Intelligent Intrusion Detection System

ACM Winter School 2019 on Cybersecurity

December 13, 2019

Kamalkanta Sethi, Research Scholar
and
Rahul Kumar, B. Tech Final-Year CSE

SCHOOL OF ELECTRICAL SCIENCES

INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

October 2019

Outline

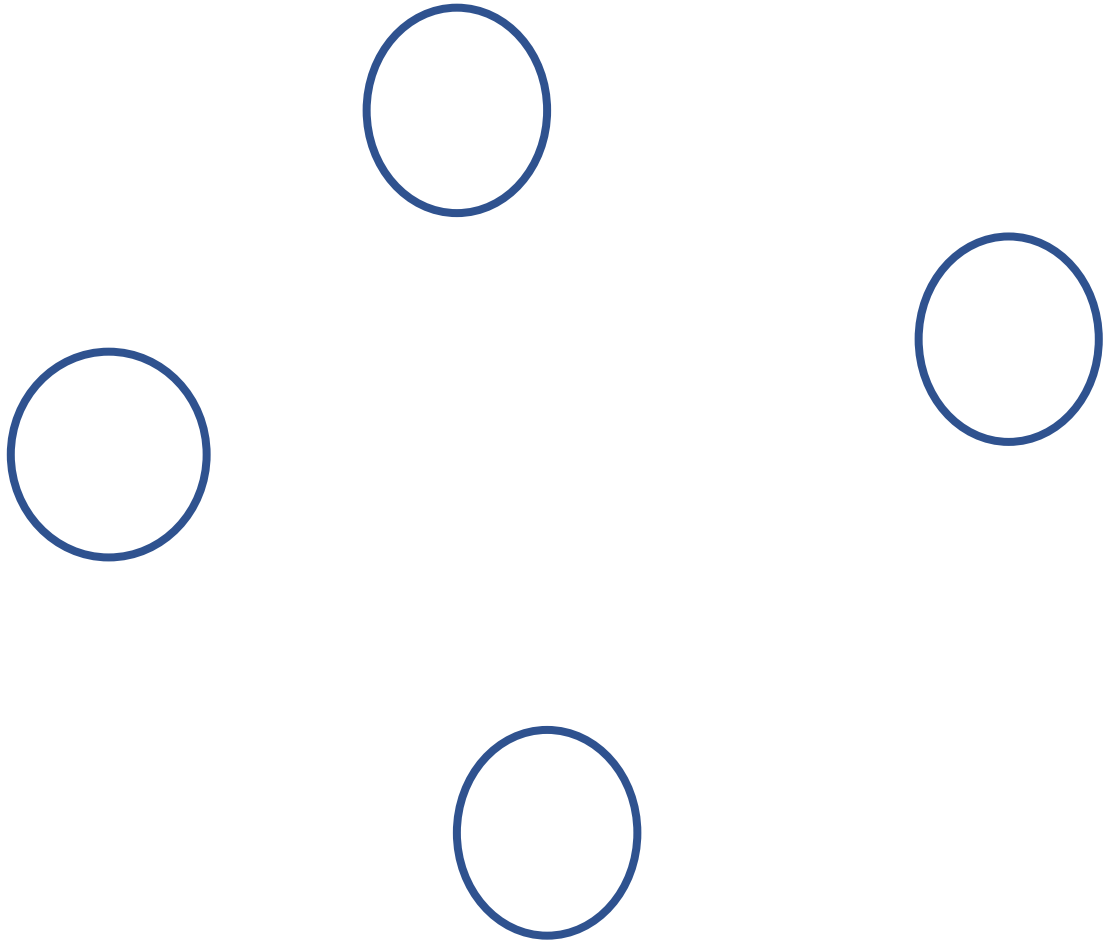
1. Introduction to Network Attacks and discussion on its different types

1. Introduction to Intrusion Detection Systems and its different types

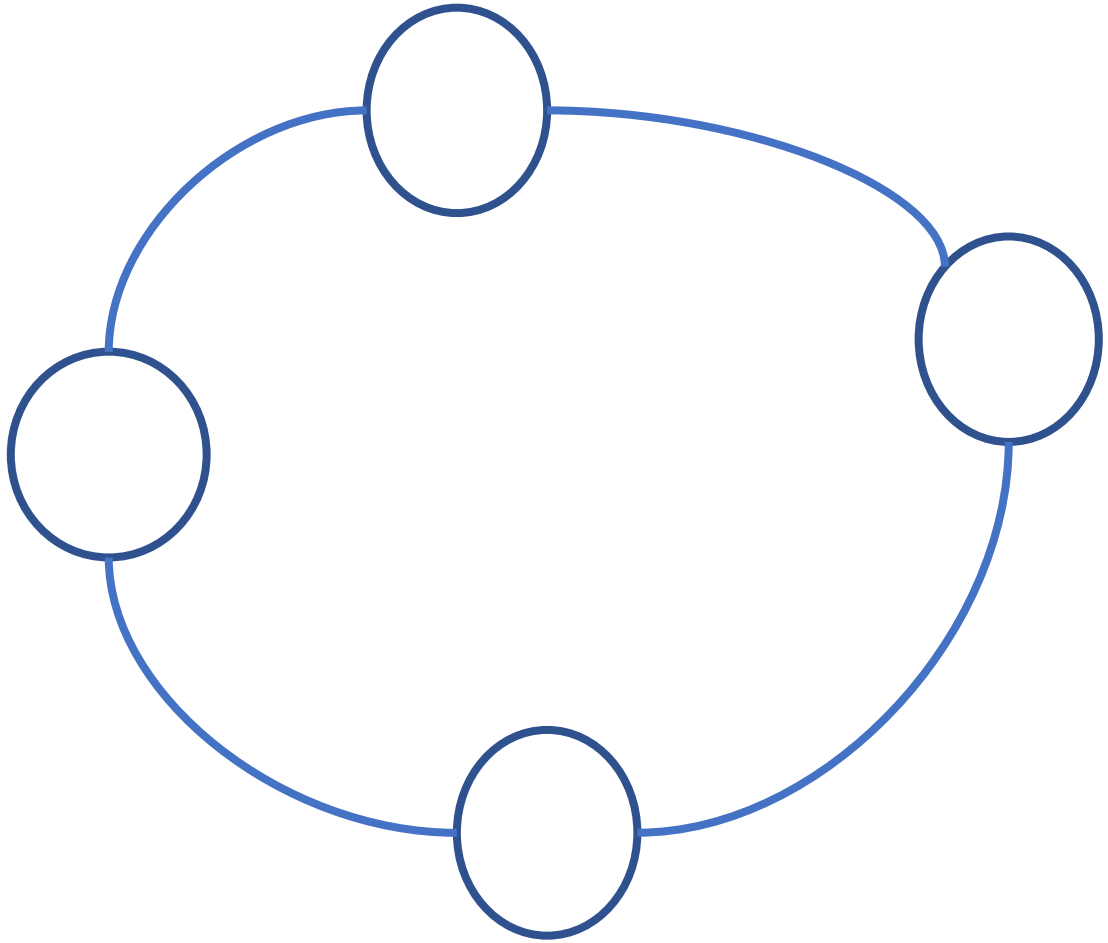
3. Machine Learning methods and related Intrusion Detection System (IDS)

