人工智慧網路防禦實務

1

魏得恩 Dwyane Wei

資訊工業策進會 資安科技研究所
Institute for Information Industry (III)
Cybersecurity Technology Institute (CSTI)

關於我

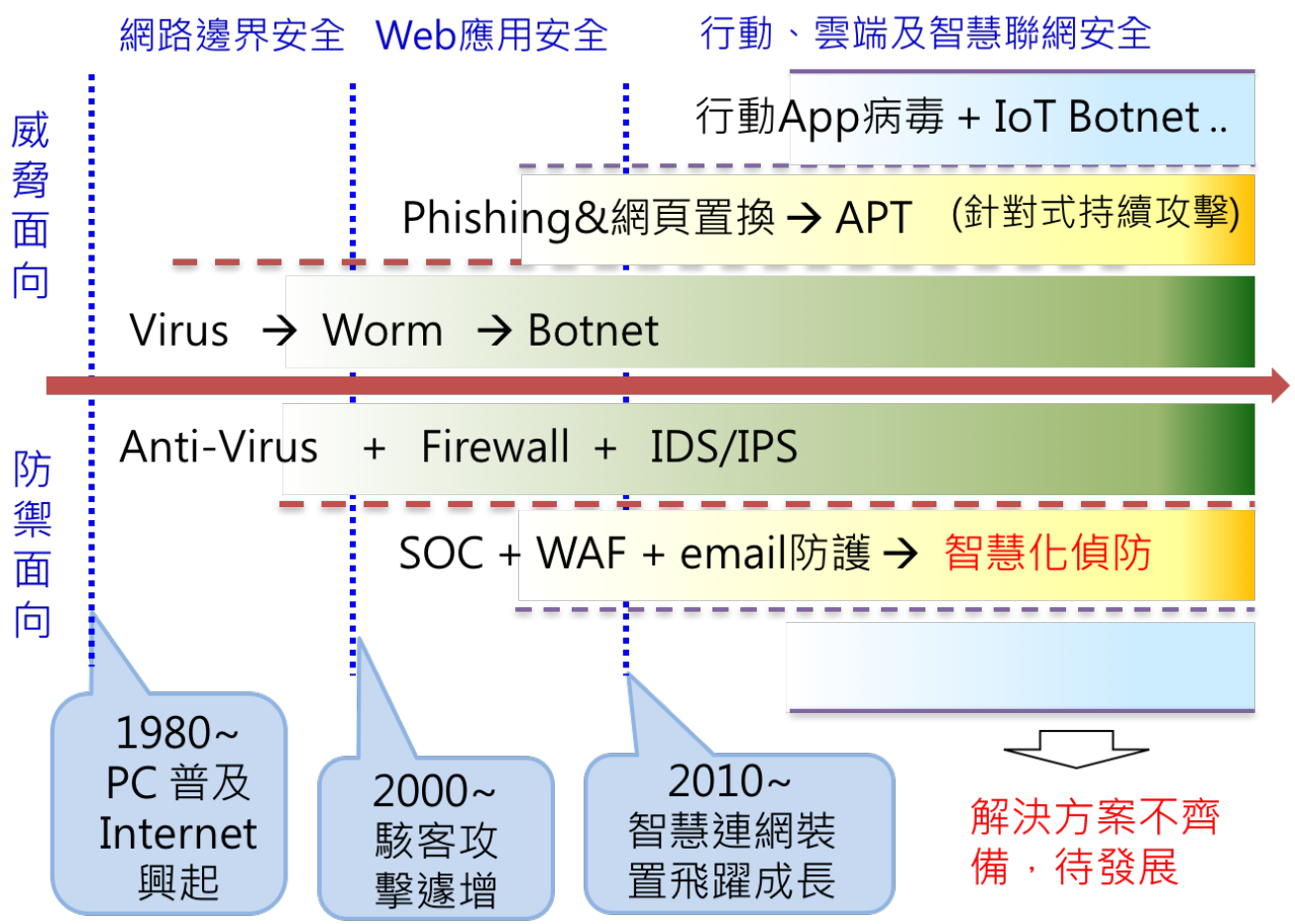
- 姓名：魏得恩 (Dwyane Wei)
- E-mail :
  - dwyanewei@iii.org.tw
  - dwei8399@gmail.com
- 工作：資策會 資安所 組長
- 經歷：
  - 資安廠商合作（e.g., 趨勢）
  - 國際合作（e.g., CMU, Harvard, UC Riverside）

背景：資安威脅與防禦趨勢

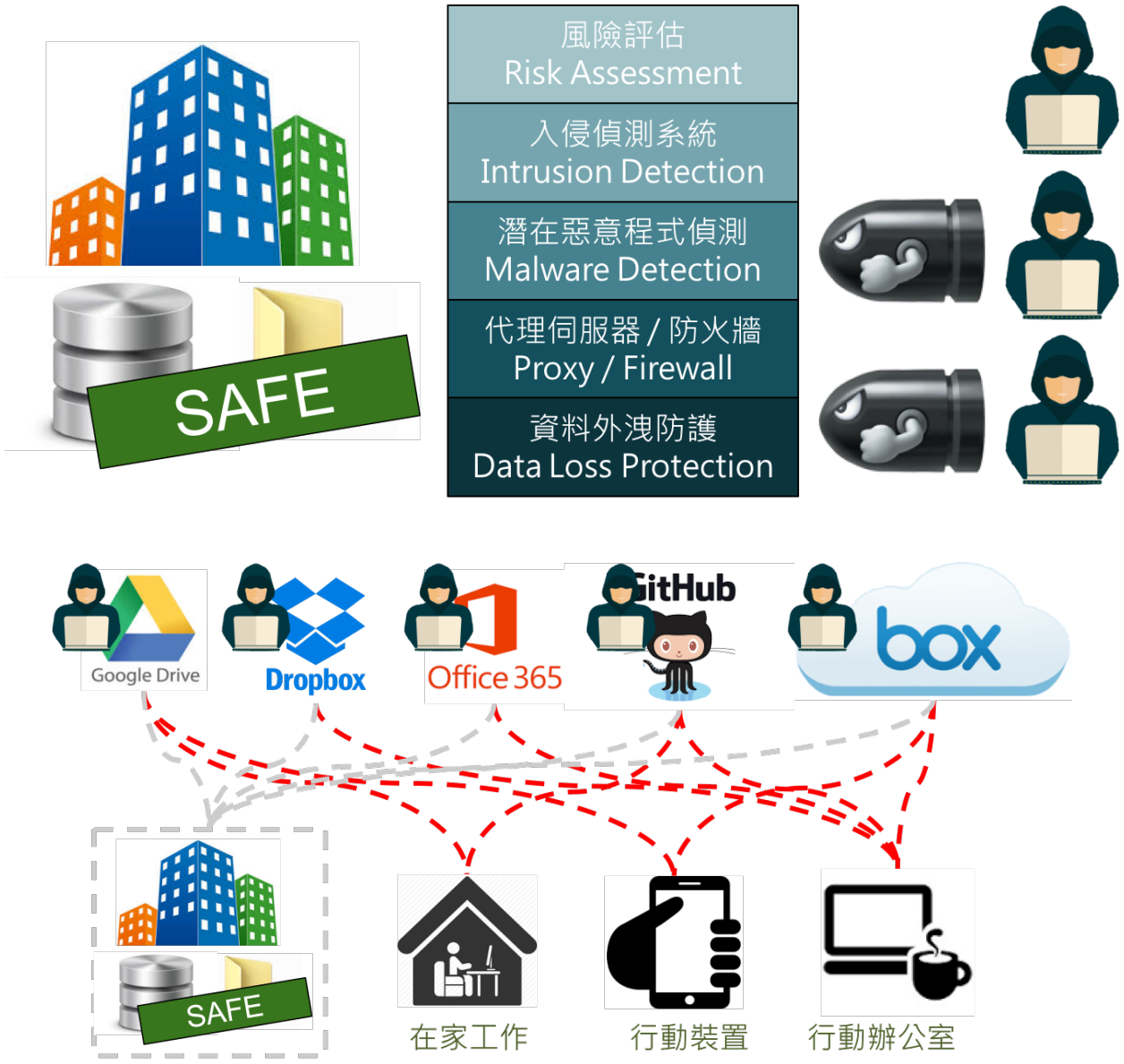
A M-trends report from Mandiant for Advanced Persistent Threat (APT)

- Varied Targeted Victims
- Less Anomaly Self-Detection
- Non-immediate Detection

背景：資安威脅與防禦趨勢

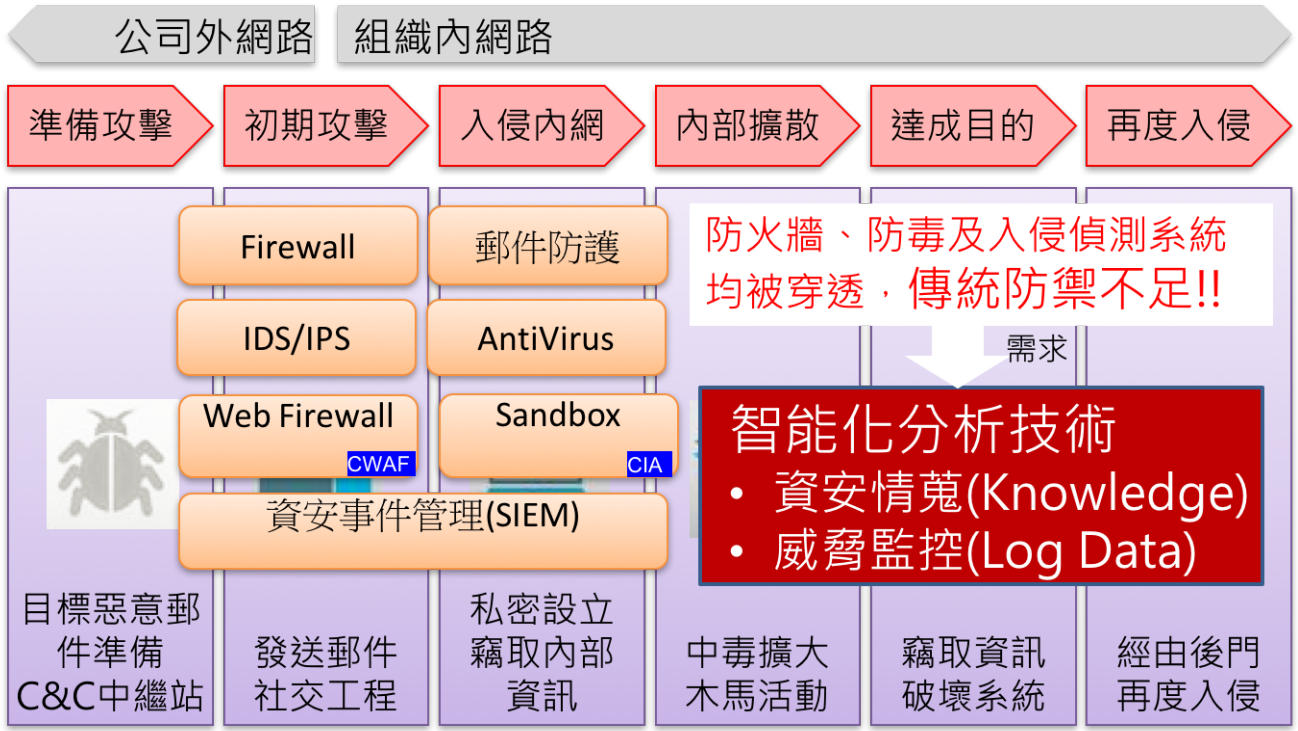


- 攻擊手法日益複雜
- 新興科技改變攻擊面貌
- 傳統防禦效能難提升
- 新興攻擊欠缺有效防禦技術



背景：
新興應用導入(E.G., 企業導入雲端服務)後所面臨的困境

- 傳統資安防禦保障企業內部安全無虞
- 雲端服務存取，導致防線形同虛設



國內廠商技術缺口-智能化分析

- 新興攻擊手法衝擊傳統邊界防禦技術
- 智能化分析偵測企業暗網、Shadow IT等橫向擴散行為

人工智慧與機器學習已成為資安攻防技術研發最新顯學

FORTSCALE

成立於2012年新創公司，以AI 為核心，分析使用者與系統互動行為，偵測內部威脅與長期潛伏攻擊。

patternex

以機器學習為基礎的異常偵測技術，強調模仿資安專家智慧進行資安事件分析，以AI driven by Analyst Intuition 為概念

DARKTRACE

以仿人體自我免疫的資安防護系統，已有超過1,200以上的使用者案例經驗，現為知名的全球資安新創公司。

CGC CYBER GRAND_CHALLENGE

- 機器對機器攻擊，使得DARPA推出CGC（Cyber Grand Challenge）機器人漏洞自動攻防競賽計畫(Berkeley, CMU, UC Santa Babara)
  機器速度修補漏洞，發展自動化、可規模化系統，打破過去修補漏洞的反應模式

OUTLINE

- What "AI" in Information Security do
- Supervised Learning
  - Decision Tree
  - Support Vector Machines (SVM)
- Unsupervised Learning
  - K-means
- Decomposition Algorithm
  - Singular Value Decomposition (SVD)
- Graph Mining
  - Pagerank
- Anomaly Detection & Graph Mining
  - ChainSpot
  - WebHound
  - Playwright
- Other Cyber Application in AI
  - Malware Analysis

WHAT "AI" IN INFORMATION SECURITY DO

- One of most representative form of data to be analyzed in information security is .LOG
- Log Analysis
  - IDS Logs
  - Firewall Logs
  - Web Server Access Logs
  - Proxy Logs
  - Active Directory Logs
- Social Media
  - Twitter, FB, …

INTRUSION DETECTION SYSTEM (IDS)

- Depends on where you deploy the IDS
  - Host IDS (HIDS)
  - Network IDS (NIDS)
- Detect the known attacking signature
  - Port, attacking vector, or sent content.
  - Highly depends on human fine tuning the rule
  - False alarm issue
- Usually contains following information:
  - Event Name, Source IP, Source Port, Dest. IP, Dest. Port, TimeStamp
- A well-studied domain:
  - False Alarm Reduction
  - Multi-steps Attacks correlation
- Applied scenario: SOC (security operation center), CERT, ISAC…

FIREWALL LOGS

- Firewall: blocking connection or transmission based on known blacklist or pre-defined policy
- Easy to deployment and use.
- Lacking of analysis ability as facing unseen attacking signature. (e.g. IP, Domain Name)

- Contain following information:
  - Source IP, Source Port, Dest. IP, Dest. Port, Permit/Deny

- Also well-studied on large-scaled connection correlation
  - Honeypot attacking pattern analysis
  - Botnet structure analysis

WEB SERVER ACCESS LOGS

- Application-layer malicious behavior detection
- Web Server Accessing Logs contain:
  - Source IP, Accessed Path, Access Status, Timestamp, Http header (optional), File Size(optional)
- Usually, a tradeoff is between log visibility and server execution performance
- Only application-layer malicious behavior such as script can be detected.
- Research emerged around 2005:
  - Analyze the accessed path to detect SQL injection
  - Graph mining on association graph of web access
  - Build a classifier to predict a risk of a given request.

PROXY LOGS

- Proxy is an agent deployed on gateway of an enterprise network.
- Recorded the web surfing behavior
- Why "web surfing behavior" is important
  - URL connection for C&C server communication and control.
  - Detect the phishing web site.
  - Detect malicious scanning attack.
- Containing:
  - Source IP, Dest. URL, TimeStamp, Web Agent Info., MIME …
- Research emerged around 2010
  - Analyze the malware behavior inside the enterprise network.
  - Correlate other logs (IDS, FW, AD logs, etc.) to evaluate risk of an account

WINDOWS EVENT LOGS (ACTIVE DIRECTORY LOGS)

- Operating system records connections or events, between client side and server side, of registered accounts.
- Containing:
  - TimeStamp, Account, Source IP, Event Name, Sub-Event Annotation, …
- Can be used to detect hidden malicious behavior:
  - Anomaly login/logout behavior
  - Anomaly resource allocation
  - Anomaly permission granting
- Recently, AD is well-used on event tracking, however, it resulted in issue that AD data is big scale
  - Companies in industry focused on AD-related research in last 2-3 years.

DIFFERENT KINDS OF MACHINE LEARNING

supervised learning (classification, regression) $\{(x_{1:n}, y_{1:n})\}$

$\updownarrow$

semi-supervised classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}, x_{test}\}$

transductive classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}\}$

$\updownarrow$

semi-supervised clustering $\{x_{1:n}$, must-, cannot-links$\}$

$\updownarrow$

unsupervised learning (clustering) $\{x_{1:n}\}$

SUPERVISED LEARNING

- Decision Tree
- Support Vector Machines (SVM)

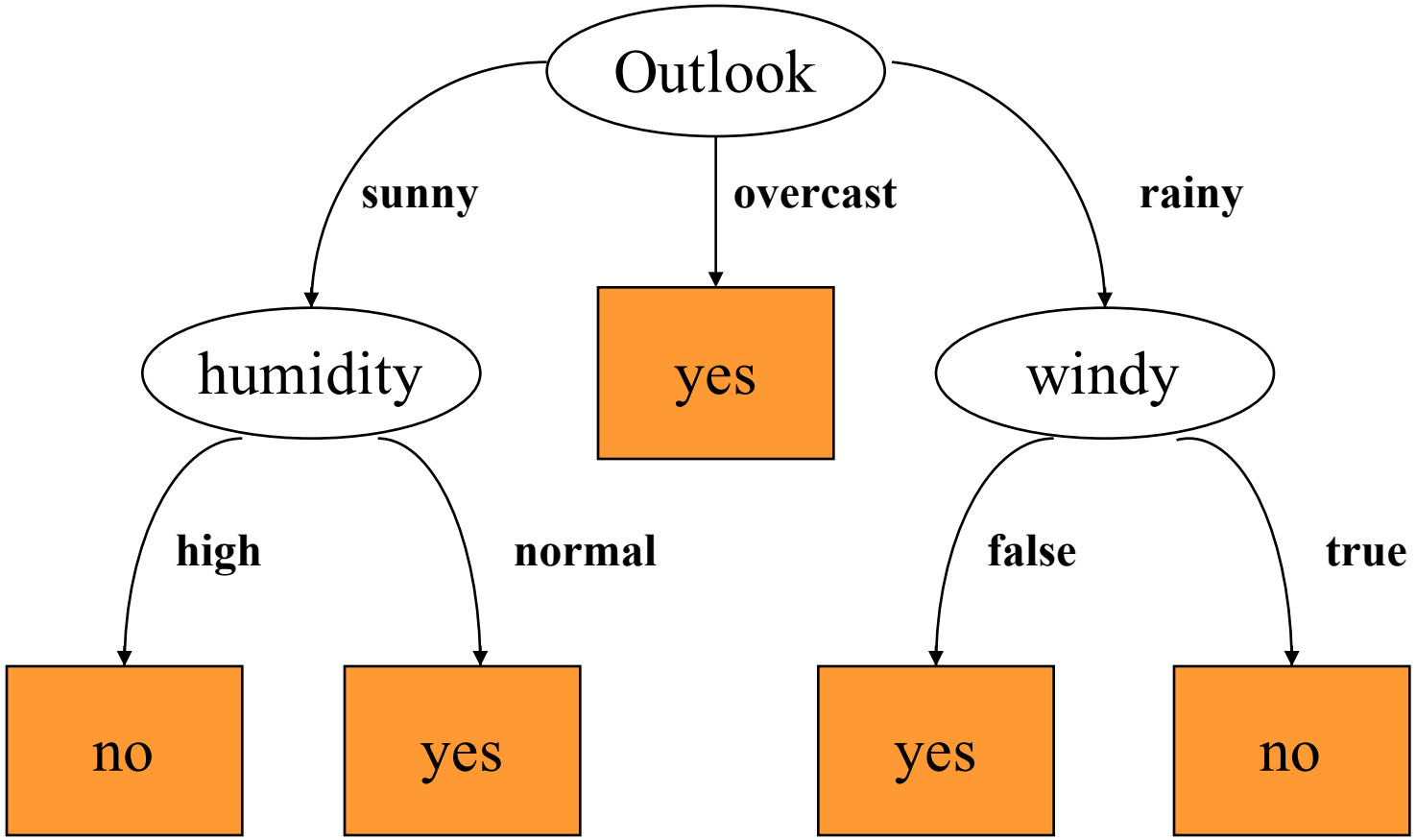
SUPERVISED LEARNING

- Decision Tree
- Support Vector Machines (SVM)

A DECISION TREE EXAMPLE – THE WEATHER DATA

ID code	Outlook	Temperature	Humidity	Windy	Play
a	Sunny	Hot	High	False	No
b	Sunny	Hot	High	True	No
c	Overcast	Hot	High	False	Yes
d	Rainy	Mild	High	False	Yes
e	Rainy	Cool	Normal	False	Yes
f	Rainy	Cool	Normal	True	No
g	Overcast	Cool	Normal	True	Yes
h	Sunny	Mild	High	False	No
i	Sunny	Cool	Normal	False	Yes
j	Rainy	Mild	Normal	False	Yes
k	Sunny	Mild	Normal	True	Yes
l	Overcast	Mild	High	True	Yes
m	Overcast	Hot	Normal	False	Yes
n	Rainy	Mild	High	True	No

A DECISION TREE EXAMPLE



Decision tree for the weather data.

THE PROCESS OF CONSTRUCTING A DECISION TREE

- Select an attribute to place at the root of the decision tree
- Make one branch for every possible value
- Repeat the process recursively for each branch

WHICH ATTRIBUTE SHOULD BE PLACED AT A CERTAIN NODE

- One common approach is based on the information gain by placing a certain attribute at this node

- The so called information gain is directly proportional to improvement in terms of outcome distribution entropy

THE GENERAL FORM FOR CALCULATING THE INFORMATION GAIN

- First we calculate outcome distribution followed by a certain decision.
- Entropy of a decision =

$$-P_1 \times \log_2 P_1 - P_2 \times \log_2 P_2 - \cdots - P_n \times \log_2 P_n$$

, where P_1, P_2, …, P_n are the probabilities of the n possible outcomes.

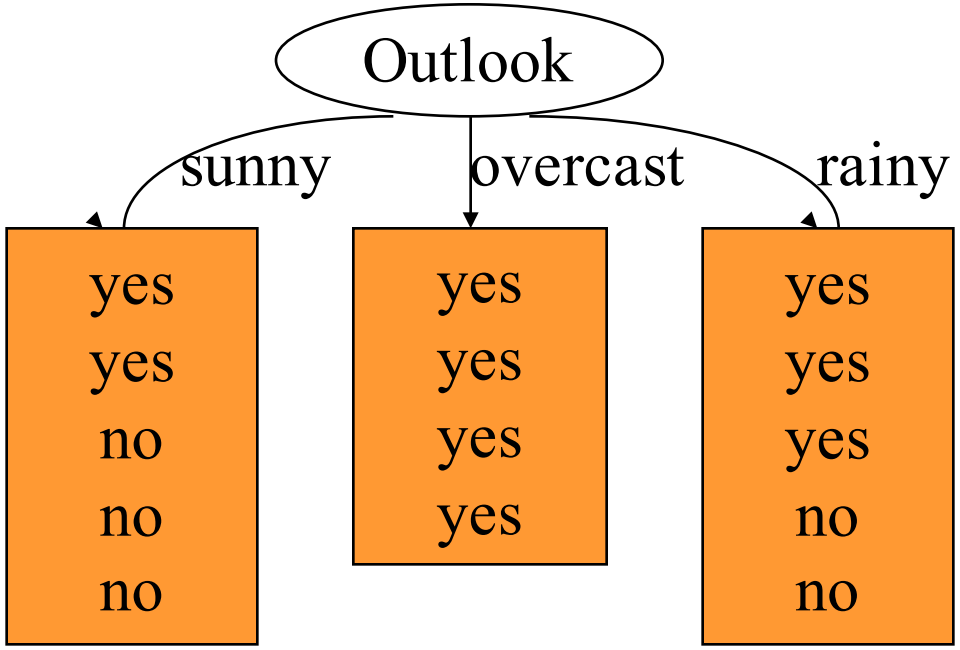
- The max. entropy happens when $P_1 = P_2 = \ldots = P_n = 1/n$
- The min. Entropy happens when one of P_is = 1

FOR EXAMPLE, THE ORIGINAL ENTROPY

- In the weather data example,
  - 9 instances of which the decision to play is "yes"
  - 5 instances of which the decision to play is "no" .
  - Then, the entropy of original distribution is

$$\frac{9}{14}\times\left(-\log_2\frac{9}{14}\right)+\left(\frac{5}{14}\right)\times\left(-\log_2\frac{5}{14}\right)=0.940\text{ bits.}$$

RESULTED INFORMATION GAIN AS WE PLACE "OUTLOOK" AT THE ROOT



Information further required =

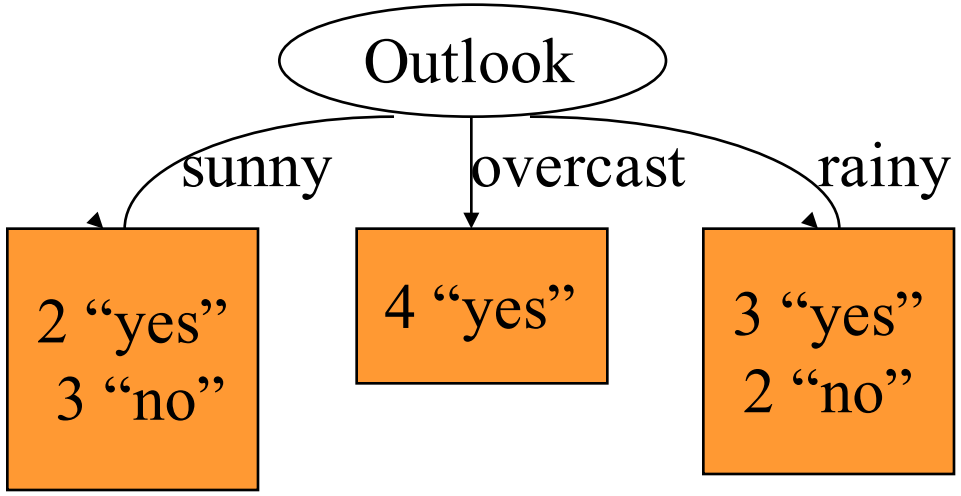
$$\left(\frac{5}{14}\right)\times 0.971+\left(\frac{4}{14}\right)\times 0+\left(\frac{5}{14}\right)\times 0.971=0.693 bits.$$

INFORMATION GAINED BY PLACING EACH OF THE 4 ATTRIBUTES

- Gain(outlook) = 0.940 bits – 0.693 bits = 0.247 bits.
- Gain(temperature) = 0.029 bits.
- Gain(humidity) = 0.234 bits.
- Gain(windy) = 0.048 bits.

THE STRATEGY FOR SELECTING AN ATTRIBUTE TO PLACE AT A NODE

- Select the attribute that gives us the largest information gain (i.e., improvement of entropy).
- In this example, it is the attribute "Outlook".



THE RECURSIVE PROCEDURE FOR CONSTRUCTING A DECISION TREE (1)

- The operation discussed above is applied to each branch recursively to construct the decision tree.

- For example, for the branch "Outlook = Sunny", we evaluate the information gained by applying each of the remaining 3 attributes.
  - Gain(Outlook=sunny;Temperature) = 0.971 – 0.4 = 0.571
  - Gain(Outlook=sunny;Humidity) = 0.971 – 0 = 0.971
  - Gain(Outlook=sunny;Windy) = 0.971 – 0.951 = 0.02

THE RECURSIVE PROCEDURE FOR CONSTRUCTING A DECISION TREE (2)

- Similarly, we also evaluate the information gained by applying each of the remaining 3 attributes for the branch "Outlook = rainy".

  - Gain(Outlook=rainy;Temperature) = 0.971 – 0.951 = 0.02
  - Gain(Outlook=rainy;Humidity) = 0.971 – 0.951 = 0.02
  - Gain(Outlook=rainy;Windy) =0.971 – 0 = 0.971

SCIKIT-LEARN EXAMPLE

1.10.1. Classification

`DecisionTreeClassifier` is a class capable of performing multi-class classification on a dataset.

As with other classifiers, `DecisionTreeClassifier` takes as input two arrays: an array X, sparse or dense, of size `[n_samples, n_features]` holding the training samples, and an array Y of integer values, size `[n_samples]`, holding the class labels for the training samples:

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

After being fitted, the model can then be used to predict the class of samples:

```
>>> clf.predict([[2., 2.]])
array([1])
```

參考資訊： http://scikit-learn.org/stable/modules/tree.html

SUPERVISED LEARNING

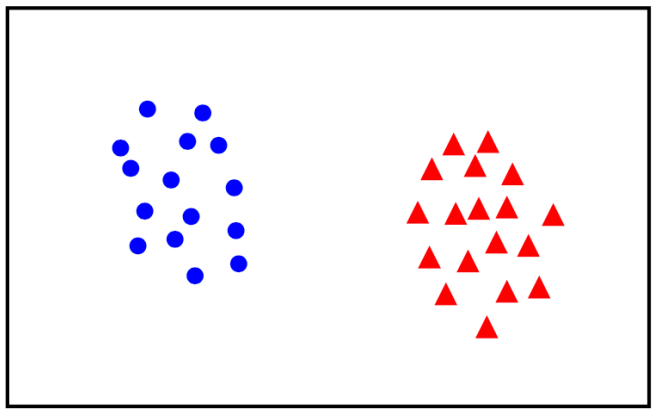
- Decision Tree
- Support Vector Machines (SVM)

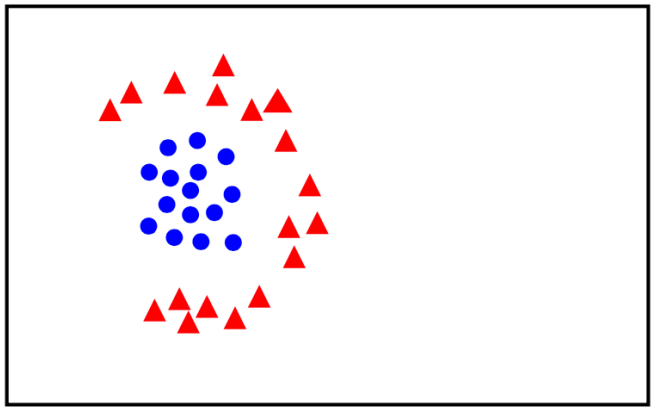
BINARY CLASSIFICATION

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots N$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

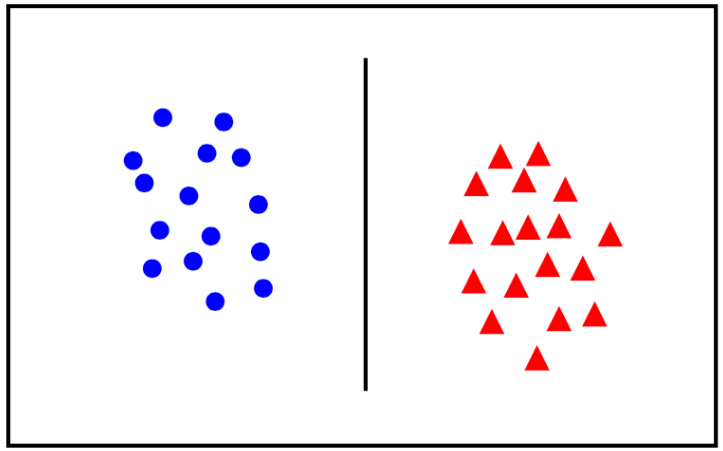
i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification.

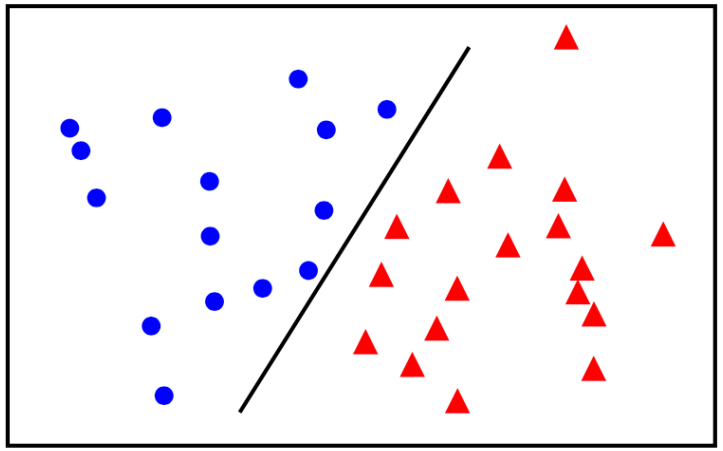




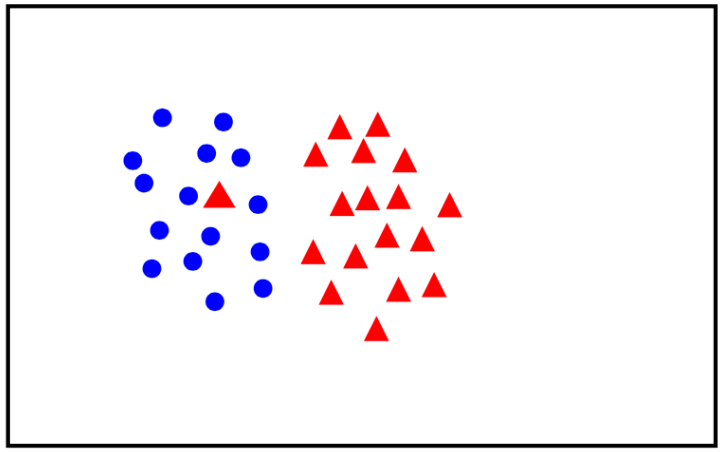
LINEAR SEPARABILITY

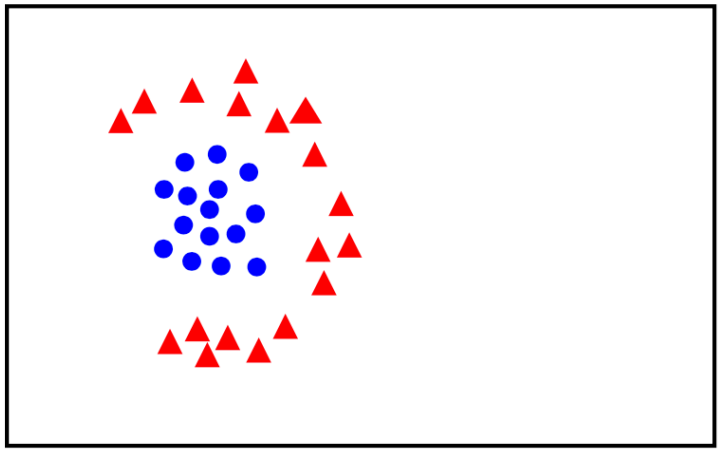
linearly separable





not linearly separable



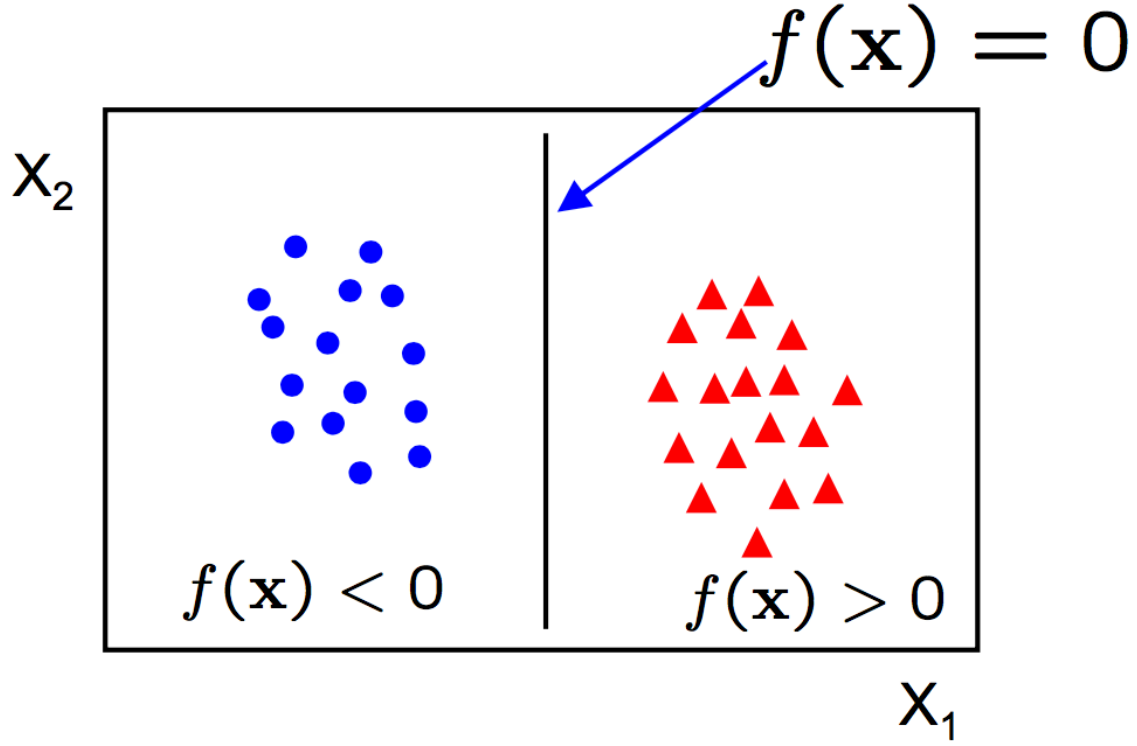


LINEAR CLASSIFIERS

- A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

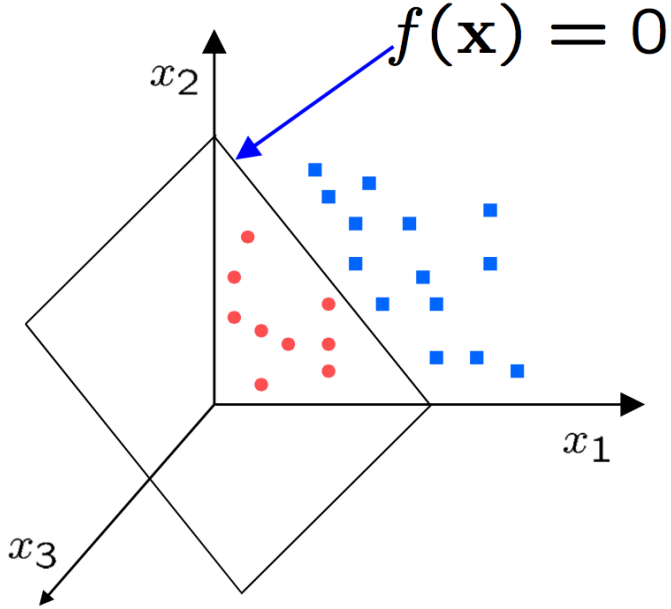
- in 2D the discriminant is a line
- is the normal to the line, and b the bias
- is known as the weight vector



