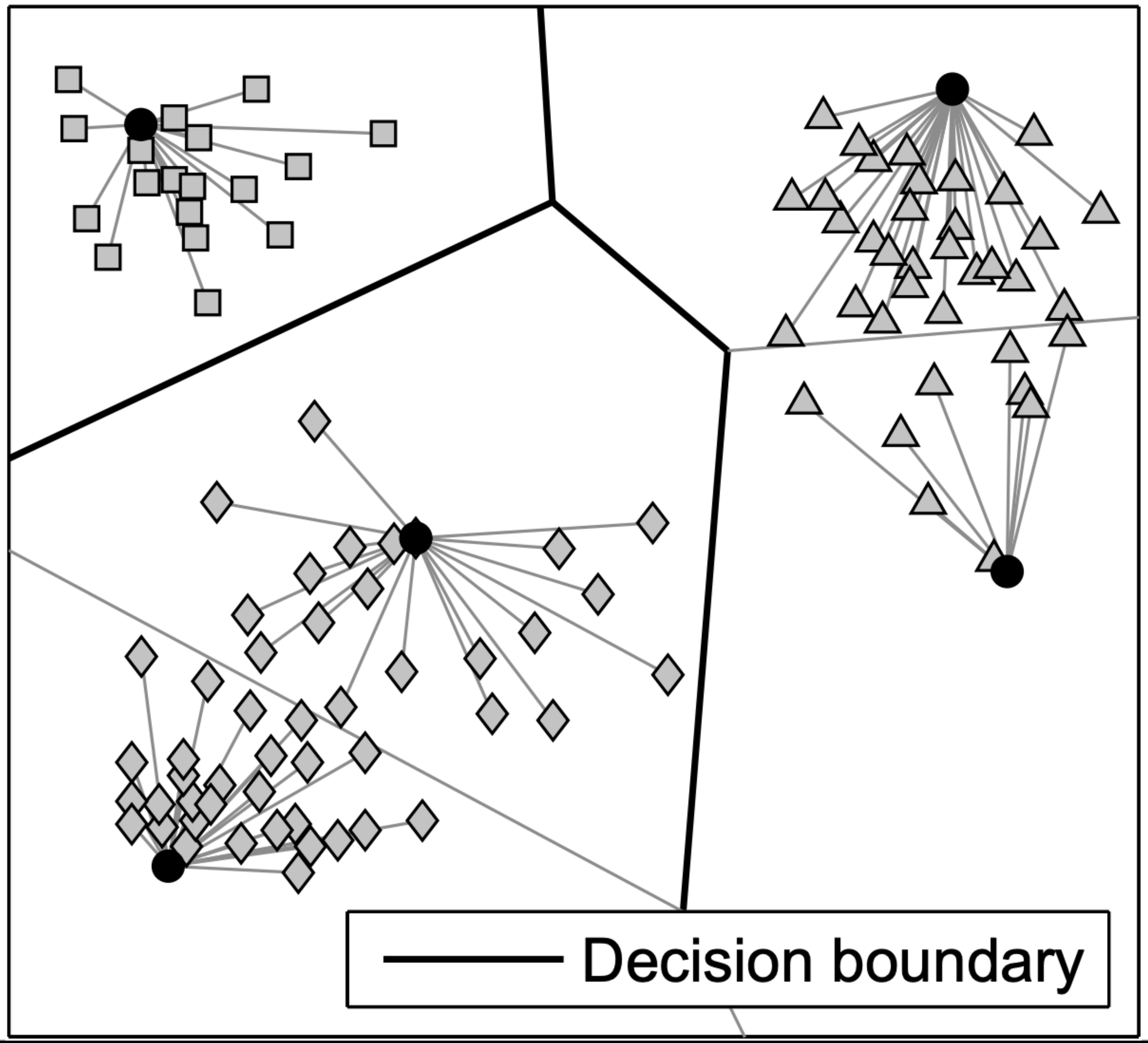
```
k: # of prototypes
n: # of reports
```

"clusters with fewer than **m** members are rejected and kept for later incremental analysis"

(Rieck et al., 2011)

Step 5: Classification using Prototypes





Algorithm 3 Classification using prototypes $O(k \cdot n)$

```
for x ∈ reports do
    z ← nearest prototype to x
    if ||φ̂(z) − φ̂(x)|| > d_r then
        reject x as unknown class
    else
        assign x to cluster of z
```