4. Our research work : Context Adaptive IDS using Reinforcement Learning

Introduction to network: nodes

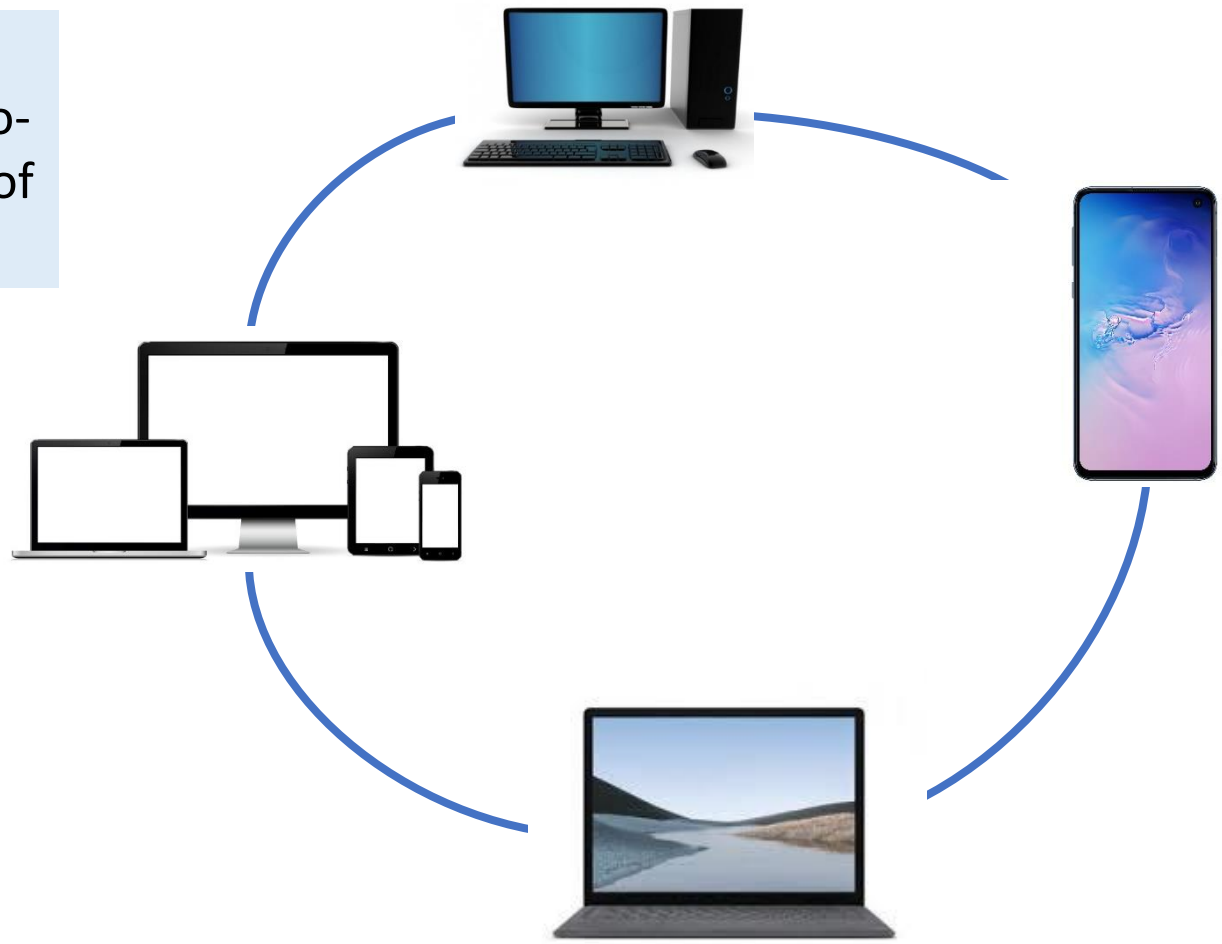


Introduction to network: nodes and vertices

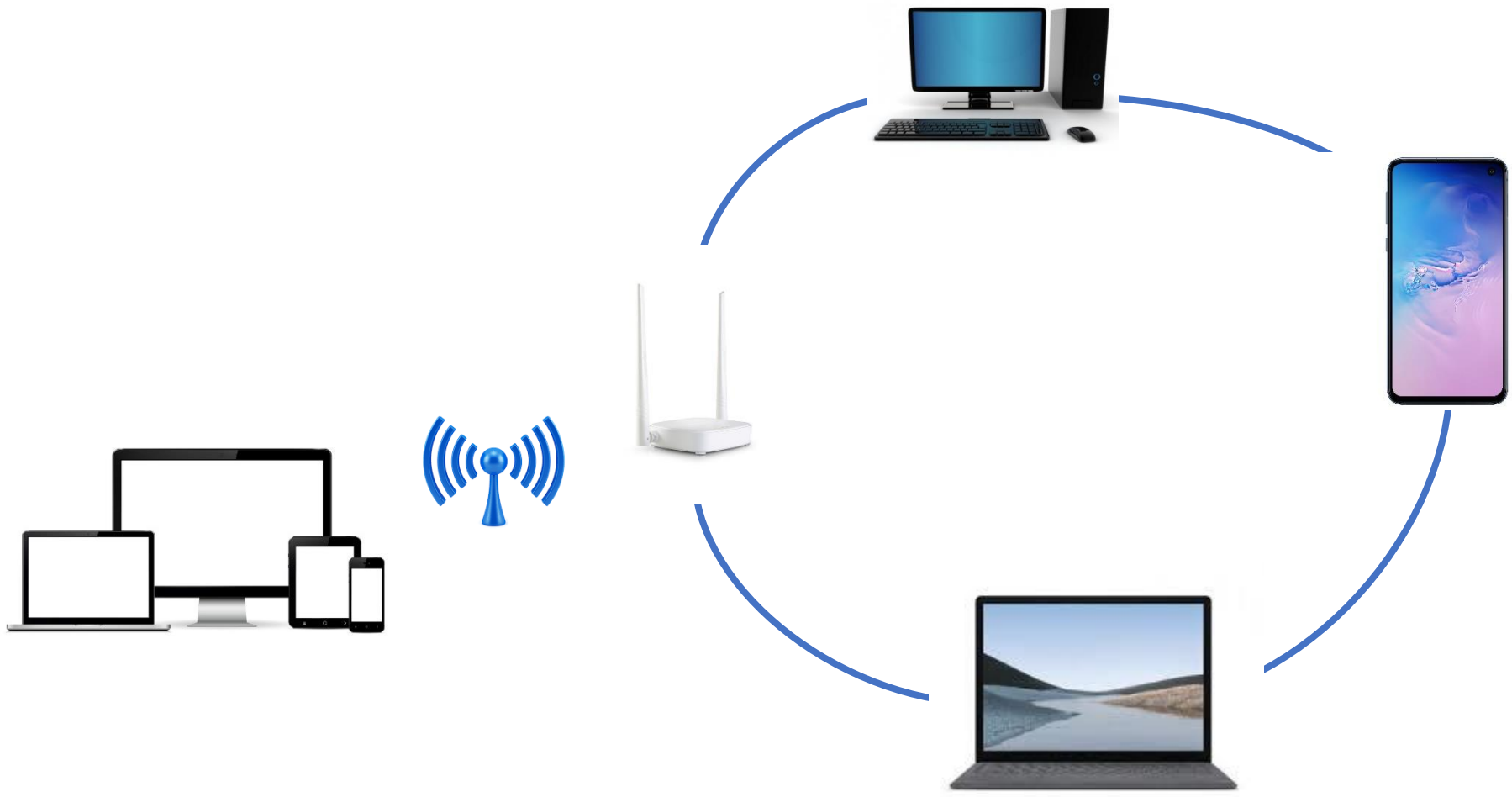


Introduction to network

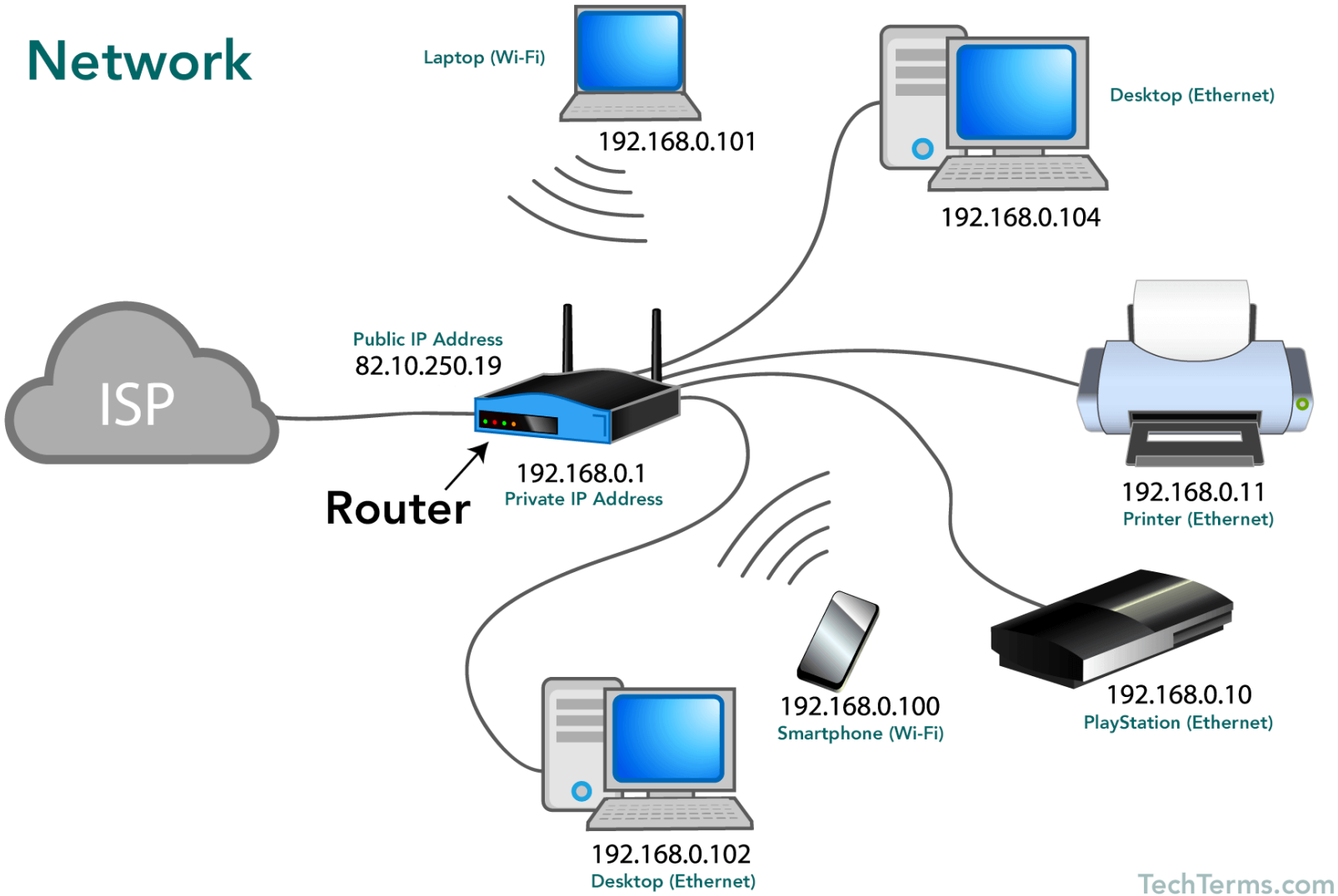
A network is a connected graph mathematically defined as a two-tuple <V,E> , where V is the set of nodes and E is the set of edges.



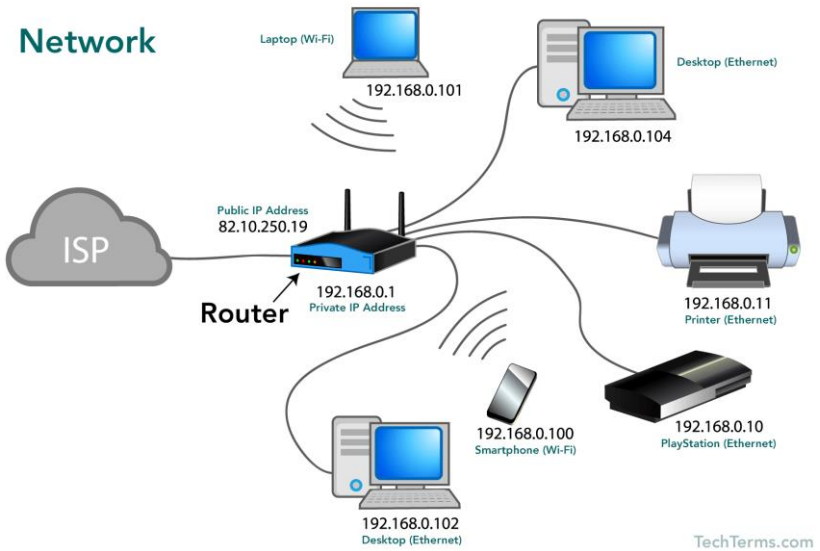
Introduction to network: wired and wireless

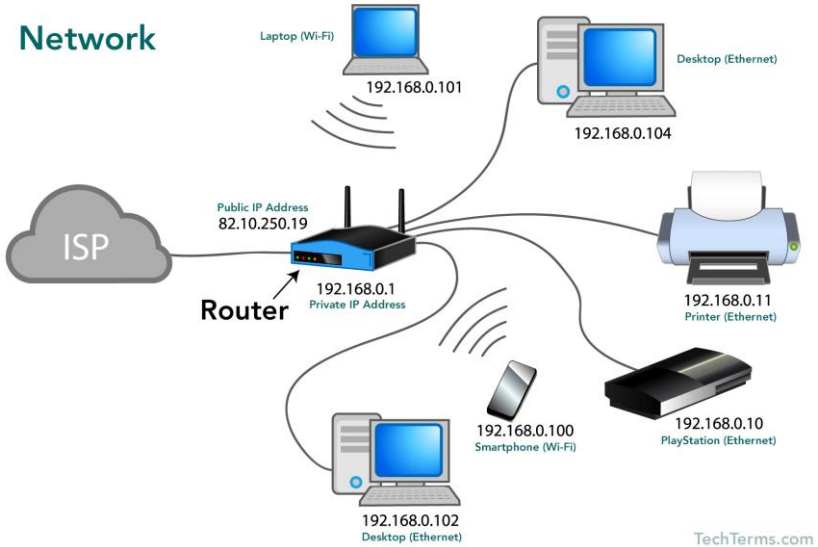


Network architecture

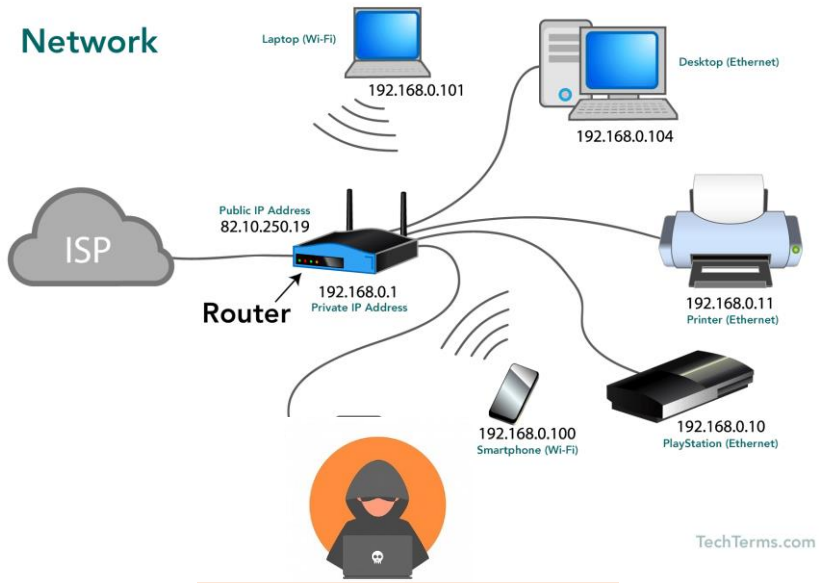


Internet

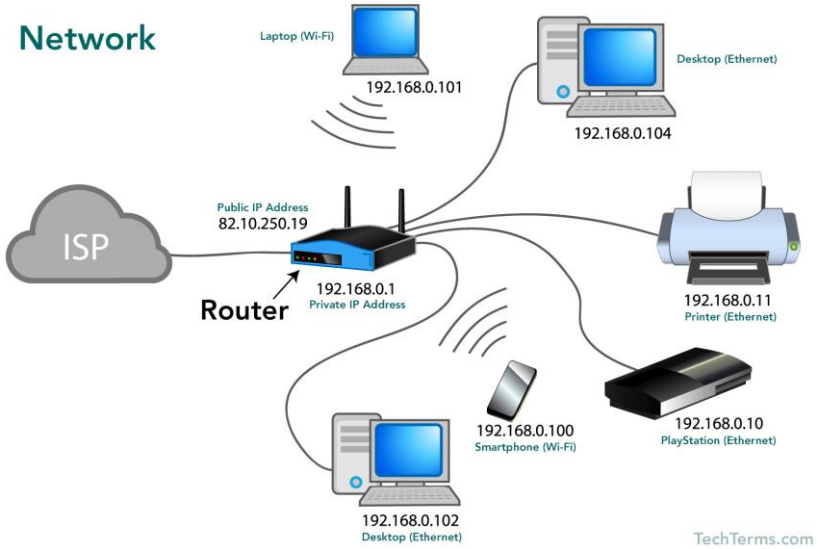

Network
Laptop (Wi-Fi)
192.168.0.101
Desktop (Ethernet)
192.168.0.104
Public IP Address
82.10.250.19
ISP
Router
192.168.0.1
Private IP Address
192.168.0.11
Printer (Ethernet)
192.168.0.100
Smartphone (Wi-Fi)
192.168.0.10
PlayStation (Ethernet)
192.168.0.102
Desktop (Ethernet)
TechTerms.com


Network
Laptop (Wi-Fi)
192.168.0.101
Desktop (Ethernet)
192.168.0.104
Public IP Address
82.10.250.19
ISP
Router
192.168.0.1
Private IP Address
192.168.0.11
Printer (Ethernet)
192.168.0.100
Smartphone (Wi-Fi)
192.168.0.10
PlayStation (Ethernet)
192.168.0.102
Desktop (Ethernet)
TechTerms.com

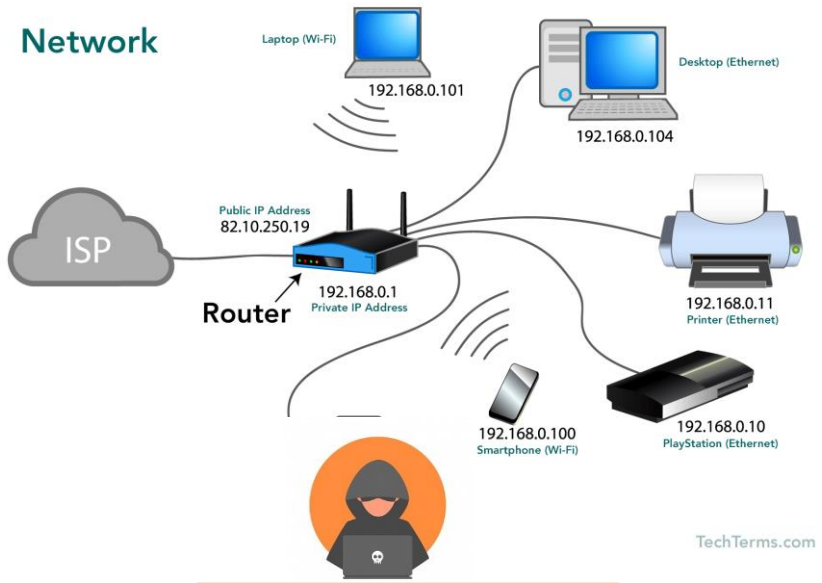
Malicious user



Attacker machine

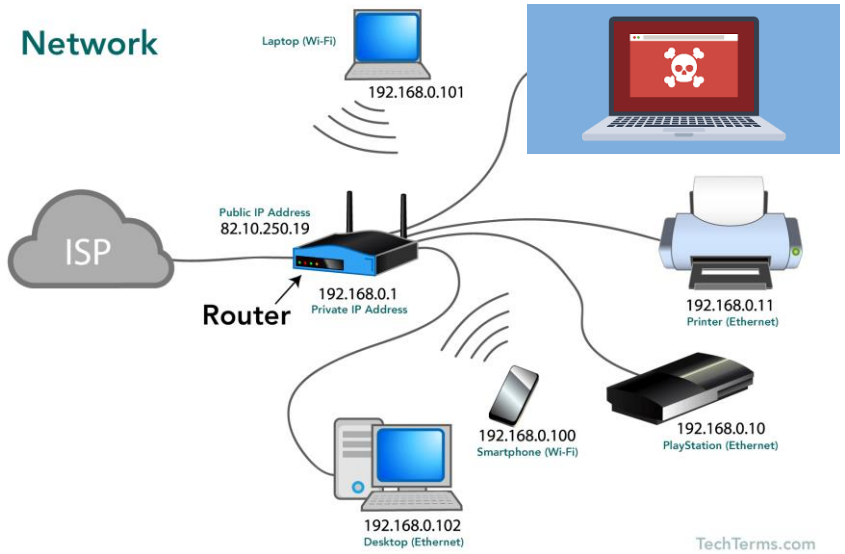


Network attack



Attacker machine

Victim machine



What is Intrusion?

- ***Intrusions:*** attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer system or network(illegal access).
- Intruders are classified into two groups.
  - External intruders do not have any authorized access to the system.
  - Internal intruders have at least some authorized access to the system. misuse their privileges or attempt to gain additional privileges for which they are not authorized.
- Although many intrusions are malicious in nature, many others are not; for example: a person might mistype the address of a computer and accidentally attempt to connect to a different system without authorization.
- Based on the target, it can be network intrusions or system intrusions. Network intrusions target to attack the network where system intrusions target to attack a particular system.