LINEAR CLASSIFIERS (CONT'D)

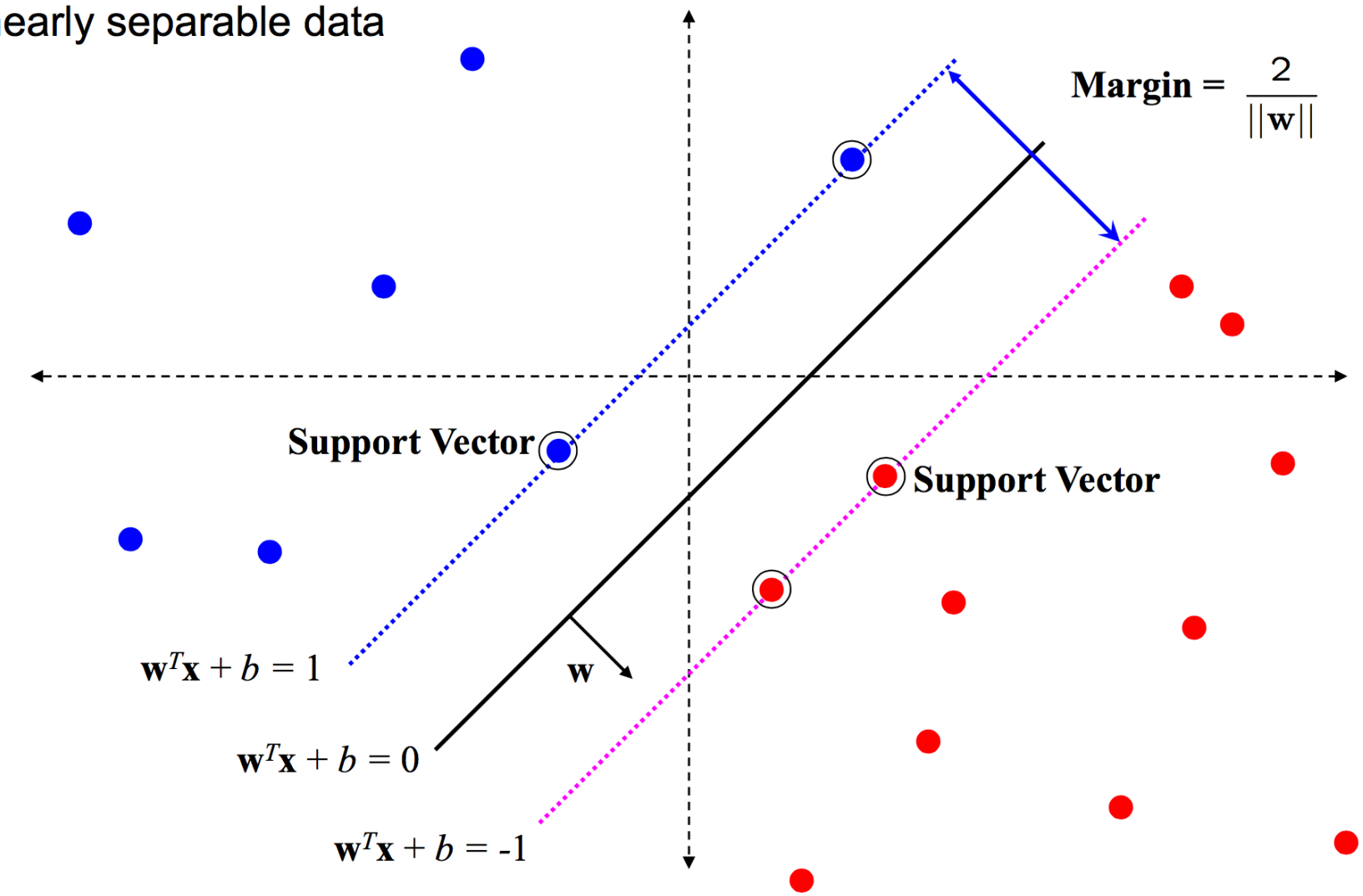
- A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



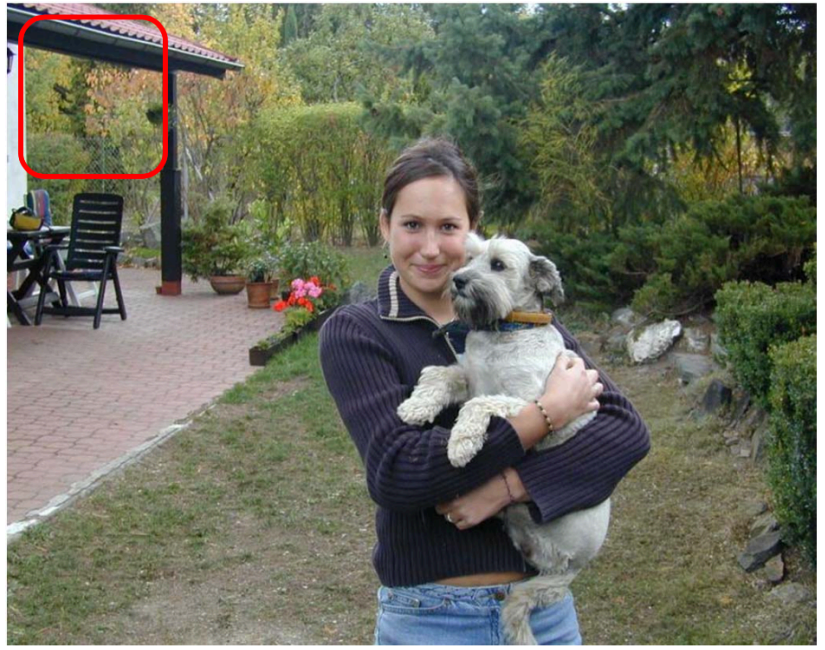
- in 3D the discriminant is a plane, and in nD it is a hyperplane
- For a K-NN classifier it was necessary to `carry' the training data
- For a linear classifier, the training data is used to learn **w** and then discarded
- Only **w** is needed for classifying new data

SUPPORT VECTOR MACHINE



APPLICATION: PEDESTRIAN DETECTION IN COMPUTER VISION

- Objective: detect (localize) standing humans in an image
  - reduces object detection to binary classification
  - does an image window contain a person or not?

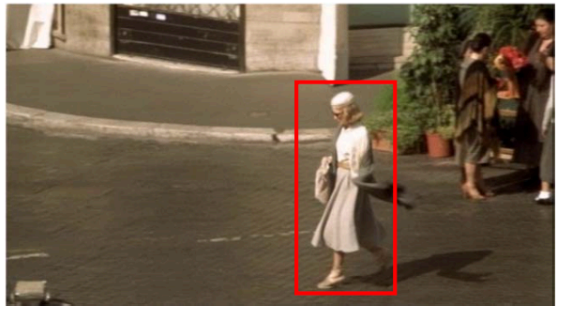


Method: the HOG detector

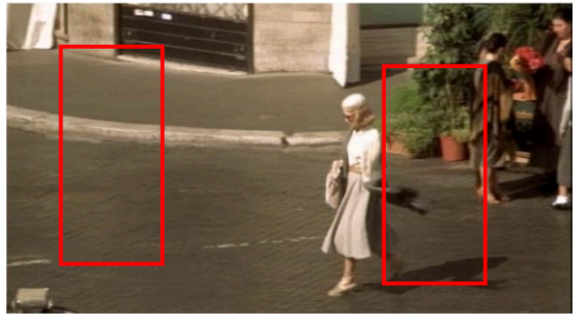
TRAINING DATA AND FEATURES

- Positive data – 1208 positive window examples

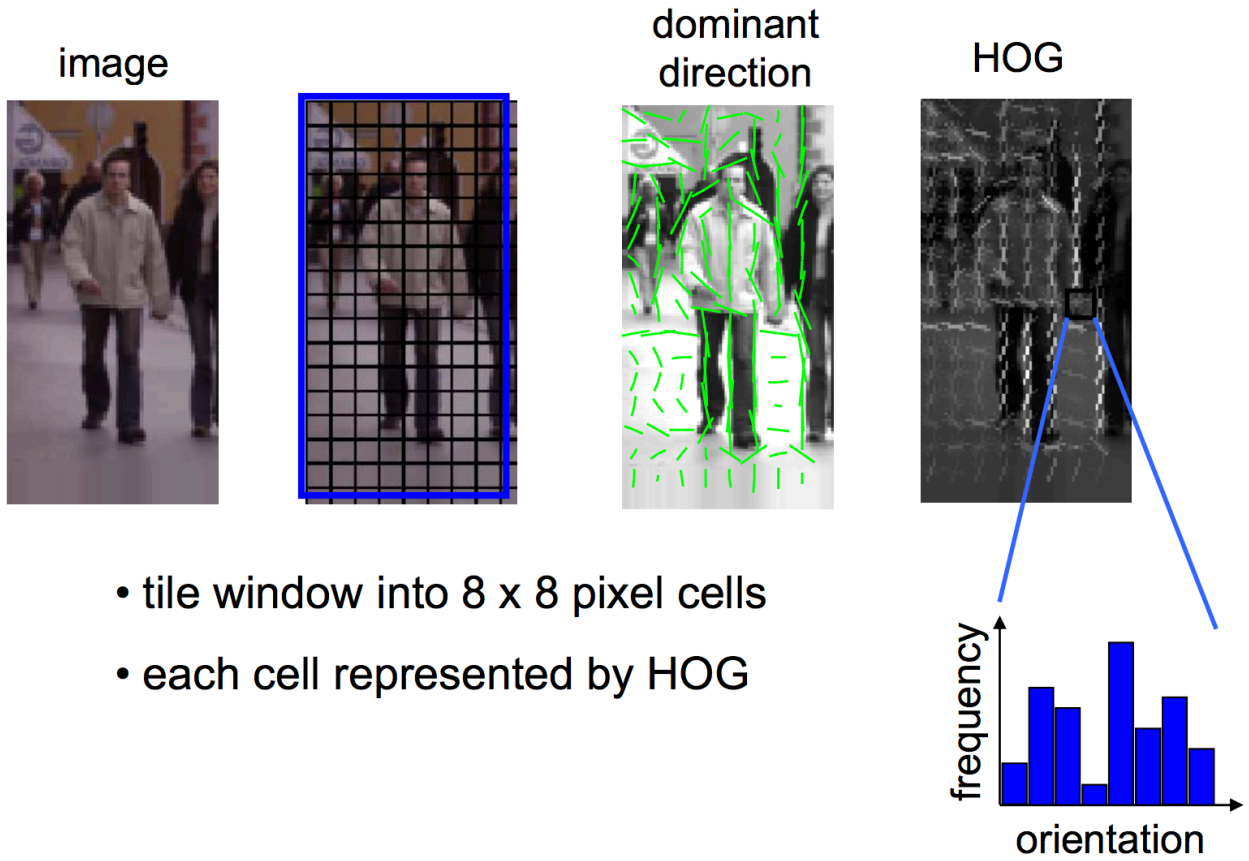


- Negative data – 1218 negative window examples (initially)

FEATURE: HISTOGRAM OF ORIENTED GRADIENTS (HOG)



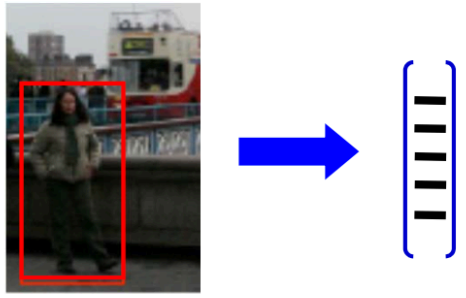
- tile window into 8 x 8 pixel cells
- each cell represented by HOG

Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024

ALGORITHM & RESULTS

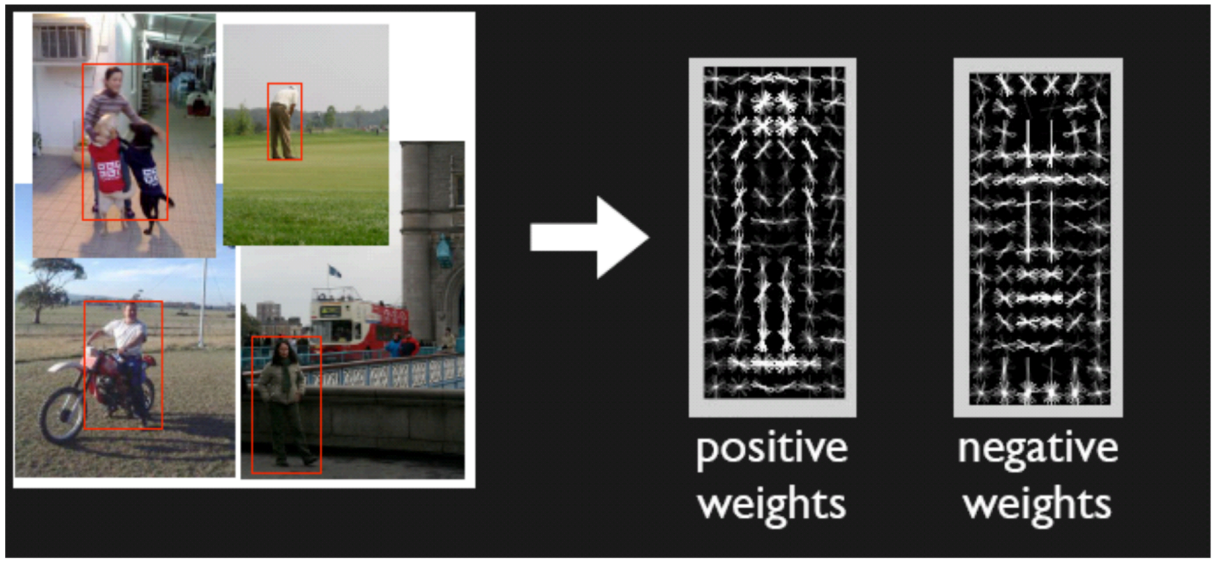
Training (Learning)

- Represent each example window by a HOG feature vector



$$\mathbf{x}_i \in \mathbb{R}^d, \text{ with } d = 1024$$

- Train a SVM classifier



Testing (Detection)

- Sliding window classifier

$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

SCIKIT-LEARN EXAMPLE

1.4.1. Classification

`SVC`, `NuSVC` and `LinearSVC` are classes capable of performing multi-class classification on a dataset.

As other classifiers, `SVC`, `NuSVC` and `LinearSVC` take as input two arrays: an array X of size `[n_samples, n_features]` holding the training samples, and an array y of class labels (strings or integers), size `[n_samples]`:

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

After being fitted, the model can then be used to predict new values:

```
>>> clf.predict([[2., 2.]])
array([1])
```

SVMs decision function depends on some subset of the training data, called the support vectors. Some properties of these support vectors can be found in members `support_vectors_`, `support_` and `n_support`:

```
>>> # get support vectors
>>> clf.support_vectors_
array([[ 0.,  0.],
       [ 1.,  1.]])
>>> # get indices of support vectors
>>> clf.support_
array([0, 1]...)
>>> # get number of support vectors for each class
>>> clf.n_support_
array([1, 1]...)
```

參考資訊： http://scikit-learn.org/stable/modules/svm.html

UNSUPERVISED LEARNING

- K-means

WHAT IS CLUSTERING?

- **Clustering** is assigning objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.

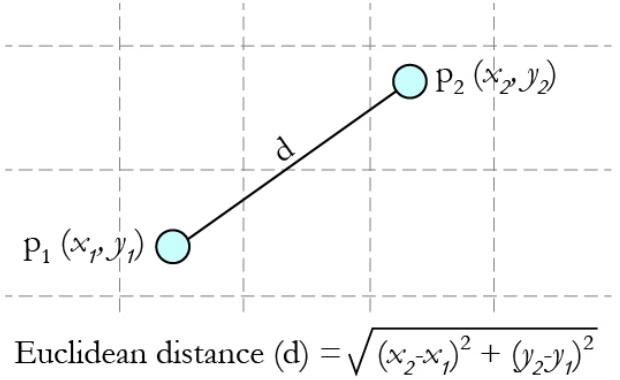
COMMON DISTANCE MEASURES

- *Distance measure* will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.

  They include:

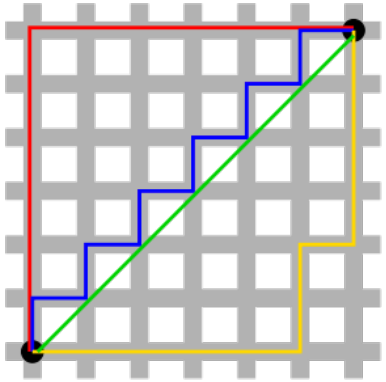
1. The Euclidean distance (also called 2-norm distance) is given by:

$$d(x, y) = \sqrt[2]{\sum_{i=1}^{p} |x_i - y_i|^2}$$



2. The Manhattan distance (also called taxicab norm or 1-norm) is given by:

$$d(x, y) = \sum_{i=1}^{p} |x_i - y_i|$$

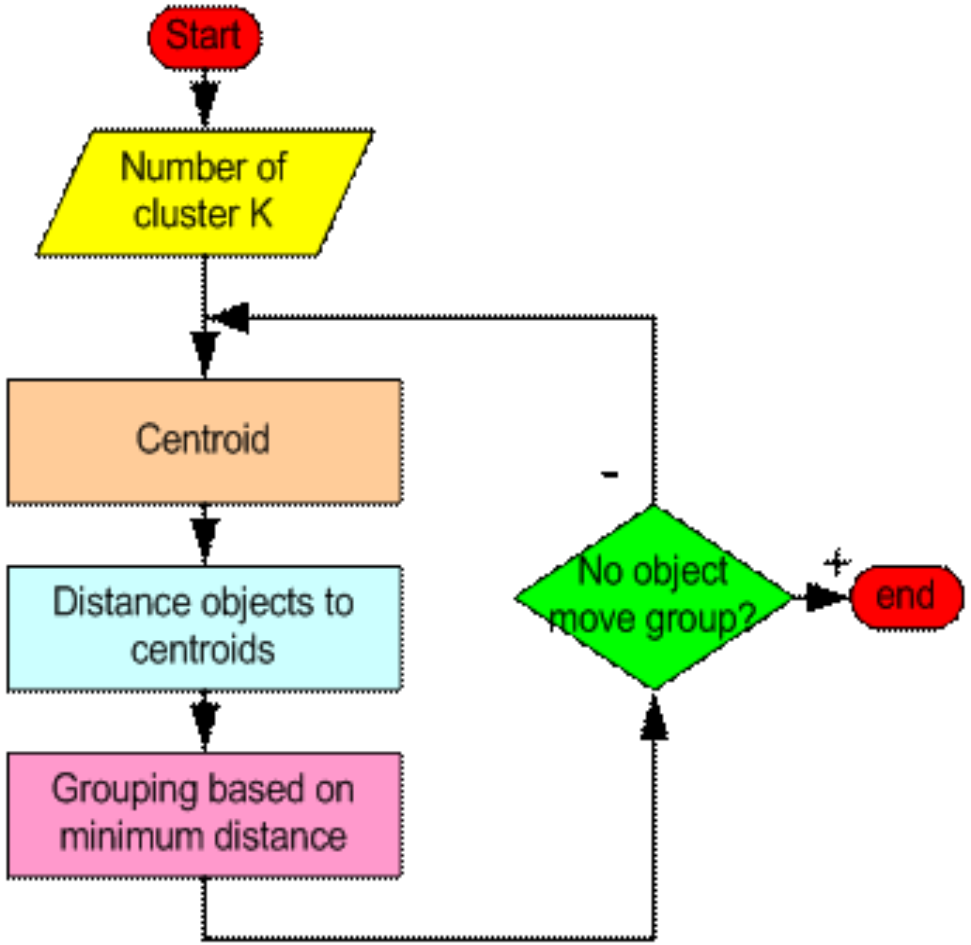


K-MEANS CLUSTERING

- The **K-means algorithm** is an algorithm to cluster *n* objects based on attributes into *k* partitions, where $k < n$.

- It assumes that the object attributes form a vector space.

HOW THE K-MEAN CLUSTERING ALGORITHM WORKS? (1)



HOW THE K-MEAN CLUSTERING ALGORITHM WORKS? (2)

- **Step 1:** Begin with a decision on the value of K = number of clusters .
- **Step 2**: Put K initial centers to form K initial clusters.
  - You may assign the training samples randomly, or systematically as the following:

1. Take the first k training sample as single-element clusters
2. Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid.
3. After each assignment, re-compute the centroid of the gaining cluster.

HOW THE K-MEAN CLUSTERING ALGORITHM WORKS? (3)

- **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.

- **Step 4 .** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

A SIMPLE EXAMPLE OF K-MEANS ALGORITHM (1)

- K = 2

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

A SIMPLE EXAMPLE OF K-MEANS ALGORITHM (2)

- **Step 1**:
  - Initialization: Randomly we choose following two centroids (k=2) for two clusters.
  - In this case the 2 centroid are: m1=(1.0,1.0) and m2=(5.0,7.0).

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

A SIMPLE EXAMPLE OF K-MEANS ALGORITHM (3)

- **Step 2:**
  - Thus, we obtain two clusters containing: {1,2,3} and {4,5,6,7}.
  - Their new centroids are:

$$m_1 = \left(\frac{1}{3}(1.0+1.5+3.0), \frac{1}{3}(1.0+2.0+4.0)\right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0+3.5+4.5+3.5), \frac{1}{4}(7.0+5.0+5.0+4.5)\right)$$

$$= (4.12, 5.38)$$

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

$$d(m_1,2) = \sqrt{|1.0-1.5|^2 + |1.0-2.0|^2} = 1.12$$

$$d(m_2,2) = \sqrt{|5.0-1.5|^2 + |7.0-2.0|^2} = 6.10$$

A SIMPLE EXAMPLE OF K-MEANS ALGORITHM (4)

- **Step 3:**
  - Now using these centroids we compute the Euclidean distance of each object, as shown in table.
  - Therefore, the new clusters are:

    {1,2} and {**3**,4,5,6,7}
  - Next centroids are: m1=(1.25,1.5) and m2 = (3.9,5.1)

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

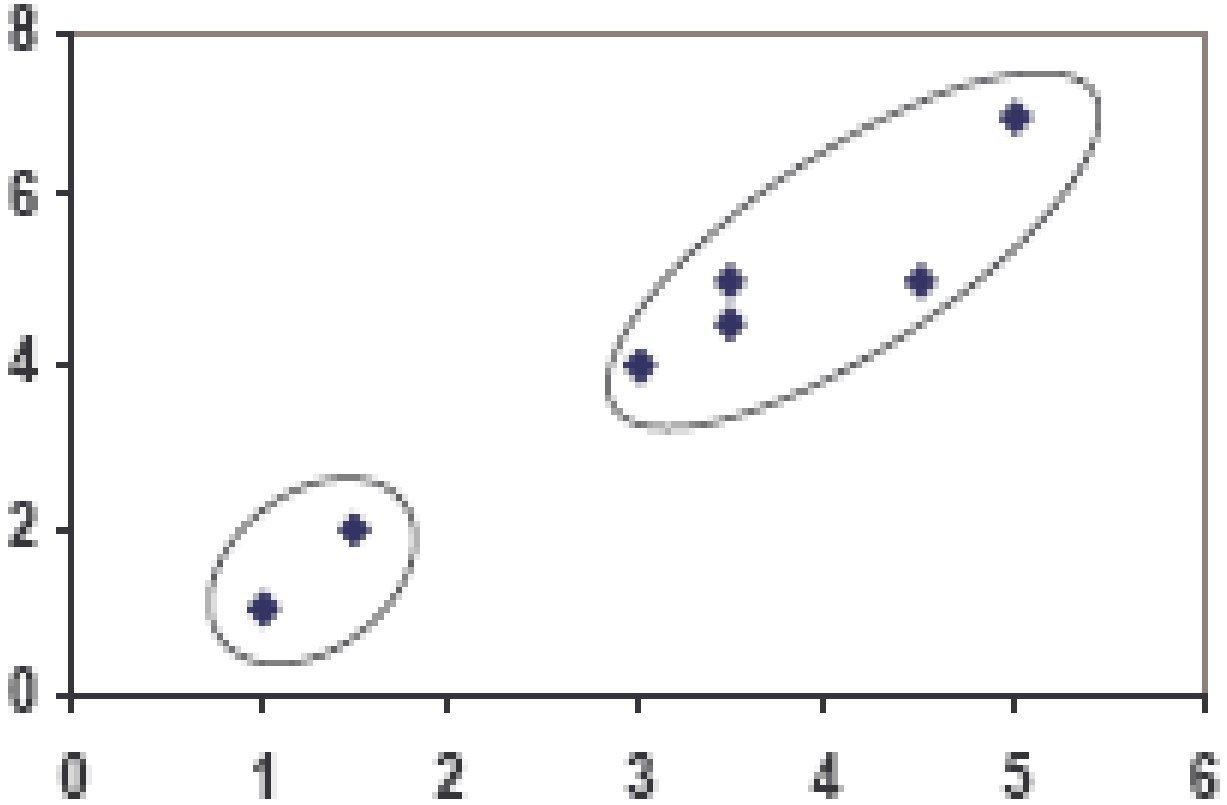
A SIMPLE EXAMPLE OF K-MEANS ALGORITHM (5)

- **Step 4 :**
  - The clusters obtained are:

    {1,2} and {3,4,5,6,7}
  - Therefore, there is no change in the cluster.
  - Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.

Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0. 56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72

PLOT OF RESULT



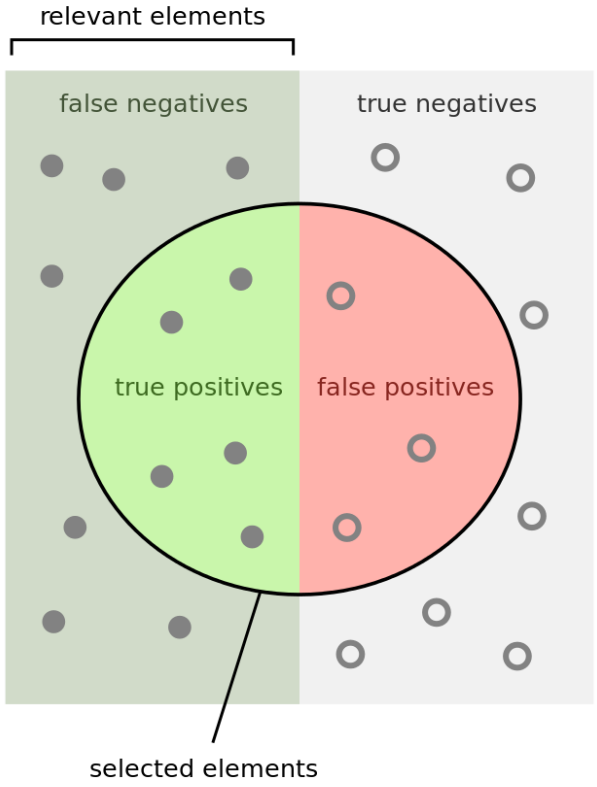
SCIKIT-LEARN EXAMPLE

Examples

```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [4, 2], [4, 4], [4, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([0, 0, 0, 1, 1, 1], dtype=int32)
>>> kmeans.predict([[0, 0], [4, 4]])
array([0, 1], dtype=int32)
>>> kmeans.cluster_centers_
array([[ 1.,  2.],
       [ 4.,  2.]])
```

參考資訊： http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

MEASURES OF QUALITY OF SUPERVISED LEARNING



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

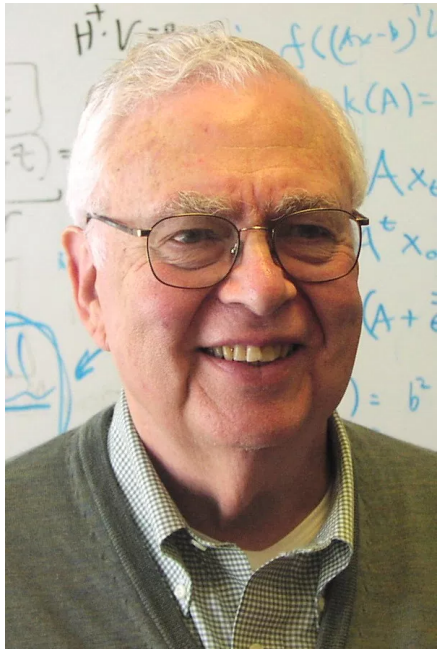
$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

DECOMPOSITION ALGORITHMS

- Singular Value Decomposition (SVD)

INTRODUCTION OF SVD

- 奇異值分解 (**S**ingular **V**alue **D**ecomposition，以下簡稱 SVD) 被譽為矩陣分解的「瑞士刀」和「勞斯萊斯」，前者說明它的用途非常廣泛，後者意味它是值得珍藏的精品。



美國史丹佛大學教授格魯布 (Gene Golub) 於矩陣運算的貢獻造就 SVD 成為今日最重要的線性代數應用
From http://www.cs.nyu.edu/overton/genearoundtheworld/gene.jpg

參考資訊： https://ccjou.wordpress.com/2009/09/01/%E5%A5%87%E7%95%B0%E5%80%BC%E5%88%86%E8%A7%A3-svd/

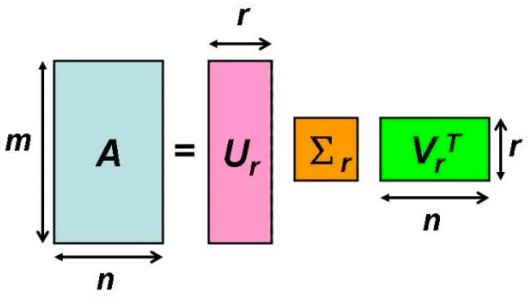
HOW TO DO MATRIX DECOMPOSITION ?!

設 A 為一個 $m \times n$ 階實矩陣，$r = \mathrm{rank}A$，SVD 具有以下形式：

$$A = U\Sigma V^T,$$

其中 U 是 $m \times m$ 階，V 是 $n \times n$ 階，Σ 是 $m \times n$ 階。特別的是，方陣 U 和 V 都是實正交矩陣 (orthogonal matrix)，也就是說，$U^T = U^{-1}$，$V^T = V^{-1}$，Σ 是 (類) 對角矩陣，如下：

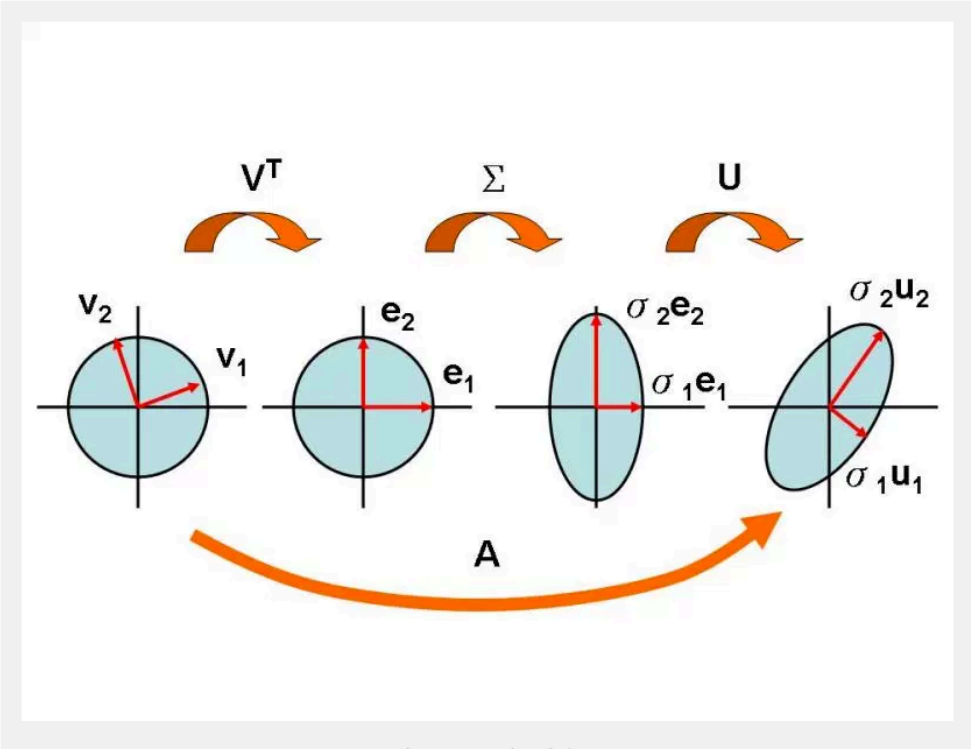
$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & \vdots & \cdots & 0 \\ 0 & 0 & \sigma_r & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}_{\circ}$$



參考資訊： https://ccjou.wordpress.com/2009/09/01/%E5%A5%87%E7%95%B0%E5%80%BC%E5%88%86%E8%A7%A3-svd/

WHAT DOES SVD MEAN ?!

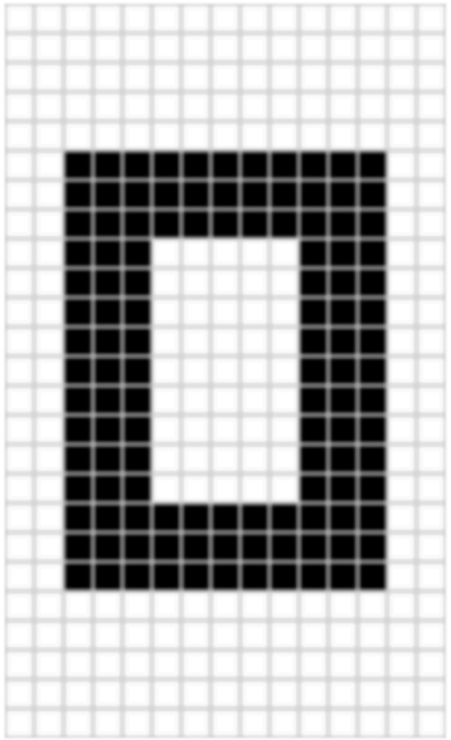
我們可以將 SVD 視為變換矩陣 A 的三個分解步驟：旋轉 V^T，伸縮 Σ，再旋轉 U，圖三顯示 2×2 階矩陣的分解變換。另一方面，Σ 也可以視為變換矩陣 A 參考了基底 $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ 和 $\{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$ 的主對角變換矩陣 (見"線性變換觀點下的奇異值分解")。



參考資訊： https://ccjou.wordpress.com/2009/09/01/%E5%A5%87%E7%95%B0%E5%80%BC%E5%88%86%E8%A7%A3-svd/

APPLICATION OF SVD

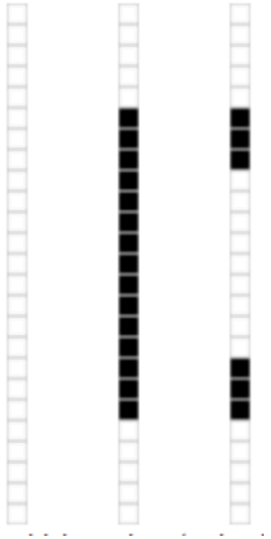
- 有一張 25×15 的圖片:



把它用矩陣表示：

$$M = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}$$

這個矩陣的秩等於 3。即矩陣只有 3 種線性無關的列，其他的列都是冗余的：



對 M 做奇異值分解，得到 3 個不為零的奇異值：

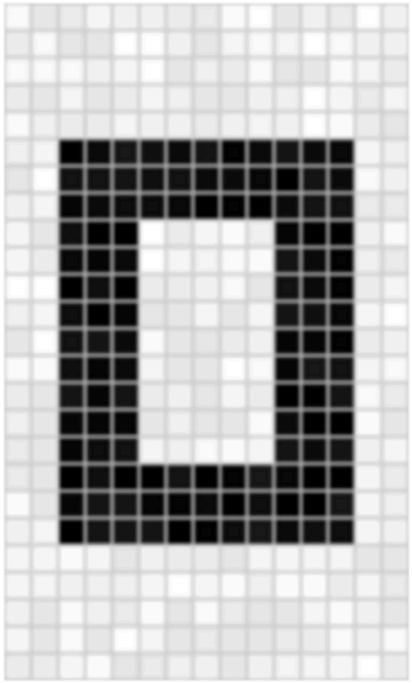
$\sigma_1 = 14.72$
$\sigma_2 = 5.22$
$\sigma_3 = 3.31$

參考資訊： https://ifun01.com/8V9KDF3.html

APPLICATION OF SVD – NOISE FILTER

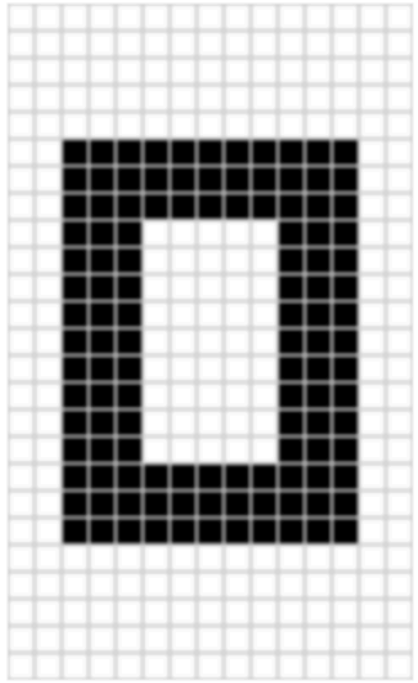
- 有一張 25×15 的圖片:

另一種更一般的情況，處理一張有雜訊的圖片：



它的奇異值為：

$\sigma_1 = 14.15$
$\sigma_2 = 4.67$
$\sigma_3 = 3.00$
$\sigma_4 = 0.21$
$\sigma_5 = 0.19$
...
$\sigma_{15} = 0.05$ $= 0$



參考資訊： https://ifun01.com/8V9KDF3.html

NUMPY EXAMPLE

Examples

```
>>> a = np.random.randn(9, 6) + 1j*np.random.randn(9, 6)
```

Reconstruction based on full SVD:

```
>>> U, s, V = np.linalg.svd(a, full_matrices=True)
>>> U.shape, V.shape, s.shape
((9, 9), (6, 6), (6,))
>>> S = np.zeros((9, 6), dtype=complex)
>>> S[:6, :6] = np.diag(s)
>>> np.allclose(a, np.dot(U, np.dot(S, V)))
True
```

Reconstruction based on reduced SVD:

```
>>> U, s, V = np.linalg.svd(a, full_matrices=False)
>>> U.shape, V.shape, s.shape
((9, 6), (6, 6), (6,))
>>> S = np.diag(s)
>>> np.allclose(a, np.dot(U, np.dot(S, V)))
True
```

參考資訊： https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.svd.html

GRAPH MINING

- Pagerank

GRAPH MINING

- Pagerank

INTRODUCTION OF PAGERANK

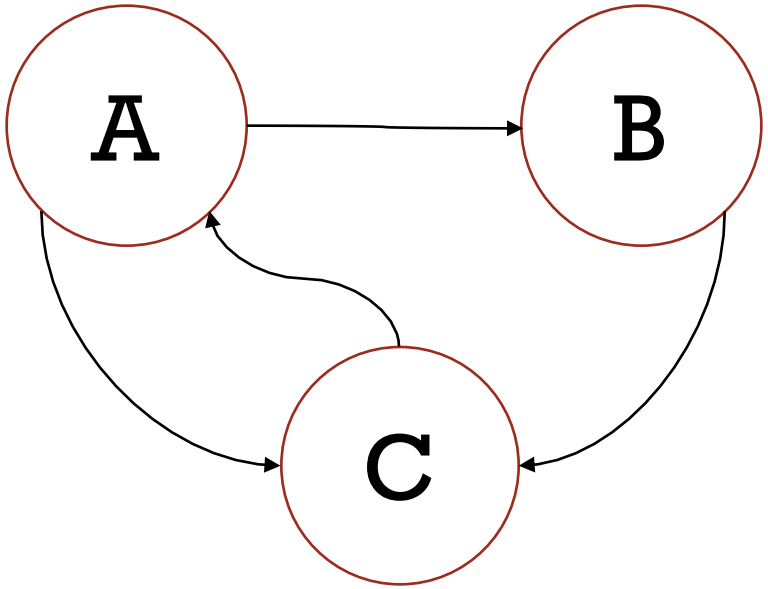
- 問題：
  - 早期搜尋引擎無法解決透過關鍵字搜尋之後的頁面排序？！

- 然而，被連結數易被操控，例如網站經營者可能為了提高自己的能見度而創 造大量垃圾連結指向同一目標網站，藉此提高被連結數。
- 另外，單純計算被連結 數，無法有效給予每個連結相對的權重，例如被獲得知名網站的連結與獲得一般 網站的連結，其重要性應該要有所區別。基於上述理由，在考量連結因素方面更 可靠的評估方法如 PageRank 便因運而生。

參考資訊： http://jpndbs.lib.ntu.edu.tw/DB/PageRank.pdf

HOW DOES PAGERANK WORK ?!