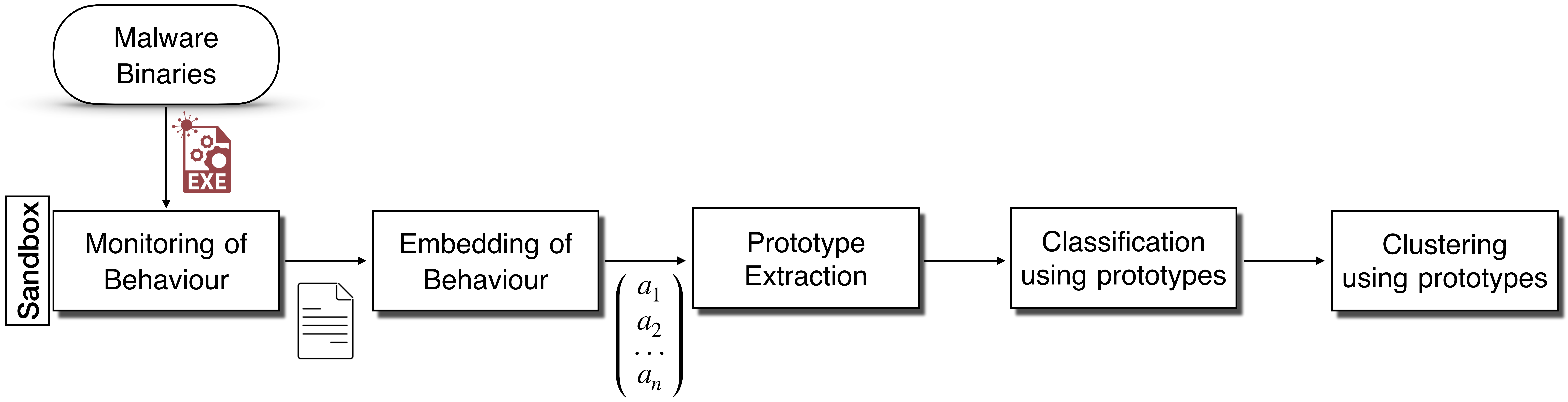
```
k: # of prototypes
n: # of reports
```

"Nearest prototype classification resembles an efficient alternative to costly k-nearest neighbor methods"

(Rieck et al., 2011)

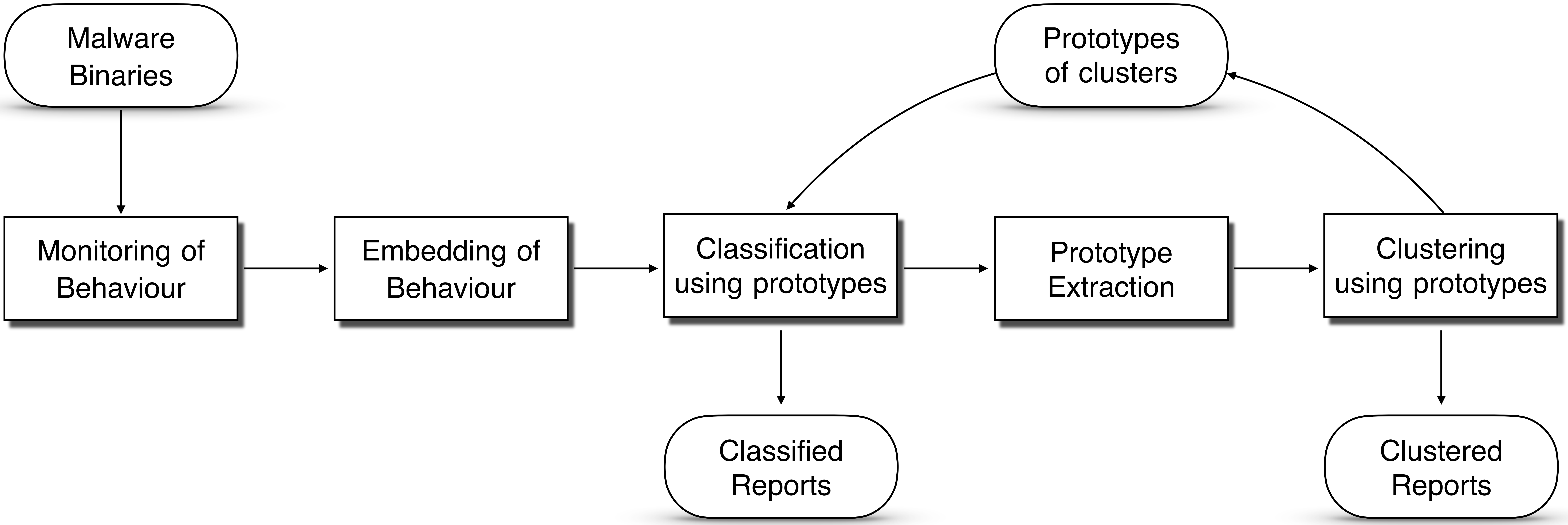
Malheur Framework Simplified

Overview



Malheur Framework

Overview



Incremental Analysis

Algorithm 4 Incremental Analysis

rejected $\leftarrow \emptyset$, *prototypes* $\leftarrow \emptyset$
for *reports* $\leftarrow$ data source $\cup$ *rejected* **do**
    classify *reports* to known clusters using *prototypes* $\triangleright$ see Algorithm 3
    extract prototypes from remaining *reports* $\triangleright$ see Algorithm 1
    cluster remaining *reports* using prototypes $\triangleright$ see Algorithm 2
    prototypes $\leftarrow$ *prototypes* $\cup$ prototypes of new clusters
    rejected $\leftarrow$ rejected reports from clustering

(Rieck et al., 2011)

Attacking the malware with AI

Where the finest concepts of Data Science & Cybersecurity meet

- The talk is based on the paper *“Automatic analysis of malware behavior using machine learning”* (Rieck et al., 2011)

Thanks to:

- Sneha Rajguru
- Bsides Munich

Q/A