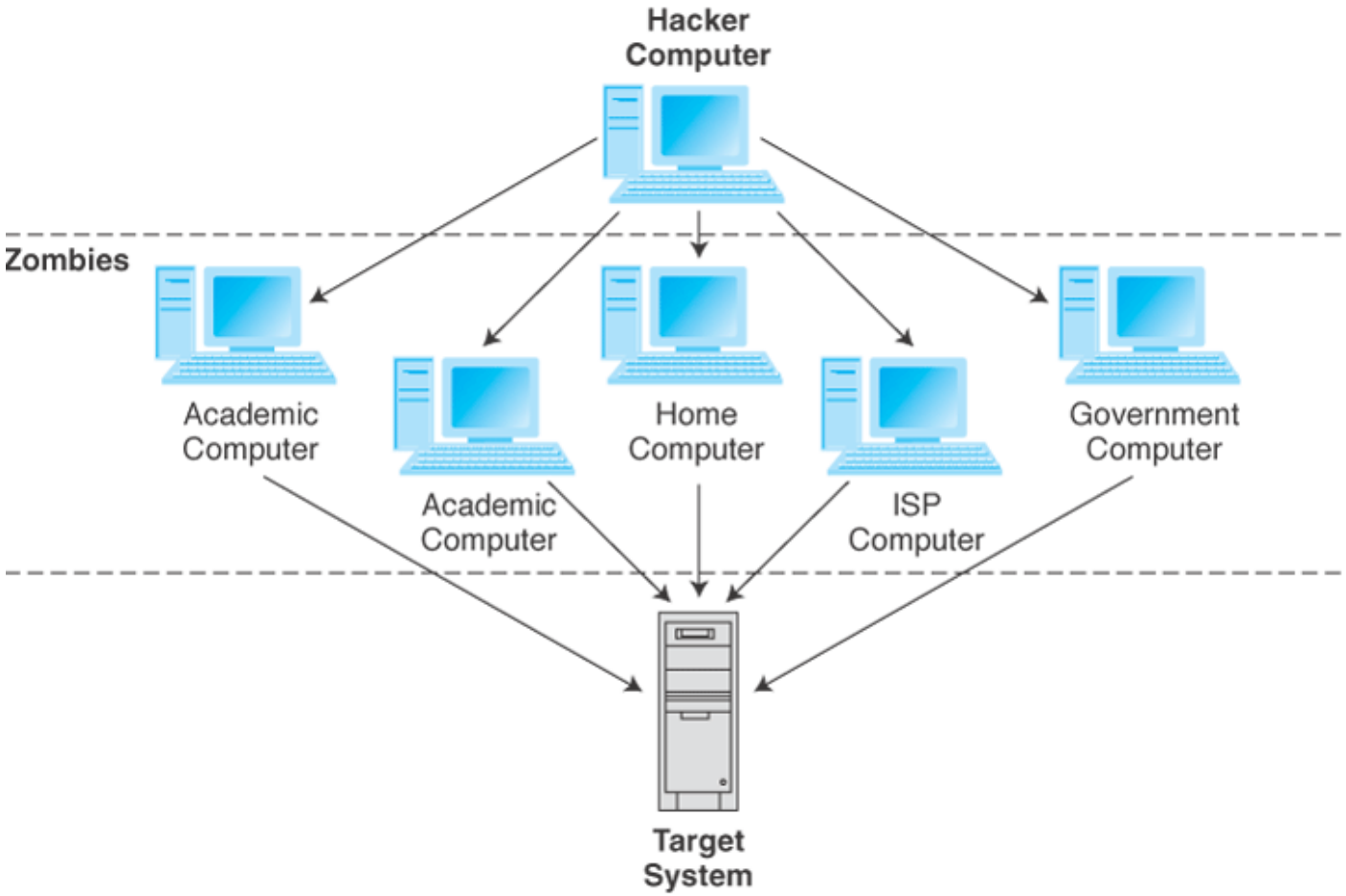
Types of Intrusions

- Network Intrusions: Intrusions are initiated based on a flow of packet sent over a network.
  - Denial of Service (DoS) attack
  - Distributed Denial of Service (DDoS) attack.
  - Man-in-the-Middle attack
  - IP Spoofing
  - Port Scanning
  - Hijack attack
  - U2R attack
  - Asymmetric routing
  - SYN-Flooding
- System Intrusions: Host attacks target specific hosts or system by running malicious software to compromise the system functionalities or corrupt it. Most host attacks are categorized under the malware category. This includes worms, viruses, adwares, ransomware.

Denial-of-service Attack

- A Denial of Service (DoS) attack: Here, the attacker aims at preventing authorized users to access service provided by server by sending a large number of packets, towards the server.
  - Here, attacker either consumes all available network bandwidth or consume system **resources** (mainly **CPU**, **memory**, storage space)
  - As a result, the server is unable to fulfill legitimate requests
- A **distributed denial of service (DDoS)** attack: It is launched from a large number of computers that are infected by malicious software (malware) controlled by attacker.

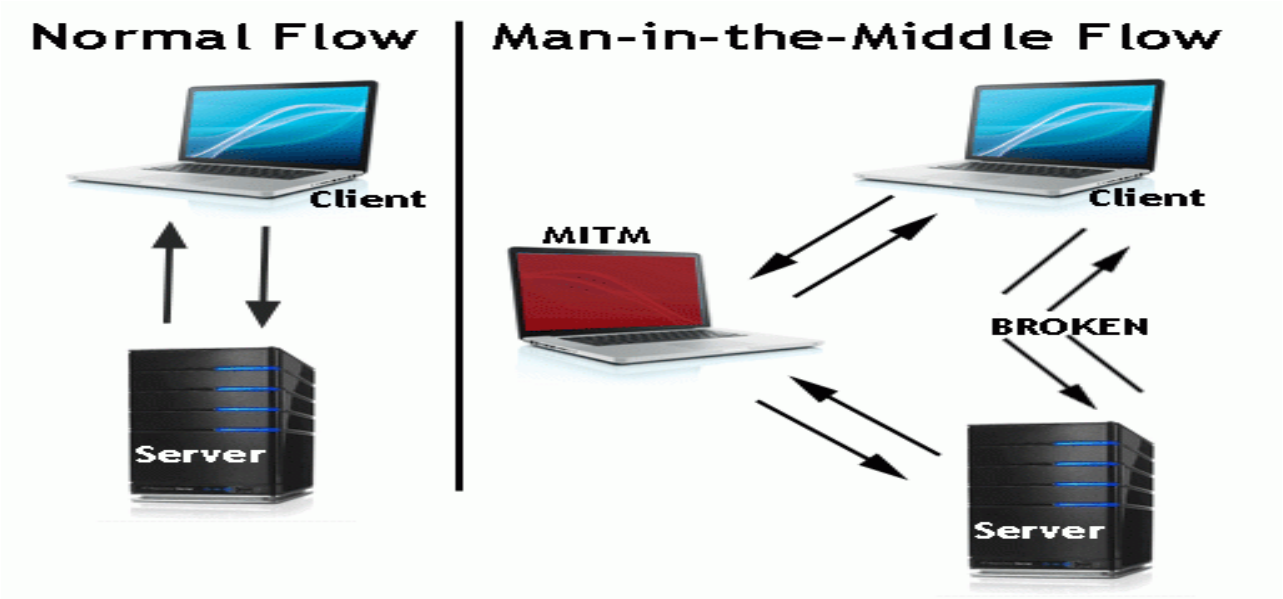
DDoS attack



Source: Scambray, J. et al. *Hacking Exposed 2e.* New York: McGraw-Hill, 2000. Copyright © McGraw-Hill Companies, Inc.

Man-in-the-middle Attack

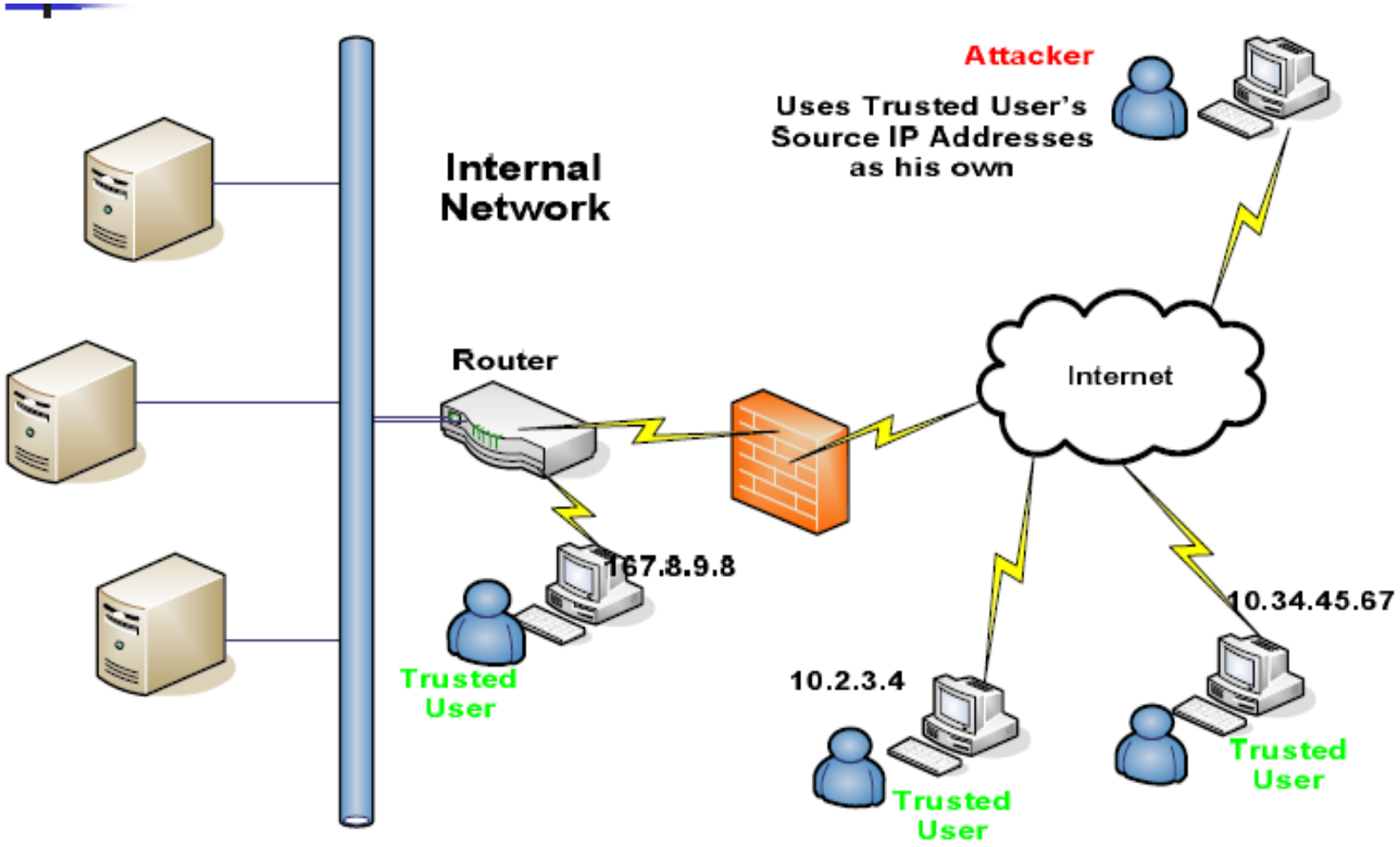
- A man-in-the-middle attack: Here, a malicious actor inserts him/herself into a conversation between two parties, impersonates both parties and gains access to information that the two parties were trying to send to each other. A man-in-the-middle attack allows a malicious actor to intercept, send and receive data meant for someone else, or not meant to be sent at all, without either outside party knowing until it is too late.



IP spoofing

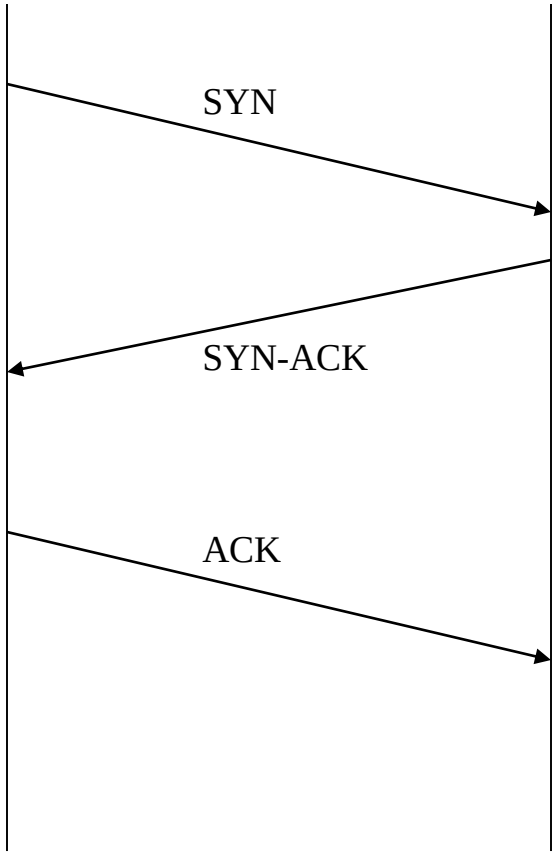
- IP spoofing is a technique used to gain unauthorized access to computers, where by the attacker sends messages to a computer with a forging IP address indicating that the message is coming from a trusted host.
- Attacker puts an internal, or trusted, IP address as its source. The access control device sees the IP address as trusted and lets it through.
- Two general techniques are used during IP spoofing:
    - A hacker uses an IP address that is within the range of trusted IP addresses.
    - A hacker uses an authorized external IP address that is trusted.

IP Spoofing


Attacker
Uses Trusted User's Source IP Addresses as his own
Internal Network
Router
Internet
167.8.9.8
Trusted User
10.2.3.4
Trusted User
10.34.45.67
Trusted User

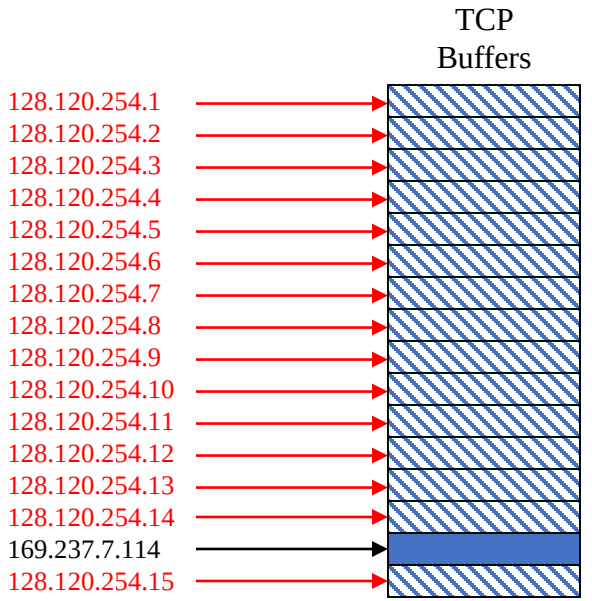
SYN Flooding Attack

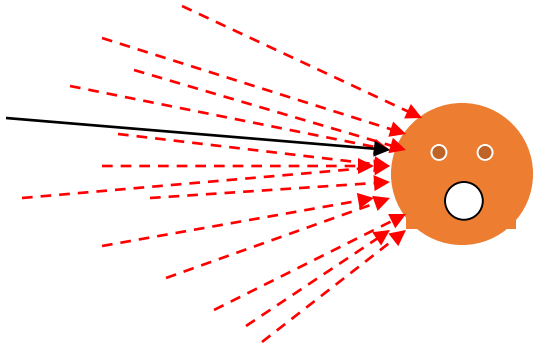
- TCP Handshake Review
  - client
    - sends SYN packet to server
    - waits for SYN-ACK from server
  - server
    - responds with SYN-ACK packet
    - waits for ACK packet from client
  - client
    - sends ACK to server
  - TCP communication between client and server begins with a SYN, a SYN-ACK response, and then an ACK response. When the handshake is complete, traffic is sent between two hosts.