在這樣隨機瀏覽的過程中，受歡迎的網頁容易被看到，因為大多數的網頁傾向連結受歡迎的網頁；被受歡迎網頁連結的網頁，能見度也大幅提升。為了要達到此種重要性、連結結構、與能見度之間交互影響的結果，PageRank 的運算公式被設計為「一個網站的 PageRank 值，來自於加總所有連結到該網站的網站之 PageRank 值除以本身的導出連結數」。以下述情形為例，假設網路上僅有 A、B、C 三個網頁，其互相連結的關係如下圖：



參考資訊： http://tul.blog.ntu.edu.tw/

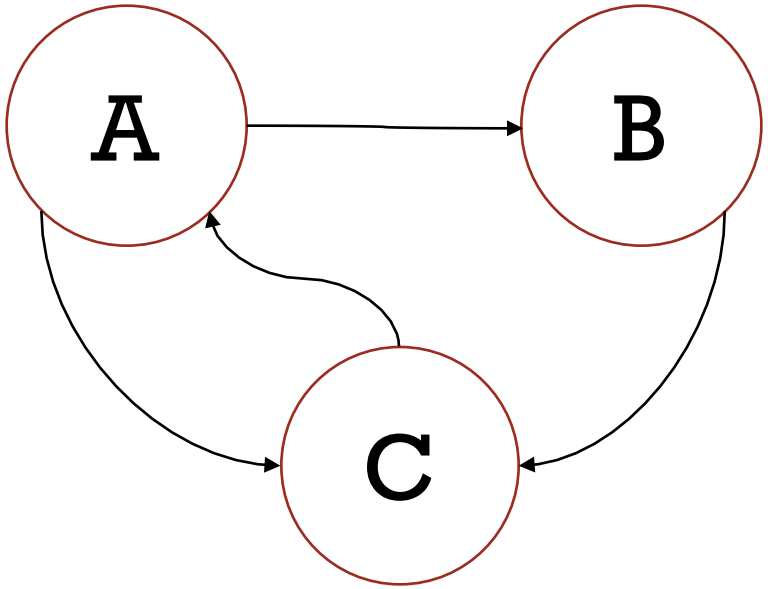
HOW DOES PAGERANK WORK ?! (CONT'D)

則 C 網站的 PageRank 值來自於連結到 C 的網站，也就是網站 A 和 B，將 A 和 B 的 PageRank 值除以其各自的導出連結數，再將此二數值加總，即可得 C 網站的 PageRank 值如下：

$$PR(C) = \frac{PR(A)}{2} + \frac{PR(B)}{1}$$

若將此公式其一般化，則可得 PageRank 值的計算方式為：

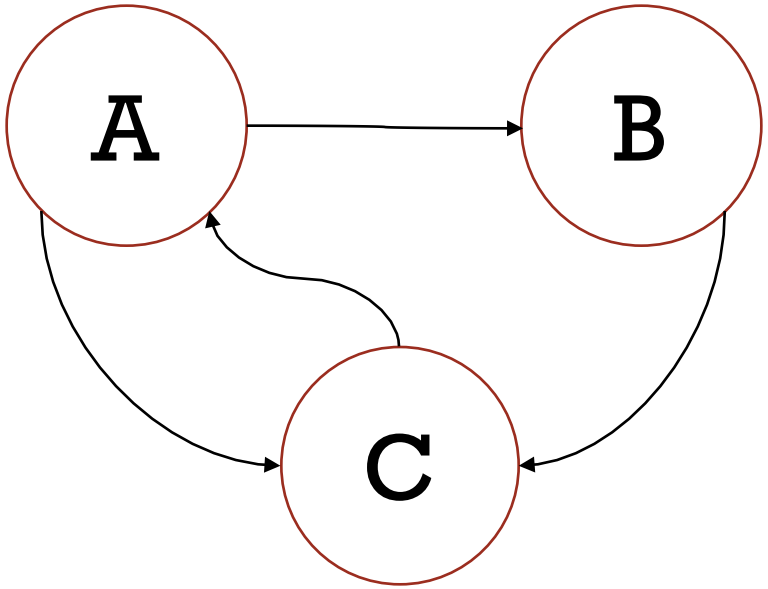
$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L_v}$$



參考資訊： http://tul.blog.ntu.edu.tw/

HOW DOES PAGERANK WORK ?! (CONT'D)

此公式是一個會收斂的運算。以上述例子而言，一開始假設每個網頁的 PageRank 值都是均等的，則計算方法如下(每階段的 PR 值使用前一階段的運算結果)：



(1) PR(A)=PR(B)=PR(C)=1/3=0.33

(2) PR(A)=0.33 PR(B)=0.33/2=0.17 PR(C)=0.33/2+0.33=0.5

(註：B 獲得 A 的連結，而 A 有 2 個導出連結，因此 PR(B)= PR(A)/2；C 獲得 A 和 B 的連結，A 有 2 個導出連結，B 只有 1 個，因此 PR(C)= PR(A)/2+ PR(B))

參考資訊： http://tul.blog.ntu.edu.tw/

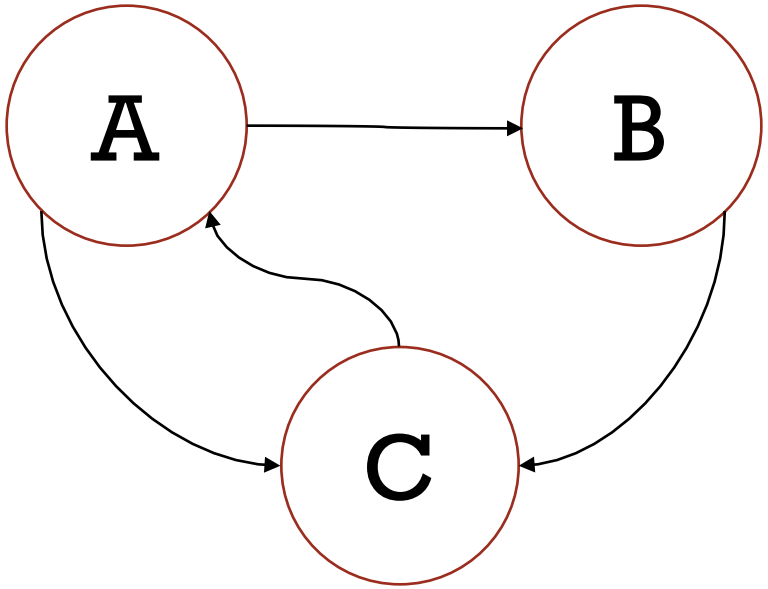
HOW DOES PAGERANK WORK ?! (CONT'D)

(3) PR(A)=0.5 PR(B)= 0.33/2=0.17 PR(C)=0.33/2+0.17=0.33

(註：A 獲得 C 的連結，C 只有 1 個導出連結，因此 PR(A)=前一階段 PR(C)的運算結果；B 獲得 A 的連結，而 A 有 2 個導出連結，因此 PR(B)=前一階段的 PR(A)/2；C 獲得 A 和 B 的連結，A 有兩個導出連結，B 只有一個，因此 PR(C)=前一階段 PR(A)/2+前一階段 PR(B))

(4) PR(A)=0.33 PR(B)=0.5/2=0.25 PR(C)=0.5/2+0.17=0.42

(5) 依此類推…



最後趨近：

PR(A)=0.4 PR(B)=0.2 PR(C)=0.4

參考資訊： http://tul.blog.ntu.edu.tw/

NETWROKX EXAMPLE

Compute pagerank with Python

The pageranks of the nodes in the example graph (see figure above) was computed in Python with the help of the *networkx* library, which can be installed with pip: `pip install networkx`. The code that creates a graph and computes pagerank is listed below:

```
import networkx as nx

# Initialize directed graph
G = nx.DiGraph()

# Add edges (implicitely adds nodes)
G.add_edge(1,6)
G.add_edge(2,6)
G.add_edge(3,6)
G.add_edge(4,6)
G.add_edge(5,6)
G.add_edge(4,5)
G.add_edge(6,7)

# Compute pagerank (keys are node IDs, values are pageranks)
pr = nx.pagerank(G)
"""
{
  1: 0.06242340798778012,
  2: 0.06242340798778012,
  3: 0.06242340798778012,
  4: 0.06242340798778012,
  5: 0.08895357136701444,
  6: 0.32374552689540625,
  7: 0.33760726978645894
}
"""
```

參考資訊： http://skipperkongen.dk/2016/08/16/how-to-compute-the-pagerank-of-almost-anything/

ANOMALY DETECTION & GRAPH MINING

- ChainSpot
- WebHound
- Playwright

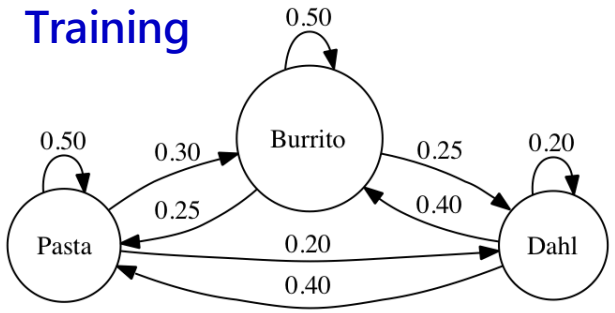
ANOMALY DETECTION & GRAPH MINING

- ChainSpot
- WebHound
- Playwright

CHAINSPOT: MINING SERVICE LOGS FOR CYBER SECURITY THREAT DETECTION

基於探勘服務日誌的個人化資安威脅偵測技術

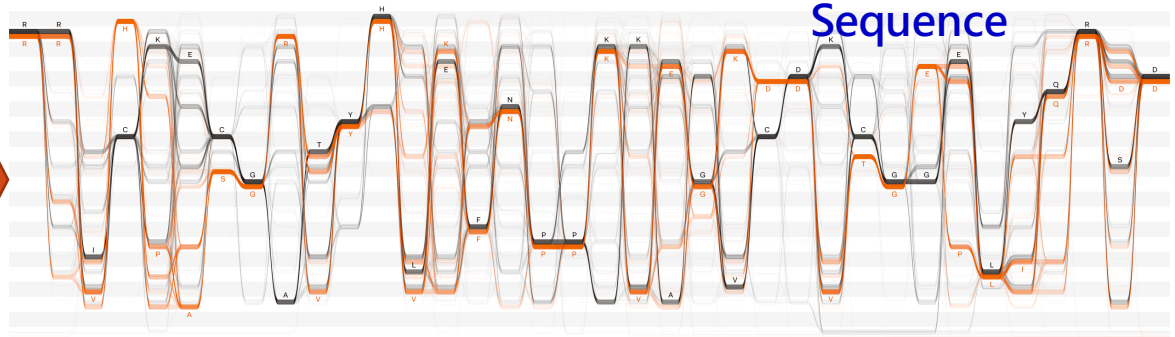
Markov Model Training



- *Problem*: Attackers usually invade industries and access sensitive data by the compromised employee.
- *Idea:* We focus on anomaly behavior detection for each employee (account) in an industry. 1) AD & Proxy Log Collection, 2) Behavioral Sequence Model, 3) Event Ticket Correlation, and 4) Anomaly Account Detection.
- *Contribution:* 85.66% average accuracy among 6 types of event tickets.

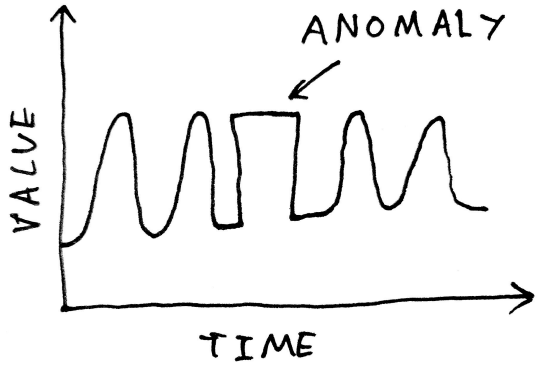
Anomaly Analysis of AD Account

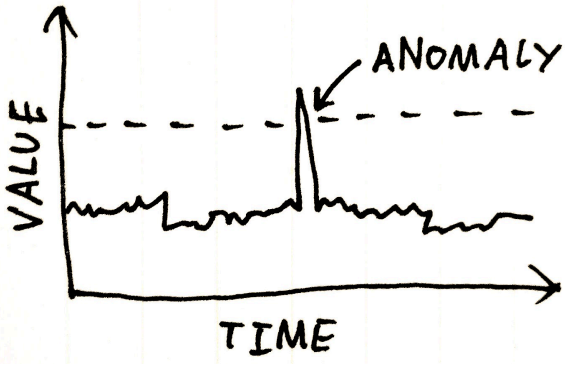
Account Behavioral Sequence

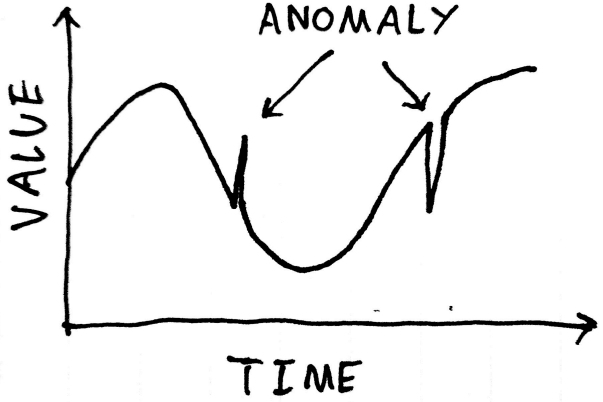


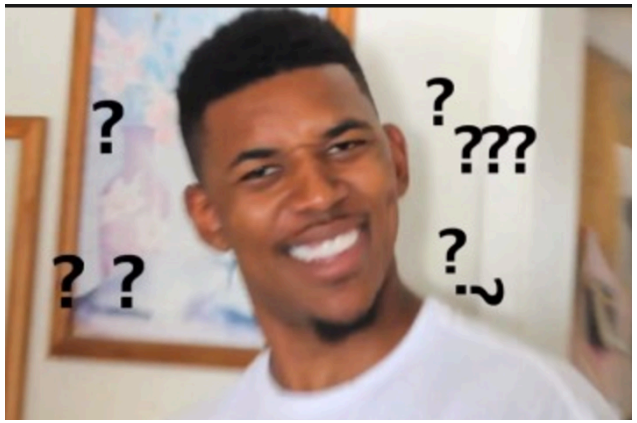
1. Obtain Ground Truth based on correlating AD account to Event Tickets.
2. Define Markov State based on AD events and Proxy activities.
3. Detect compromised accounts based on their behavioral change which are represented in Markov Model.

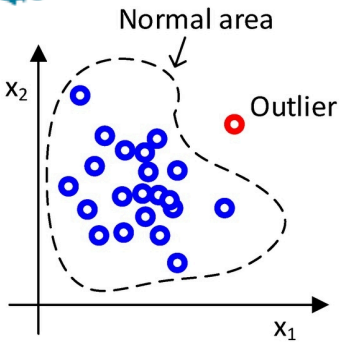
WHAT IS ANOMALY DETECTION ?!











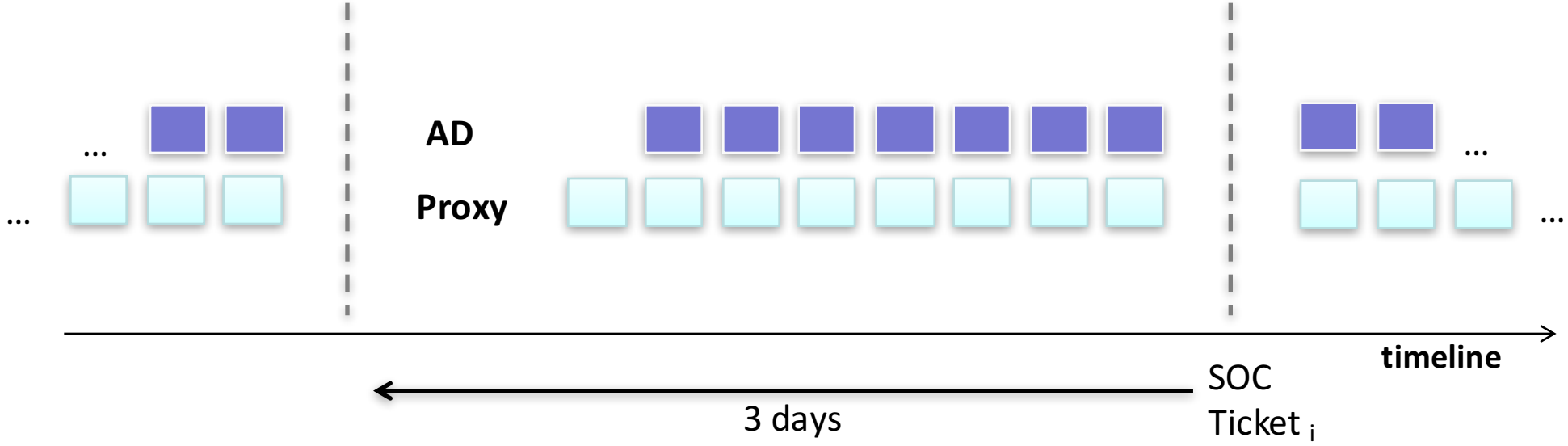
MOTIVATION

- Target of APT is usually specific and personal
  - Attackers usually invade enterprise and access sensitive data by using compromised accounts.

- We focus on detecting anomaly behavior or behavioral deviation for each employee (account) in an enterprise.

METHOD (1)

- We concern the anomaly behaviors extracted from Active Directory (Kerberos or NTLM authentication) & Proxy Server (web surfing usage) for each account.
- Sequential Data Synchronizer
  - Correlate heterogeneous data (AD and Proxy) based on IP Address and interval time (3 days) of SOC Tickets



METHOD (2)

- ChainSpot model sequential behaviors as a probabilistic model as baseline, and any change on employee's behaviors will results in an anomaly which may imply:
  - Account has been compromised
  - APT exists in an employee's host
- To properly model the sequential behaviors as a probabilistic model for each account, and to detect the anomalies based on deviation of account's behaviors.
  - Markov Model for building probabilistic model
  - Graph Edit Distance for estimating behavioral deviation

MARKOV MODEL

For each account, we will build his Markov model using his normal action sequences

Transition Probability Matrix

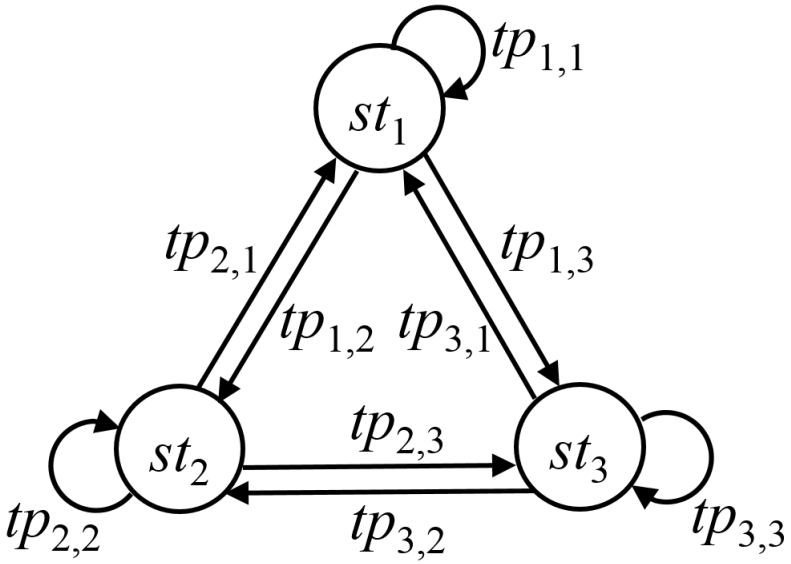
An $ns \times ns$ transition probability matrix (TPM), as following:

$$TPM = \begin{bmatrix} tp_{1,1} & \dots & tp_{1,j} & \dots & tp_{1,ns} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ tp_{i,1} & \dots & tp_{i,j} & \dots & tp_{i,ns} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ tp_{ns,1} & \dots & tp_{ns,j} & \dots & tp_{ns,ns} \end{bmatrix}$$

where for each i and j, $tp_{i,j}$ represents the transition probability from i^{th} state to j^{th} state, with constraints that $\sum_{j=1}^{ns} tp_{i,j} = 1$, and $i = 1, ..., ns$.

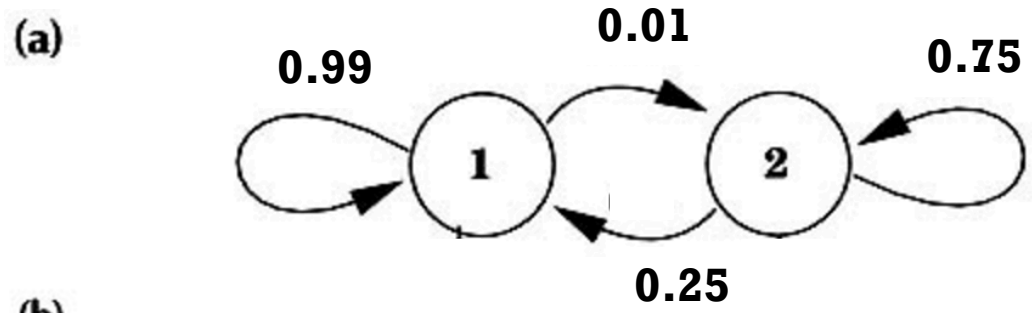
$$\forall\ i, j = 1, ..., ns,$$
$$tp_{i,j} = \frac{\#\text{transitions from } st_i \text{ to } st_j \text{ in } D_i}{\#\text{transitions starting from } st_i \text{ in } D_i}$$

An example of Markov Model



SEQUENTIAL BEHAVIORS TO MARKOV MODEL

- Markov States: 1, 2
- Markov Model:
  - P(1|1) = 0.99, P(2|1) = 0.01, P(1|2) = 0.25, P(2|2) = 0.75



99 x 1 → 1 : 0.99

3 x 2 → 2 : 0.75

1 x 1 → 2 : 0.01

1 x 2 → 1 : 0.25

GRAPH EDIT DISTANCE

- Graph Edit Distance is used to evaluate the difference between two Markov Models.

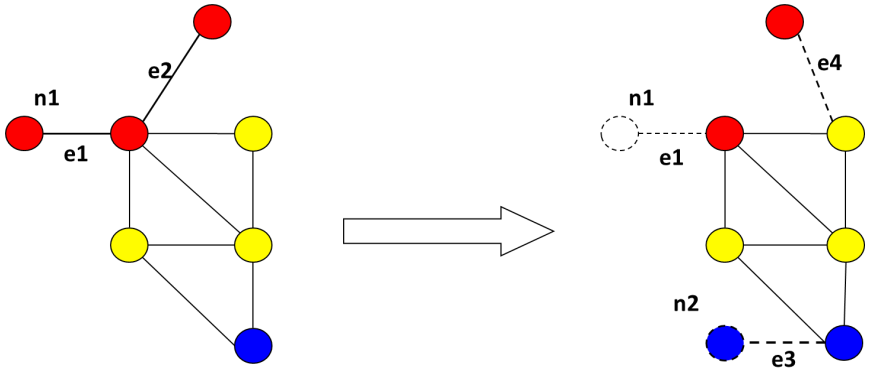
$$GED(G_1, G_2) = \min_{(e_1,\ldots,e_k)\in P(G_1,G_2)} \sum_{i=1}^{k} cost(e_i), \quad (1)$$

- Simplification of GED

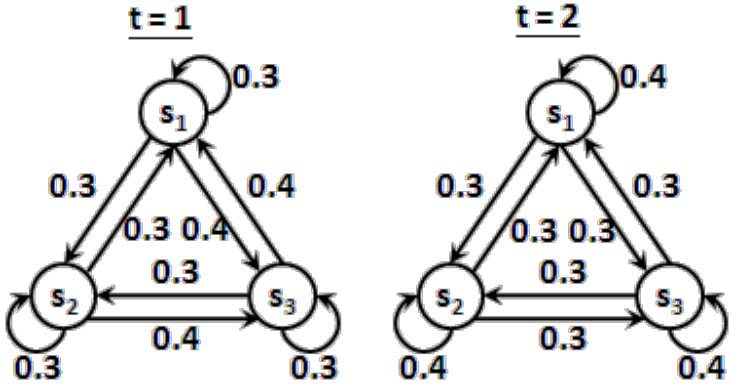
$$GED(TPM^1, TPM^2) = \sum_{i=1}^{ns}\sum_{j=1}^{ns} \left| tp_{i,j}^1 - tp_{i,j}^2 \right|, \quad (2)$$

HOW TO COMPARE THE DIFFERENCE BETWEEN TWO MARKOV MODELS ?!

- Graph Edit Distance (GED)
  - n1 + n2 + e1 + e2 + e3 + e4



- Examples
  - $P(S_1|S_1)$: **0.1** (0.4 - 0.3)
  - $P(S_1|S_3)$: **0.1** (0.4 - 0.3)
  - $P(S_3|S_1)$: **0.1** (0.4 - 0.3)
  - so on.



DATASET COLLECTIONS OF REAL WORLD (1)

- For each account, we build his personal profile of state sequences which describe this account's sequential behaviors.
- Each state in a sequential data consists of followings
  - For AD log sequence
    - Event (e.g., No 4624, 4634)
    - Reply Code (Only 4771 -> 0x12, 0x18)
  - For Proxy log sequence
    - A Meta Behavior describing web surfing.

      (e.g., GET and Download on microsoft.com then Failed)

      - HTTP Method
        - E,g., GET, POST, …
      - Download or Upload
        - Download when size in > size out
        - Upload when size in < size out
      - Domain Name
        - Second Level Domain
      - Access Result
        - E.g., Allowed, Failed, …

EXPERIMENT REGARDING TO REAL ENVIRONMENT

- Environment & Dataset Description:
  - Contains about 1,089 accounts.
  - Duration from 2015/08/01 to 2015/08/31.
  - The active directory domain service of Windows Servers ver. 2008-R2 results in 27,902,857 logs.
  - The proxy service in the same duration generates 78,044,332 logs.
- Dataset
  - Number of Categories in SOC Tickets: 6

Index	Type of SOC ticket	#tickets
1^{st}	Single account failed too many times when logon	4
2^{nd}	Multiple accounts logon from single IP in short time	5
3^{rd}	Host connects to malicious domain	11
4^{th}	Buffer overflow attack from outside	19
5^{th}	DoS attack from outside	59
6^{th}	Try to logon in non-working hours	1

EXPERIMENTS (CONT'D)

- Evaluation
  - We divide Aug. logs of each account into three partitions:
    - Training data
      - A half of unlabeled data mapped by SOC Tickets
    - Abnormal Testing data
      - Labeled data mapped by SOC Tickets
    - Normal Testing data
      - Selection from all unlabeled data except Training data
  - Compare the difference of Training data between Anomaly and Normal Testing data
  - Experiment Hypothesis

EFFECTIVENESS OF CHAINSPOT

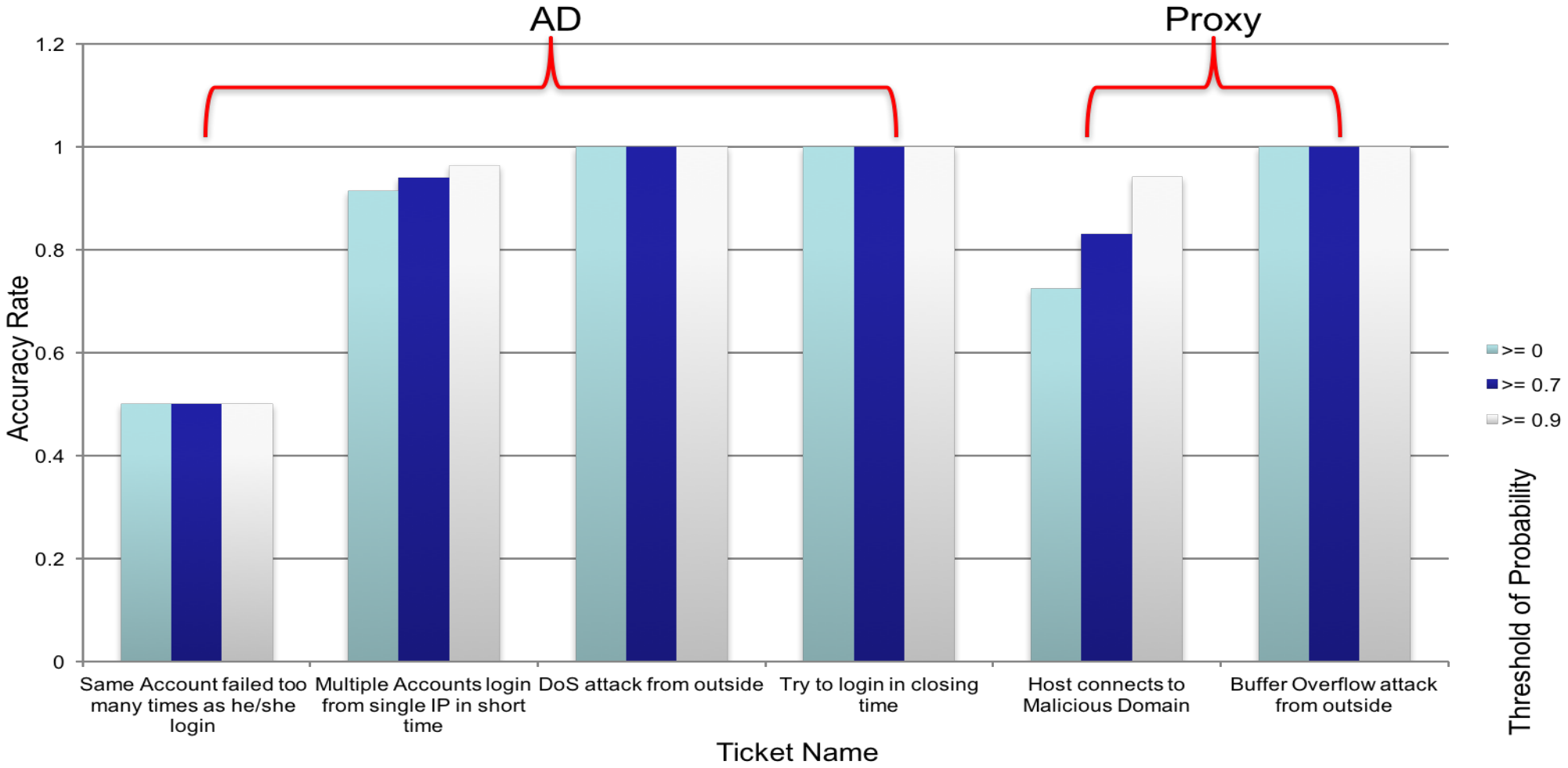
Hypothesis: $\forall\ i = 1, ..., E,\ i^{th}$ account is regarded as:
$Successful,$ If $GED(M^i_{abnor.}, M^i_{tr.}) > GED(M^i_{nor.}, M^i_{tr.})$.
$Failed,$ otherwise.

The general effectiveness measuring gives 85.66% and 87.17% success rates in terms of averaging on various types or averaging on different accounts.