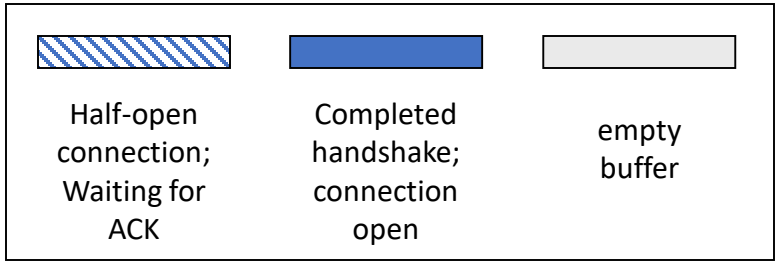


SYN Flood Attack

- But in SYN-Flooding attack, the attacker host will send a flood of SYN packet but will not respond with an ACK packet
- Attacker causes TCP buffer to be exhausted with half-open connections
- After Buffer becomes full, server will not serve requests made by legitimate users.







Port Scanning Attack

- What is Port scanner?
  - **A port scanner is a software application designed to probe a server or host for open ports. This is often used by administrators to verify security policies of their networks and by attackers to identify running services on a host with the view to compromise it.**
- In this attack, the attacker gets list of open and closed ports. Then attacker lunch attacks against service running on open ports.
- This leads attackers to gain entry into organizations and steal their private data.

TCP Session Hijacking

- TCP session hijacking is when an attacker takes over a TCP session between your computer A and another computer B.
- After that, it disconnects another computer B from the communication.
- Your computer A behaves that he is communicating with B and shares your private information to the attacker by accident.

Phishing

- In a phishing attack, an attacker create a fake website that looks similar to popular website site like SBI Bank or PayPal.
- Then the attacker send you an email that appears to be from someone you trust, like your boss or a company you do business with. The email will seem legitimate, and it will have some urgency to it (e.g. fraudulent activity has been detected on your account). In the email, there will be a link to click. If you click the link, it will redirect you to a legitimate-looking website, but it is fake website.
- When user attempts to log in with their account information, the hacker records user name and password and tries that information in actual website.

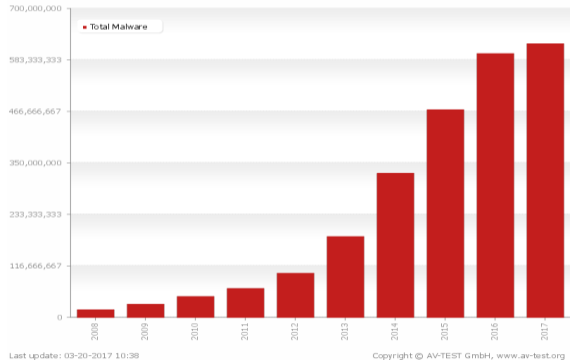
Asymmetric Routing

It is a situation in which packet flowing from source to destination follow one route and packet flowing from destination to source follow another route.

Example: Suppose Host A and B located in different networks communicating through TCP Connection. Segments flowing from Host A to Host B follow one route and segments flowing from Host B to Host A follow another route.

Malware

- **MALWARE is**
  - Malicious Software.
  - Adware, virus,Worm, Ransomware
- **Malware can do**
- allow cybercriminals to get into other people's computers without their permission.
  - -steal personal information
  - -delete files
  - -steal software serial numbers
- According to AV-Test in March 2017, the total number of malwares is increasing exponentially since 2008.
- Due to such increasing ,it is necessary to detect these files before they harm

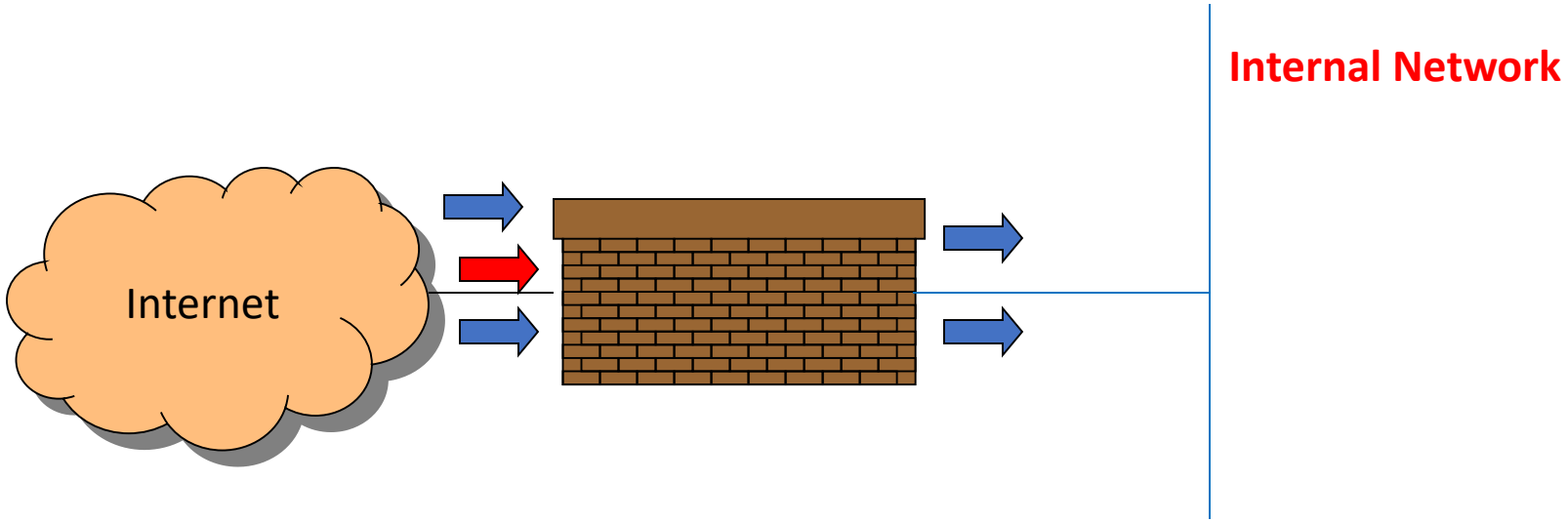


Solutions to Intrusions

- Firewall: Firewall filters network traffic between your network and outside network based on some rules configured by the administrator.
- Intrusion Detection System:
  - Filtered traffic may contain malicious data
  - It is a software that monitors the events occurring in a computer system or network and analyzing them for signs of possible *intrusions (incidents).*
- Intrusion Prevention System
  - It is a software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents.

Firewall

- Interconnects two networks with differing trusts.
- Firewall inspects traffic passing through two networks
- Allows and Block traffic based on some rules
- It works like a security guard.



Firewall

- We specify rules based on following information
  - IP source and destination addresses
  - Transport protocol (TCP, UDP, or ICMP)
  - TCP/UDP source and destination ports
  - ICMP message type
  - Packet options (Fragment Size etc.)
- **Actions Available**
  - Allow the packet to go through
  - Drop the packet

What firewall can do and can not do ?

- Firewall can not protect against insider attack.
- A firewall can take action on traffic that pass through it but it cannot do anything on traffic that does not pass through it.
- A firewall cannot protect you against complete new attacks.
- A firewall can take decision based on source IP address, destination IP address, source port number and destination port number but it can not inspect payload or data inside the packet. This may contain virus.

Intrusion Detection System(IDS)

- Intrusion detection system (IDS) monitors the network traffic and system-level applications to detect malicious activities in the network
- An IDS detects attacks as soon as possible and takes appropriate action.
- An IDS does not usually take preventive measures when an attack is detected.
- It is a reactive rather than a pro-active agent.
- It plays a role of informant rather than a police officer.
- It can deal with insider and outsider attacks.
- The most popular way to detect intrusions has been using the audit data generated by the operating system.
  - And audit trail is a record of activities on a system that are logged to a file in chronologically sorted order

IDS Requirements

- run continually with minimal human supervision
- be fault tolerant
- resist subversion
- minimal overhead on system
- scalable, to serve a large numbe of users
- Provide graceful degradation of service
- configured according to system security policies
- allow dynamic reconfiguration

IDS Architecture

- Basically, a sophisticated audit system
  - *Agent* like logger; it gathers data for analysis. It is also known as sensor.
  - *Director* like analyzer; it analyzes data obtained from the agents according to its internal rules
  - *Notifier* obtains results from director, and takes some action
    - May simply notify security officer
    - May reconfigure agents, director to alter collection, analysis methods
    - May activate response mechanism

Agents

- Obtains information and sends to director
- May put information into another form
  - Preprocessing of records to extract relevant parts
- May delete unneeded information
- Director may request agent send other information

Example

- IDS uses failed login attempts in its analysis
- Agent scans login log every 5 minutes, sends director for each new login attempt:
  - Time of failed login
  - Account name and entered password
- Director requests all records of login (failed or not) for particular user
  - Suspecting a brute-force cracking attempt

Director

- Reduces information from agents
  - Eliminates unnecessary, redundant records
- Analyzes remaining information to determine if attack under way
  - Analysis engine can use a number of techniques, discussed before, to do this
- Usually run on separate system
  - Does not impact performance of monitored systems
  - Rules, profiles not available to ordinary users

Notifier

- Accepts information from director
- Takes appropriate action
  - Notify system security officer
  - Respond to attack
- Often GUIs
  - Well-designed ones use visualization to convey information

Types of IDS

- Detection Model
  - signature detection vs. anomaly detection
- Monitoring Environment
  - Host based, vs. network based
- Operation
  - Off-line vs. real-time
- Architecture
  - Centralized vs. distributed

Types of IDS: host-based

Deployed in a **single target computer**.

Monitors the **Operating System logs** and **Analyse** the **audit information** to detect trails of **intrusion**.

Audit information includes events like identification and authentication mechanism (like logins, accessed registries), file access information (like time, permissions), program executions, admin activities etc.

Deployment options:

- Key servers that contain mission-critical and sensitive information.
- Web servers.
- FTP and DNS servers.
- E-commerce database servers, etc.

Host-based

Types of IDS: host-based

PROS:

- No additional hardware
- Less volume of traffic so less overhead
- Better for detecting attacks from the inside
- Near real-time detection and response
- System specific activities
- Encrypted traffic is also available for analysis
- Lower entry cost

CONS:

- **Detection is based on what any single host can record**
- **Narrow in scope (watches only specific host activities)**
- **Reduce performance of host system.**
- **Vulnerable to situation like when host operating system is compromised**
- **More expensive to implement.**
- **Deployment is challenging**
- **OS dependent**
- **Does not see packet headers.**

Types of IDS: Network-based

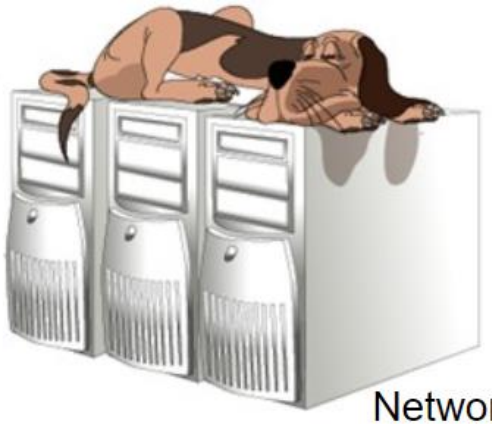
- Instead of analyzing information that originates and resides on a host, Network-based IDS analyses traffic passing through the network to detect intrusions.
- uses packet sniffing techniques to pull **packet header information** from TCP/IP packets or other protocols that are traveling along the network.
- **Packet header information** includes events like packet size (like 20KB), protocol (TCP, UDP etc.), port (like 1024), IP addresses, purpose (response or request).

Deployment options:

Outside firewall

- Just inside firewall

  -Combination of both will detect attacks getting through firewall and may help to refine firewall rule set.
- Behind remote access server
- Routers



Network-based

Types of IDS: Network-based

PROS:

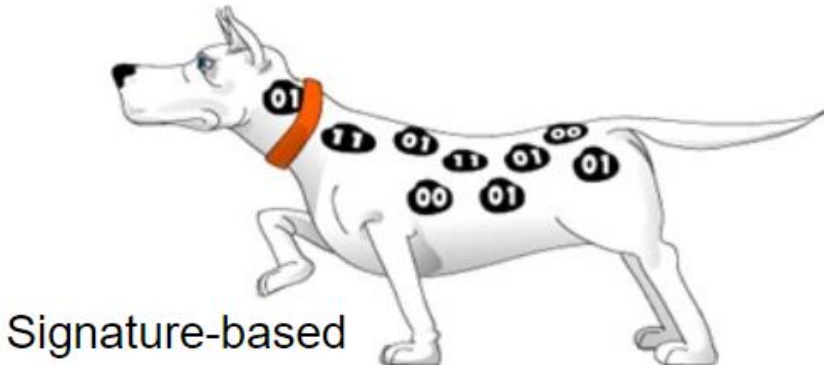
- Protect the whole network and detect network based attacks (like DOS)
- Broad in scope (watches all network activities)
- Easier setup: Easy to deploy
- Better for detecting attacks from the outside
- Less expensive to implement
- Detection is based on what can be recorded on the entire network
- Examines packet headers
- Near real-time response
- OS-independent
- Detects unsuccessful attack attempts

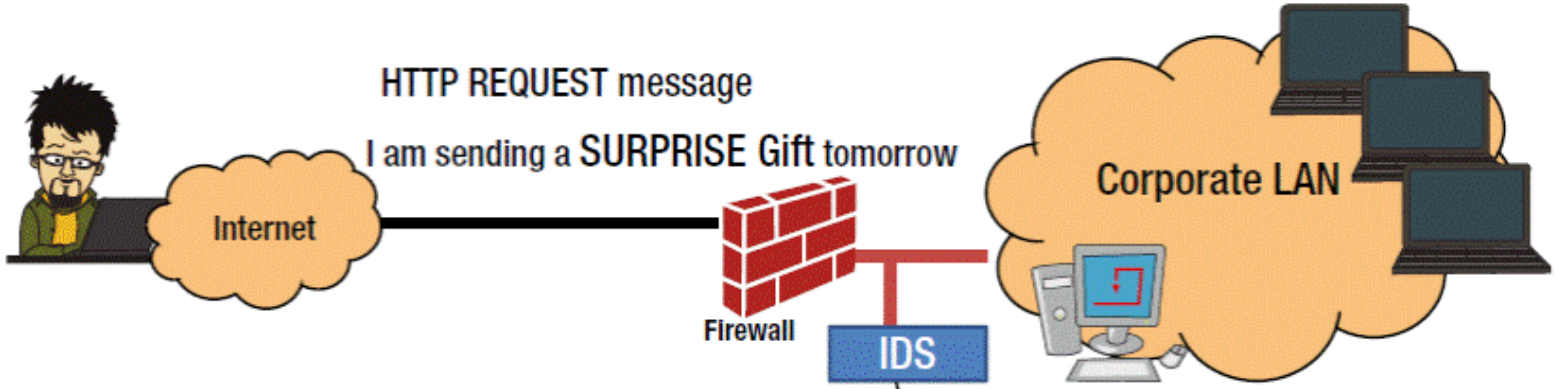
CONS:

- Require all traffic information.
- Generates an enormous amount of data to be analyzed
- Cannot monitor traffic at higher network traffic rates
- Cannot deal with encrypted network traffic
- Can not detect system-specific attacks (like trojan)

Types of IDS: signature-based

Compares **signatures** against **observed events** to identify possible incidents. In case of any matching, an alert is issued.





Signature to Fire:

Header : IPv4	Protocol: HTTP	Destination Port: 80	Signature String: SURPRISE

Source: networkingsphere.com

Types of IDS: signature-based

PROS:

- High accuracy
- Low FPR (false Positive Rate),
- Widely available
- Fairly fast
- Easy to implement
- Easy to update

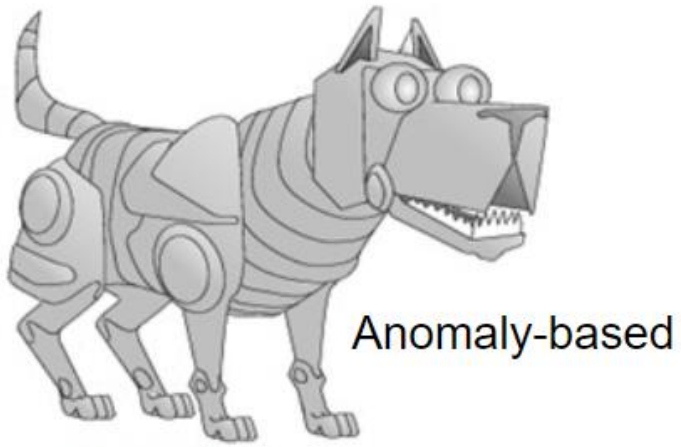
CONS:

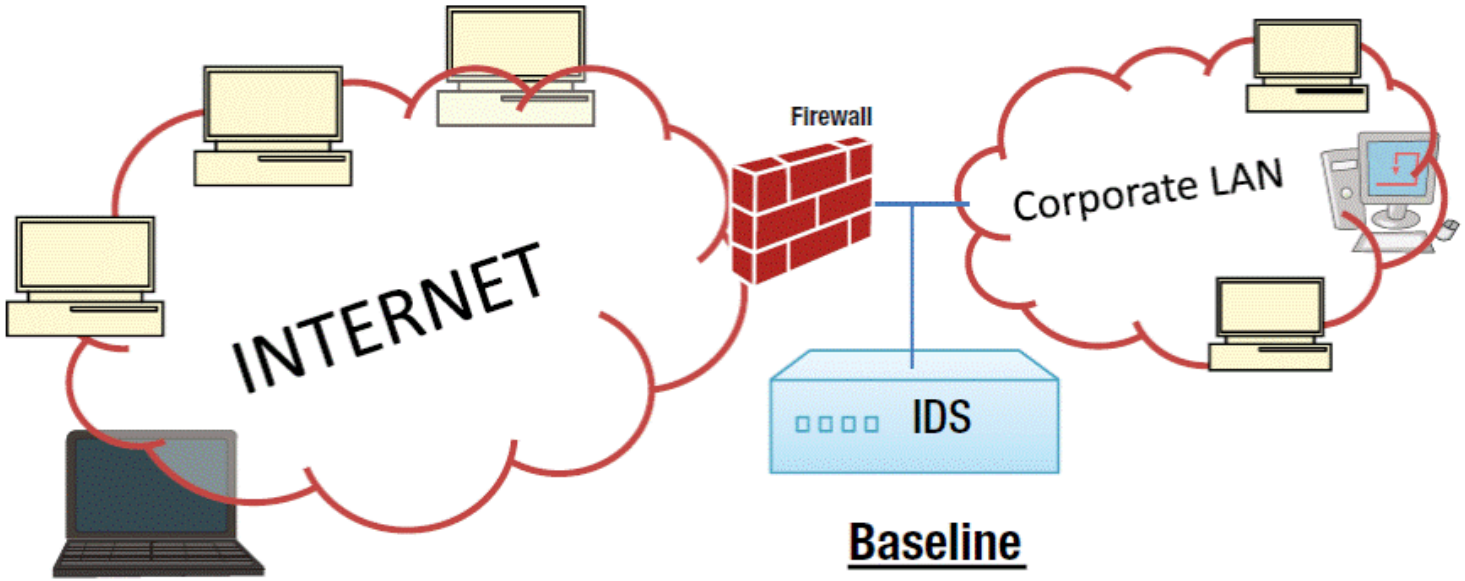
- Zero-day or unknown attack
- Cannot detect attacks for which it has no signature
- frequent update of signatures

Source: networkingsphere.com

Types of IDS: anomaly-based

Compares definitions of what is considered **normal activity** with **observed events** to identify significant deviations.





Baseline

TCP Pkts = 100/sec
UDP Pkts = 112/sec
ICMP Pkts = 43/sec

Alert !!! UDP Pkts = 425/sec !!!

Source: networkingsphere.com

Types of IDS: anomaly-based

Threshold detection

- Checks excessive event occurrences over time
- Compute statistics of certain system/network activities
- Report an alert if statistics outside range
- Example:

  For each user, store daily count of certain activities

  For example, fraction of hours spent reading email

  Maintain list of counts for several days

  Report anomaly if count is outside weighted norm

Profile based

- Characterize past behavior of users and groups
- Then, detect significant deviations
- Build it manually (this is hard)
- Analysis method (mean and standard deviation, multivariate, etc.) require machine learning and data mining techniques

Types of IDS: anomaly-based

PROS:

- Zero-day attack detection

CONS:

- High FPR (False Positive Rate)
- Greater complexity
- slower, more resource intensive

Source: networkingsphere.com

Types of IDS: specification-based

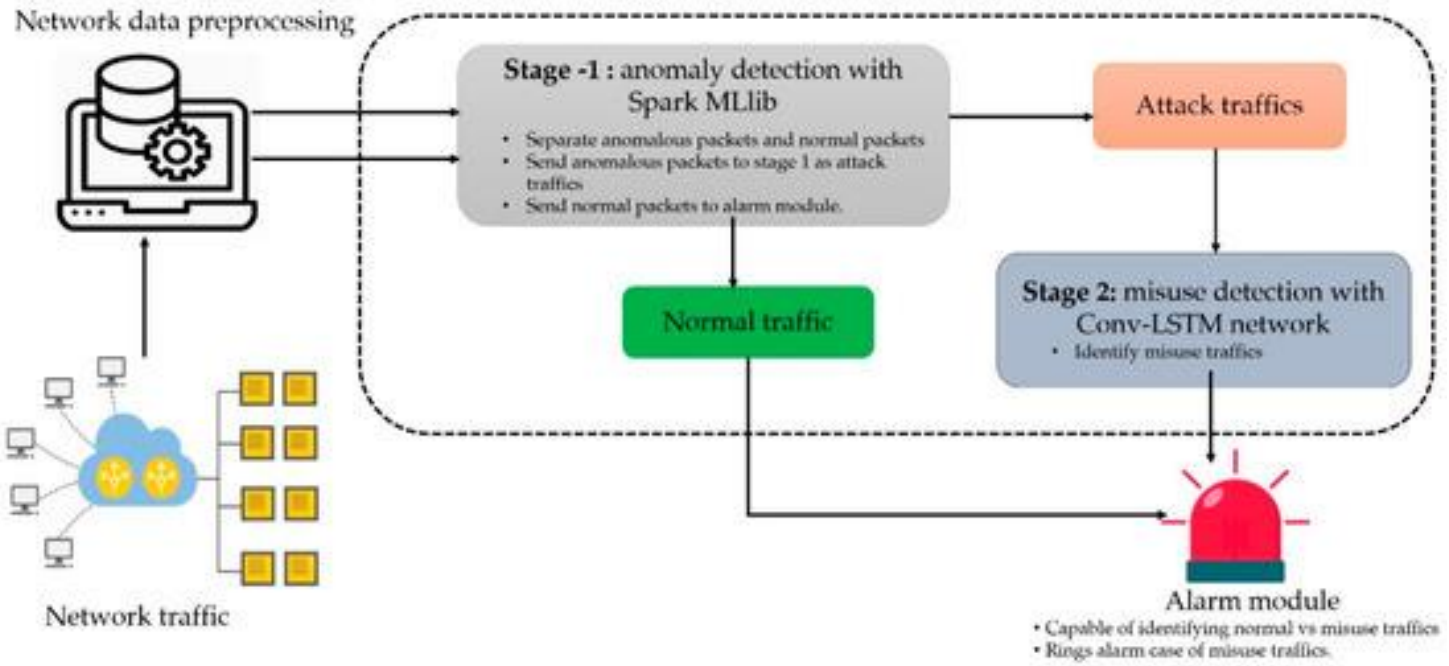
Combines the strength of both signature and anomaly based to form a **hybrid model**.

PROS:

- Able to achieve low FPR and high accuracy
- Detection of zero-day attacks

CONS:

- Implementation and deployment is very challenging
- Only theoretical results



IDS Types: Centralized vs. Distributed

Centralized

- Data collected from single or multiple hosts
- All data shipped to a central location for analysis

Distributed

- Data collected at each host
- Distributed analysis of the data

IDS Performance metrics

True positive (TP): Intrusions that are successfully detected by the IDS.
False positive (FP): Normal/non-intrusive behavior that is wrongly classified as intrusive by the IDS.
True Negative (TN): Normal/non-intrusive behavior that is successfully labeled as normal/non-intrusive by the IDS.
False Negative (FN): Intrusions that are missed by the IDS, and classified as normal/non-intrusive.

Accuracy: It is defined as the ratio of correctly classified instances and the total number of instances.

$$ACCURACY = \frac{TP + TN}{TP + TN + FP + FN}$$

Detection Rate (DR): It is computed as the ratio between the number of correctly detected attacks and the total number of attacks

$$\mathrm{D}R = \frac{TP}{TP + FN}$$

False positive rate (FPR): It is defined as the ratio between the number of normal instances detected as attack and the total number of normal instances.

$$FPR = \frac{FP}{TN + FP}$$

Efficient IDS: High Accuracy and low FPR.

Dataset Description

Table 2: UNSW-NB15 dataset classes distribution

Attack Types	Training Set	Testing Set
Analysis	2000	677
Backdoor	1746	583
DoS	12264	4089
Exploits	33,393	11,132
Fuzzers	18,184	6062
Reconnaissance	10491	3496
Shellcode	1133	378
Worms	130	44
Normal	56000	37000
Total	175341	82332

In 2015, Moustafa and Slay created a dataset called **UNSW-NB15 dataset** for research purposes. This dataset contains normal and malicious activities of modern network traffic.

Table 1: NSL-KDD dataset classes distribution

Dataset	Total	Normal	Dos	Probe	U2R	R2L
Training	125973	67343	45927	11656	52	995
Testing	22544	9711	7458	2421	67	2887

The NSL-KDD data set is the refined version of the **KDD cup99 data set.**

Table 3: AWID-CLS-R dataset classes distribution

Dataset	Total	Normal	flooding	impersonation	injection
Training	1795575	1633190	48484	48522	65379
Testing	530643	530785	8097	20079	16682

In 2015, Kolias et al. created a dataset named AWID (Aegean WiFi Intrusion Dataset) which contain real traces of both normal and intrusion activities of a **802.11 Wi-Fi network.**

Dataset Description: NSL-KDD

Attack Type	Attack Description	Subclass
DOS	system is flooded with connection requests that it doesn't have sufficient buffer to take actual authentic requests. e.g. Syn flood	Neptune, Smurf, Pod, Teardrop, Landback, Teardrop, Mailbomb, Processtable, Udpstorm, Apache2, Worm
PROBE	attacker claims to avoid the security controls of a network and gather possible information under this context. e.g. Port scanning.	Portsweep, IPSweep, Nmap, Satan, Mscan, Saint
U2R	attacker tries to gain access to the root privileges of the system by various means.	BufferOverflow, LoadModule,Perl, Rootkit, Sqlattack, Xterm, Ps
R2L	intruder has the right to send packets into network. This is used to manipulate and gain access. e.g. Guessing password.	Guesspassword, Ftpwrite, Imap, Phf, Multihop, Warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named

Dataset Description: UNSW-NB15

Attack Type	Attack Description
FUZZERS	Attempting to cause a program or network suspended by feeding it the randomly generated data
Analysis	It contains different attacks of port scan, spam and html files penetrations.
Backdoor	A technique in which a system security mechanism is bypassed stealthily to access a computer or its data.
DoS	A malicious attempt to make a server or a network resource unavailable to users, usually by temporarily interrupting or suspending the services of a host connected to the Internet
Exploit	The attacker knows of a security problem within an operating system or a piece of software and leverages that knowledge by exploiting the vulnerability
Generic	A technique works against all block ciphers (with a given block and key size), without consideration about the structure of the block-cipher.
Reconnaissance	Contains all Strikes that can simulate attacks that gather information.
Shellcode	A small piece of code used as the payload in the exploitation of software vulnerability.
Worms	Attacker replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it.

Dataset Description: AWID

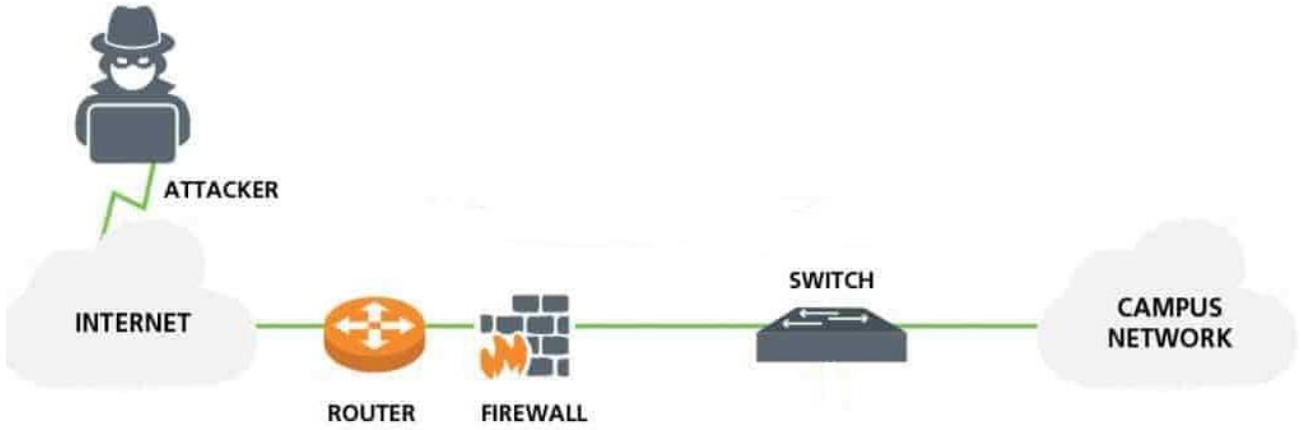
Attack Type	Attack Description	Subclass
Flooding	designed to cause an interruption or suspension of services of a specific host/server by flooding it with large quantities of useless traffic or external communication requests. When the Denial of Service (DoS) attack succeeds the server is not able to answer even to legitimate requests any more - this can be observed in numbers of ways: slow response of the server, slow network performance, unavailability of software or web page, inability to access data, website or other resources.	deauthentication, authentication request, amok, probe request, probe response
Impersonation	Impersonation takes the form of device cloning, address spoofing, unauthorized access, rogue base station (or rogue access point) and replay. For example, in an evil twin setup, the client(s) unknowingly connect to them under the pretext that they are connected to a genuine access point. Once a client is connected, an attacker eavesdrops on its communication to hijack clients communication, re-direct clients to malicious websites, steal credentials of the clients connecting to it	evil-twin, caffe-latte, hirte
Injection	Attacker uses existing vulnerabilities in the applications to inject a code/string for execution that exceeds the allowed and expected input to the system requested, e.g. inject a client-side script onto the webpage or an SQL database	arp, fragmentation, chop chop

Till Now: Lot of Cyber Attacks !