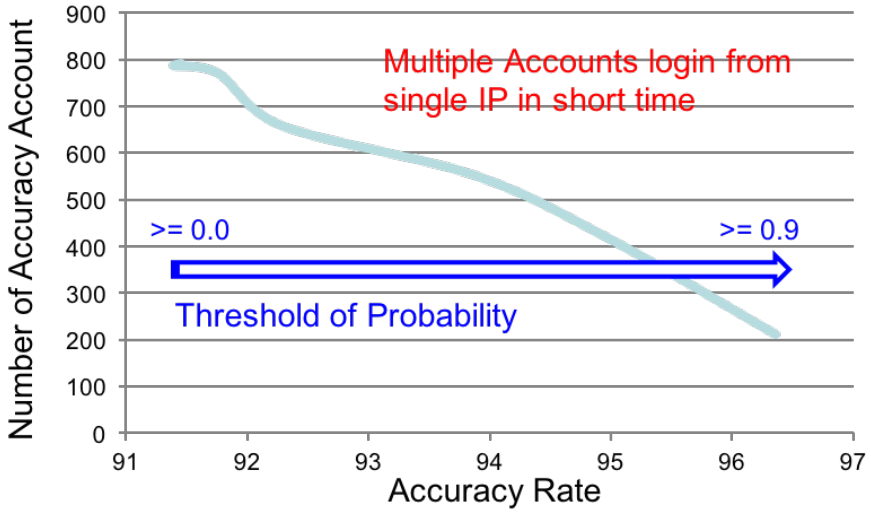
AD logs related			
Ticket index	#Related Accounts	#Succ. Accounts	Succ. Rate
1^{st}	2	1	50.00%
2^{nd}	865	791	91.45%
5^{th}	3	3	100.00%
6^{th}	2	2	100.00%

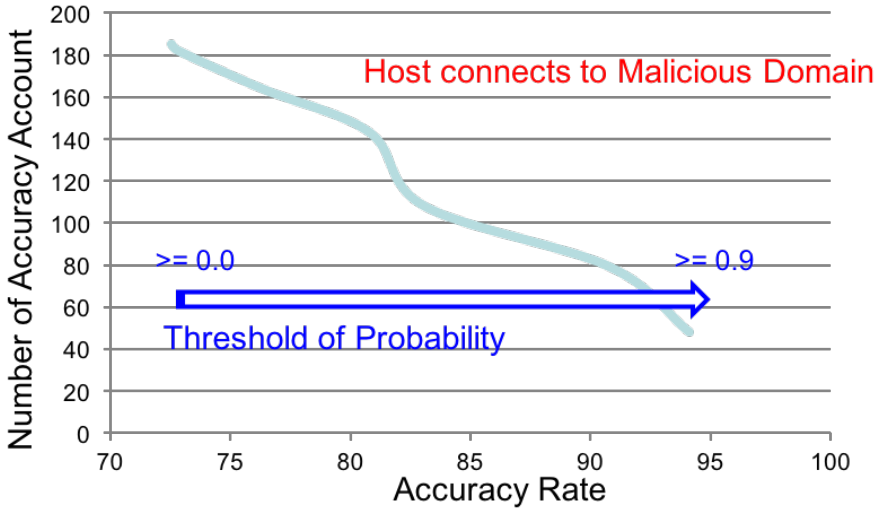
Proxy logs related			
Ticket index	#Related Accounts	#Succ. Accounts	Succ. Rate
3^{rd}	255	185	72.50%
4^{th}	3	3	100.00%

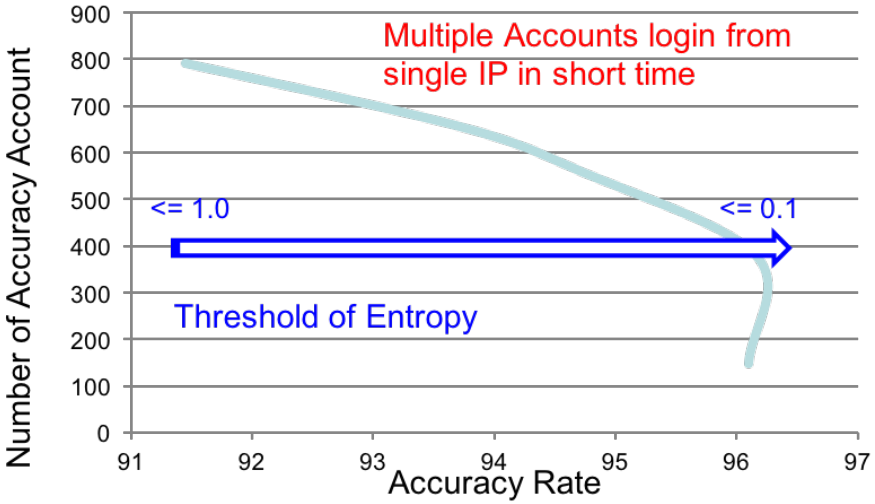
HOW TIGHT BETWEEN ACCOUNTS (AD, PROXY) AND IP

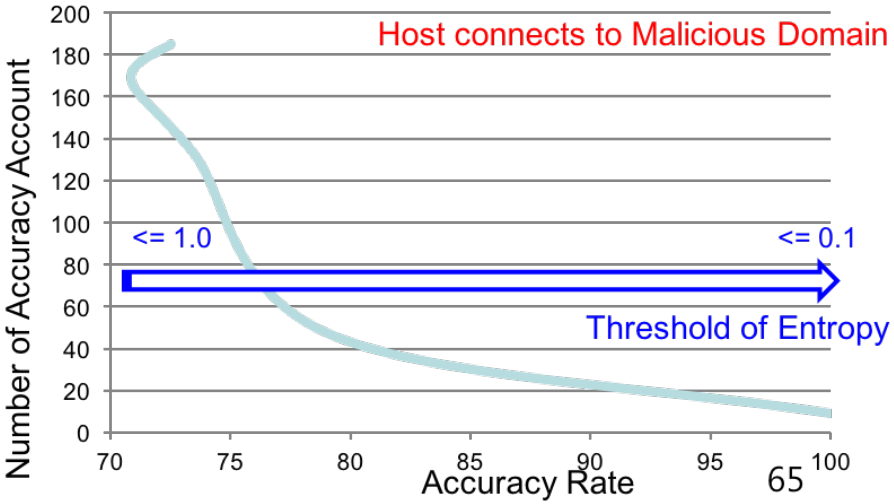


EXPERIMENT RESULTS (CONT'D)









PERFORMANCE OF CHAINSPOT

Hypothesis:

$$\begin{array}{l} Abnormal, \text{ once } GED(M^i_{tr.}, M^i_{unseen}) \geq \delta, \\ Normal, \text{ for otherwise,} \end{array} \quad (4)$$

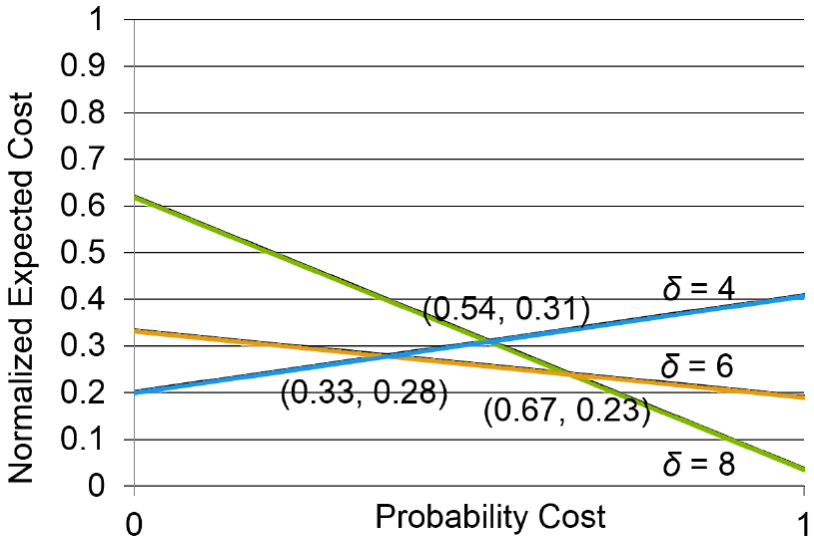
Generally speaking, ChainSpot deliver the well performance as prediction with 0.71 precision and 0.81 recall rates.

Measure-ments	$\delta = 4$	$\delta = 6$	$\delta = 8$
$\#TP$	1050	882	645
$\#FP$	674	362	218
$\#TN$	415	727	871
$\#FN$	39	207	444
$Precision$	0.61	0.71	0.75
$Recall$	0.96	0.81	0.59

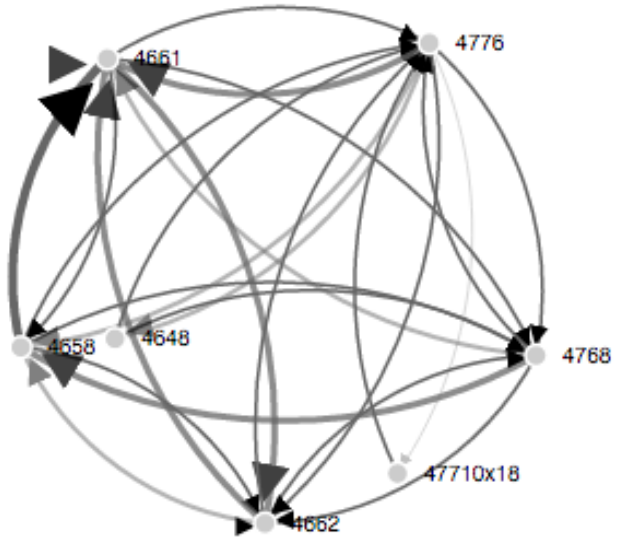
Cost curve helps optimally customize the sensitivity of ChainSpot to making alert.

$$PC(+) = \frac{p(+)cost(-|+)}{p(+)cost(-|+) + (1 - p(+))cost(+|-)}, \quad (5)$$

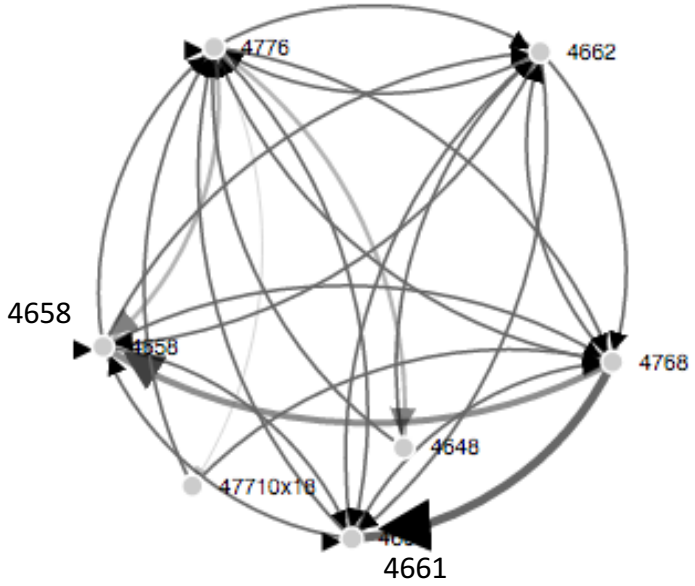
$$NEC = Rate_{FN} * PC(+) + Rate_{FP} * (1 - PC(+)), \quad (6)$$



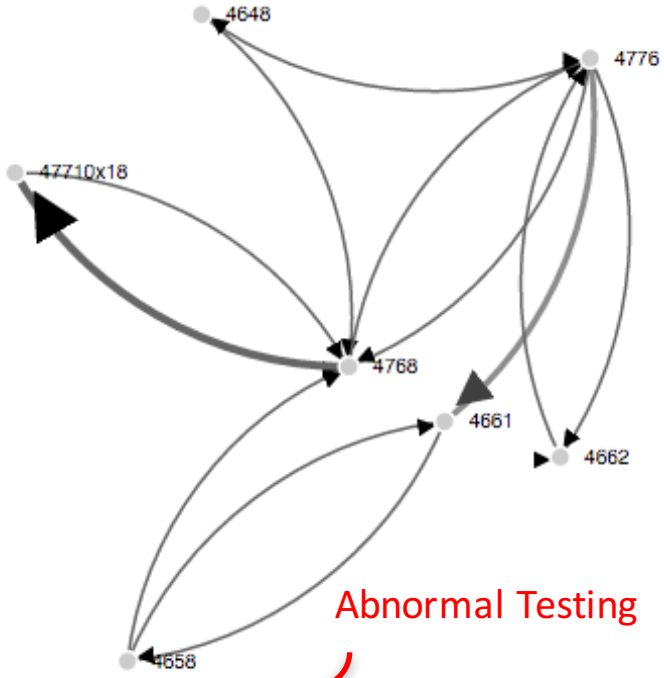
AD CASE STUDY - MULTIPLE ACCOUNTS LOGIN FROM SINGLE IP IN SHORT TIME



Normal Testing



Training



Abnormal Testing

Abnormal Data has no 4661 -> 4662, 4662 -> 4658

A handle to an object was requested (4661)

↓

An operation was performed on an object (4662)

↓

The handle to an object was closed (4658)

Abnormal Data has 4768 -> 4771 0x18, 4771 0x18 -> 4768

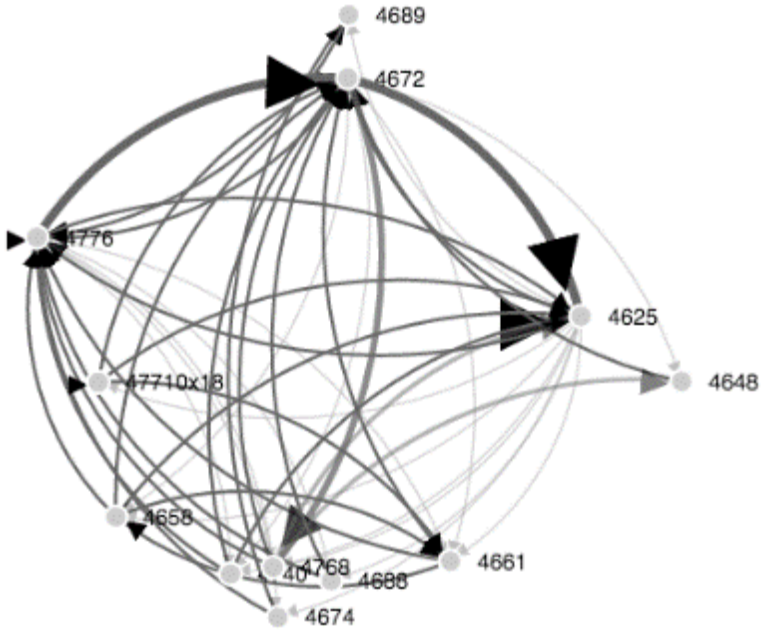
A Kerberos authentication ticket (TGT) was requested (4768)

↓ ↑

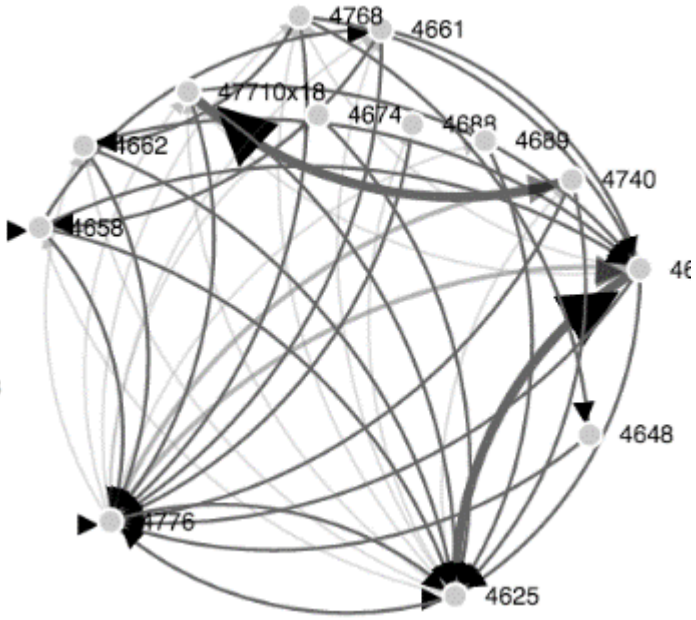
Kerberos pre-authentication failed (Bad Password) (4771 0x18)

There is no complete service path in abnormal data, and which has error authentication by bad password

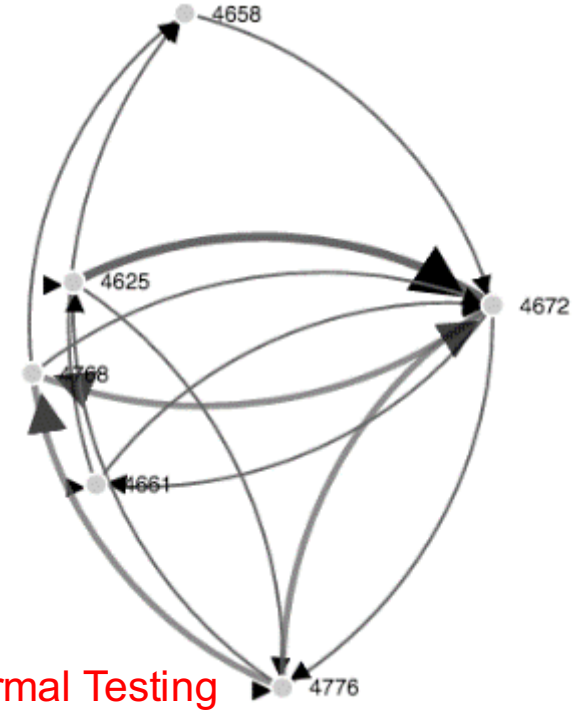
AD CASE STUDY - TOO MANY LOGINS IN CLOSING TIME



Normal Testing



Training



Abnormal Testing

Abnormal Data has no followings :

An account failed to log on (4625)

- The handle to an object was closed (4658)
  A handle to an object was requested (4661)
- A new process has been created (4688)
- A Kerberos authentication ticket (TGT) was requested (4768)
  Kerberos pre-authentication failed (Bad PWD) (4771 0x18)
  A user account was locked out (4740)

Abnormal and Normal Data both have followings :

An account failed to log on (4625)

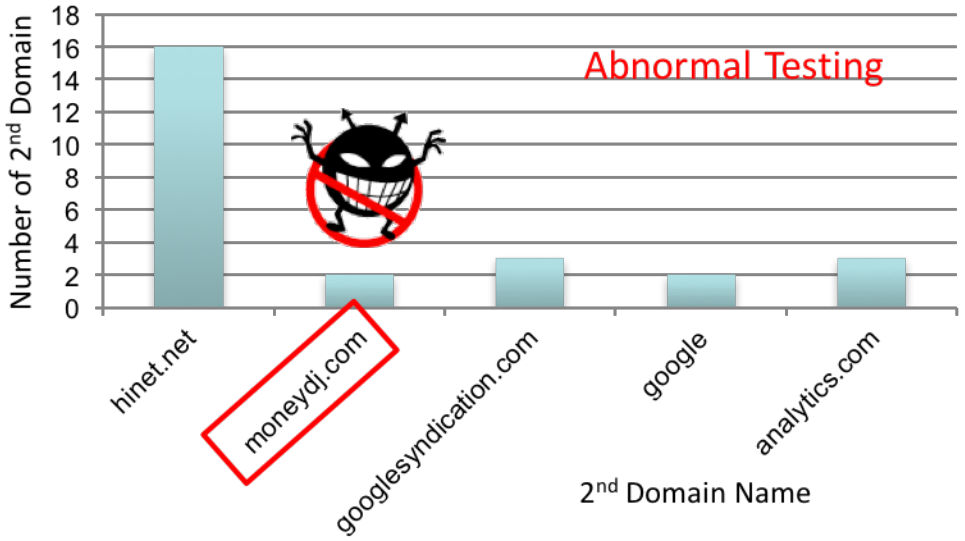
- Special privileges assigned to new logon (4672)
- The domain controller attempted to validate the credentials for an account (4776)

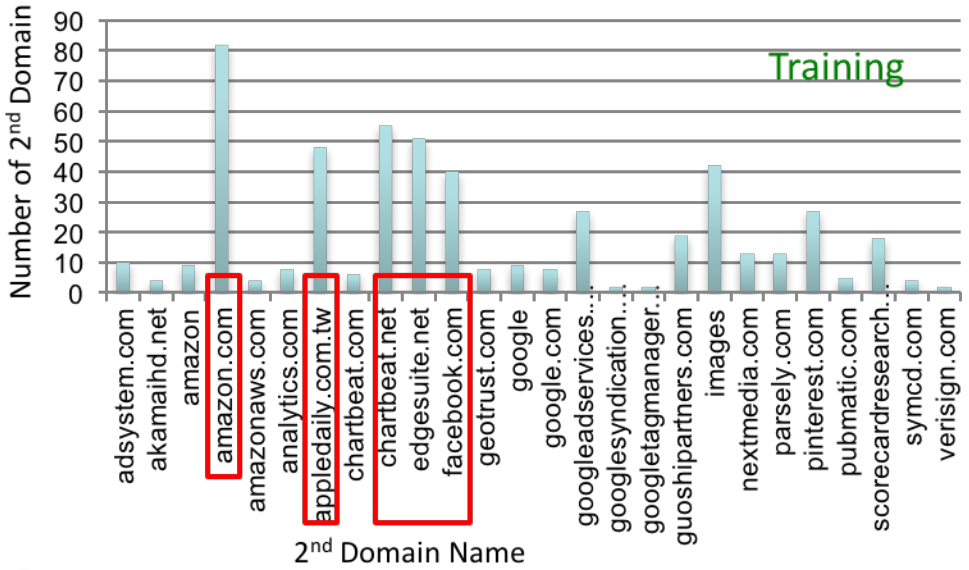
PROXY CASE STUDY - HOST CONNECTS TO MALICIOUS DOMAIN

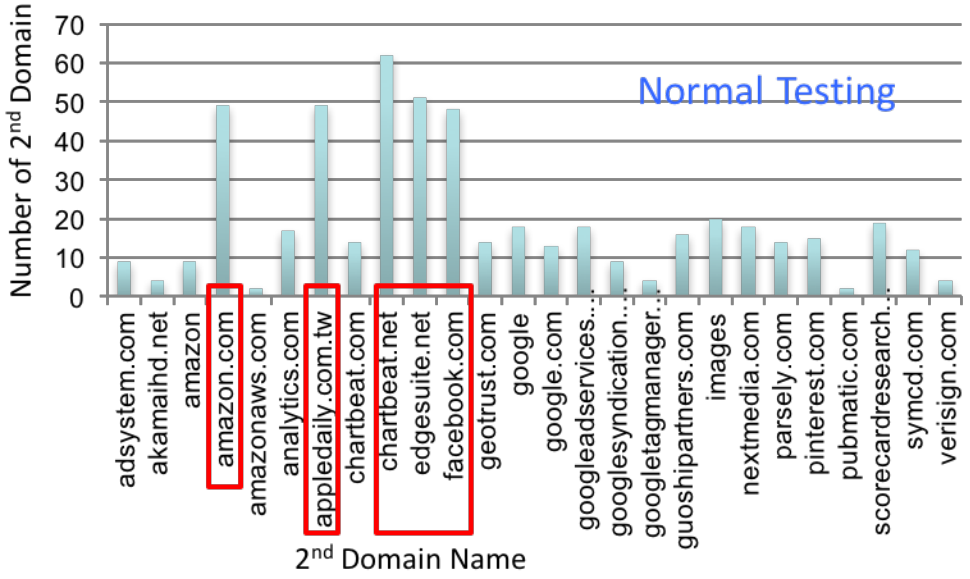
1. Abnormal Testing has ***five different*** 2nd Level Domains, especially ***hinet.net***

2. moneydj.com***blacklist***

3. In Normal Testing and Training, user usually browses ***five*** 2nd Level Domains which are ***never*** appear in Abnormal Testing (Ex. amazon.com, facebook,com, and so on)





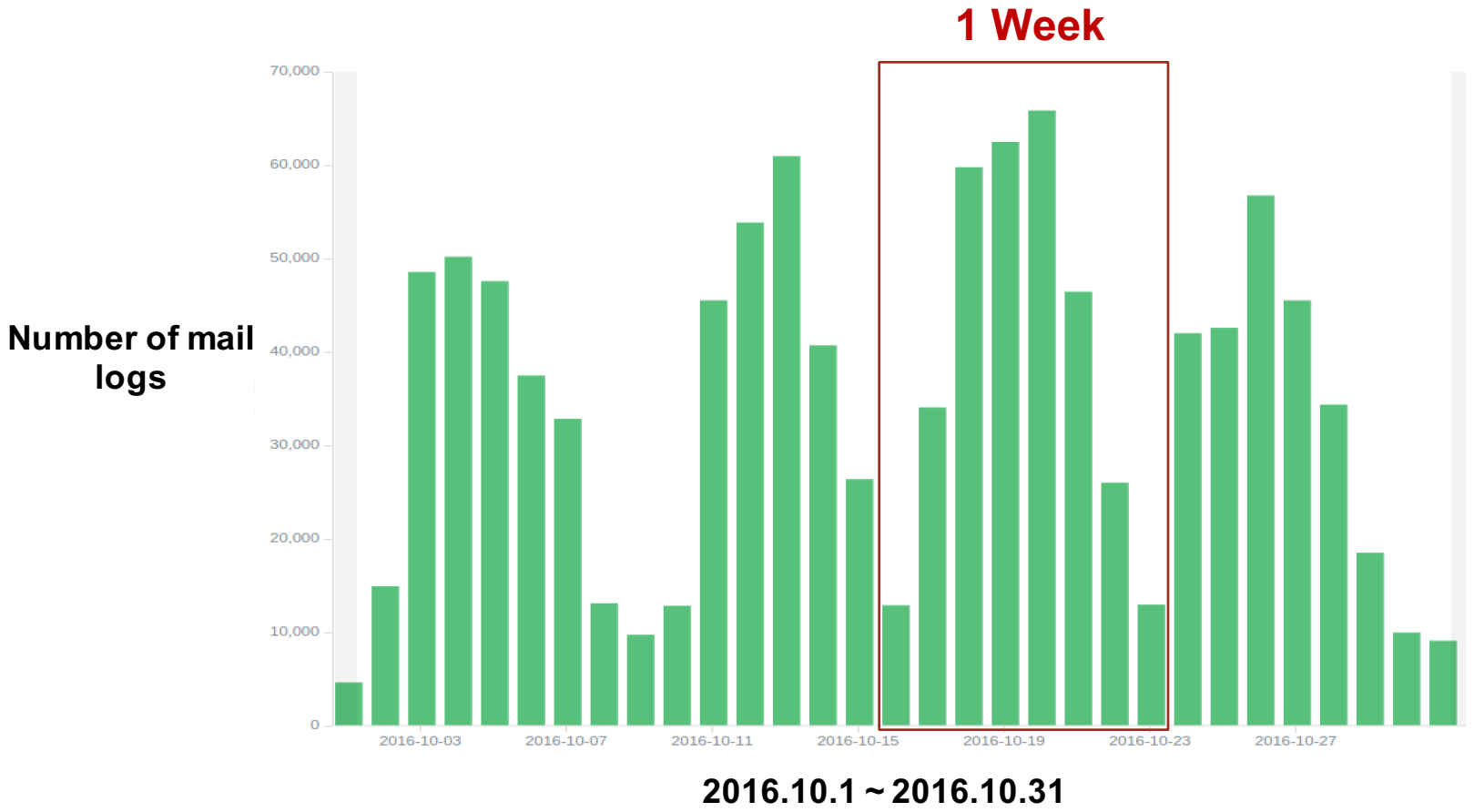


DATASET COLLECTIONS OF REAL WORLD (2)

- We collected e-mail logs of an university from Oct. to Dec. (3 moths)
  - Account Login Logs
    - Docs : 156,840,553
    - Size : 15.13 GB
    - Schema
      - Service, Account, Server IP, Client IP, Device, City, Region, Country, Timestamp
  - Sender / Receiver Logs
    - Docs : 1,229,039
    - Size : 195 MB
    - Schema
      - Client IP, Server IP, Sender, Subject, Receiver, Subject, Mail Time

HOW LONG SHOULD WE TRAIN A MARKOV MODEL ?!

- Depend on accounts' number of mail logs in each day during 3 months.



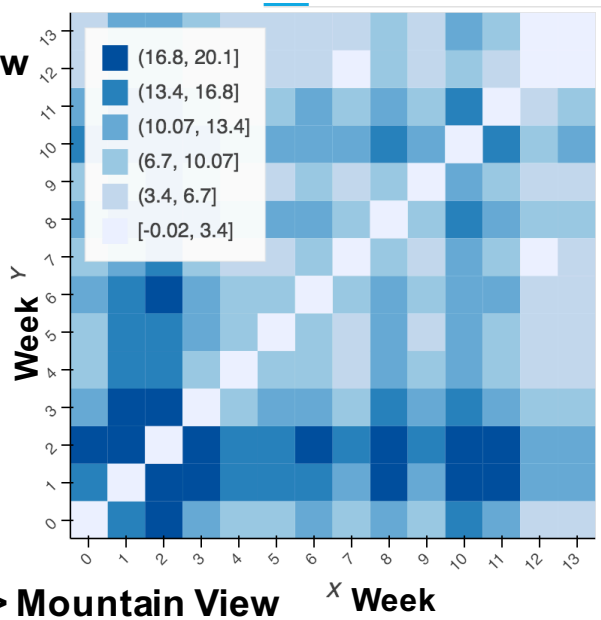
MARKOV EXPERIMENT RESULTS – CASE 1

IP + Protocol + Location

Mountain View -> Mountain View


```
{
  "state :  , 209.85.217.X , pop3 , Mountain View": {
    "state :  , 209.85.217.X , pop3 , Mountain View": 0.574468085106383,
    "state :  , 209.85.213.X , pop3 , Mountain View": 0.425531914893617
  },
  "state :  , 209.85.213.X , pop3 , Mountain View": {
    "state :  , 209.85.217.X , pop3 , Mountain View": 0.7272727272727273,
    "state :  , 209.85.213.X , pop3 , Mountain View": 0.2727272727272727
  }
}
```


Week12




```
{
  "state :  , 209.85.217.X , pop3 , Mountain View": {
    "state :  , 209.85.217.X , pop3 , Mountain View": 0.7333333333333333,
    "state :  , 209.85.213.X , pop3 , Mountain View": 0.26666666666666666
  },
  "state :  , 209.85.213.X , pop3 , Mountain View": {
    "state :  , 209.85.217.X , pop3 , Mountain View": 1.0
  }
}
```


Week13

Mountain View -> Mountain View


```
  "state :  , 124.89.13.X , bogon , Xian": {
    "state :  , 124.89.13.X , ip73-164.vpnip.cc.ntu.edu.tw , Xian": 0.07142857142857142,
    "state :  , 124.89.13.X , bogon , Xian": 0.14285714285714285,
    "state :  , 124.89.13.X , ip73-141.vpnip.cc.ntu.edu.tw , Xian": 0.07142857142857142,
    "state :  , 209.85.213.X , pop3 , Mountain View": 0.5,
    "state :  , 124.89.13.X , ip73-207.vpnip.cc.ntu.edu.tw , Xian": 0.07142857142857142,
    "state :  , 124.89.13.X , ip73-61.vpnip.cc.ntu.edu.tw , Xian": 0.14285714285714285
  },
```


week2 Xian (西安) -> Xian (西安)

MARKOV EXPERIMENT RESULTS – CASE 2

(間隔時間 HR , 間隔距離 海哩)

```
"(2, 0)": {
  "(2, 0)": 0.8407079646017699,
  "(1, 0)": 0.1592920353982301
},
```

```
"1481340014": [
  {
    "city": "Mountain View",
"1481348544": [
  {
    "city": "Mountain View",
```

```
"1481344454": [
  {
    "city": "Mountain View",
```

一小時多登入一次

Week10

```
"(2, 0)": {
  "(3, 0)": 0.008403361344537815,
  "(2, 0)": 0.8235294117647058,
  "(1, 0)": 0.16806722689075632
},
```

```
"1481911182": [
  {
    "city": "Mountain View",
"1481919369": [
  {
    "city": "Mountain View",
```

```
"1481915011": [
  {
    "city": "Mountain View",
```

一小時多登入一次

Week11

```
"(2, 0)": {
  "(3, 0)": 0.05263157894736842,
  "(10, 0)": 0.02631578947368421,
  "(7, 0)": 0.02631578947368421,
  "(17, 0)": 0.02631578947368421,
  "(1, 0)": 0.07894736842105263,
  "(2, 0)": 0.7894736842105263
}
```

```
"1480183351": [
  {
    "city": "Mountain View",
```

```
"1480191140": [
  {
    "city": "Mountain View",
```

兩次登入時間差了快3小時

```
"1480195233": [
  {
    "city": "Mountain View",
```

```
"1480203868": [
  {
    "city": "Mountain View",
```

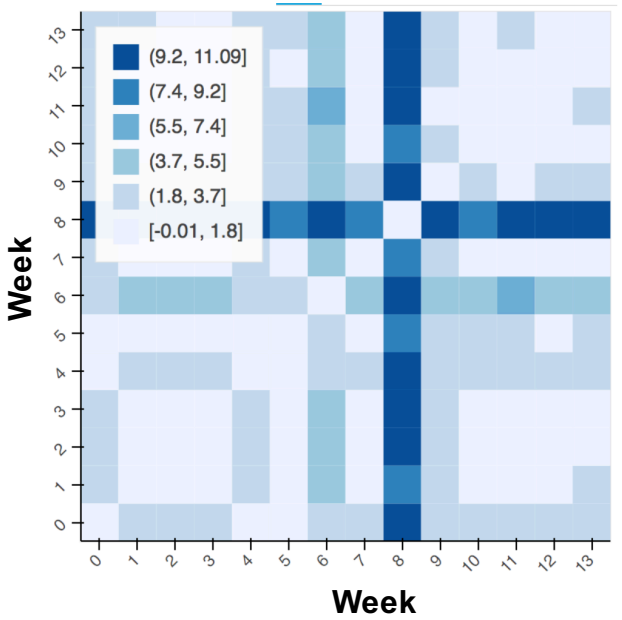
兩次登入時間差了快3小時

```
"1480294979": [
  {
    "city": "Mountain View",
```

```
"1480327981": [
  {
    "city": "Mountain View",
```

兩次登入時間差了快10小時

Week8



MARKOV EXPERIMENT RESULTS – CASE 2

(間隔時間 HR , 間隔距離 海哩)


```
"(1, 0)": {
  "(2, 0)": 0.04084720121028744,
  "(1, 0)": 0.9591527987897126
}
```



```
"1483127491": [
  {
    "city": "Taipei",
```


台北 -> 台北


```
"1483130865": [
  {
    "city": "Taipei",
```


Week12


```
"(1, 0)": {
  "(2, 0)": 0.02654867256637168,
  "(1, 0)": 0.9734513274336283
}
```

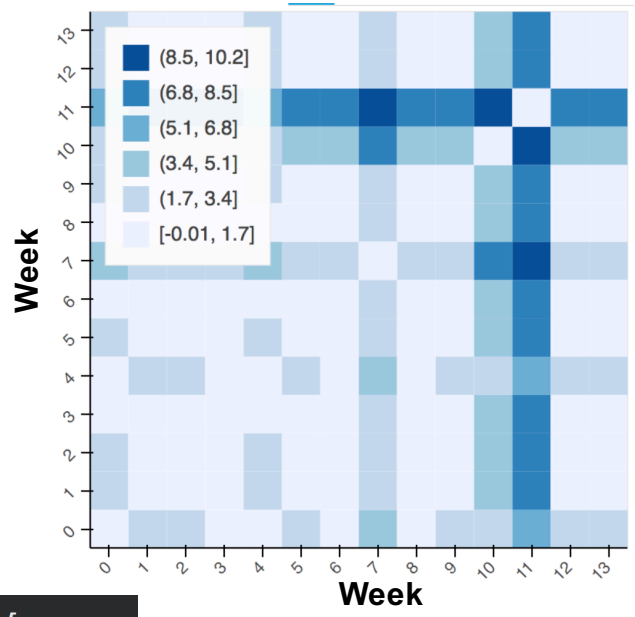


```
"1483142129": [
  {
    "city": "Taipei",
```


台北 -> 台北


```
"1483144014": [
  {
    "city": "Taipei",
```


Week13




```
"(1, 0)": {
  "(3, 0)": 0.00522466039707419,
  "(4, 0)": 0.0020898641588296763,
  "(1, 11)": 0.0020898641588296763,
  "(2, 13)": 0.0010449320794148381,
  "(1, 0)": 0.9592476489028213,
  "(1, 13)": 0.0020898641588296763,
  "(1, 12)": 0.008359456635318705,
  "(2, 0)": 0.017763845350052248,
  "(11, 11)": 0.0010449320794148381,
  "(8, 0)": 0.0010449320794148381
},
```



```
"1482419197": [
  {
    "city": "Sai Ying Pun",
```


西營盤, 上海 -> Absecon, New Jersey


```
"1482419859": [
  {
    "city": "Absecon",
```


Week11


```
"1482442499": [
  {
    "city": "Guangzhou",
```


廣州 -> 台北

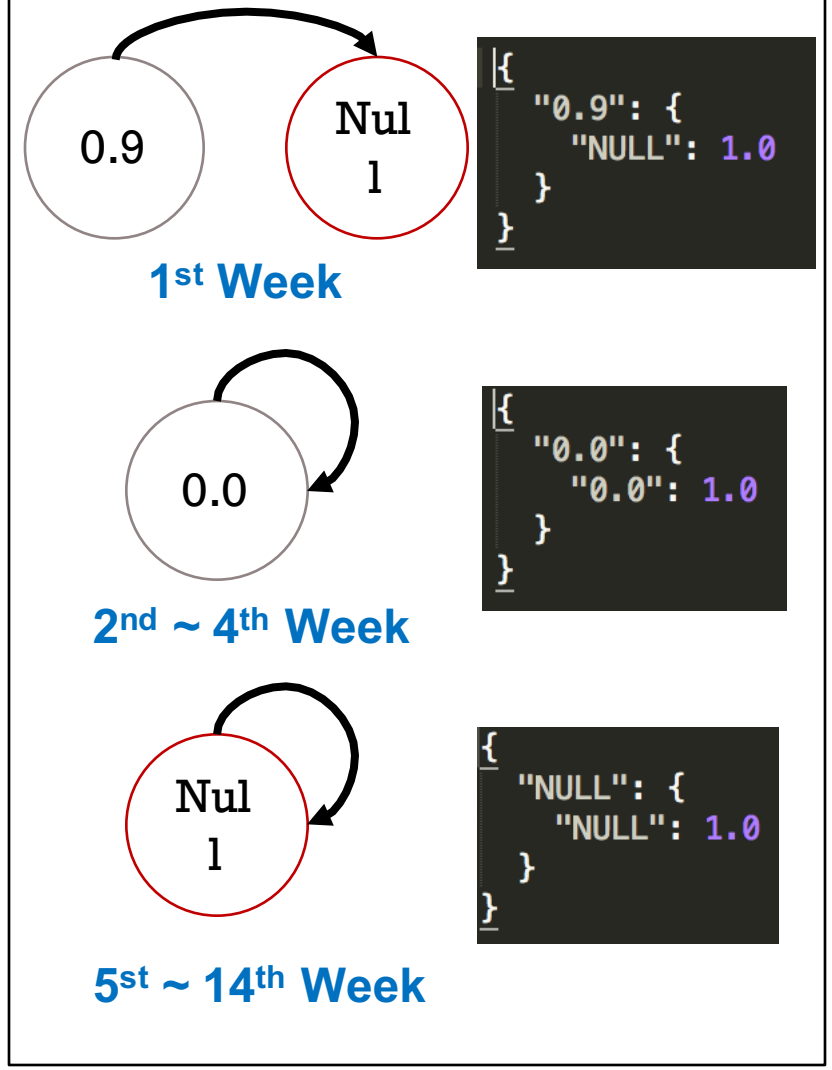

```
"1482451919": [
  {
    "city": "Taipei",
```