How to protect our devices ?

Firewall: first line of defense

Rashtrapati Bhavan.(Reuters File Photo)



Source: digitalplanet

Intrusion Detection System: Second line of defense

Rashtrapati Bhavan(Reuters File Photo)

Source: digitalplanet

Motivation for Machine Learning based IDS

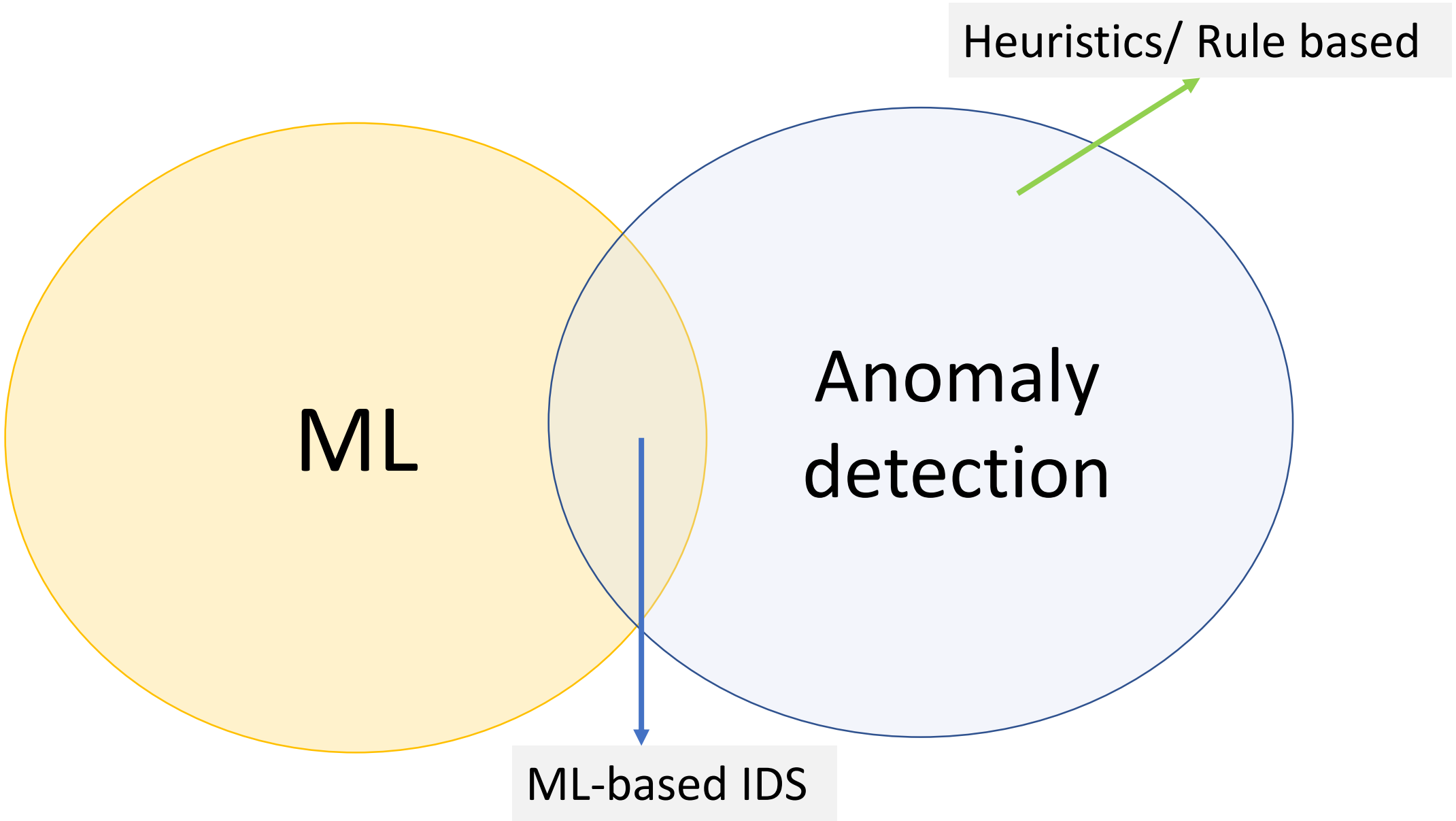
Current network environments are **complex and dynamic** due to the emergence of **novel attacks** and continuous changes in attack pattern.

CONS of rule based methods:

1. **regular updates**
2. **static**
3. **frequent human intervention** for creating new rules for novel attacks

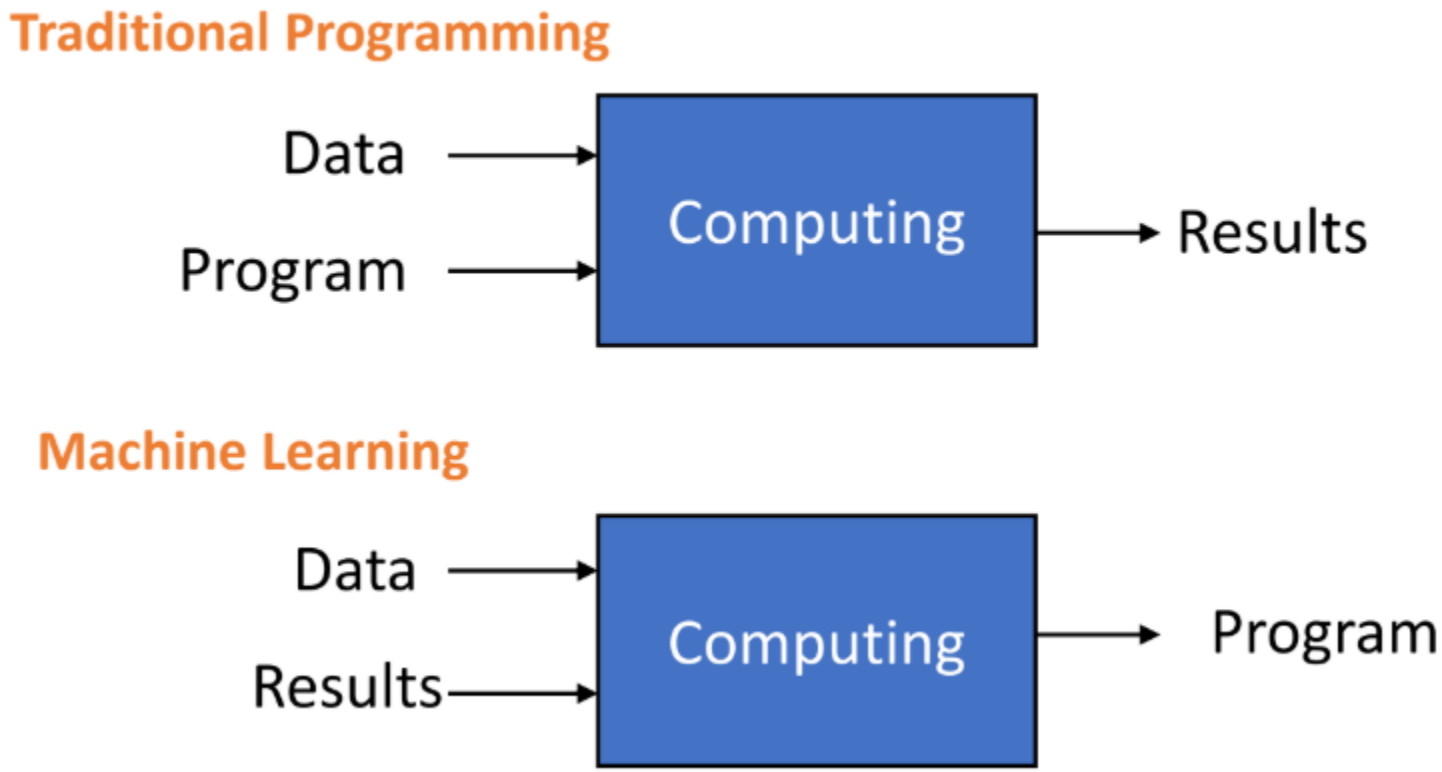
Therefore, **need** for a **IDS** that is

1. **adaptive**
2. **dynamic**
3. **require least human intervention**

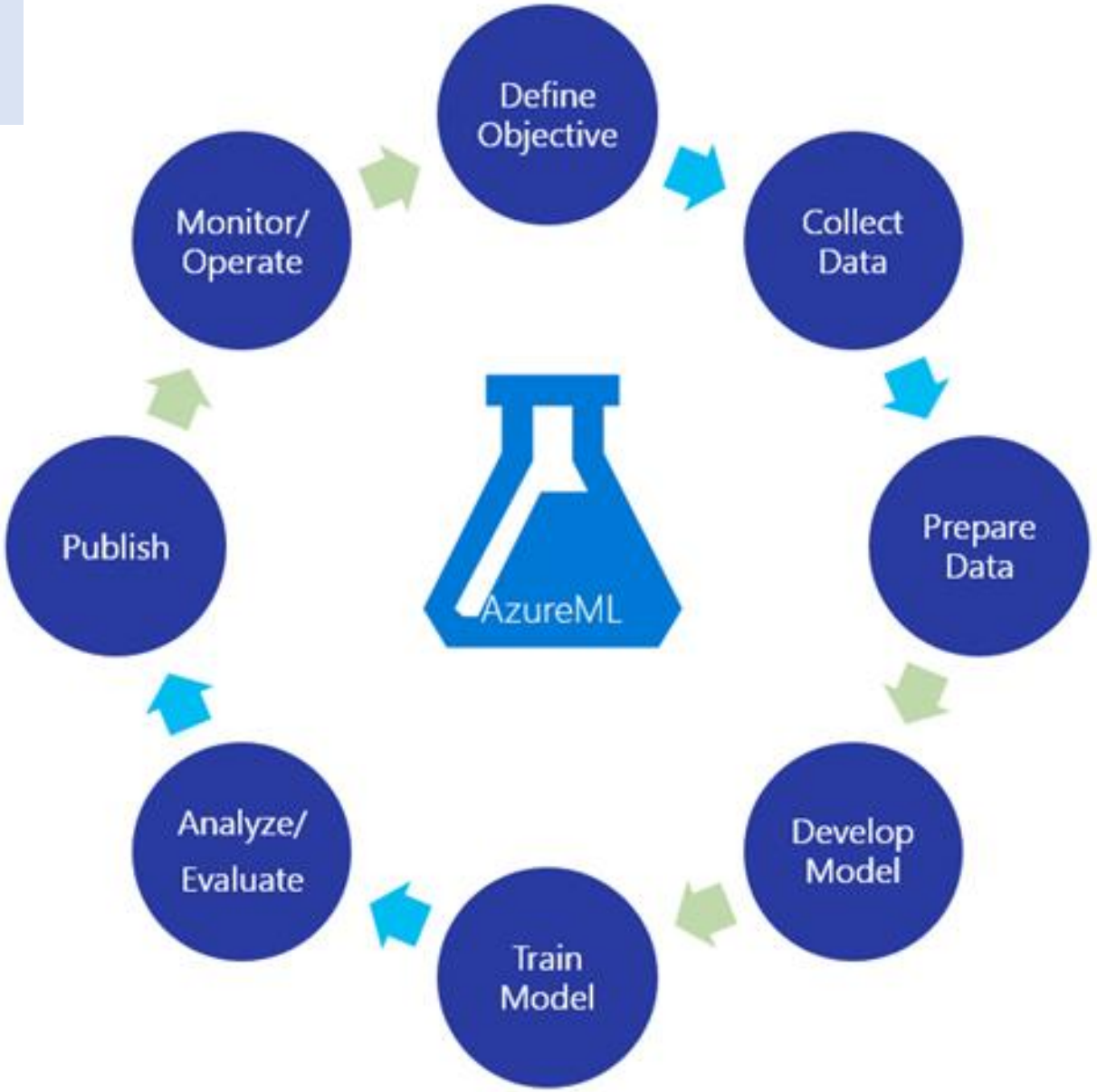

Heuristics/ Rule based
ML
Anomaly detection
ML-based IDS

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed.

ML vs Traditional programming



ML workflow


Define Objective
Collect Data
Prepare Data
Develop Model
Train Model
Analyze/ Evaluate
Publish
Monitor/ Operate
AzureML

ML evaluation metric: Confusion metric

Binary labelled data

False Positive Rate (FPR)

FPR = 1 - Specificity

$$= \frac{FP}{TN + FP}$$

		Actual	
		Positive	Negative
Predicted	Positive	True Positive TP	False Positive FP
	Negative	False Negative FN	True Negative TN

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	TP (a)	FP (b)	*Positive Predictive Value*	a/(a+b)
	Negative	FN (c)	TN (d)	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

TPR True Positive Rate (TPR)

ML evaluation metric: TPR, FPR, ROC, AUC

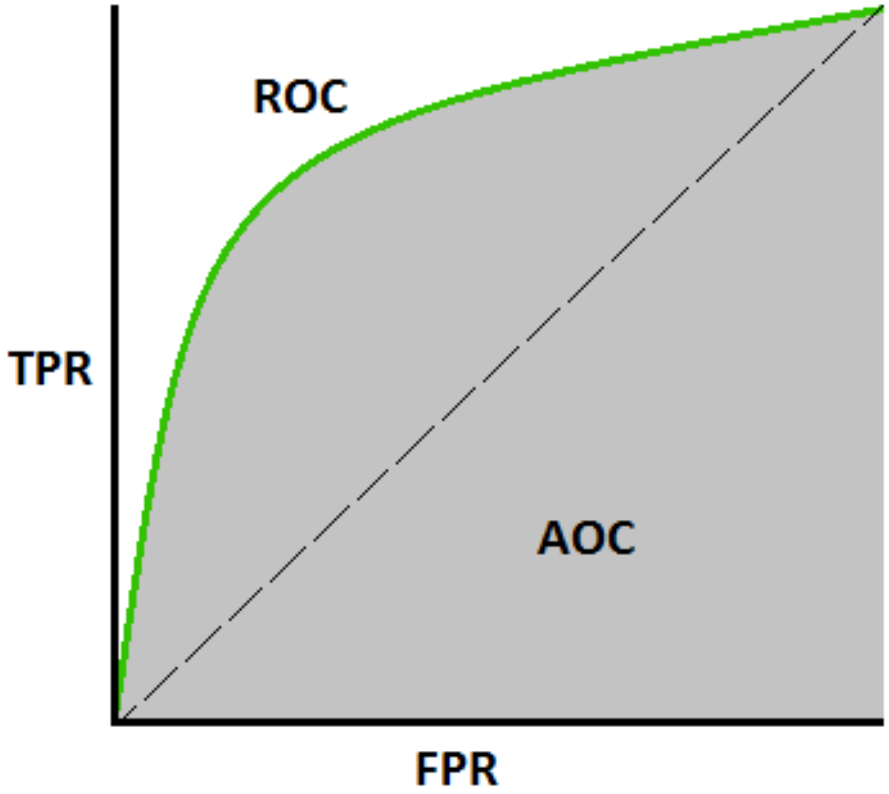
Binary labelled data

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{FPR} = 1 - \text{Specificity}$$

$$= \frac{FP}{TN + FP}$$



https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Anomaly Detection:

find novel attacks, identify **never seen before attacks** (zero-day attacks)

Traditional Machine Learning:

Learn **patterns**, identify **similar things.**

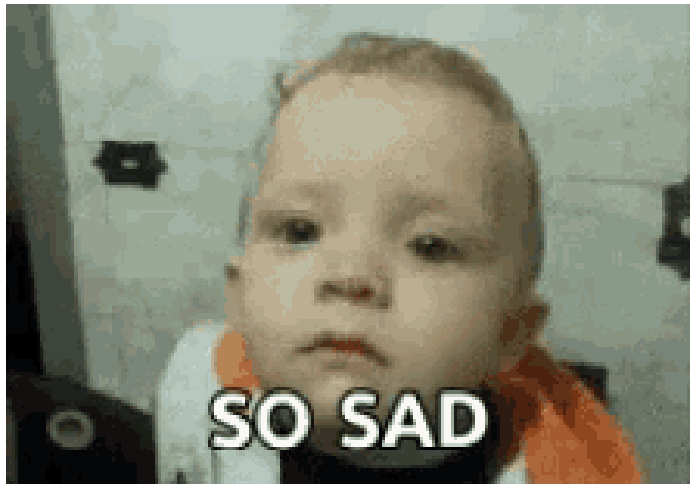
Adversarial impact

Advanced actors can create perturbation to bypass the system

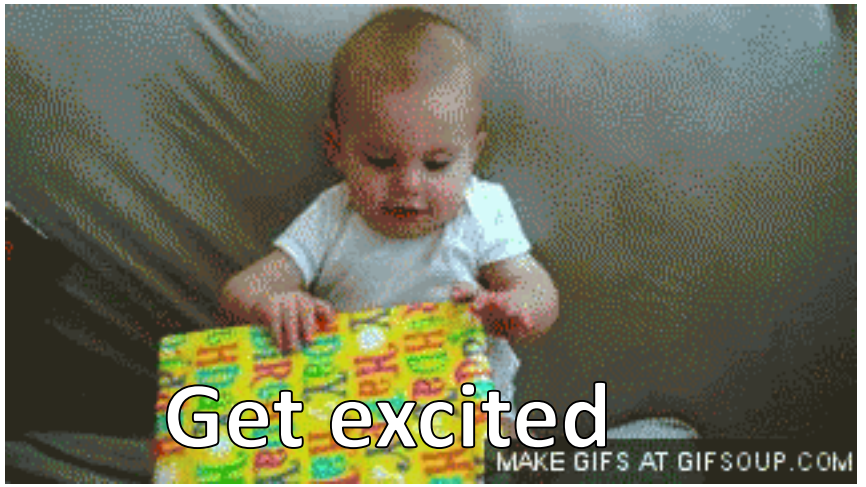
1. Many **false positives**

2. Lack of **proper training data**
   I. What data to train model on?
   II. Clean input data

3. Used without **deep understanding of model**



Advanced ML-IDS



Improper understanding leads to **gap** between **requirements** and **applications** of the model.

Misaligned bridge

Reducing **gap** between **requirements** and **applications** of the model.

Misaligned bridge

The evaluation problem

1. Devising a **sound evaluation scheme.**
2. Getting the **right features** (most difficult task)
3. Getting the **right model**

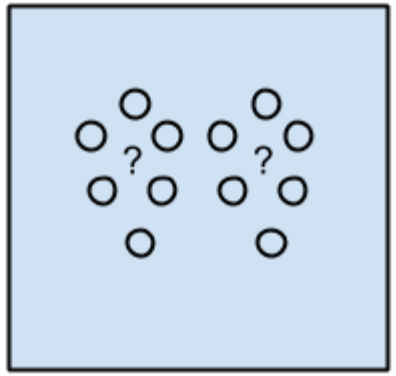
Let’s begin!

Taxonomy of ML

ML

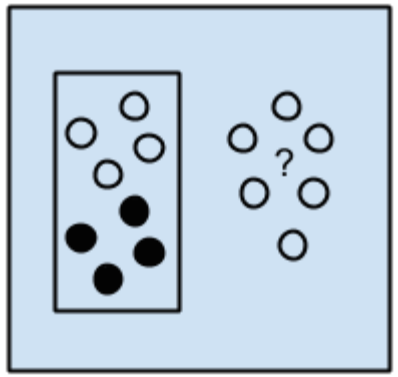
Unsupervised Learning

Input data is **not labelled**.



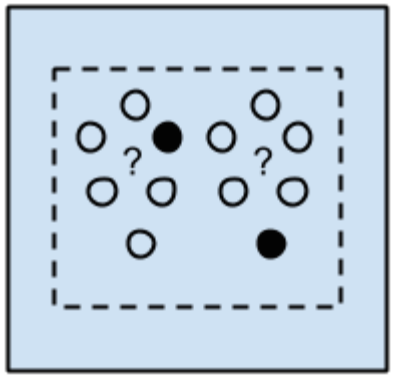
Supervised Learning

Input data is **labelled.**



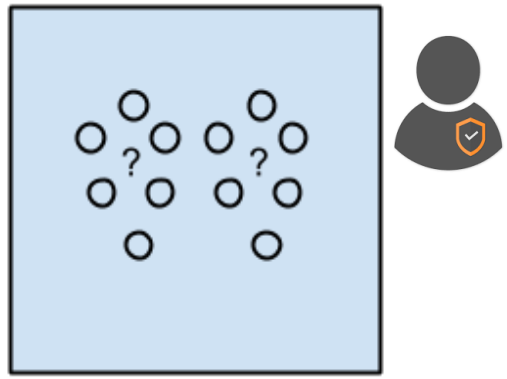
Semi-Supervised Learning

Some Input data is **labelled.**



Reinforcement Learning

Input data is **labelled dynamically**



https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

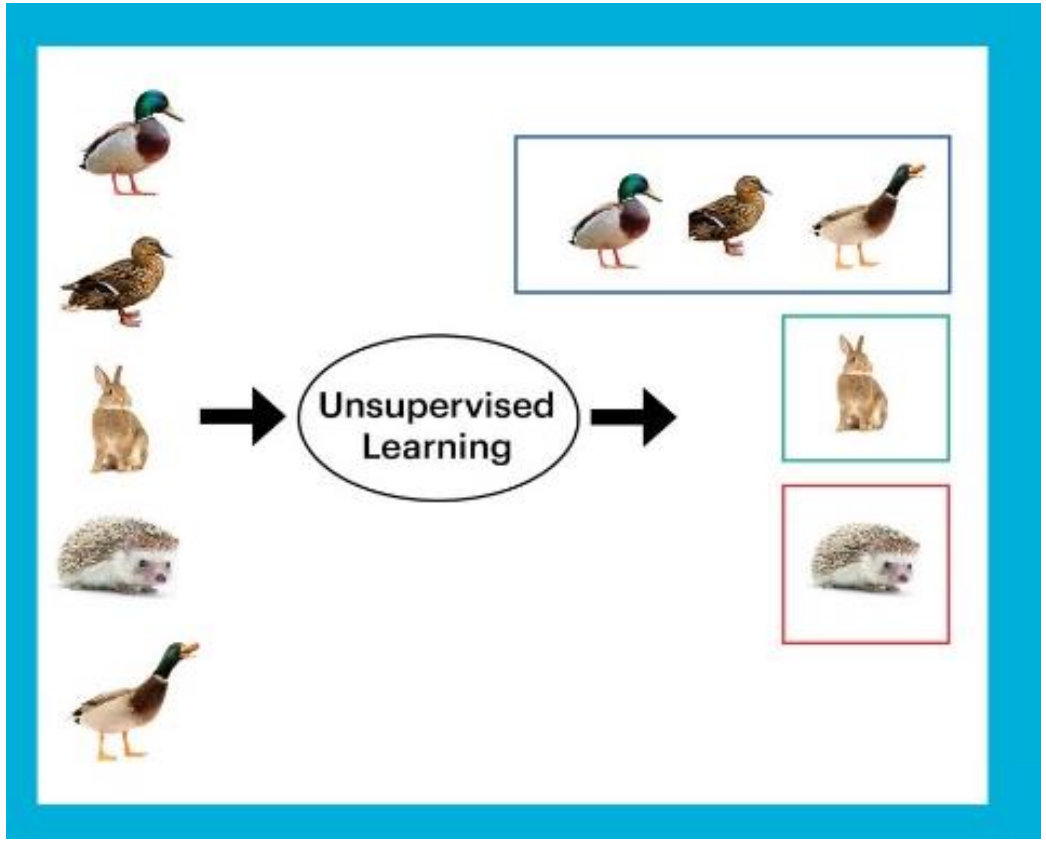
Unsupervised learning

Supervised learning

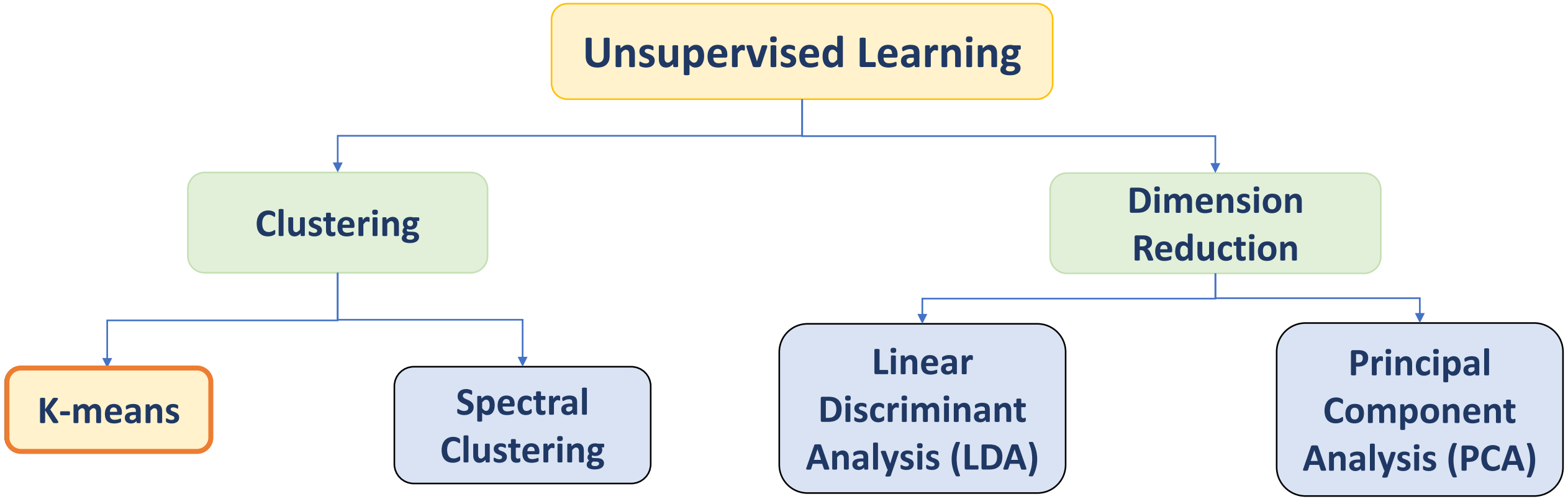
Reinforcement learning

- A model is prepared by **deducing structures present** in the input data.

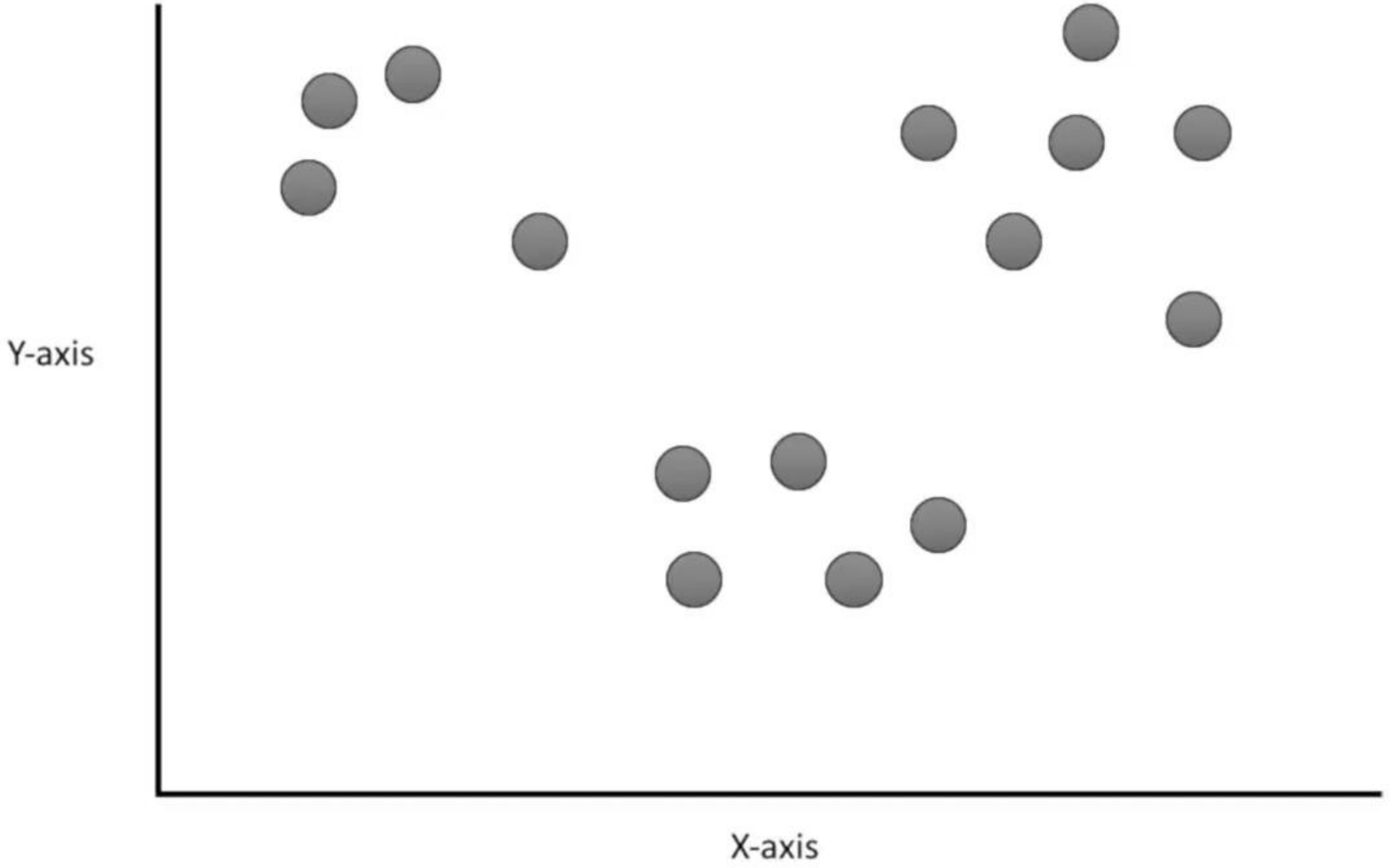
- extract **general rules** through a mathematical process to systematically reduce redundancy, or it may be to organize data by similarity.