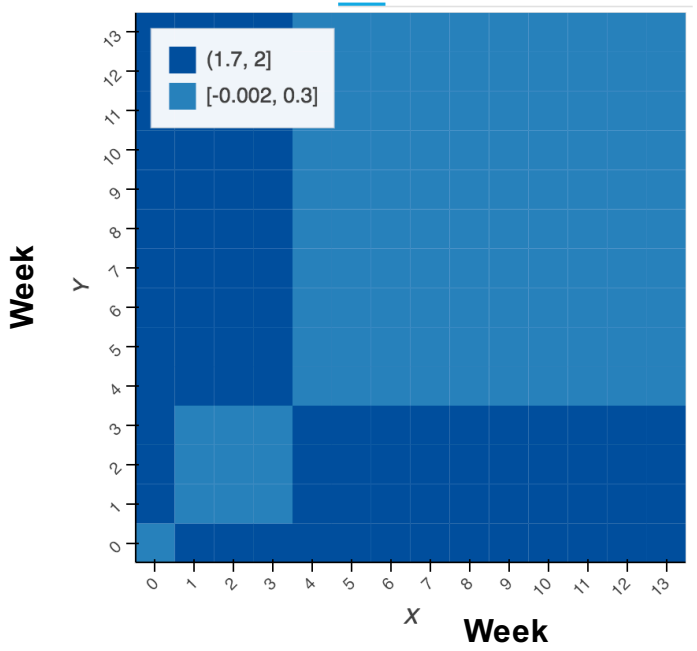
MARKOV EXPERIMENT RESULTS – CASE 3

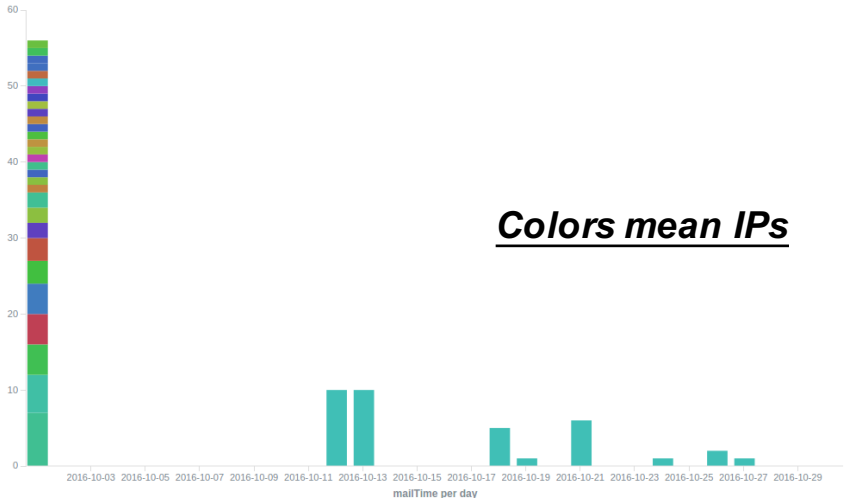

```
{
    "118.170.1.169": 1.0,
    "27.76.6.111": 1.0,
    "119.76.137.252": 7.0,
    "187.1.13.213": 1.0,
    "104.42.232.136": 1.0,
    "111.251.100.82": 1.0,
    "74.116.186.87": 1.0,
    "14.173.28.200": 3.0,
    "77.169.193.71": 1.0,
    "116.111.125.229": 1.0,
    "111.250.131.14": 1.0,
    "201.220.183.105": 1.0,
    "180.164.119.39": 5.0,
    "8.26.250.180": 1.0,
    "208.180.216.131": 2.0,
    "114.36.80.173": 1.0,
    "5.44.113.77": 4.0,
    "111.255.140.145": 4.0,
    "113.175.204.102": 1.0,
    "14.177.137.173": 2.0,
    "186.216.247.26": 4.0,
    "120.43.253.220": 1.0,
    "113.190.210.177": 1.0,
    "125.77.65.99": 1.0,
    "115.84.90.157": 1.0,
    "177.128.32.128": 2.0,
    "27.105.239.96": 1.0,
    "113.177.135.91": 1.0,
    "212.164.32.7": 1.0,
    "121.63.59.34": 3.0
}
```


IP List in 1st Week

Entropy of IPs



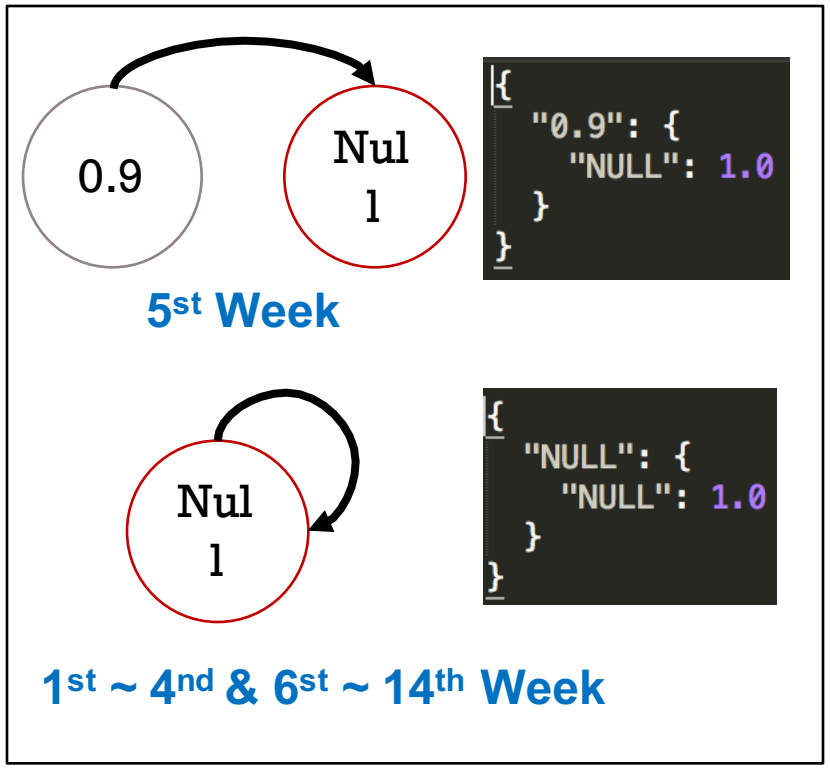




Colors mean IPs

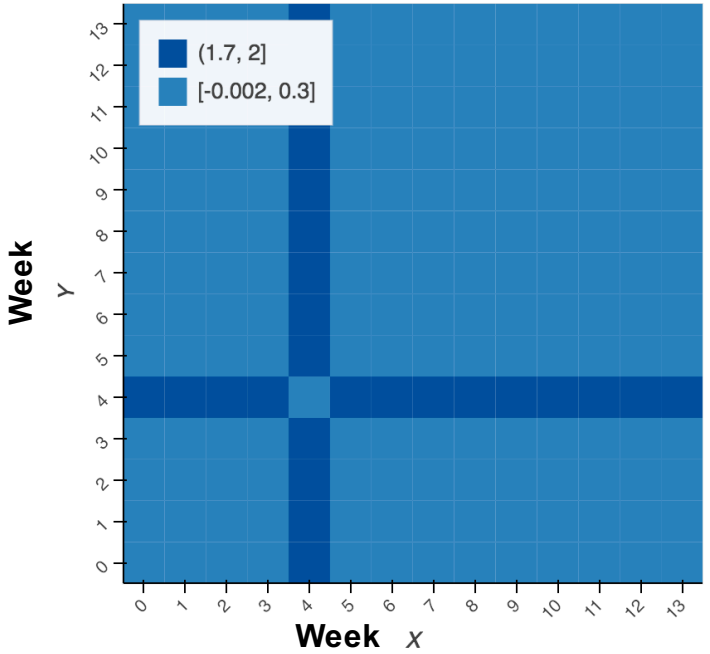
MARKOV EXPERIMENT RESULTS – CASE 3

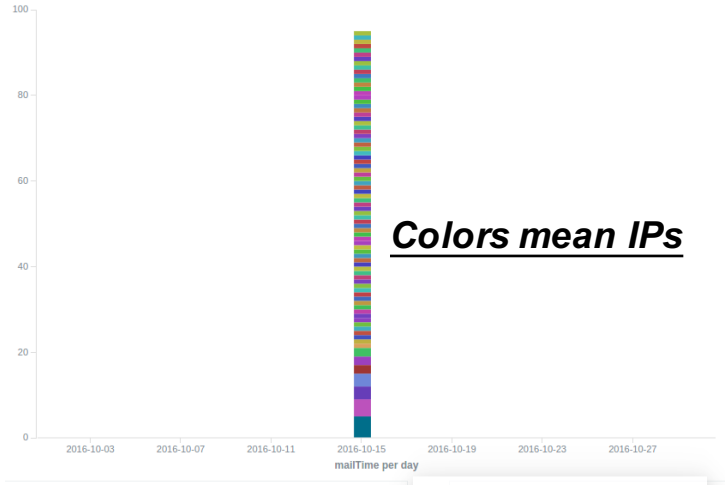

```
{
    "85.234.189.213": 30.0,
    "109.160.79.238": 11.0,
    "62.249.178.245": 16.0,
    "131.72.131.41": 73.0,
    "37.26.32.119": 16.0,
    "109.199.8.122": 66.0,
    "95.65.50.243": 16.0,
    "46.49.42.167": 6.0,
    "130.204.54.240": 17.0,
    "77.70.96.251": 10.0,
    "46.238.40.151": 17.0,
    "43.227.246.6": 22.0,
    "81.191.87.156": 83.0,
    "188.138.138.36": 11.0,
    "186.71.212.39": 31.0,
    "181.175.161.138": 26.0,
    "113.166.175.119": 11.0,
    "85.226.142.42": 1.0,
    "197.248.186.26": 11.0,
    "190.52.35.32": 11.0,
    "84.238.98.194": 22.0,
    "78.90.136.246": 11.0,
    "91.139.189.127": 9.0,
    "186.227.27.6": 1.0,
    "78.90.122.16": 111.0,
    "138.255.209.140": 26.0,
    "109.87.30.196": 46.0,
    "92.247.144.70": 18.0,
    "124.123.214.254": 10.0,
    "85.239.158.8": 30.0,
    "137.59.194.182": 11.0,
    "91.200.195.204": 10.0,
    "103.18.21.86": 6.0,
    "124.41.238.24": 18.0,
    "46.9.223.197": 27.0,
    "94.52.191.115": 8.0,
    "31.148.252.136": 39.0,
    "138.99.10.191": 30.0,
    "87.100.180.178": 6.0,
    "190.155.215.200": 27.0,
    "130.204.107.165": 8.0,
    "82.140.152.249": 14.0,
    "78.90.102.108": 8.0,
    "212.5.37.47": 11.0,
    "59.153.37.182": 35.0
}
```

Entropy of IPs

IP List in 5st Week



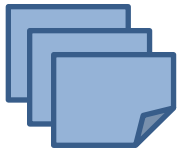


Colors mean IPs

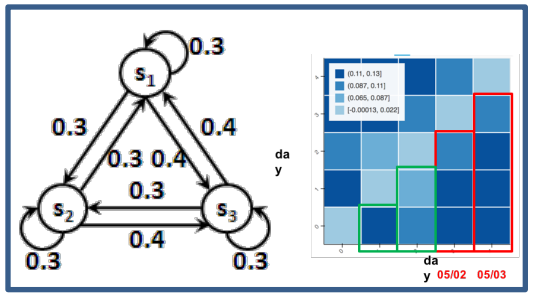
DATASET COLLECTIONS OF REAL WORLD (3)

- Data:
  - – 05/02、05/03、05/22、05/23、05/24
    - – Each account have several EventID sequences that extracted from 5 days
    - – Each Segment is a length-100 event code sequences.
- Goal: Find anomaly sequence between those days

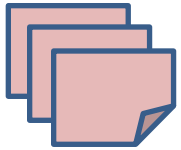
A 2-STAGED ANOMALY ANALYSIS FRAMEWORK



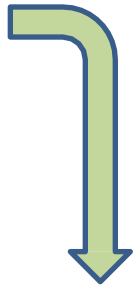
Logs of All Accounts

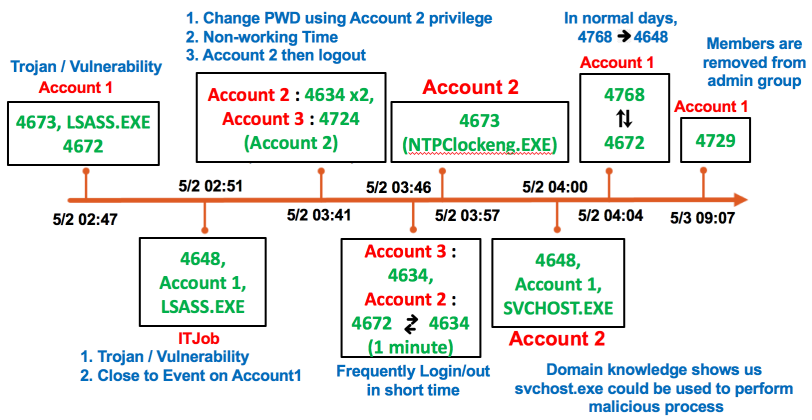


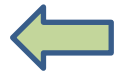
1st Stage: Highlight abnormal accounts as suspect candidates.

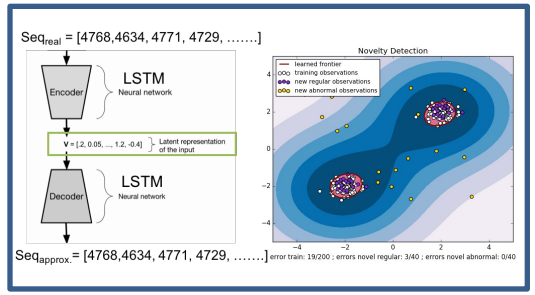


Logs of Suspect Accounts





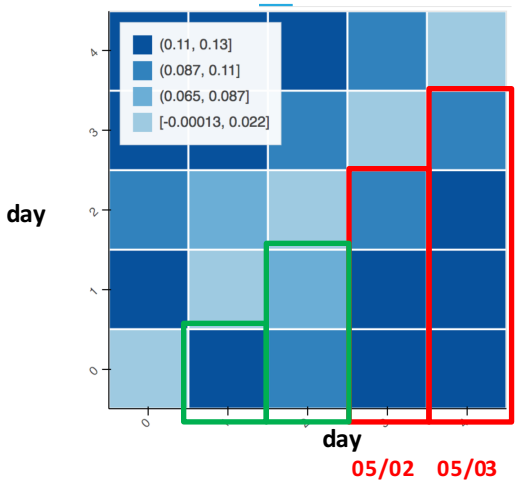




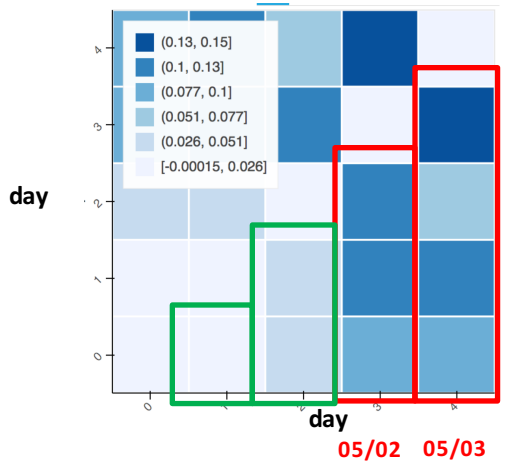
2nd Stage: Further track behaviors of suspect candidates along the timeline.

Summarize our report

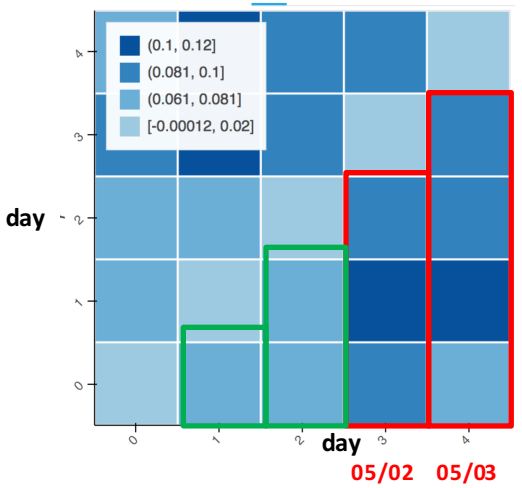
1ST STAGE: HIGHLIGHT SUSPECT ACCOUNT - FILTER OUT ABNORMAL ACCOUNTS



Account 1



Account 3

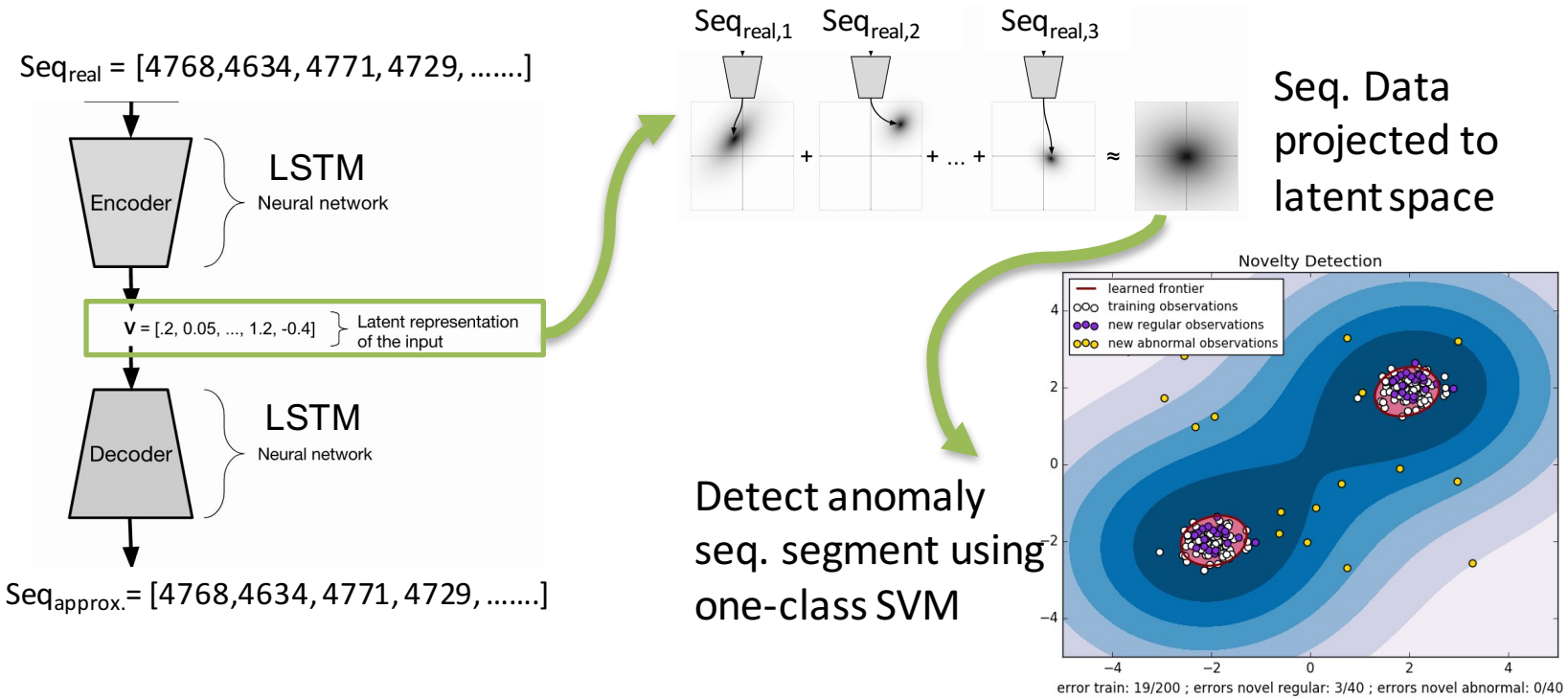


Account 2

1. Use GED to differentiate normal days and abnormal ones.
2. Highly Different between 5/2 ~ 5/3 (abnormal days) and normal days with regard to f variances of "2-gram" transitions (4673 -> 4762) and "1-gram" unseen event codes (4688).
3. Filter out 3 suspicious accounts:{**Account 1**, **Account 3**, **Account 2**} fed to Stage 2.

2ND STAGE: TRACK ABNORMAL BEHAVIORS OF TARGET ACCOUNTS

- Use LSTM-Autoencoder & One-Class SVM to find the outlier sequence



EXPERIMENT RESULTS

Outlier eventID sequence

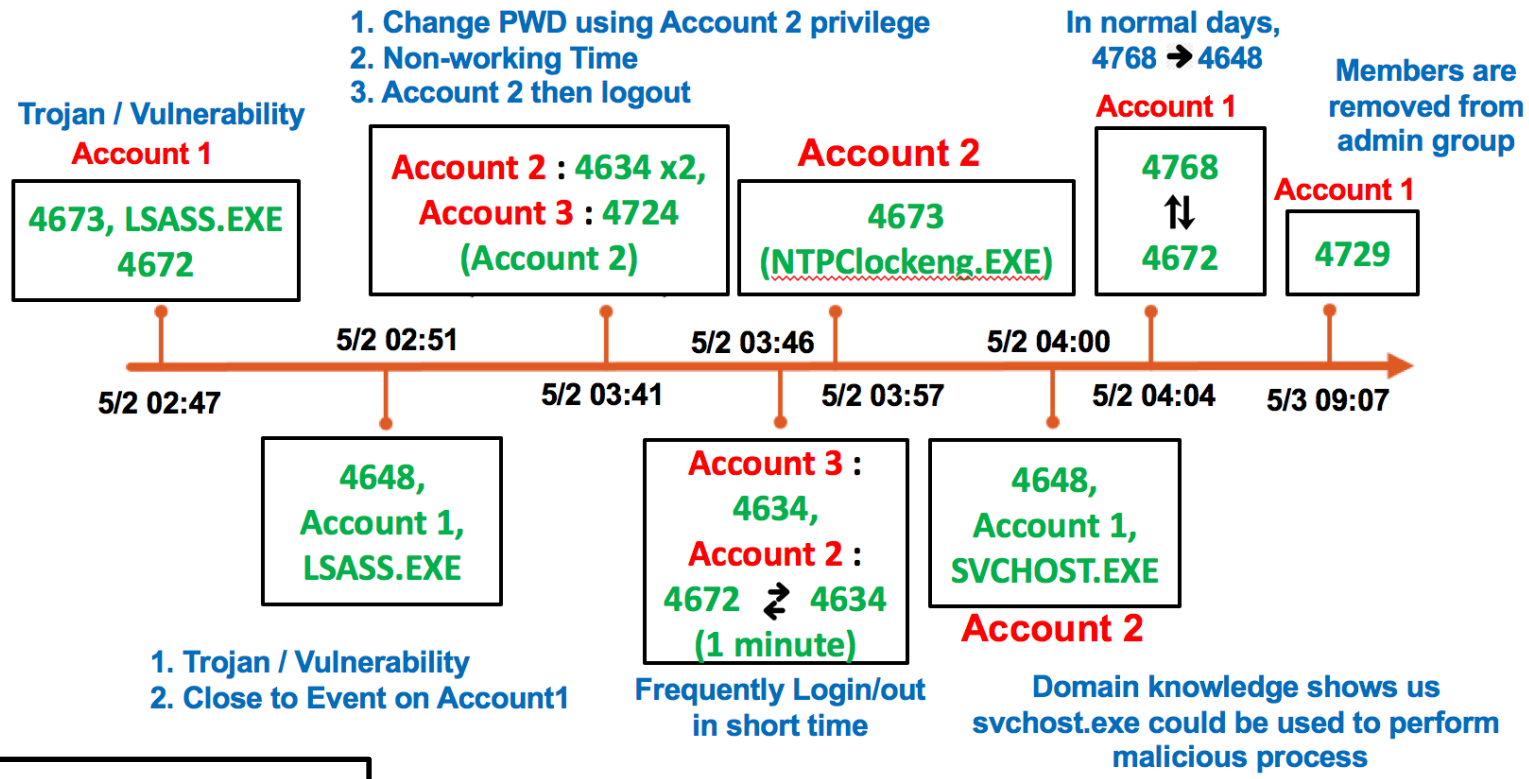

```
{
  "start_time": "2017-05-02 02:38:27.389839",
  "end_time": "2017-05-02 02:44:59.586924",
  "sequence": [
    "4624",
    "4634",
    "4672",
    "4624",
    "4634",
    "4672",
    "4624",
    "4634",
    "4634",
    "4634",
    "4634",
    "4688<->C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",
    "4688<->C:\\Windows\\System32\\conhost.exe",
    "4673<->C:\\Windows\\System32\\lsass.exe",
    "4672",
    "4624",
    "4672",
    "4624",
    "4634",
    "4634",
    "4634",
    "4672",
    "4624",
    "4662",
    "4662",
    "4662",
    "4662",
    "4672",
    "4624",
    "4672",
    "4624"
```


E.g., We find that **Account 1** has an outlier sequence in 5/2 (before 5/3) 02:38:27~02:44:59

This sequence contain EventID
4688 <-> powershell.exe (create new process)
4688 <-> conhost.exe
4673 <-> lsass.exe (執行特權程式) related with Trojan or 漏洞
4762 (特權登入)

Use privileged service lsass.exe to login with special privileges

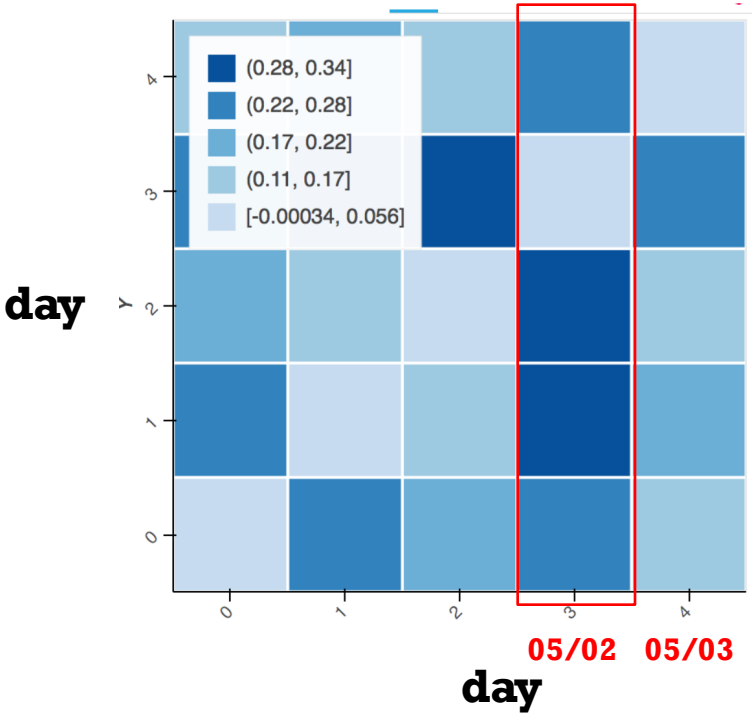
EVENTS ON THE TIMELINE



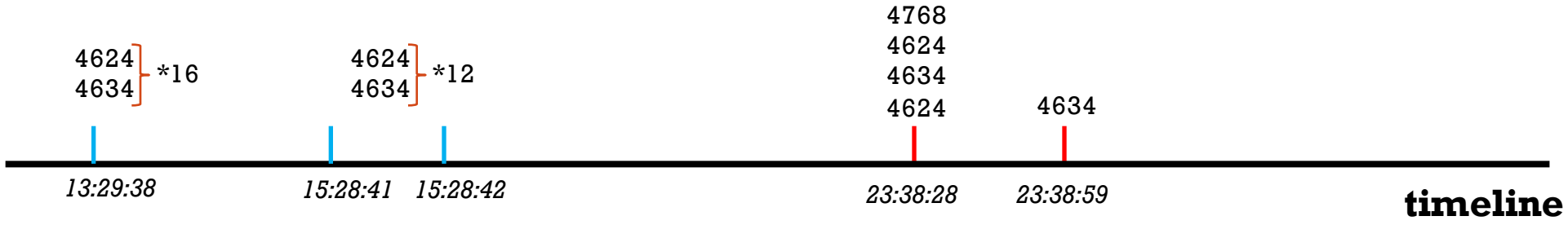
4673: A privileged service was called
4672: Special privileges assigned to new logon
4648: A logon was attempted using explicit credentials
4634: An account was logged off
4724: An attempt was made to reset an account's password
4768: A Kerberos authentication ticket (TGT) was requested
4729: A member was removed from a security-enabled global group

MARKOV - 短時間內頻繁登入登出

EventID	Remark
4768	A Kerberos authentication ticket (TGT) was requested
4624	An account was successfully logged on
4634	An account was logged off
4624	An account was successfully logged on
4634	An account was logged off



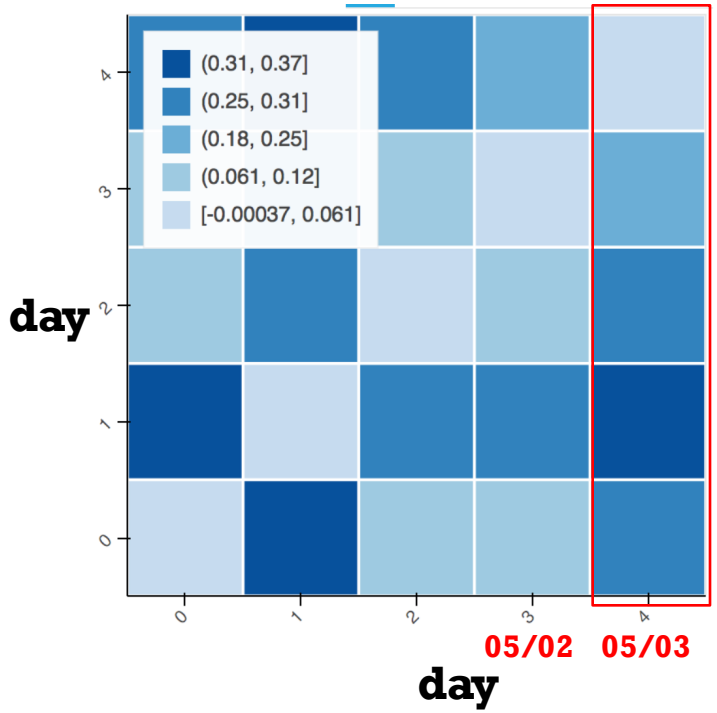
05/02



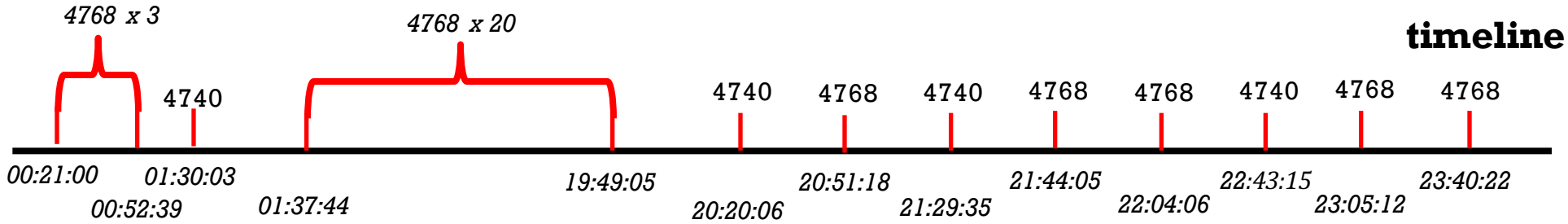
MARKOV - 帳號頻繁地要求登入並遭到封鎖

05/03 only these two kind of logs

EventID	Remark
4768	A Kerberos authentication ticket (TGT) was requested
4740	A user account was locked out



05/03

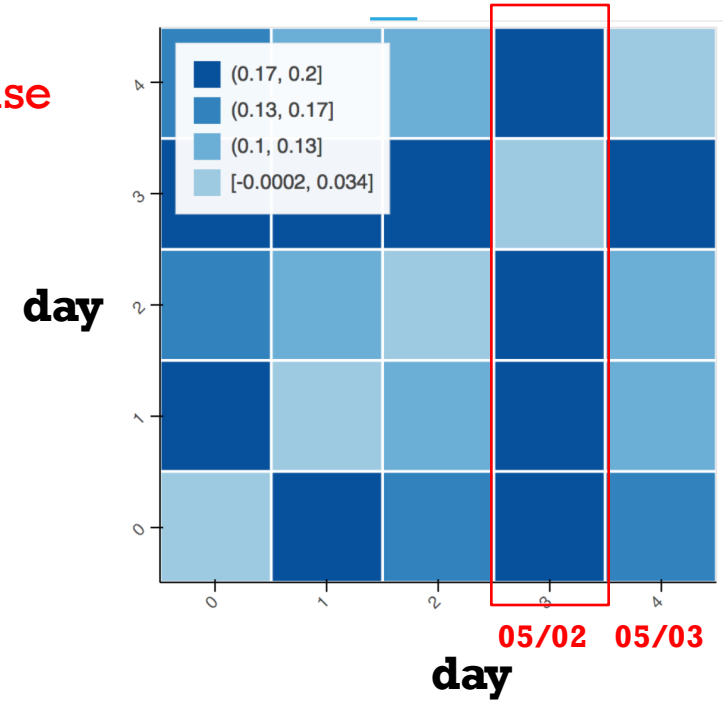


MARKOV - NTLM 登入頻繁 & NETWORK POLICY ACCESS

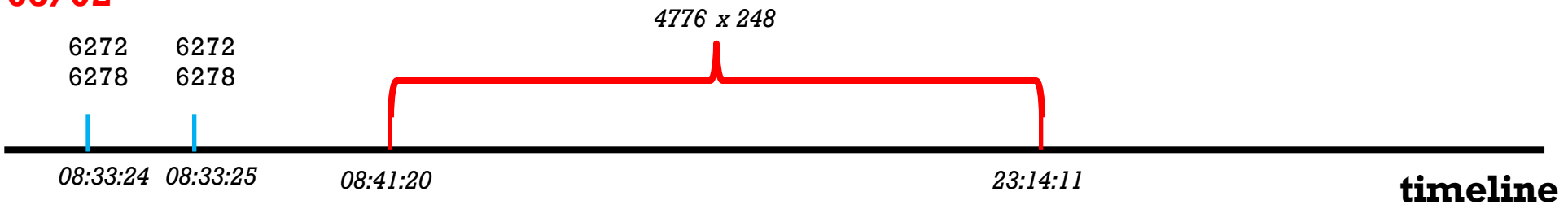
05/02 the mount of "EventID 4776" has a substantial increase

EventID	Remark
4776	The domain controller attempted to validate the credentials for an account
6272	Network Policy Server granted access to a user
6278	Network Policy Server granted full access to a user because the host met the defined health policy

Normal days	EventID 4776
05/22	114
05/23	82
05/24	65



05/02

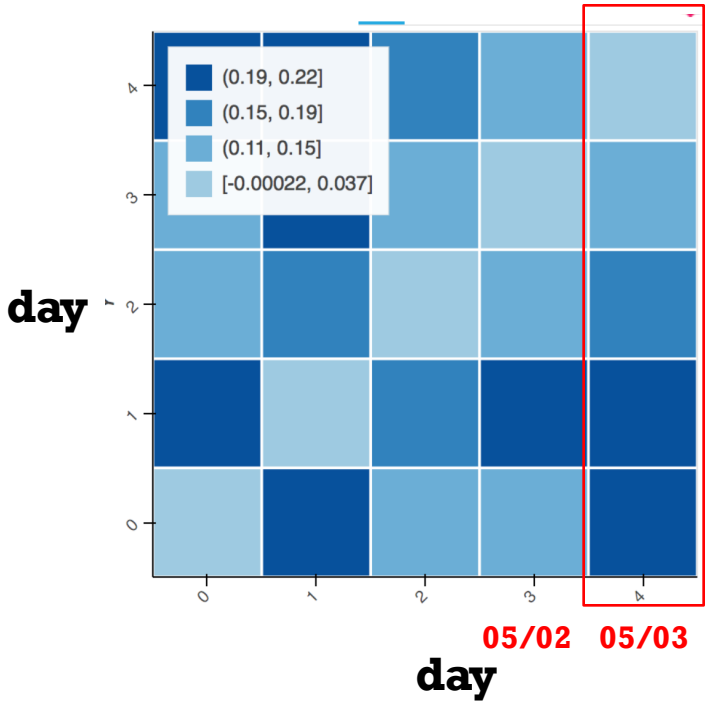


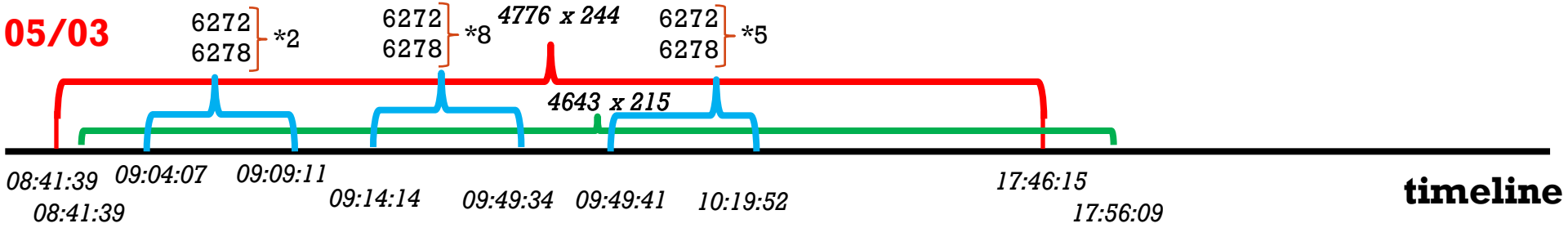
MARKOV - NTLM 登入登出頻繁 & NETWORK POLICY ACCESS

05/03 the mount of "EventID 4776/4634" has a substantial increase

EventID	Remark
4776	The domain controller attempted to validate the credentials for an account
4634	An account was logged off
6272	Network Policy Server granted access to a user
6278	Network Policy Server granted full access to a user because the host met the defined health policy

Normal days	EventID 4776	EventID 4634
05/22	27	62
05/23	62	44
05/24	48	111



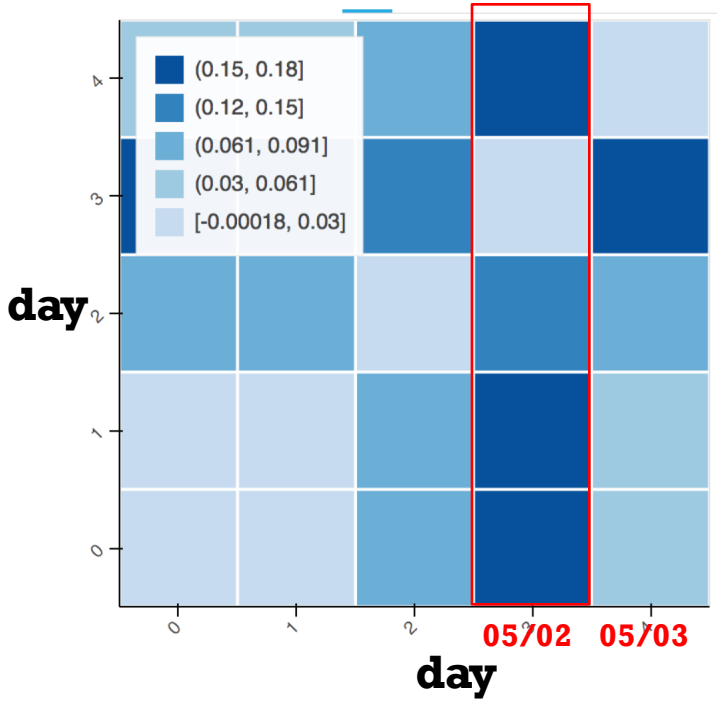


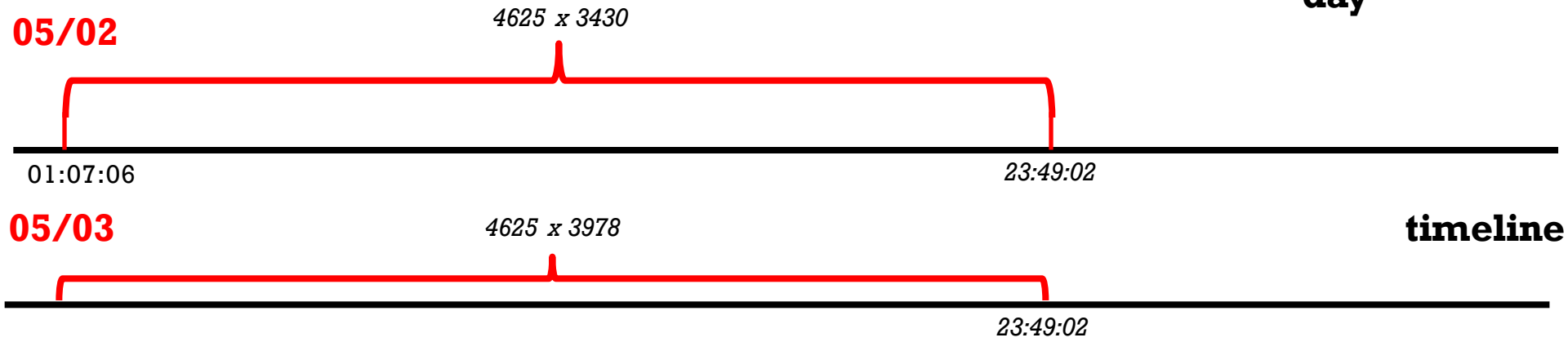
MARKOV - 大量的登入失敗

05/02 and 05/03 log failed up to 3000 times

EventID	Remark
4625	An account failed to log on

Normal days	EventID 4625
05/22	804
05/23	1215
05/24	1068





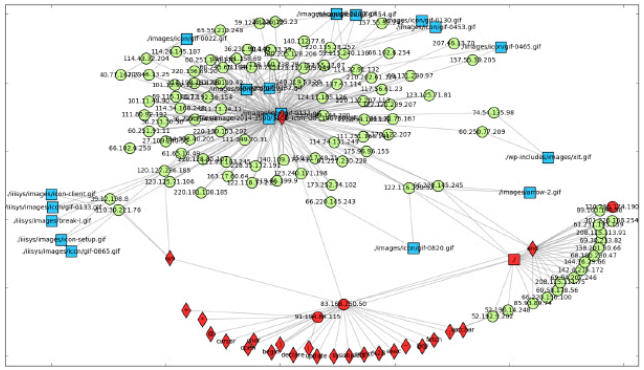
ANOMALY DETECTION & GRAPH MINING

- ChainSpot
- WebHound
- Playwright

WEBHOUND - DETECTING CYBERCRIME FROM REAL-WORLD WEB-ACCESS LOGS

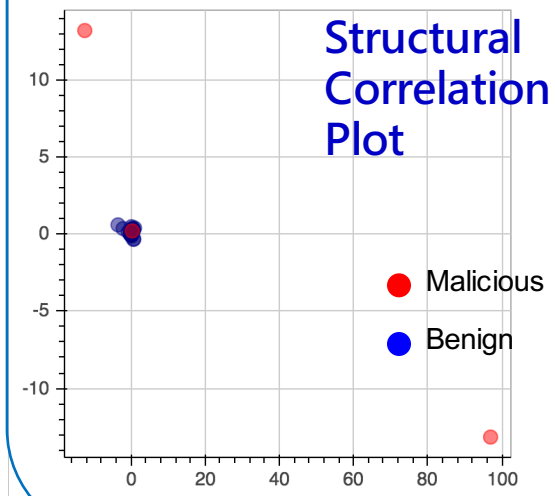
基於存取日誌及行為比對之伺服器入侵偵測技術

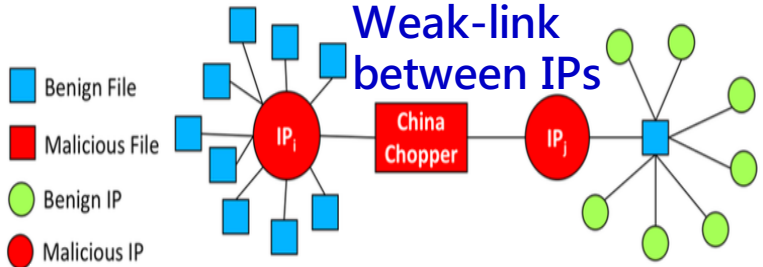
Activity Graph

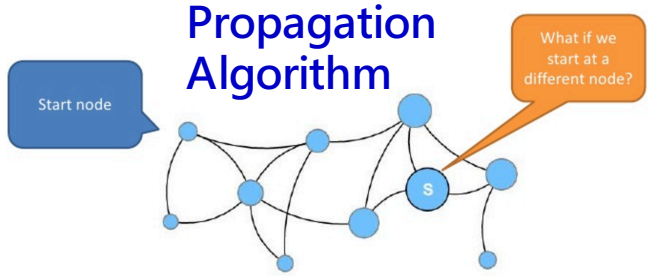


- *Problem*: Web Server is usually the first attacked target for hackers, and also is necessary for every enterprise.
- *Idea:* Detect malware based on structural correlation between source IPs. 1) Web-access Log Collection, 2) Activity Graph Reconstruction, 3) Structural correlation representation, and 4) Malicious IP Detection.
- *Contribution:* *WebHound* could additionally discover malicious IPs which are False Negative by forensics.

Structural Correlation Analysis of Hacker Activities

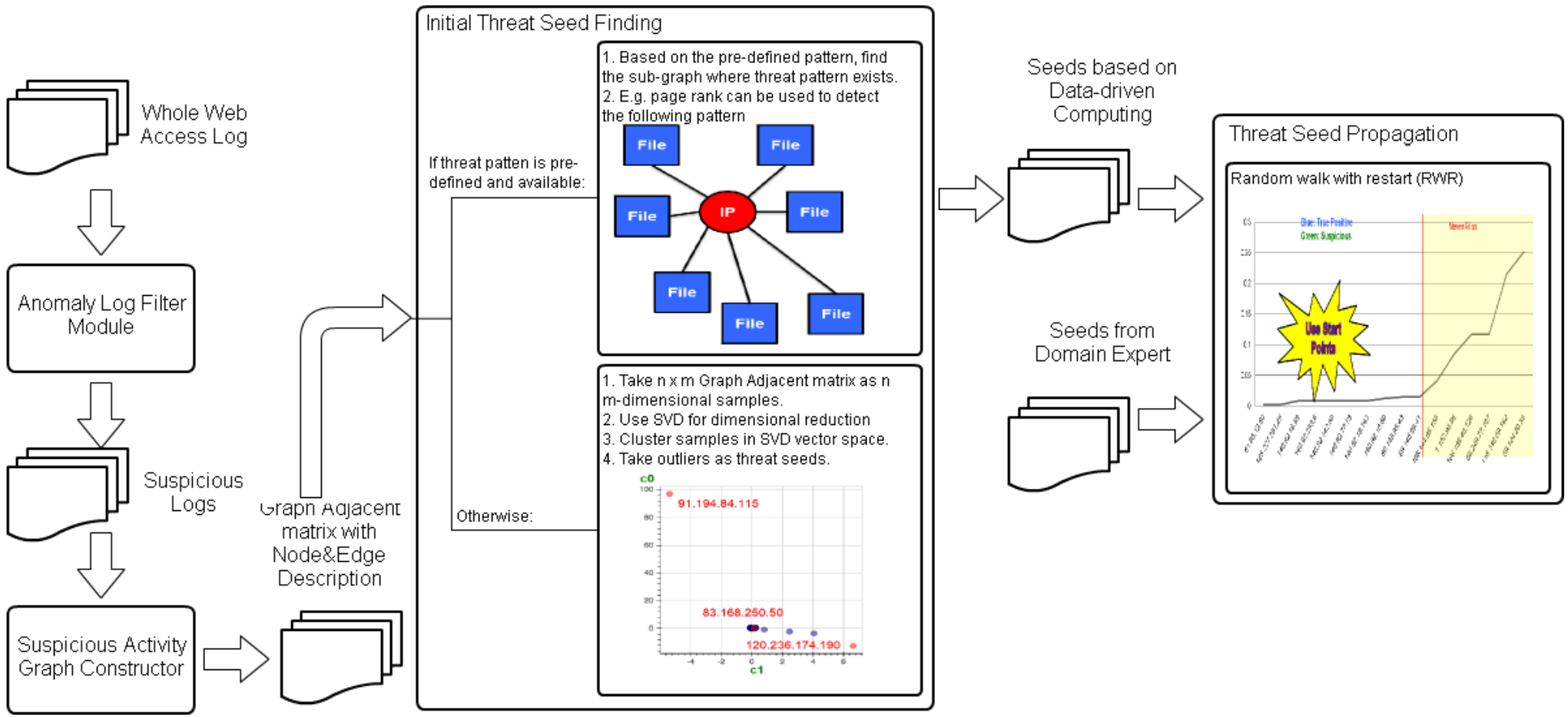






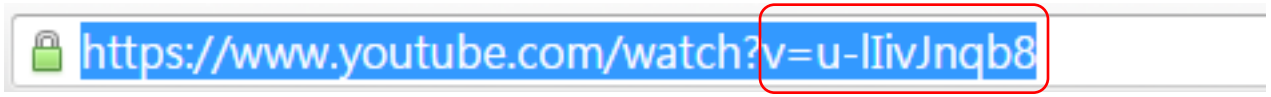
1. Detect Outlier IPs which have differential correlations. (SVD Algo.)
2. Detect Weak-link IPs which are hidden source used by hackers. (TrustRank Alog.)
3. Use Propagation Alog. to detect malicious IPs based on structural correlations.

CURRENT FRAMEWORK OF WEBHOUND



THE ANOMALY LOGS FILTER MODULE

- Intention 1
  - Web log volume too large to inspect with bare eyes
- Input
  - Raw logs from web access record
- Output
  - Filtered logs
- Example
  - Illegal Characters Verify Module
  - Unusual Parameter Usage Verify

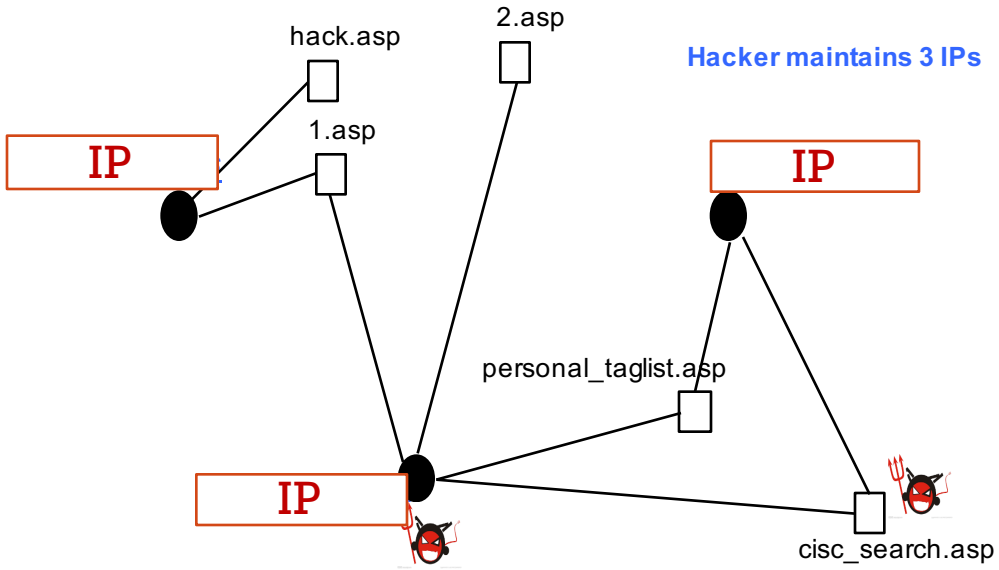


THE ATTACK SCENE MATCHING MODULE

- **Intention 2**
  - There may not be only one candidate of attacking Tactics, Techniques, and Processes (TTPs).
- **Input**
  - Kinds of filtered logs
- **Output**
  - Attacking Scene with Associated Filtered logs

THE GRAPH CONSTRUCTOR MODULE

- **Intention 3**
  - Concretize entities and associated relations of suspicious activities.
- **Input**
  - Each Attacking Scene with Associated Filtered logs
- **Output**
  - **Suspicious Activity Graph**
    - Heterogeneous Graph
    - Bipartite Graph
    - Homogenous Graph



THE GRAPH PATTERN MATCHING MODULE

- **Intention 4**
  - Find trigger points (or threat seeds) where attacker is in an attempt to invade target
- **Input**
  - **Suspicious Activity Graph**
    - Heterogeneous Graph
    - Bipartite Graph
    - Homogenous Graph
- **Output**
  - Entity labeled as suspect

CASE STUDIES

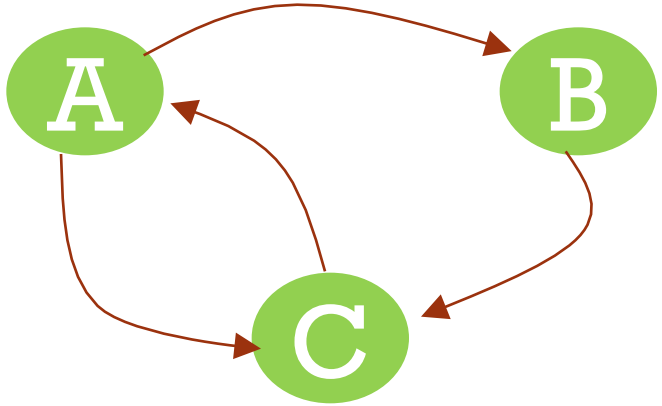
- Case 1 : IIS Logs – System Compromise
- Case 2 : Apache Logs – SQL Injection for Data Leakage
- Case 3 : Apache Logs – System Compromise

CASE STUDIES

- Case 1 : IIS Logs – System Compromise
- Case 2 : Apache Logs – SQL Injection for Data Leakage
- Case 3 : Apache Logs – System Compromise

INITIAL SEEDS FINDING REGARDING TO KNOWN PATTERN

- How to ranking nodes in a graph based on their referring to each other.



Pagerank Algorithm

- Originally be used to rank Google retrieved pages

$$PR(C) = \frac{PR(A)}{2} + \frac{PR(B)}{1}$$

$$PR(C) = \frac{\lambda}{3} + (1-\lambda) \cdot (\frac{PR(A)}{2} + \frac{PR(B)}{1})$$

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

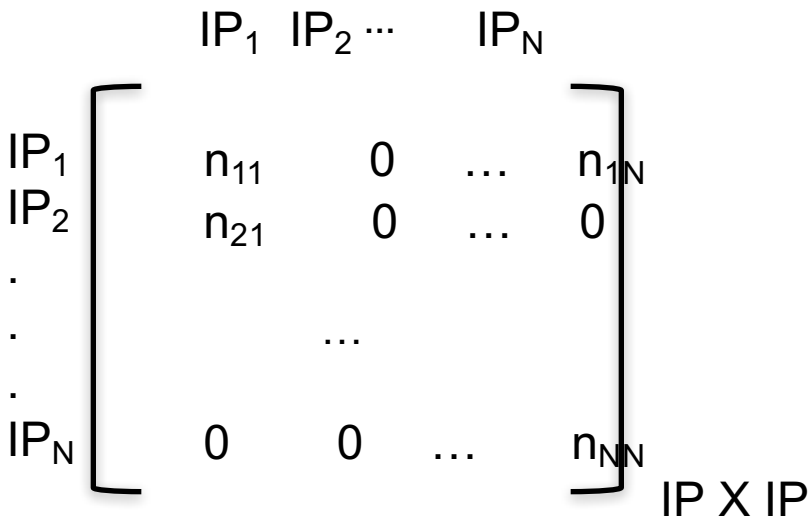
$$PR(u) = \frac{\lambda}{N} + (1-\lambda) \cdot \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

PR(A)=PR(B)=PR(C)=1/3=0.33

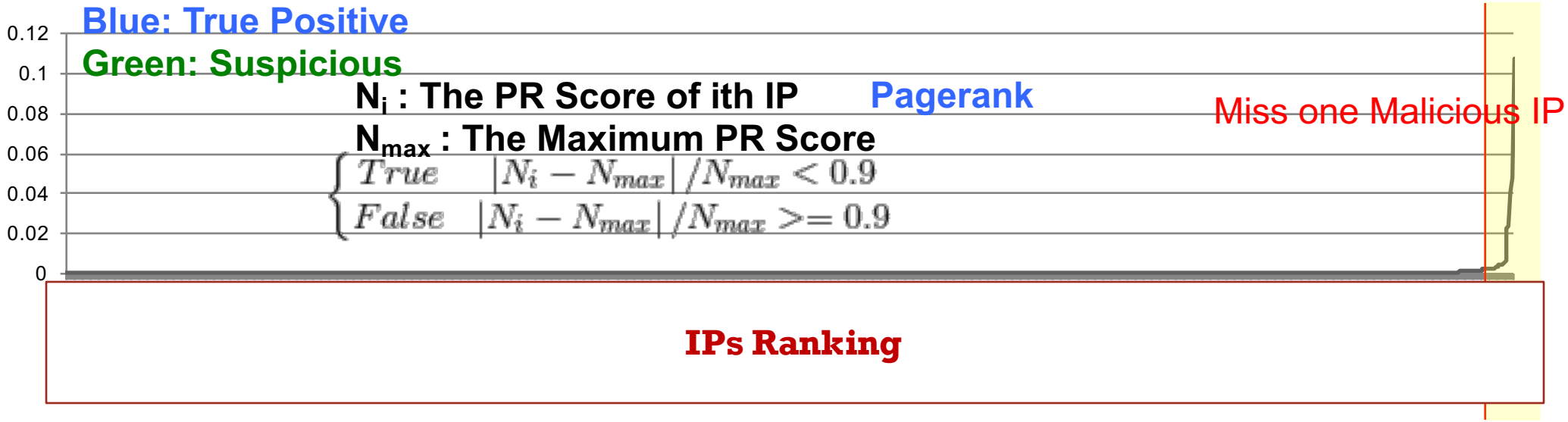
PR(A)=0.4 PR(B)=0.2 PR(C)=0.4

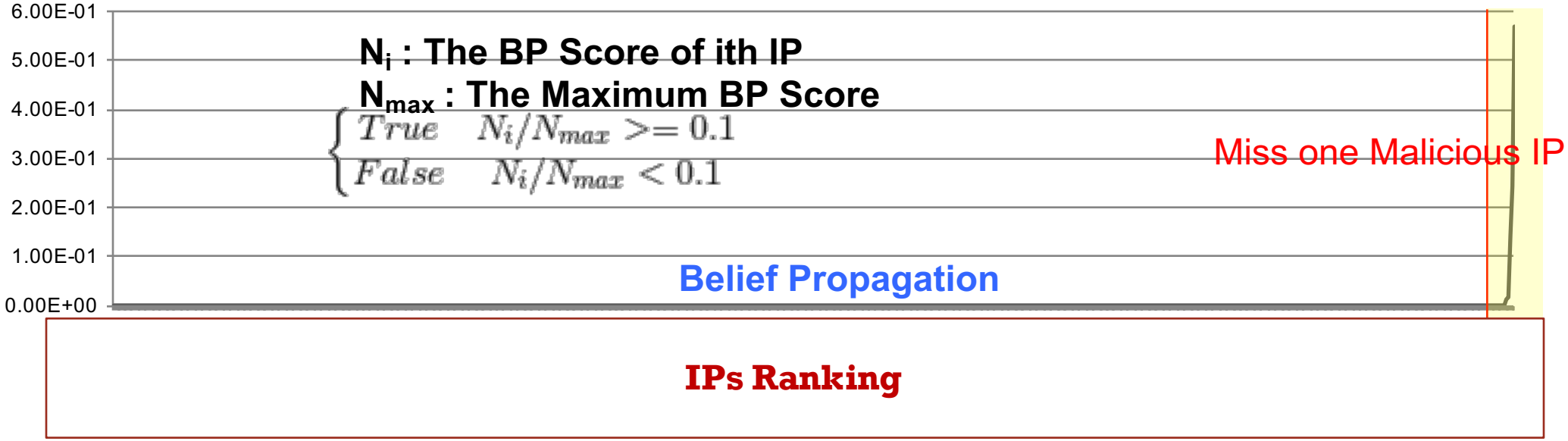
IS IP WHICH HACKER USED DIFFERENT FROM OTHERS IN SUSPICIOUS ACTIVITY GRAPH ?! - IIS

- Problem:
  - Investigate the role of IP Address in Suspicious Activity Graph
- Input:
  - |IPs| x |IPs| Matrix
    - n_{ij}: num of Queried same File
      - IP_i to IP_j
    - 0: Queried by different File
- Output:
  - Clustering Result



INITIAL THREAT SEED FINDING: WITH PRIORI KNOWLEDGE ABOUT THE THREAT PATTERN

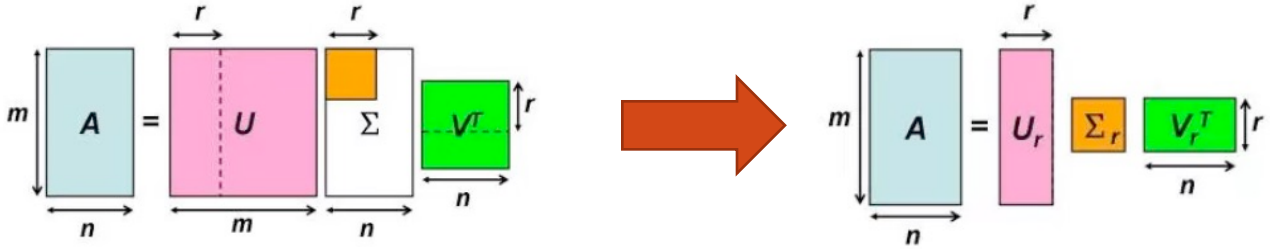




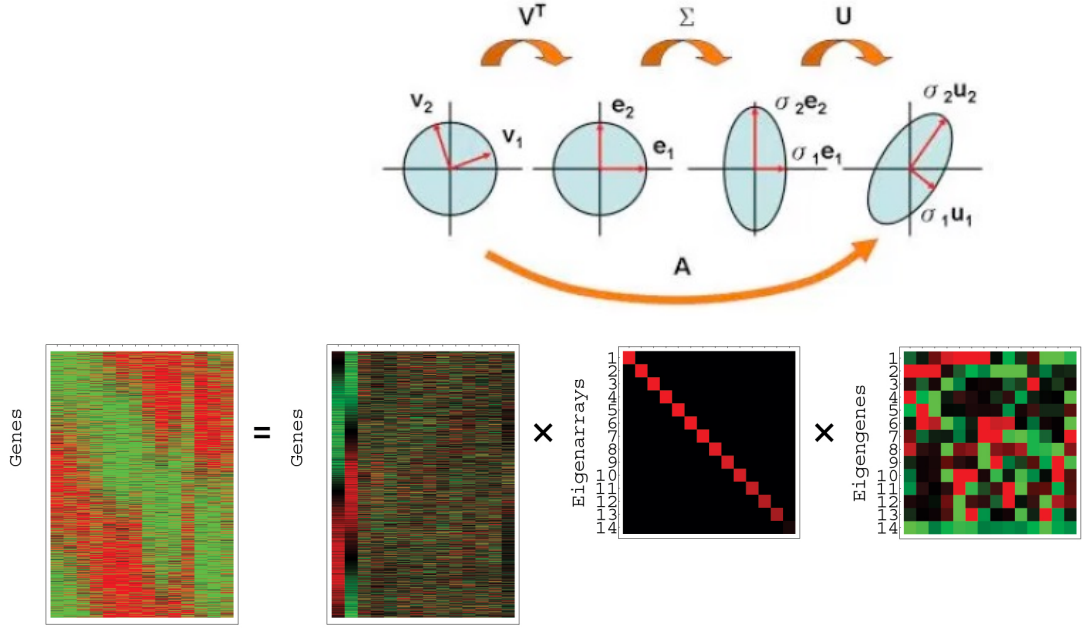
INITIAL SEEDS FINDING WITHOUT KNOWN PATTERN

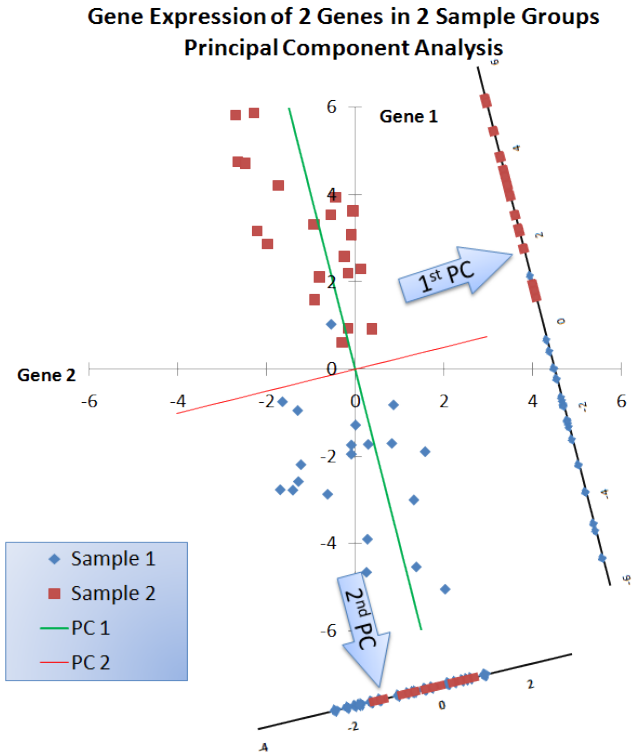
- How about the case that you don’ t have any priori knowledge

SVD



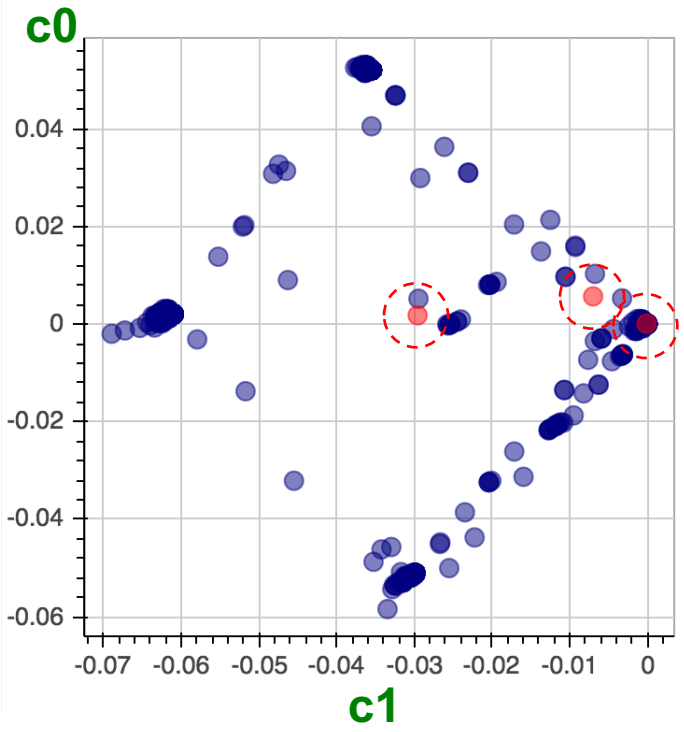
- Effect of SVD

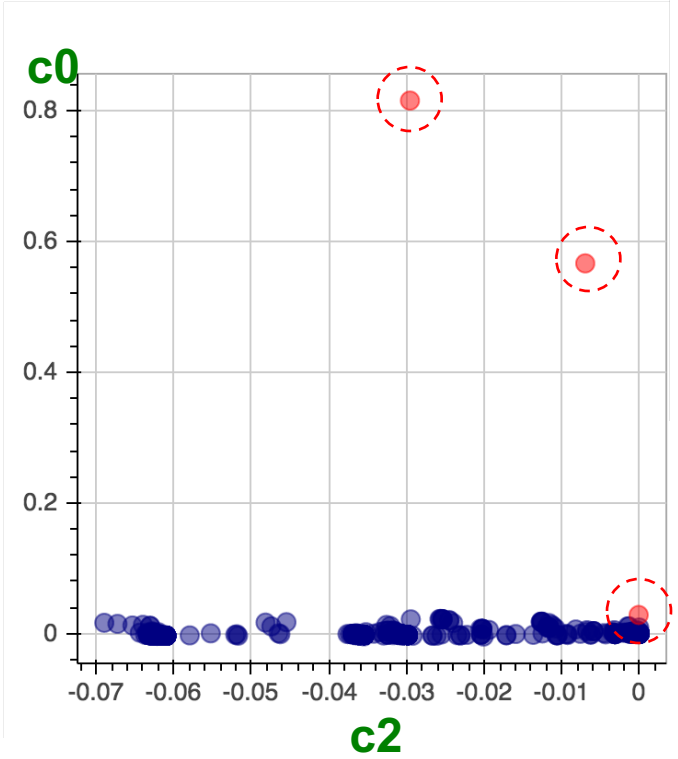


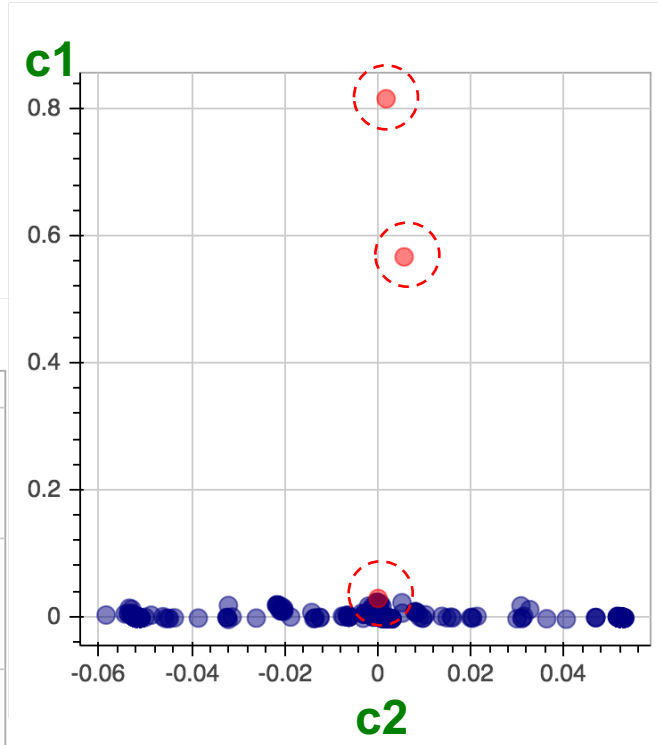


IS IP WHICH HACKER USED DIFFERENT FROM OTHERS IN SUSPICIOUS ACTIVITY GRAPH ?! (CONT'D)

SVD



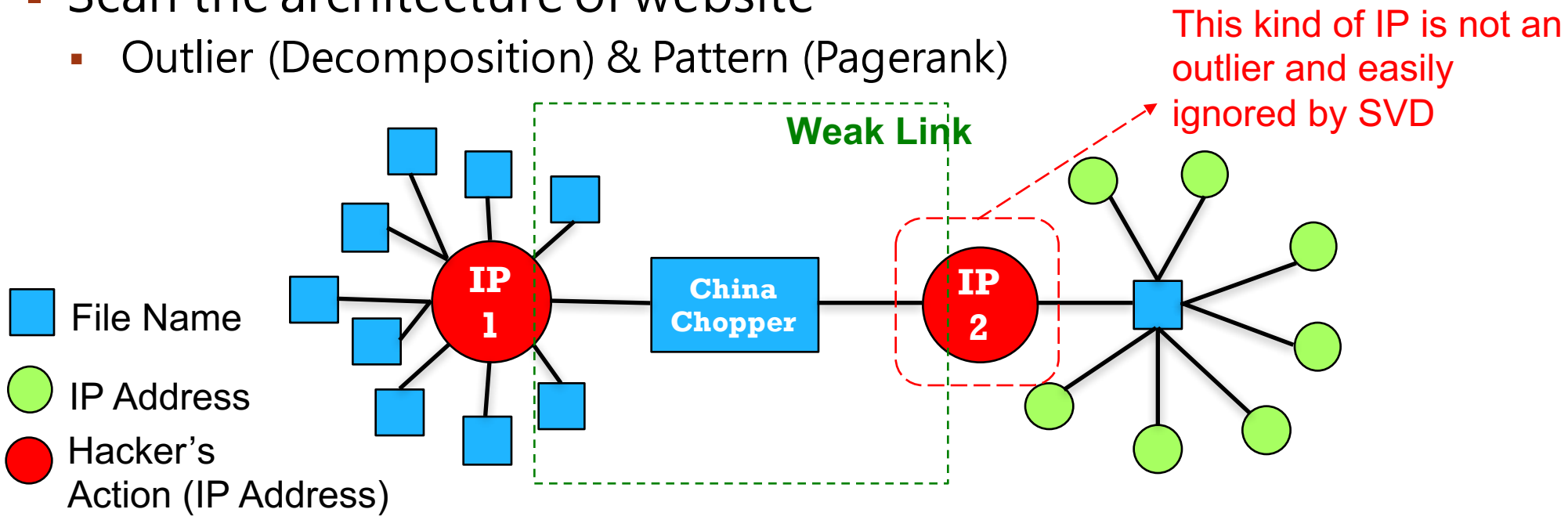




- Malicious
- Benign

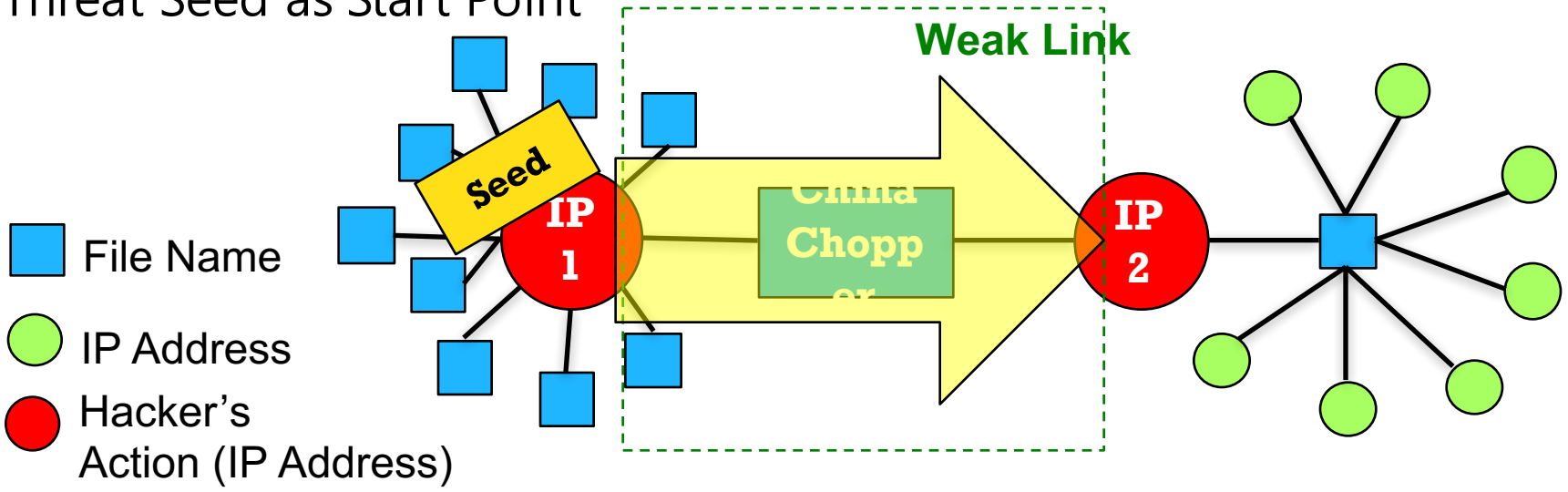
LIMITATION - SVD

- Hackers usually use different IPs to achieve their goal
  - Test the Trojan (China Chopper)
    - Weak Link IP
  - Scan the architecture of website
    - Outlier (Decomposition) & Pattern (Pagerank)

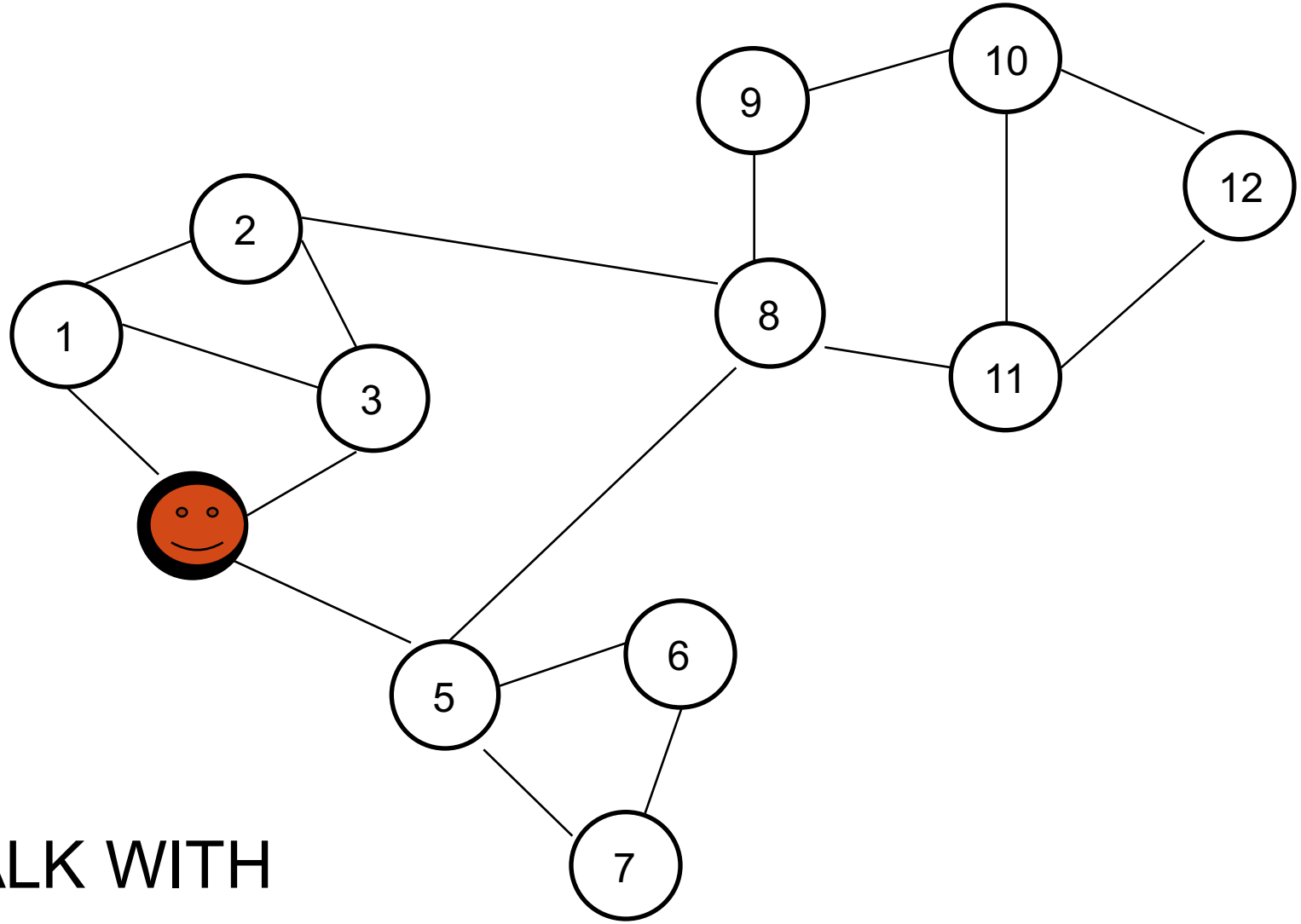


LIMITATION - SVD (CONT'D)

- How to resolve Weak Link problem?!
  - Use Graph Mining Algo. to propagate the threat score from IP1 (Seed) to IP2
  - Random Walk with Restart (RWR)
    - Threat Seed as Start Point



HOW TO PROPAGATE THE CONFIDENCE ONCE YOU HAVE SUSPECT CANDIDATE



RANDOM WALK WITH RESTART

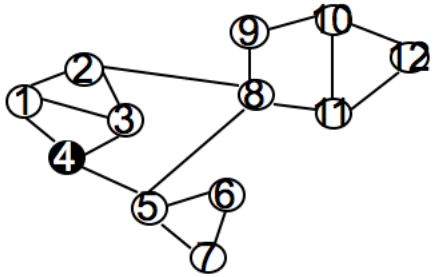
RWR COMPUTATION

$$\overrightarrow{r_{t+1}} = c\tilde{W}\overrightarrow{r_t} + (1-c)\overrightarrow{e_t}$$

Ranking vector — Adjacent matrix — Restart p — Starting vector

$$\begin{pmatrix}0.13\\0.10\\0.13\\0.22\\0.13\\0.05\\0.05\\0.08\\0.04\\0.03\\0.04\\0.02\end{pmatrix} = 0.9\times \begin{pmatrix}0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 & 0\\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 1/3 & 0 & 1/3 & 0 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1/3 & 0 & 1/2 & 1/2 & 1/4 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1/4 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1/4 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1/3 & 0 & 0 & 1/4 & 0 & 0 & 0 & 1/2 & 0 & 1/3 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 1/3 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 1/3 & 1/2\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 1/3 & 0 & 1/2\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 0\end{pmatrix} \begin{pmatrix}0.13\\0.10\\0.13\\0.22\\0.13\\0.05\\0.05\\0.08\\0.04\\0.03\\0.04\\0.02\end{pmatrix} + 0.1\times \begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$$

n x 1 — n x n — n x 1



	IP_1	IP_2	...	IP_N
IP_1	n_{11}	0	...	n_{1N}
IP_2	n_{21}	0	...	0
.				
.		...		
.				
IP_N	0	0	...	n_{NN}

IP X IP

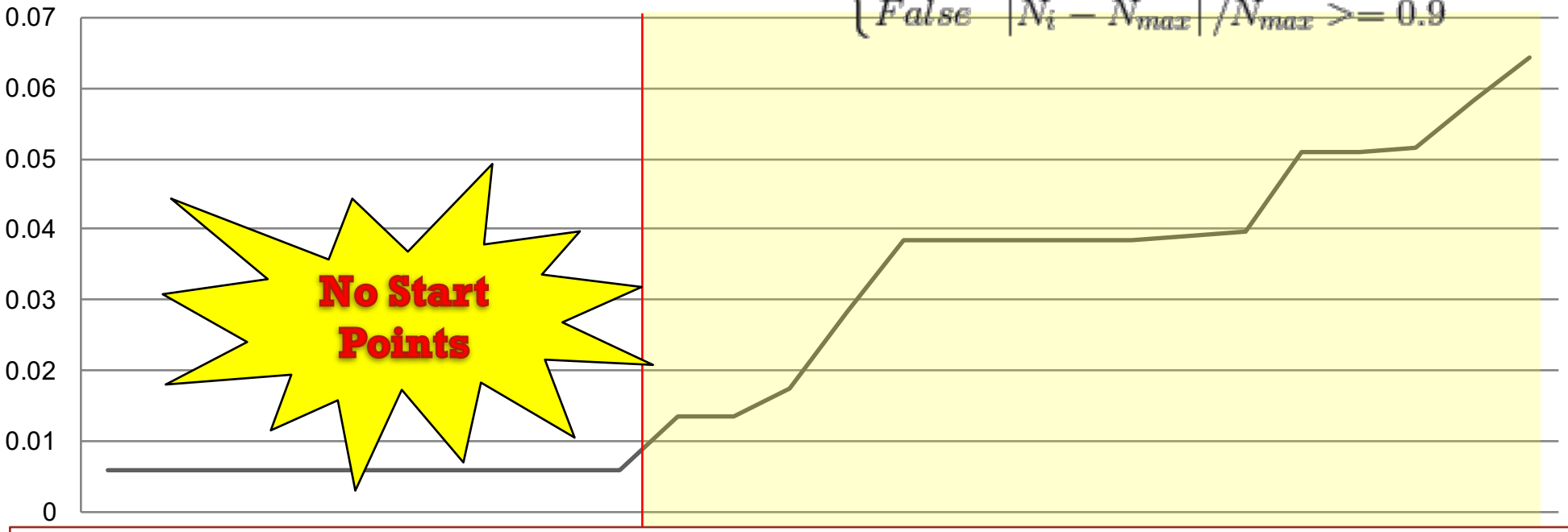
THREAT SEED PROPAGATION – GIVEN DATA- OR KNOWLEDGE-DRIVEN THREAT SEEDS

N_i : The RWR Score of ith IP

N_{max} : The Maximum PR Score

RWR

$$\begin{cases} True & |N_i - N_{max}| / N_{max} < 0.9 \\ False & |N_i - N_{max}| / N_{max} >= 0.9 \end{cases}$$



THREAT SEED PROPAGATION – GIVEN DATA- OR KNOWLEDGE-DRIVEN THREAT SEEDS

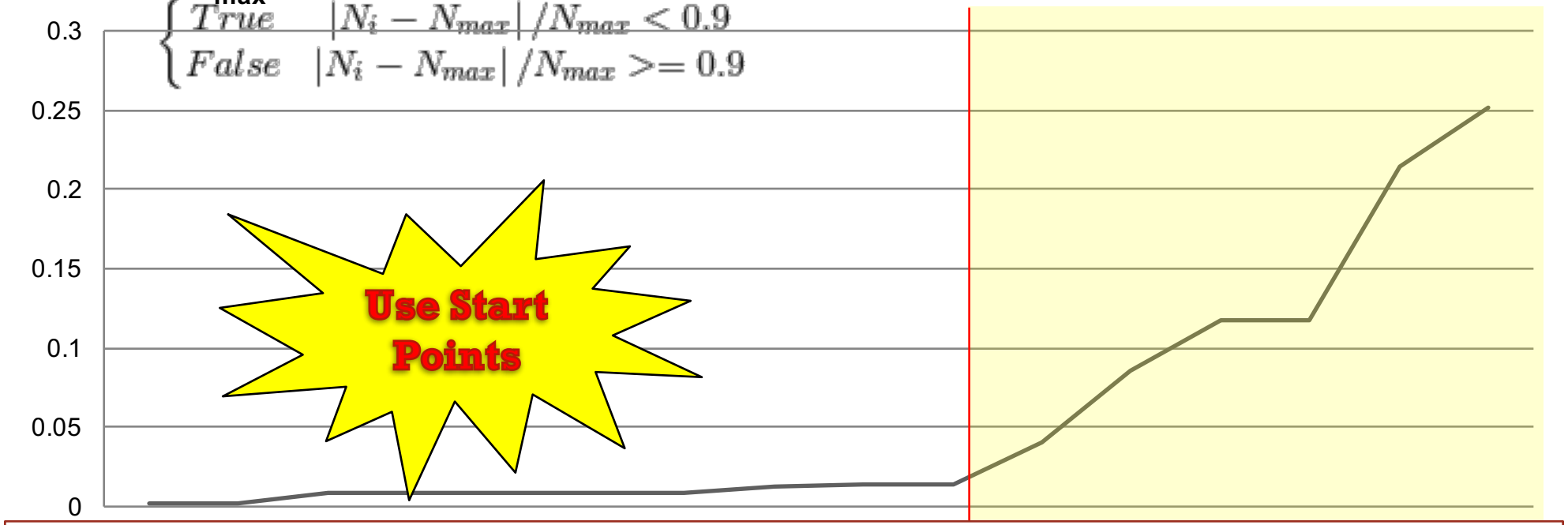
RWR

Low False Positive !!!

N_i : The RWR Score of ith IP

N_{max} : The Maximum PR Score

$$\begin{cases} True & |N_i - N_{max}| / N_{max} < 0.9 \\ False & |N_i - N_{max}| / N_{max} >= 0.9 \end{cases}$$



CASE STUDIES

- Case 1 : IIS Logs – System Compromise
- Case 2 : Apache Logs – SQL Injection for Data Leakage
- Case 3 : Apache Logs – System Compromise

INITIAL THREAT SEED FINDING: WITHOUT PRIORI KNOWLEDGE

- Problem:
  - Investigate the role of IP Address in Suspicious Activity Graph
  - Outliers may be the candidates of initial threat seeds
- Input:
  - $|IPs| \times |(File_m, IllegalChar_n)|$ Matrix
    - n_{ij}: num of queried same $(File_m, IllegalChar_n)$ by IP_i and IP_j
    - 0: Never Queried by $(File_m, IllegalChar_n)$
- Output:
  - Clustering Result

$(File_m, IllegalChar_n)_1 \cdots (File_m, IllegalChar_n)_{mXn}$

$$\begin{matrix} IP_1 \\ IP_2 \\ . \\ . \\ . \\ IP_N \end{matrix} \begin{bmatrix} n_{11} & 0 & \dots & n_{mXnX1} \\ n_{21} & 0 & \dots & 0 \\ & \dots & & \\ 0 & 0 & \dots & n_{mXnXN} \end{bmatrix}$$

IP X $(File_m, IllegalChar_n)$

INITIAL THREAT SEED FINDING: WITH PRIORI KNOWLEDGE ABOUT THE THREAT PATTERN

N_i : The PR Score of ith IP

N_{max} : The Maximum PR Score

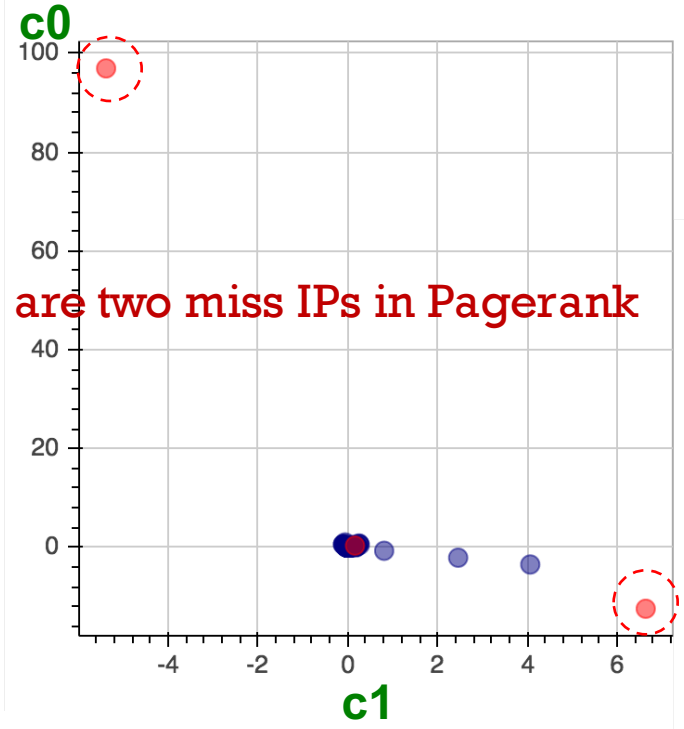
PAGERANK

$$\begin{cases} True & |N_i - N_{max}| / N_{max} < 0.9 \\ False & |N_i - N_{max}| / N_{max} >= 0.9 \end{cases}$$

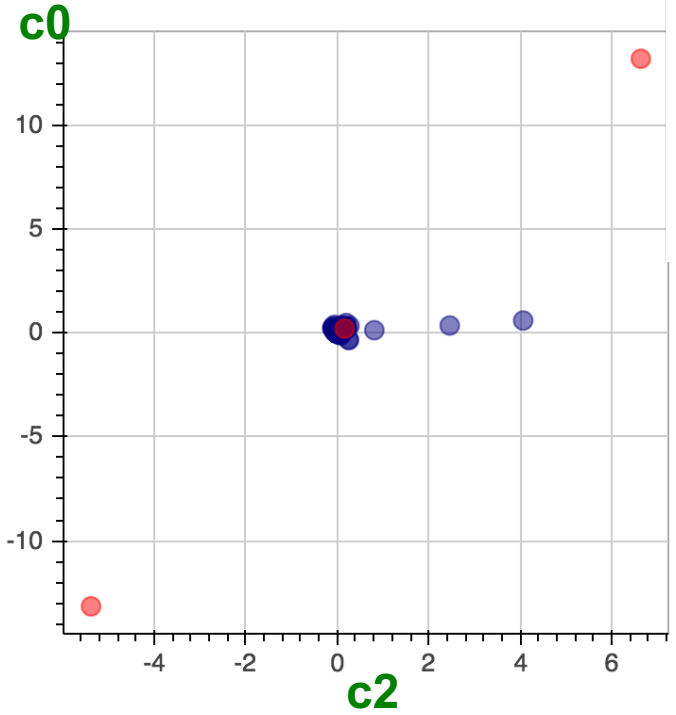
Miss two Malicious IP

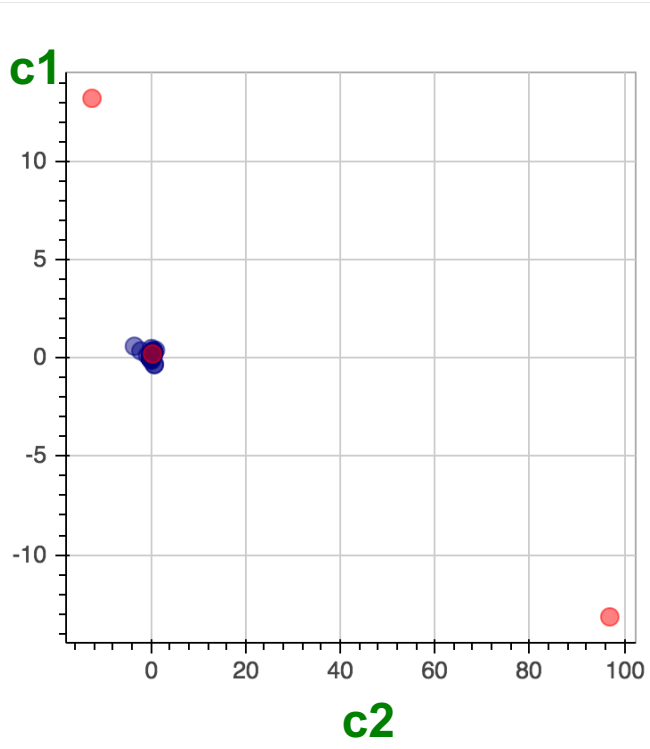
INITIAL THREAT SEED FINDING: WITHOUT PRIORI KNOWLEDGE

These are two miss IPs in Pagerank



SVD



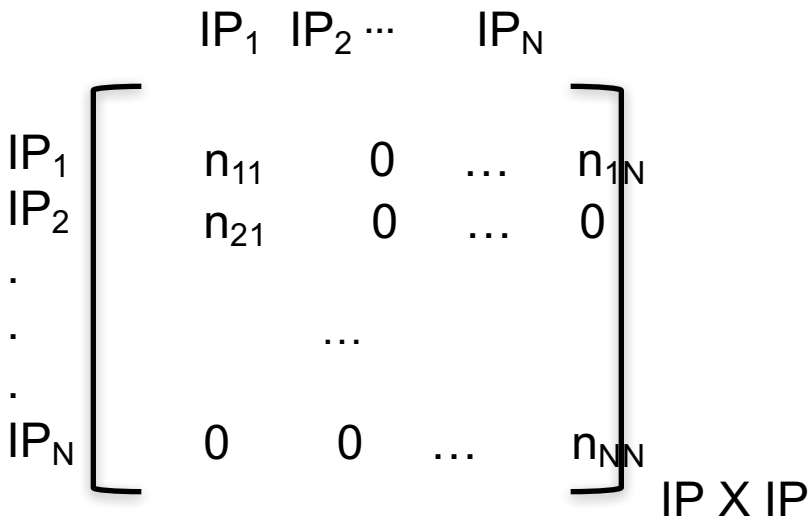


CASE STUDIES

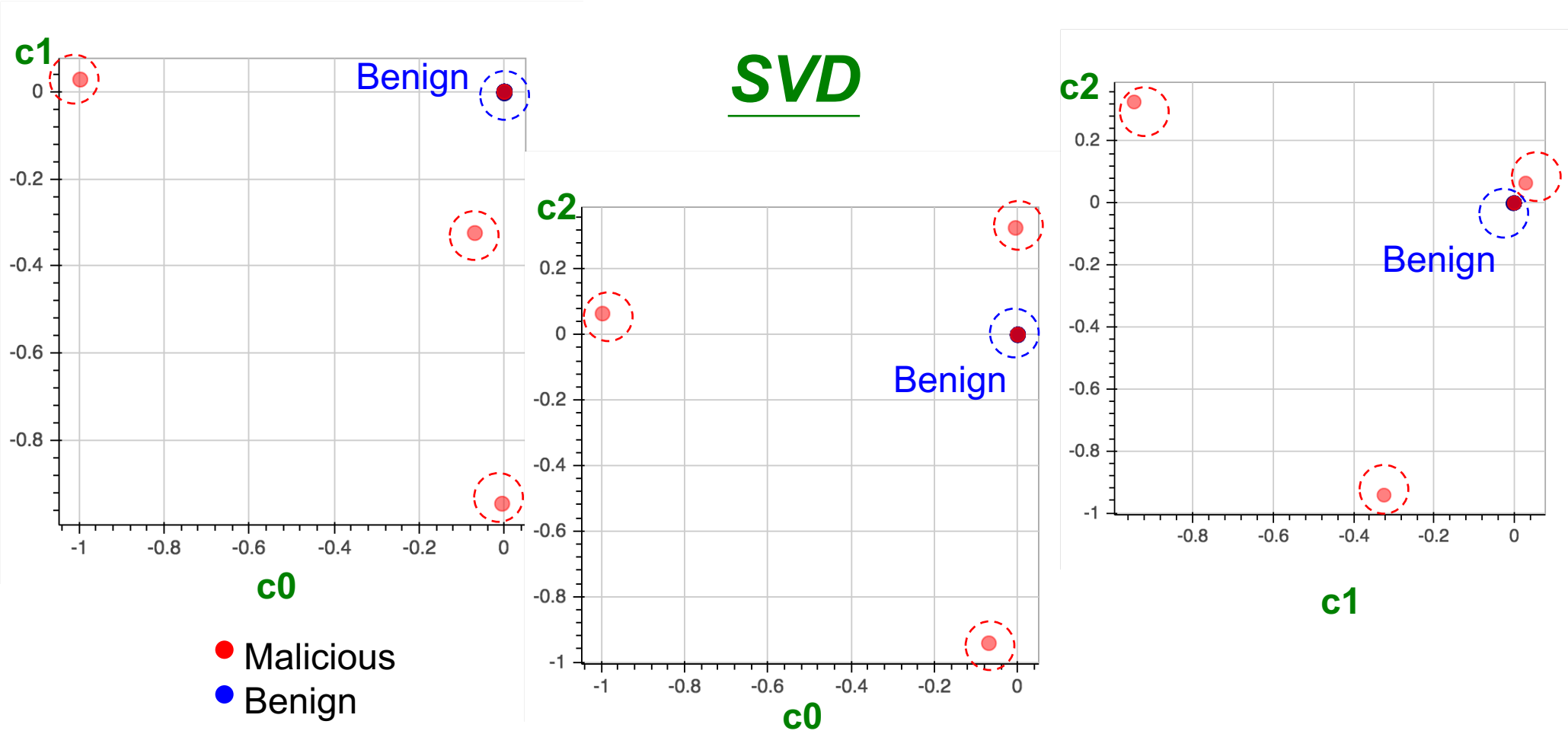
- Case 1 : IIS Logs – System Compromise
- Case 2 : Apache Logs – SQL Injection for Data Leakage
- Case 3 : Apache Logs – System Compromise

CASE 3: IS ROLE OF IP DIFFERENT FROM OTHERS FILE HACKER USED IN SUSPICIOUS ACTIVITY GRAPH ?1 - APACHE

- Problem:
  - Investigate the role of IP Address in Suspicious Activity Graph
- Input:
  - |IPs| x |IPs| Matrix
    - n_{ij}: num of Queried same File
      - IP_i to IP_j
    - 0: Queried by different File
- Output:
  - Clustering Result

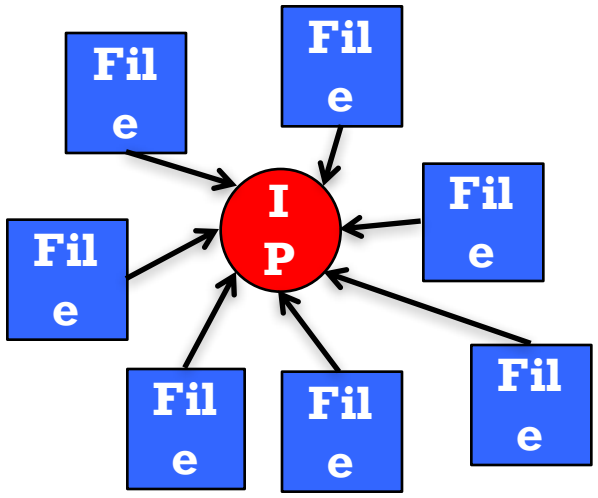


CASE 3: IS ROLE OF IP DIFFERENT FROM OTHERS FILE HACKER USED IN SUSPICIOUS ACTIVITY GRAPH ?1 - APACHE (CONT'D)



CASE 3: WHY USING PROPAGATION TO EVALUATE SUSPICIOUS IP?! - APACHE (CONT'D)

- Design a directed homogeneous graph to illustrate above patterns we found
- In this pattern, we expect to obtain high score IPs from graph when using propagation algorithm
  - Pagerank



CASE 3: WHY USING PROPAGATION TO EVALUATE SUSPICIOUS IP?! (CONT'D) – HIGH SCORE IPS

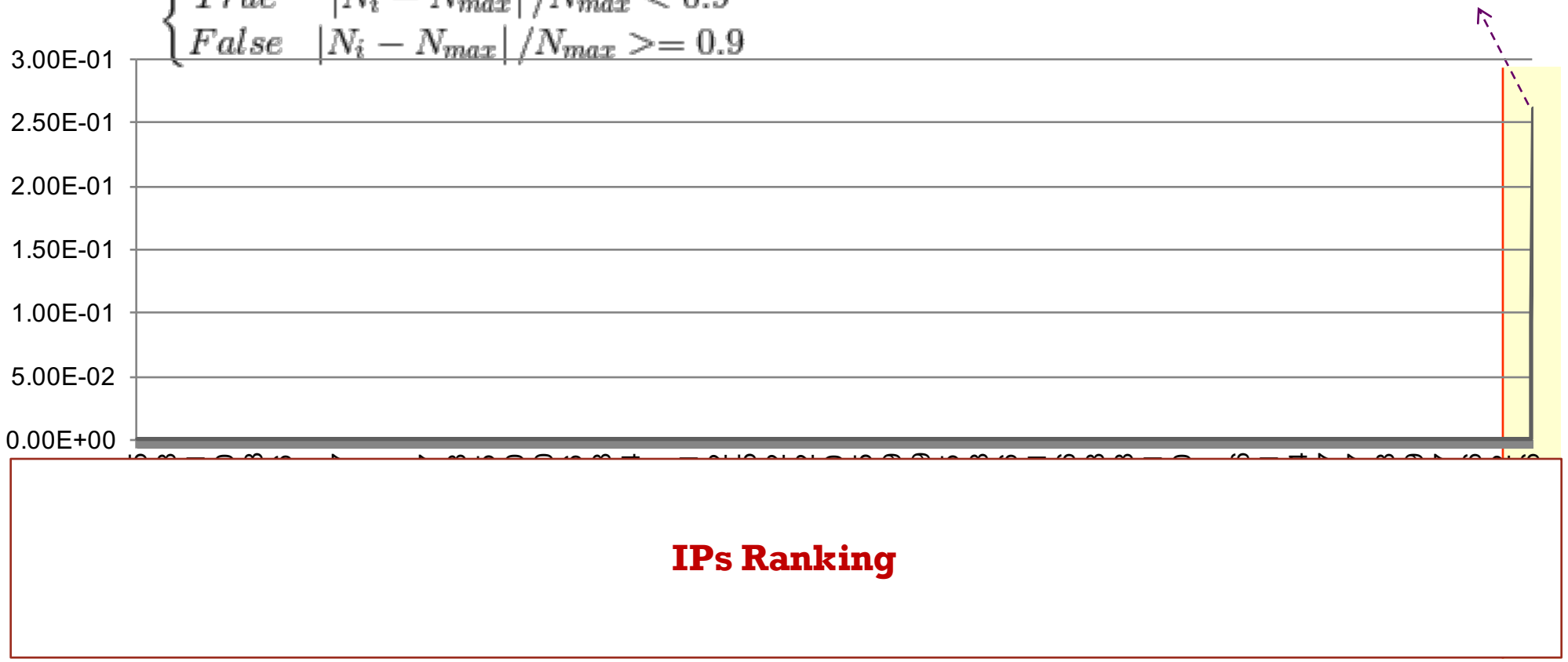
PAGERANK

N_i : The PR Score of ith IP

N_{max} : The Maximum PR Score

$$\begin{cases} True & |N_i - N_{max}| / N_{max} < 0.9 \\ False & |N_i - N_{max}| / N_{max} >= 0.9 \end{cases}$$

Miss two Malicious IPs



ADDITIONAL ENHANCING

- In Case 2 (SQL Injection), we can figure out all malicious IPs using Threat Seed Finding
- In Case 1 (IIS, System Compromise), the RWR result is better (as Table) when only focus on Web Files (e.g., .php, .asp and so on)

Weak Link IP	All_Files		Ignore_Img		Web_File_Only	
	Place	Total	Place	Total	Place	Total
IPs List	1	804	1	241	***1***	***15***
	1	804	1	241	***1***	***15***
	6	804	24	241	***8***	***15***
	804	804	241	241	***7***	***15***
	700	804	5	241	***6***	***15***

ADDITIONAL ENHANCING (CONT'D)

- However, the experiment result of Case 3 (Apache, System Compromise) isn't good enough to detect Weak Link IP
  - Non-directed Graph in RWR algorithm
    - It can't limit threat score propagating direction only from seeds to unknow IP through weak linked file
  - Benign Pattern (Multiple IPs to Single File) File affects the detection results
    - It equally divides the threat score from seed to benign IPs
- We use Trustrank Algorithm (Directed Graph) and ignore Popular(frequently-accessed) Benign Files to resolve above issues

1. Ignore popular benign pattern file which has connection with Treat Seeds, except the following case.
2. keep popular file whose entropy isn't close to 0
   - The equation of **Single File's Entropy (SFP)**

$$SFP = -p_i \sum_{i}^{n} log\, p_i$$

   - P_i : The probability of i-th IP accessing this file, and n is the number of IPs which have accessed to this file

ADDITIONAL ENHANCING (CONT'D)

- In Case 3 (Apache, System Compromise), we show the Weak Link IP's place in ***Top 40 IPs*** of two scenarios (e.g., Web File Only and Ignore Popular Benign Pattern) through RWR and Trustrank Algo.

Weak Link IP	Web_File_Only		Ignore_Benign_Pattern	
	RWR	Trustrank	RWR	Trustrank
IPs List	32th	14th	2nd	2nd
	31th	24th	1st	1st
	30th	37th	4th	4th
	29th	13th	3rd	3rd

- In the above results, Trustrank can further raise the place of Weak Link IP than RWR
- Moreover, ignore popular benign pattern is ***most important step*** for propagation algorithm

ADDITIONAL ENHANCING (CONT'D)

- In our goal, we need to completely detect all possible Weak Link IPs
- Hence, ***Recall*** is most important for us, and we obtain the great recall rate when using Trustrank algo.

Web_File_Only		TP	FP	TN	FN	Precision	Recall
RWR	Top 7 is Positive	3	4	29	4	0.43	0.43
	Top 10 is Positive	3	7	26	4	0.3	0.43
	Top 20 is Positive	3	13	16	4	0.15	0.43
	Top 30 is Positive	3	23	6	4	0.1	0.43
Trustrank	Top 7 is Positive	3	4	29	4	0.43	0.43
	Top 10 is Positive	3	7	26	4	0.3	0.43
	Top 20 is Positive	5	15	18	2	0.25	0.72
	Top 30 is Positive	6	24	9	1	0.2	0.86
Ignore_Benign_Pattern		**TP**	**FP**	**TN**	**FN**	**Precision**	**Recall**
RWR	Top 7 is Positive	7	0	33	0	1.0	1.0
	Top 10 is Positive	7	3	30	0	0.7	1.0
	Top 20 is Positive	7	13	20	0	0.35	1.0
	Top 30 is Positive	7	23	10	0	0.24	1.0
Trustrank	Top 7 is Positive	7	0	33	0	1.0	1.0
	Top 10 is Positive	7	3	30	0	0.7	1.0
	Top 20 is Positive	7	13	20	0	0.35	1.0
	Top 30 is Positive	7	23	10	0	0.24	1.0

ADDITIONAL ENHANCING (CONT'D)

- Try to perform the script of second scenario: **SQL Injection** on Case 1 data
- Perhaps, we can discover interesting events

Detected IP	True / False Positive	Reason
.asp	FALSE	
.php	TRUE	Multiple webpages accessed by same parameter
.asp	FALSE	
.php	TRUE	Internal Error by Illegal Character
.php	TRUE	SQL Injection
.php	TRUE	SQL Injection
.asp	FALSE	
.php	TRUE	SQL Injection
.asp	TRUE	SQL Injection
.asp	TRUE	SQL Injection
.asp	TRUE	SQL Injection
.php	TRUE	SQL Injection

IPs List

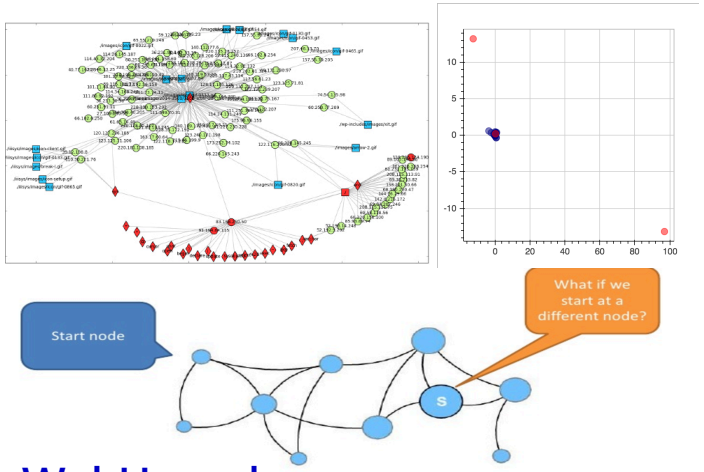
There are two IPs were detected by the script of System Compromise

ANOMALY DETECTION & GRAPH MINING

- ChainSpot
- WebHound
- Playwright

PLAYWRIGHT - RECOVERY CYBERCRIME FROM REAL-WORLD WEB-ACCESS LOGS

基於存取日誌及攻擊重塑之伺服器入侵還原技術



WebHound

- *Problem*: How to interpret and understand what hacker did or wanted based on created web access-logs in different attacking strategy ?
- *Idea:* Using Storyline to individually check what hacker does against the Web Sever. 1) Identify web-access logs of malicious IPs , 2) Attacking Event identification, 3) Intelligence Collection & Match, 4) OWASP Rules Reconstruction, and 5) Represent attacks on the story-line.
- *Contribution:* *Playwright* could globally let forensics realizing what hacker did in the Web Server.

Recovery Hacker Attacks on the Storyline

OWASP The Open Web Application Security Project

URI Parameter Analyzer

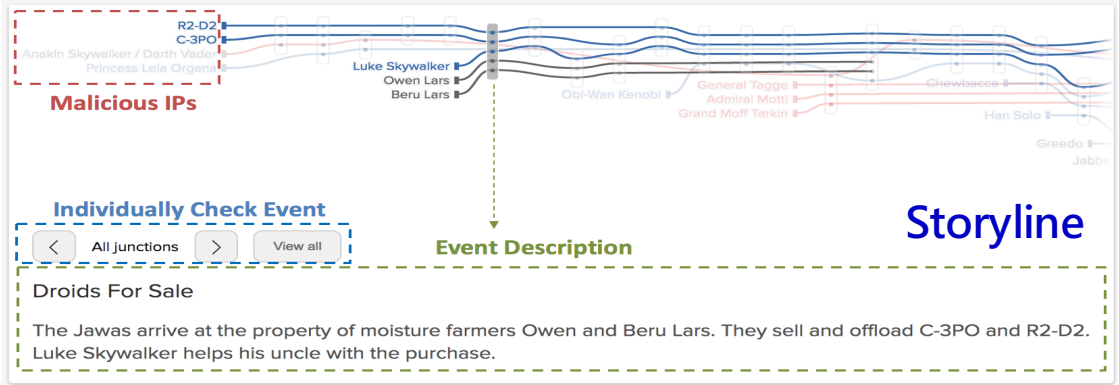
cve.mitre.org

acunetix

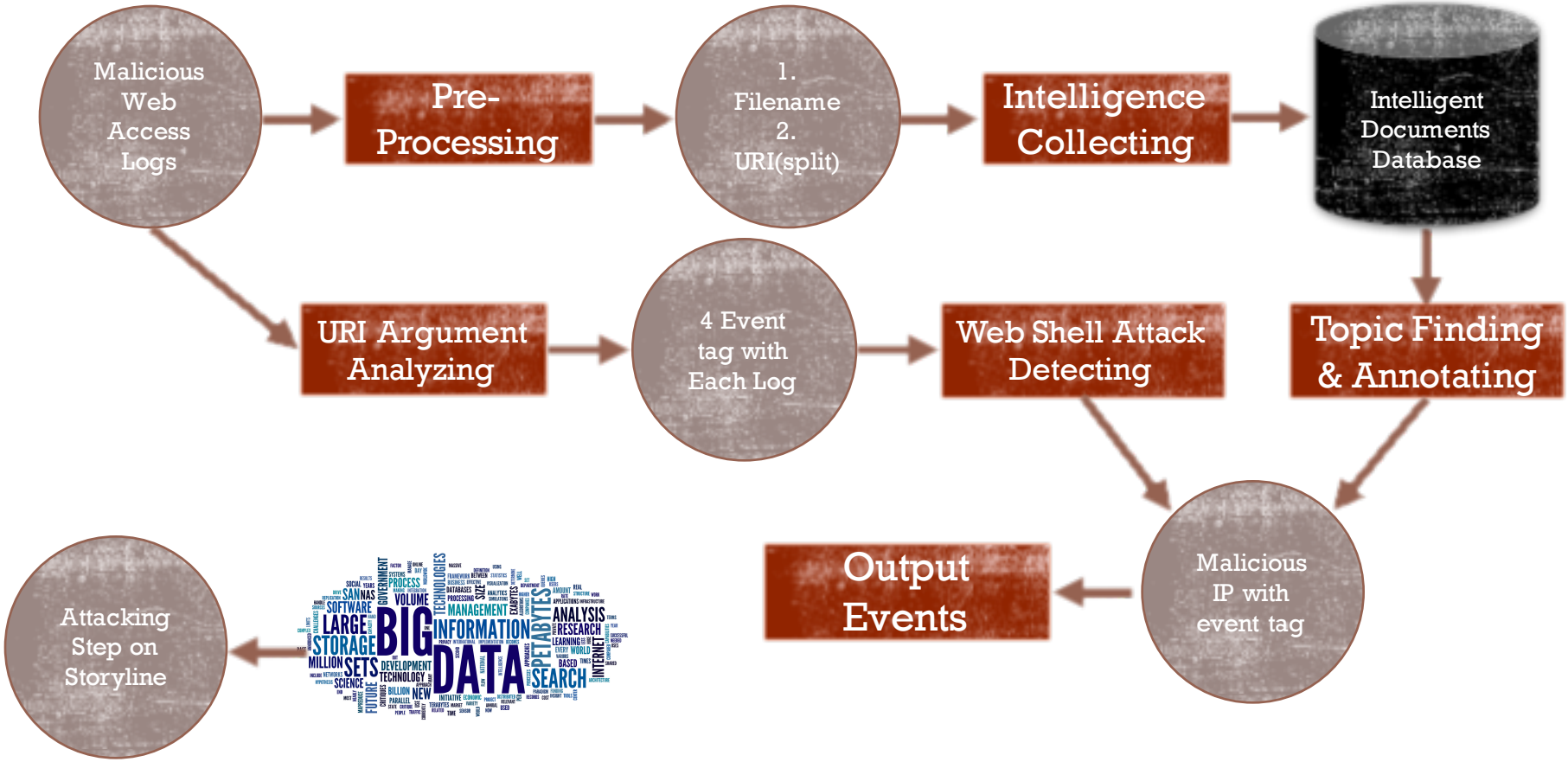
Nmap

Intelligence Collection

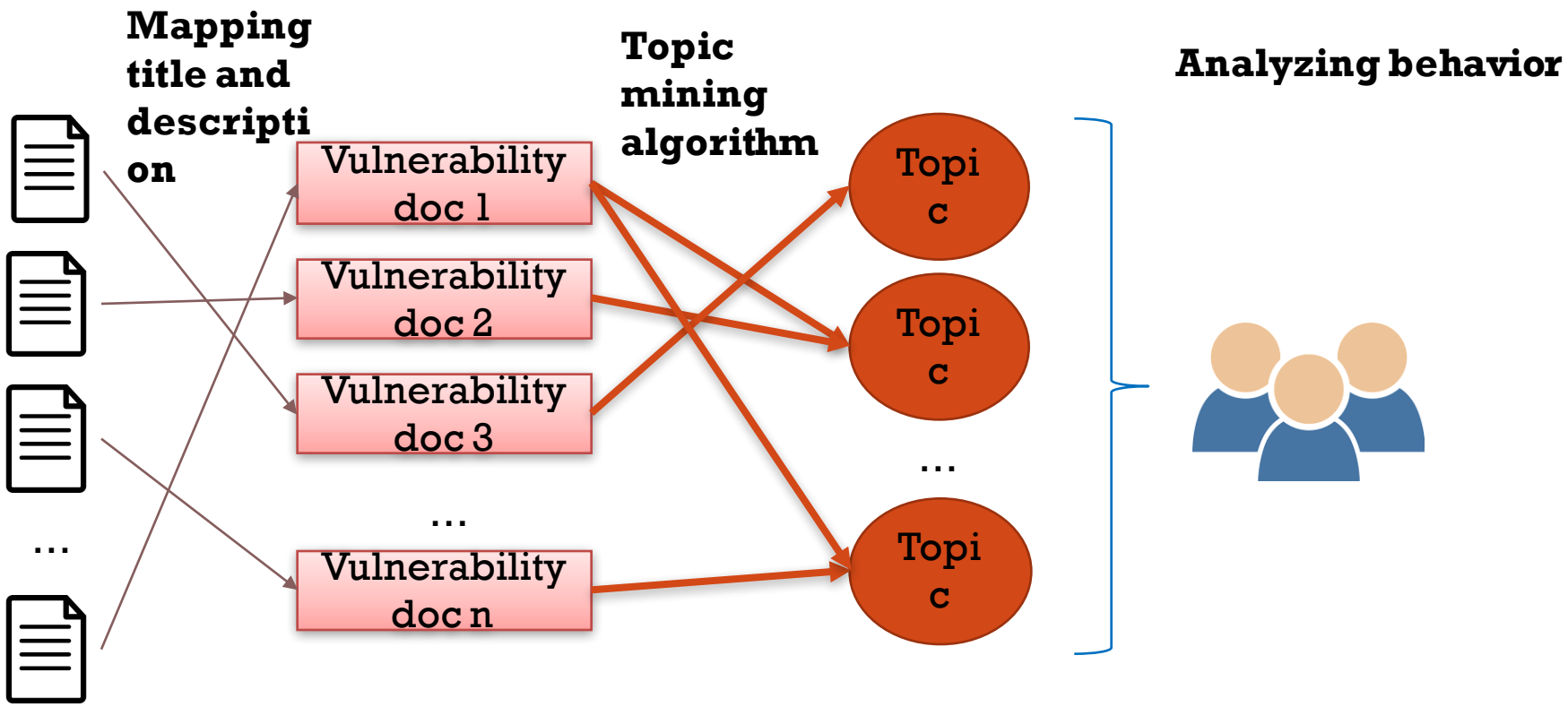
1. Use Topic Mining & Parameter analysis to define attacking events which are drawn on the time-line.



SYSTEM ARCHITECTURE



INTELLIGENCE COLLECTION & MATCH



DATA PREPROCESSING

- Input: Vulnerability documents
  - (528 docs from: https://www.acunetix.com/vulnerabilities/web/)
- Output: Doc-term matrix
- Use n-gram(n=4) and security terms(from Sans) to generate terms in each vulnerability documents

Sample sequence	1-gram sequence	2-gram sequence	3-gram sequence
… to be or not to be …	…, to, be, or, not, to, be, …	…, to be, be or, or not, not to, to be, …	…, to be or, be or not, or not to, not to be, …

WEIGHTING FUNCTION

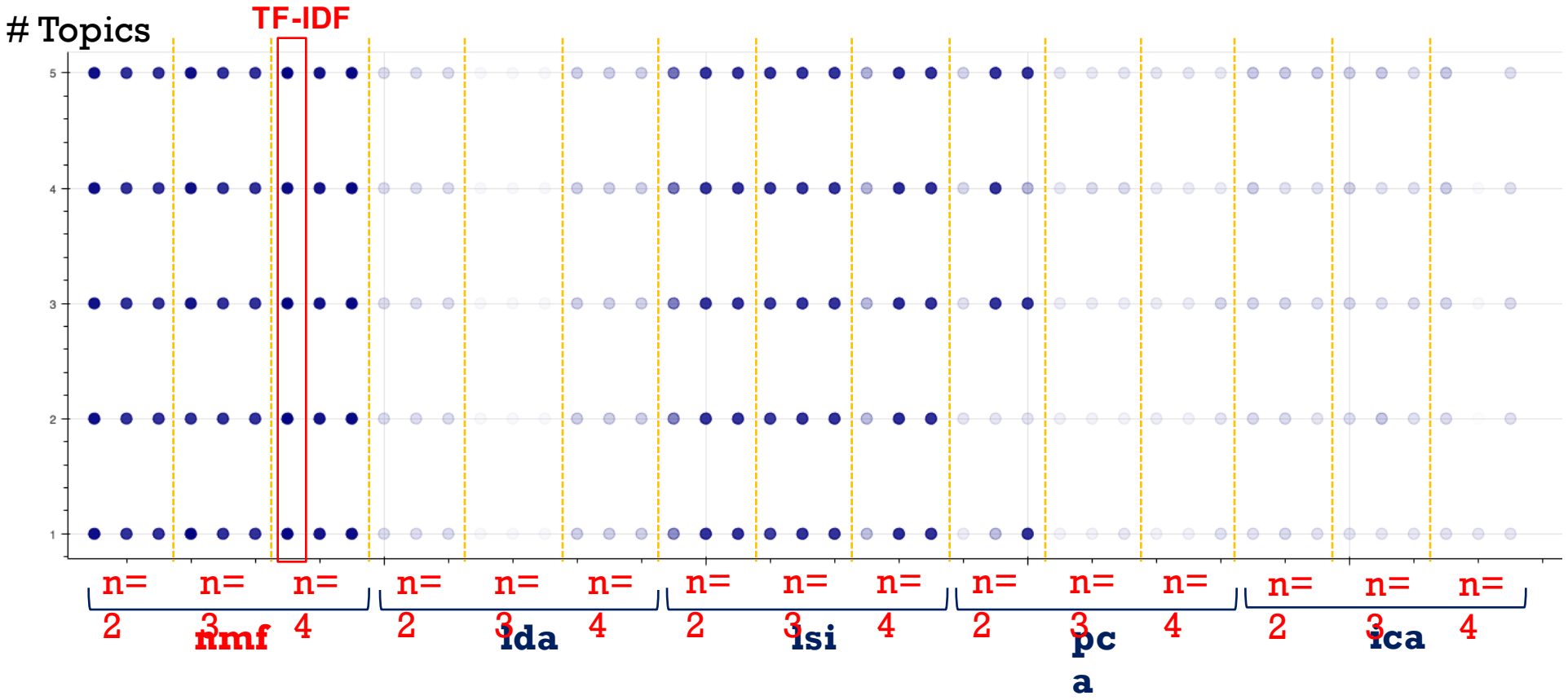
- How to choose suitable weighting function?
  - TF-IDF
  - OKAPI BM25
  - DTB

TOPIC FINDING

- How to choose suitable topic model?
  - ICA
  - PCA
  - LSI
  - LDA
  - NMF

EXPERIMENTAL RESULTS (1)

- Use **meta-score** to select suitable weighting function and model



EXPERIMENTAL RESULTS (2)

- Use n-gram(n=1~4) and find 10 topics from those vulnerability documents
- We find that NMF with n=4 have high meta-score, and it is easy to annotate the topics.

TOPICS & DOCS MAPPING TABLE (1)

- In order to remove some document not relative to the topic its belong to, we need to select important terms in the term set.

- How to select important terms in every topic-term set?

TOPICS & DOCS MAPPING TABLE (2)

- Find a threshold of **term weight** to keep more important terms in topic

- Term weight:
  - Each term consists of several words
  - The term weight is the sum of each word frequency in the topic

TOPICS & DOCS MAPPING TABLE (3)

- Find k largest term weight to get more important terms, $\lambda_1, \lambda_1, \ldots \lambda_k$

  such that $\sum_{i=1}^{k} \lambda_i \geq \alpha \times \sum_{j=1}^{n} \lambda_j$ and

  $(\lambda_k - \lambda_{k+1}) \geq \beta \times (\lambda_{k-1} - \lambda_k)$

  - $\alpha = 0.8$
  - $\beta = 1.0$

Alpha 越接近 1 涵蓋的範圍越廣
Beta 越大轉折點的斜率差越大

URI ARGUMENT ANALYSIS

- Analyzing the arguments in URI query
  - E.g., http://yahoo.com.tw/index.asp?page=1
- Which event we focus:
  - SQLI
    - Rule: {.yml | .sql} - SQL configure file
    - Case: /config/database.yml
  - XXS
    - Rule: =<script>.*.</script> - Script Language
    - Case
      - sections=All<script>alert(12345)</script>

URI ARGUMENT ANALYSIS (CONT'D)

- Which event we focus:
  - Entry Point Identification
    - Rule: {../ | cmd= | dir } - print and change the director
    - Case
      - /mailer/?mid=../../../%00
  - Command Execution
    - Rule: "java.*.\..*." - java language import lib
    - Case
      - pageTitle=${new%20java.lang.Integer(100116%2b100028)}

HOW MANY EVENTS WE WOULD LABEL ?!

- ***Intelligence Collection & Comparison***
  1. SQL Injection
  2. Cross-site Script
  3. Apache Remote Execution
  4. Older Server Version
  5. Weak Password
  6. PHP Remote Execution
  7. Sensitive Info. Disclosure
  8. TCP/UDP Service Identification
  9. Repository Identification
  10. Entry Point Identification

- *URI Argument Analysis*
  1. SQL Injection
  2. Cross-site Script
  3. Entry Point Identification
  4. Command Execution

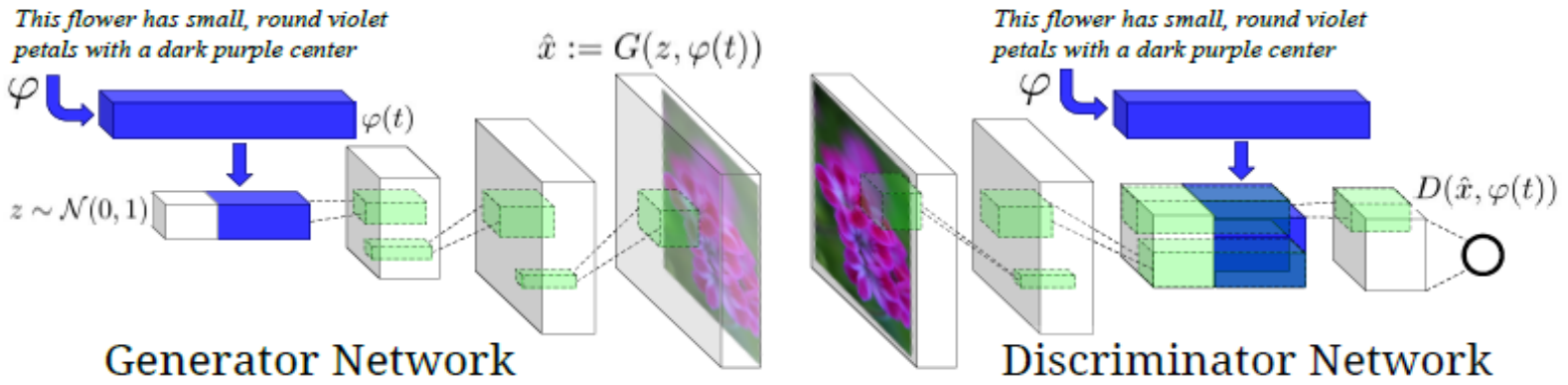
EVENT TAGGING ON STORYLINE

OTHER CYBER APPLICATION IN AI

- Malware Analysis

MALWARE CLASSIFICATION WITH CONDITIONAL GANS (CGAN)

- Conditional GAN (CGAN)
- Reed, etc.: Synthesize image conditioned on text



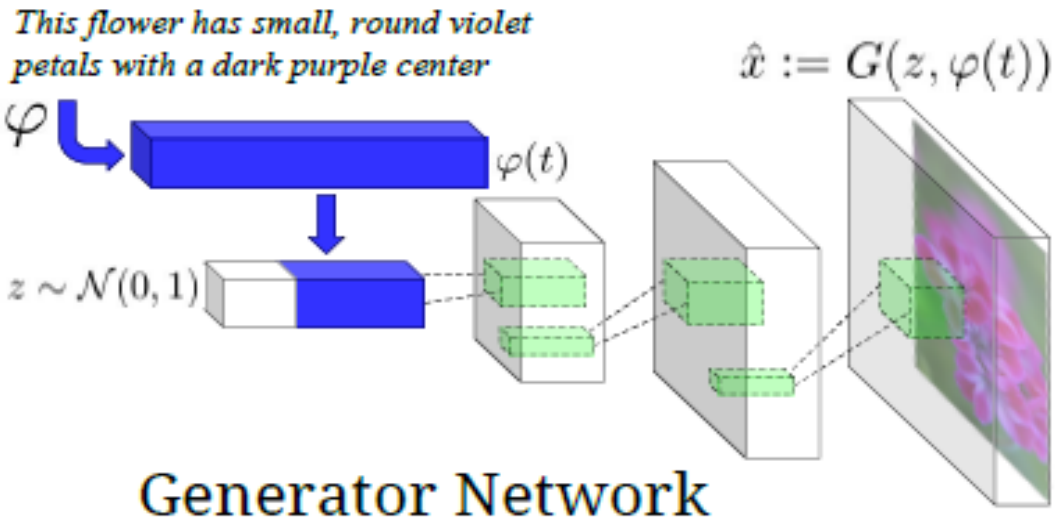
a large bird has a stumpy bill, large tufts of black feathers on its breast, and a black crown.
a small bird with a black eye, black head, and dark bill.
a solid black bird with long tail feathers and a rounded beak that looks vey unusual.
medium black white and brown bird with medium black tarsus and medium black and white beak
all black bird with a small bird and all black eyes.
this particular bird has all black feathers and a black bill and black eyes
the bird is completely black with a small head and rounded beak that blends into the head.
this bird has shiny black feathers and a curved, short beak.
this bird is all black and has a very short beak.
this bird has wings that are black and has a thick bill

NOW, WE FOCUS ON "APIMDS" AS OUR POC TARGET

- APIMDS (API-based malware detection system)
  - Ki, Youngjoon, Eunjin Kim, and Huy Kang Kim. "A novel approach to detect malware based on API call sequence analysis." *International Journal of Distributed Sensor Networks* (2015).
- It contains hash strings as malware IDs of {Trojan(1000), Adware(1000), Worm(865), Packed(964)} types.
- There exist one API sequence, whose length is 50~300, corresponding to Each malware IDs. E.g., "localalloc"->"createsemaphore"->"globaladdatomw"->…
- And we downloaded corresponding binary sample for each malware ID from VirusTotal.
- Here we take binary sample as "image" while API sequence as "description".

THE 1ST STEP: MAKE SURE DESCRIPTION BEING PRECISE AND COMPACT

- The encoded vector is responsible for parameterizing and restricting synthesized samples
- The representability of encoded vector significantly affects the effectiveness of generating customized synthesized samples.
- Such that we need to verify the correctness of word2vector encoding.



BASIC IDEA

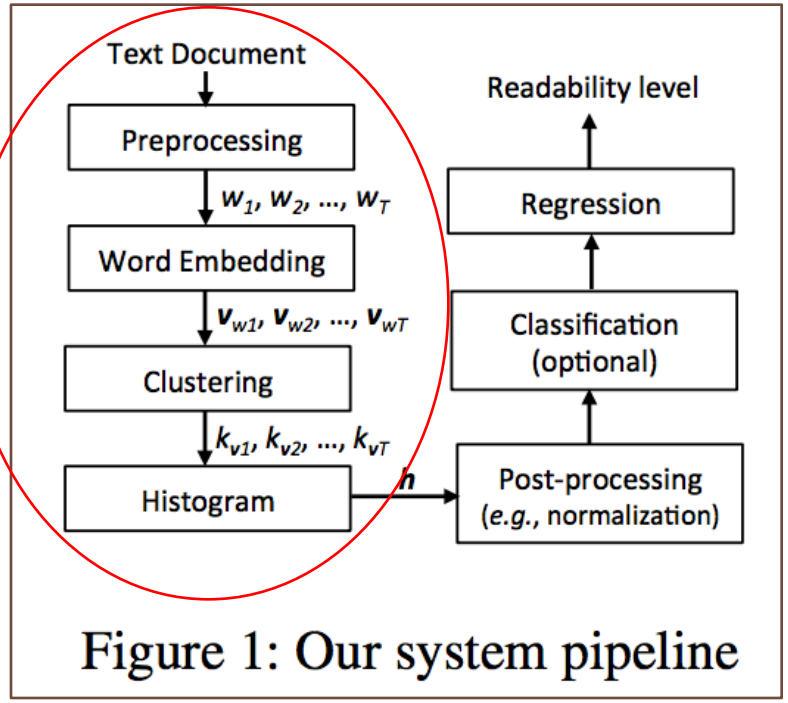
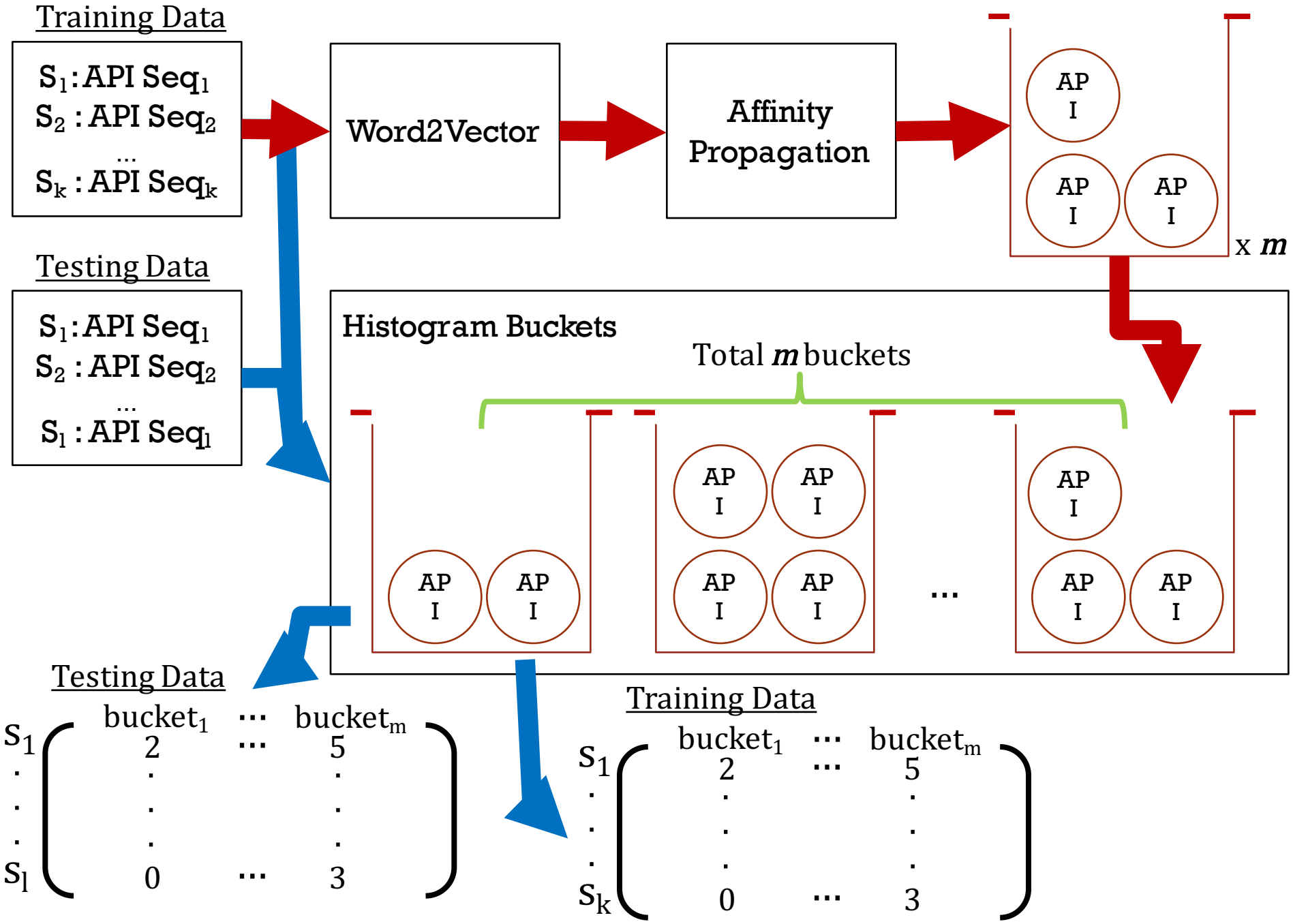


Figure 1: Our system pipeline

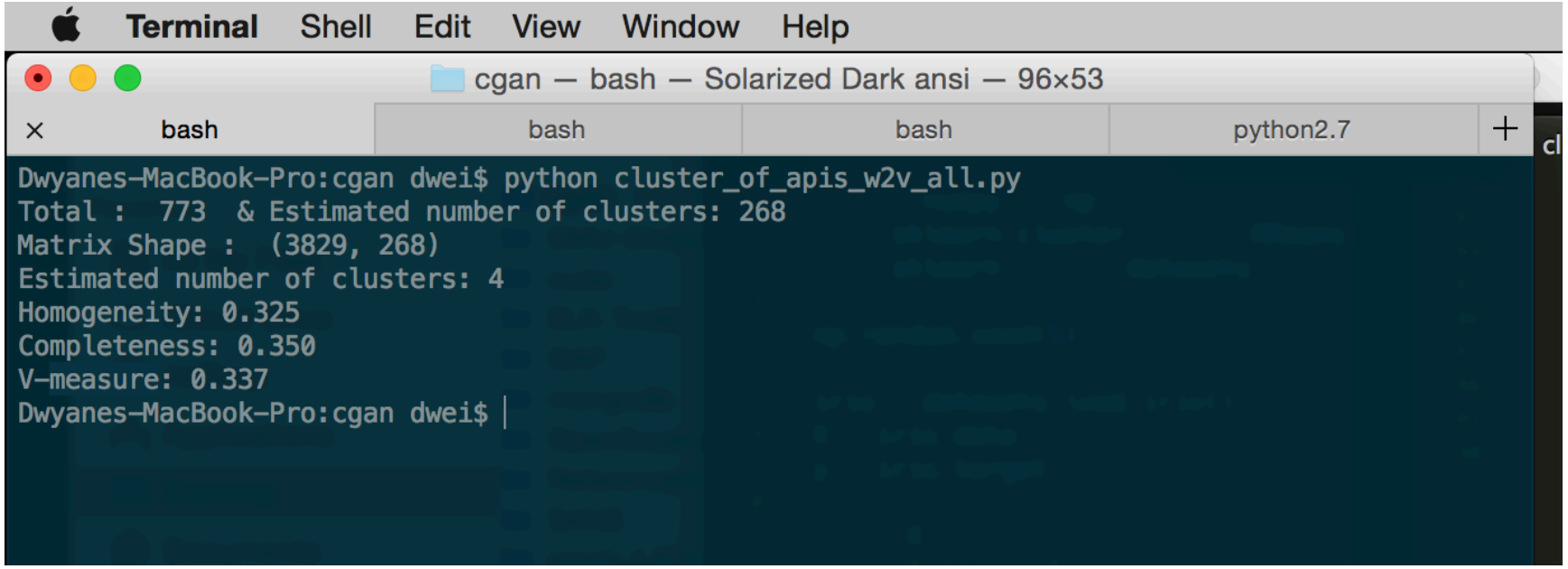
Paper: Language Modeling by Clustering with Word Embeddings for Text Readability Assessment

Based on this idea, we performed 4 experiment with various setting.

SYSTEM ARCHITECTURE



WORD2VEC + AP + K-MEANS ON ALL API SEQUENCES



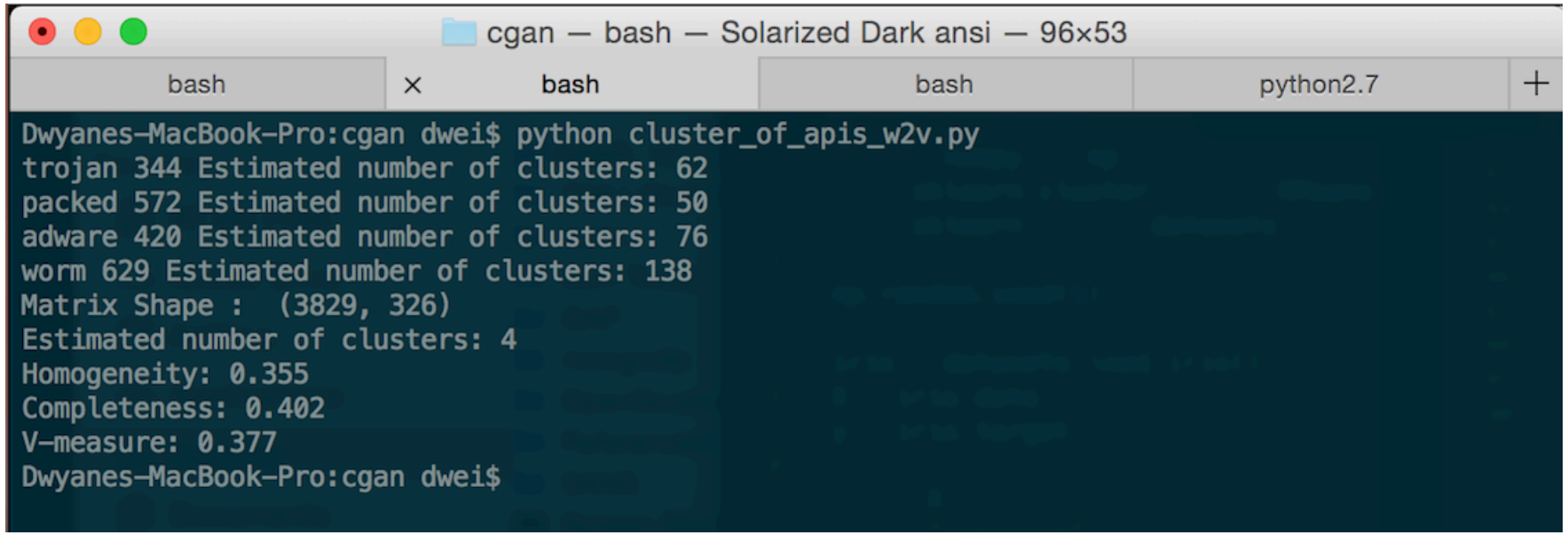
Homogeneity: A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class.

Completeness: A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.

The V-measure is the harmonic mean between homogeneity and completeness:

```
v = 2 * (homogeneity * completeness) / (homogeneity + completeness)
```

GIVEN LABEL THEN WORD2VEC +AP+ K-MEANS



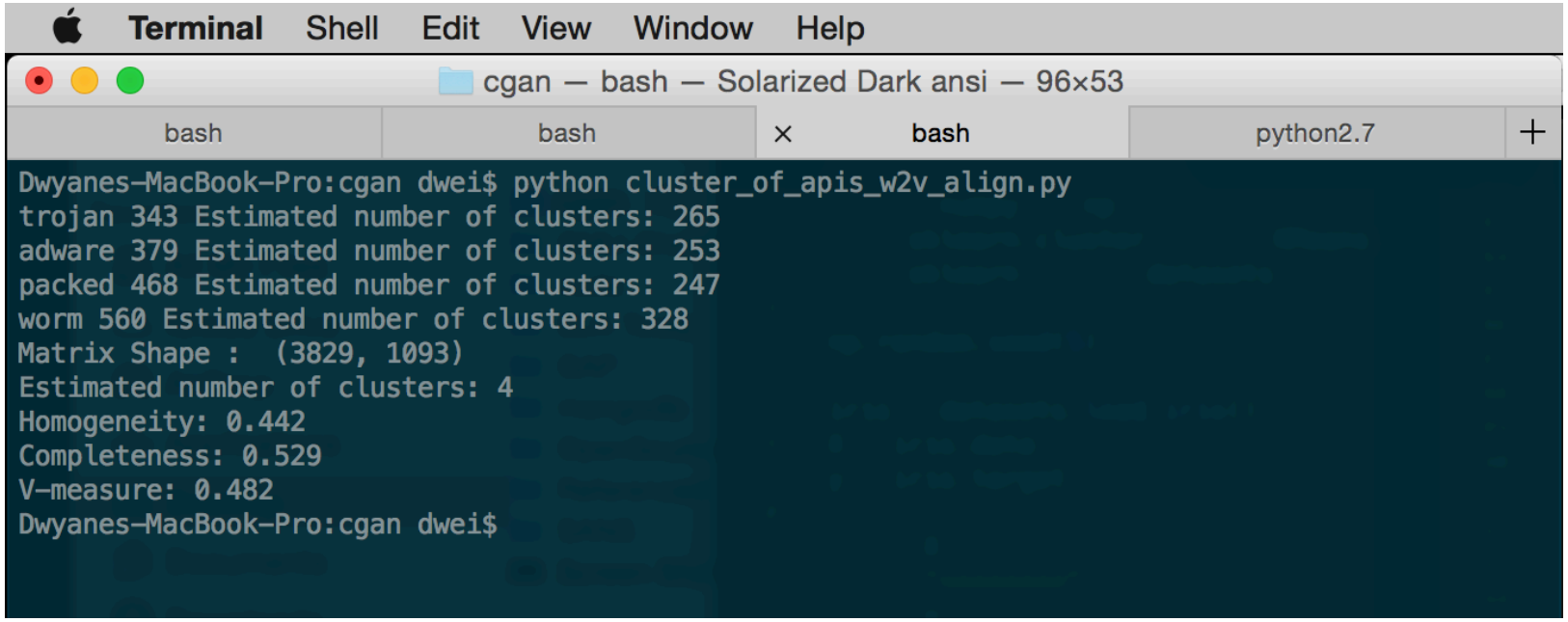
Homogeneity: A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class.

Completeness: A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.

The V-measure is the harmonic mean between homogeneity and completeness:

```
v = 2 * (homogeneity * completeness) / (homogeneity + completeness)
```

GIVEN LABEL ALIGNMENT + WORD2VEC+AP+ K-MEANS



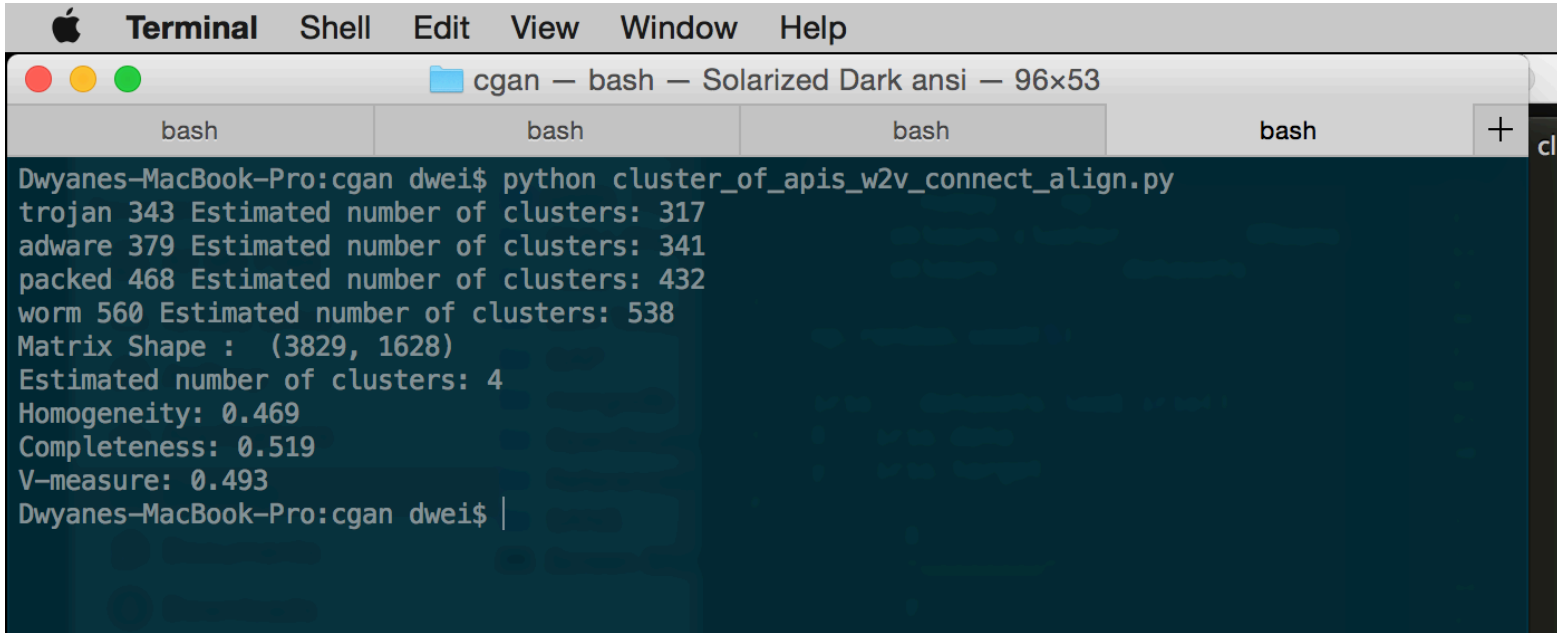
Homogeneity: A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class.

Completeness: A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.

The V-measure is the harmonic mean between homogeneity and completeness:

```
v = 2 * (homogeneity * completeness) / (homogeneity + completeness)
```

GIVEN LABEL THEN ALIGNMENT+WORD2VEC +CONNECTED COMPONENT + K-MEANS



Homogeneity: A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class.

Completeness: A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.

The V-measure is the harmonic mean between homogeneity and completeness:

```
v = 2 * (homogeneity * completeness) / (homogeneity + completeness)
```

VERIFICATION OF ABOVE 4 APPROACHES (TRAIN : TEST = 1 : 1)

1. Word2vec + AP + SVM on All API sequences

True/Pred	Trojan	Packed	Adware	Worm	Acc.
Trojan	500	0	0	0	100%
Packed	0	482	0	0	100%
Adware	0	3	495	2	99.6%
Worm	0	0	0	433	100%

2. Given label then Word2vec + AP + SVM

True/Pred	Trojan	Packed	Adware	Worm	Acc.
Trojan	500	0	0	0	100%
Packed	0	481	1	0	99.8%
Adware	0	3	495	2	99%
Worm	0	0	4	429	99.1%

3. Given Label Alignment + Word2vec + AP + SVM

True/Pred	Trojan	Packed	Adware	Worm	Acc.
Trojan	500	0	0	0	100%
Packed	0	481	1	0	99.8%
Adware	0	3	496	1	99.2%
Worm	0	0	4	429	99.1%

4. Given label then Alignment + Word2vec + Connected Component + SVM

True/Pred	Trojan	Packed	Adware	Worm	Acc.
Trojan	500	0	0	0	100%
Packed	0	481	1	0	99.8%
Adware	0	3	496	1	99.2%
Worm	0	0	4	429	99.1%


THANK YOU