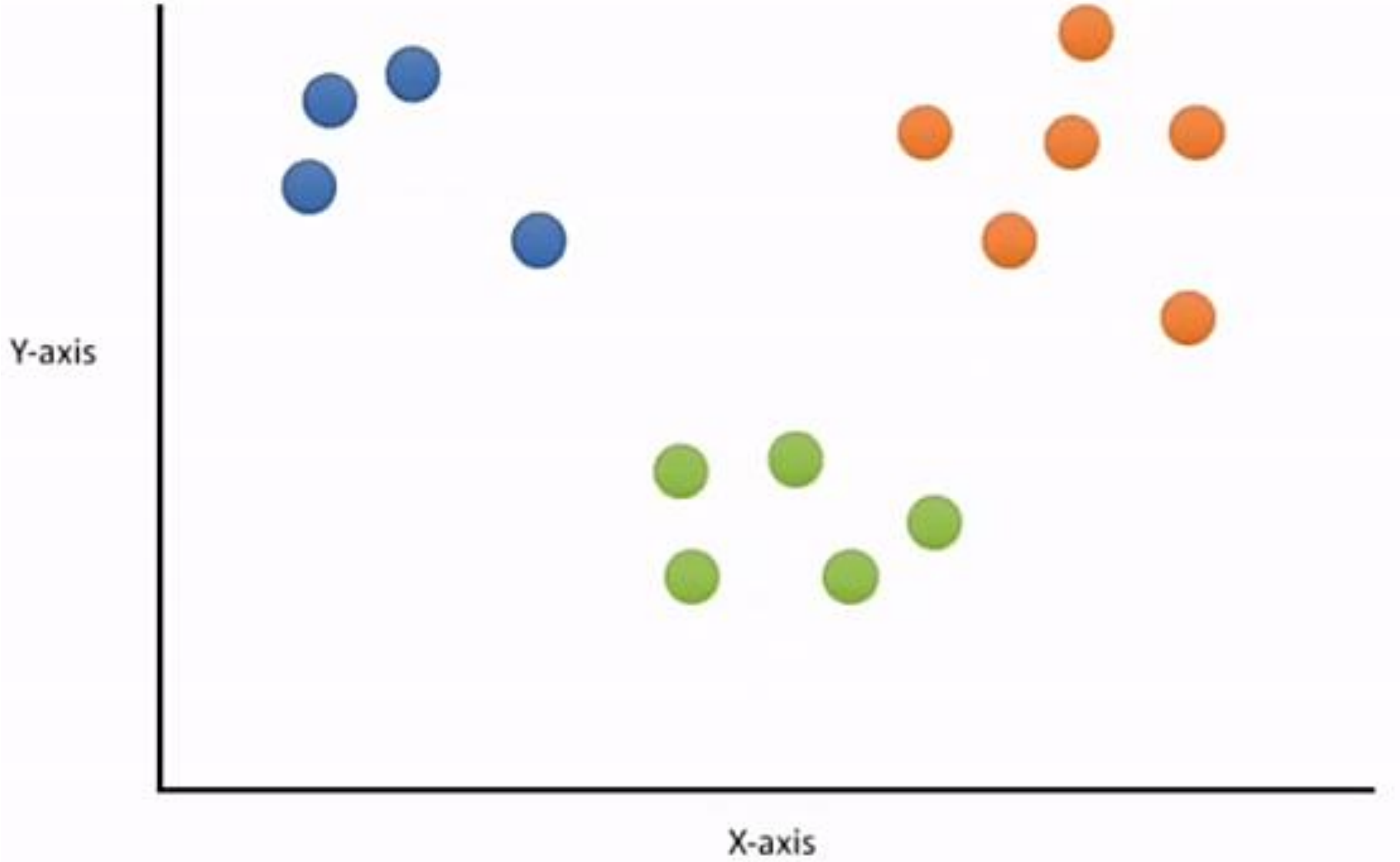
Courtesy: Western Digital https://www.datanami.com/2018/09/05/how-to-build-a-better-machine-learning-pipeline/


Unsupervised Learning
Clustering
Dimension Reduction
K-means
Spectral Clustering
Linear Discriminant Analysis (LDA)
Principal Component Analysis (PCA)

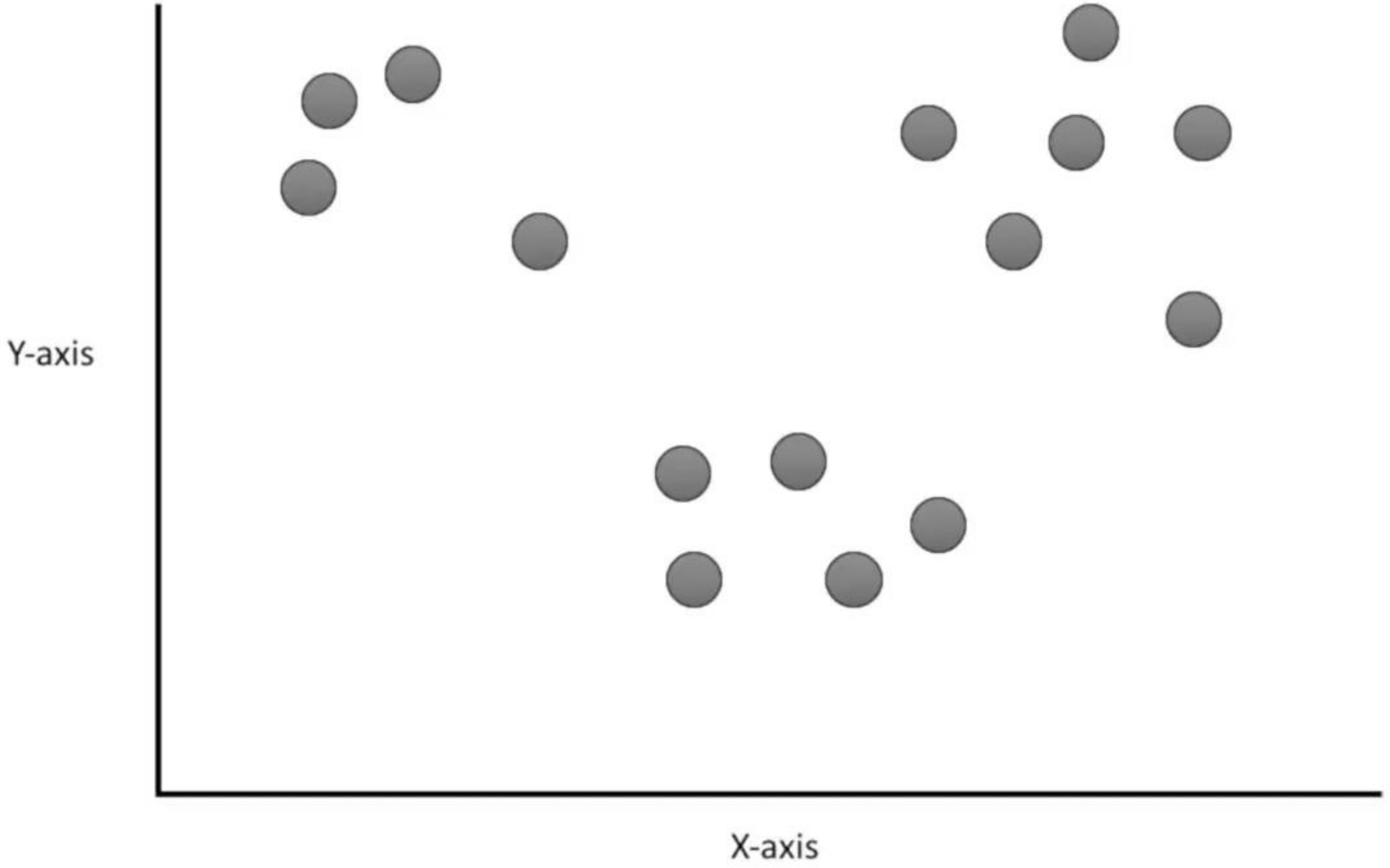
K-means Algorithm: finding cluster centers



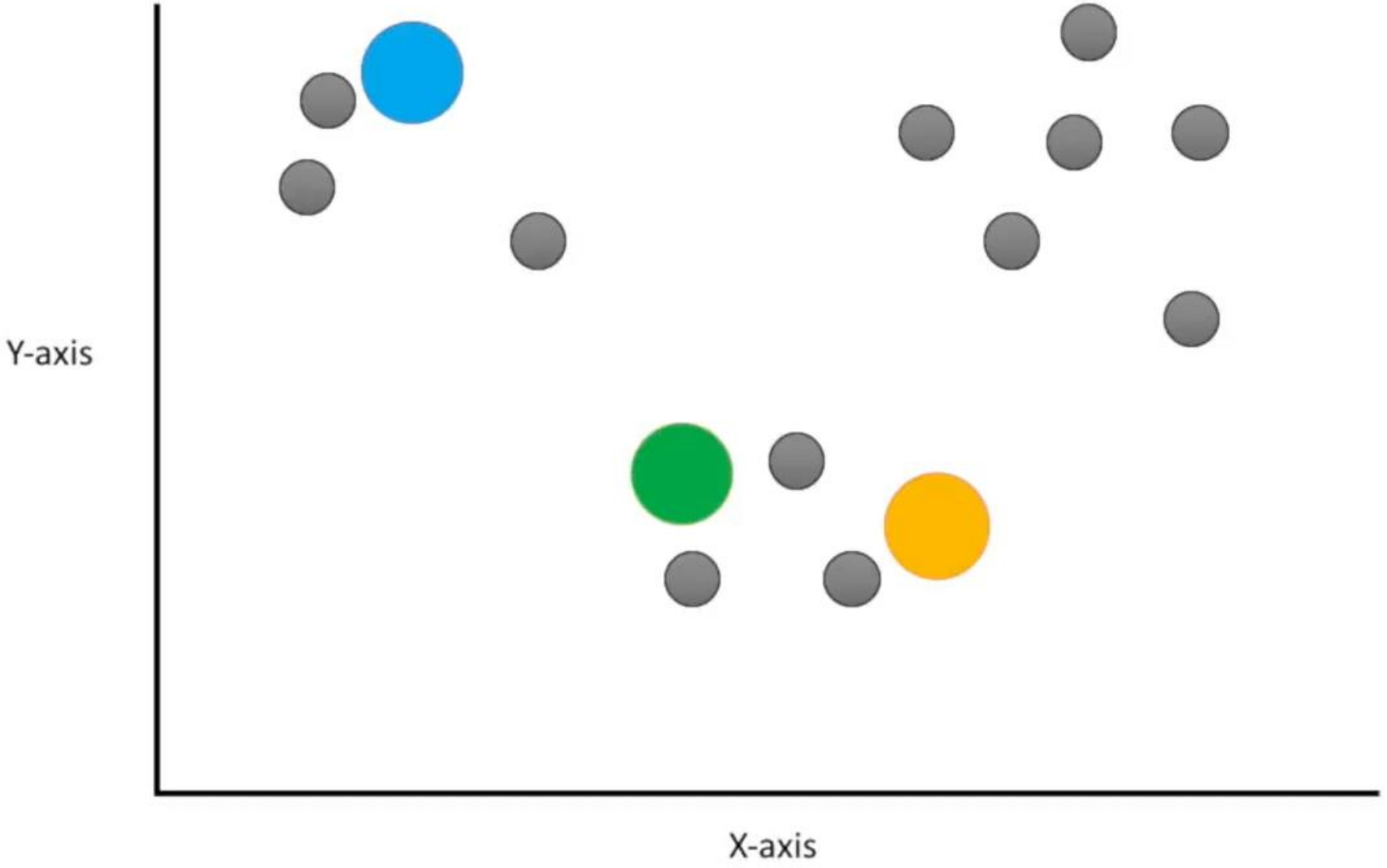
K-means Algorithm: finding cluster centers



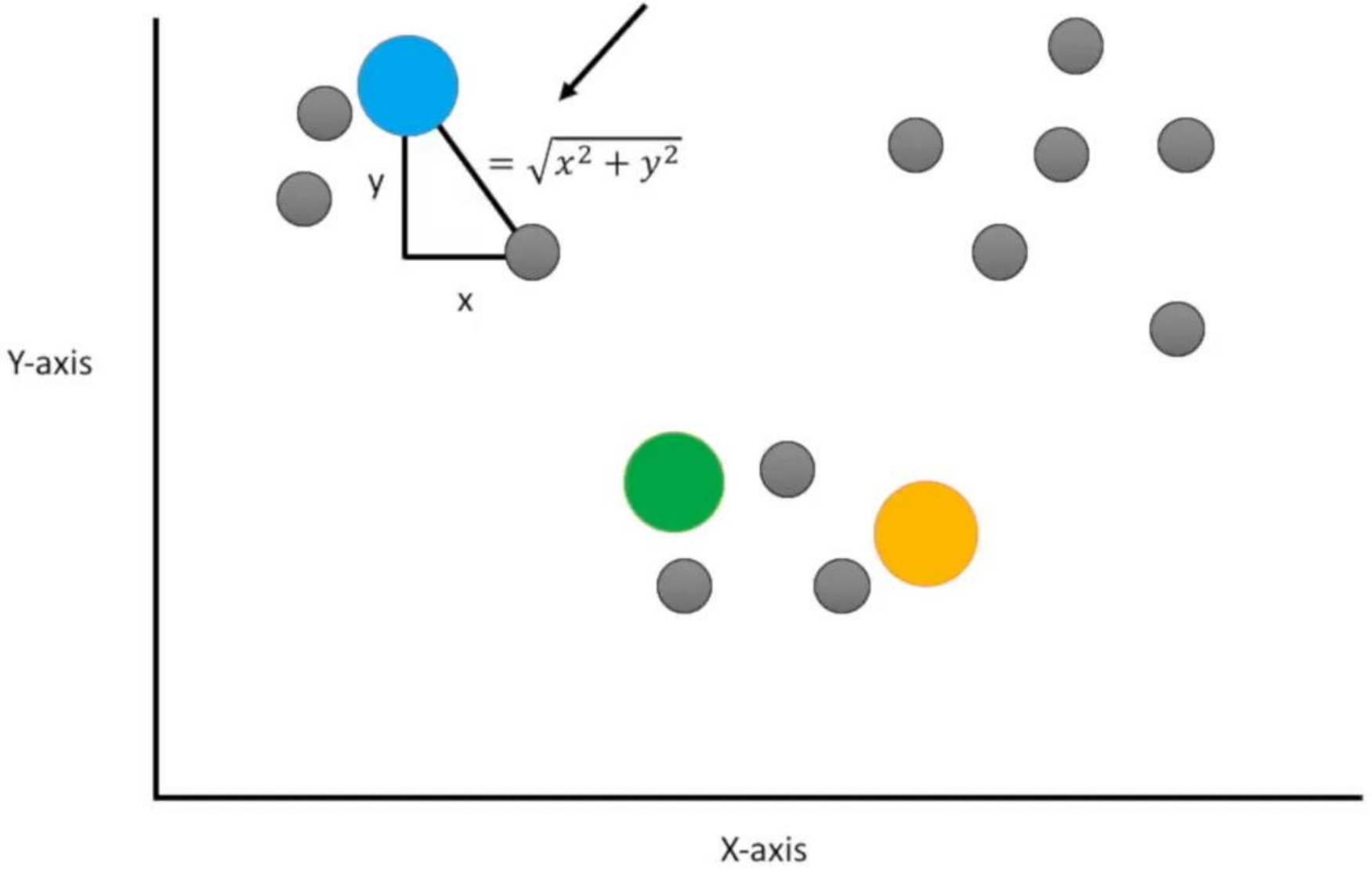
K-means Algorithm: finding cluster centers



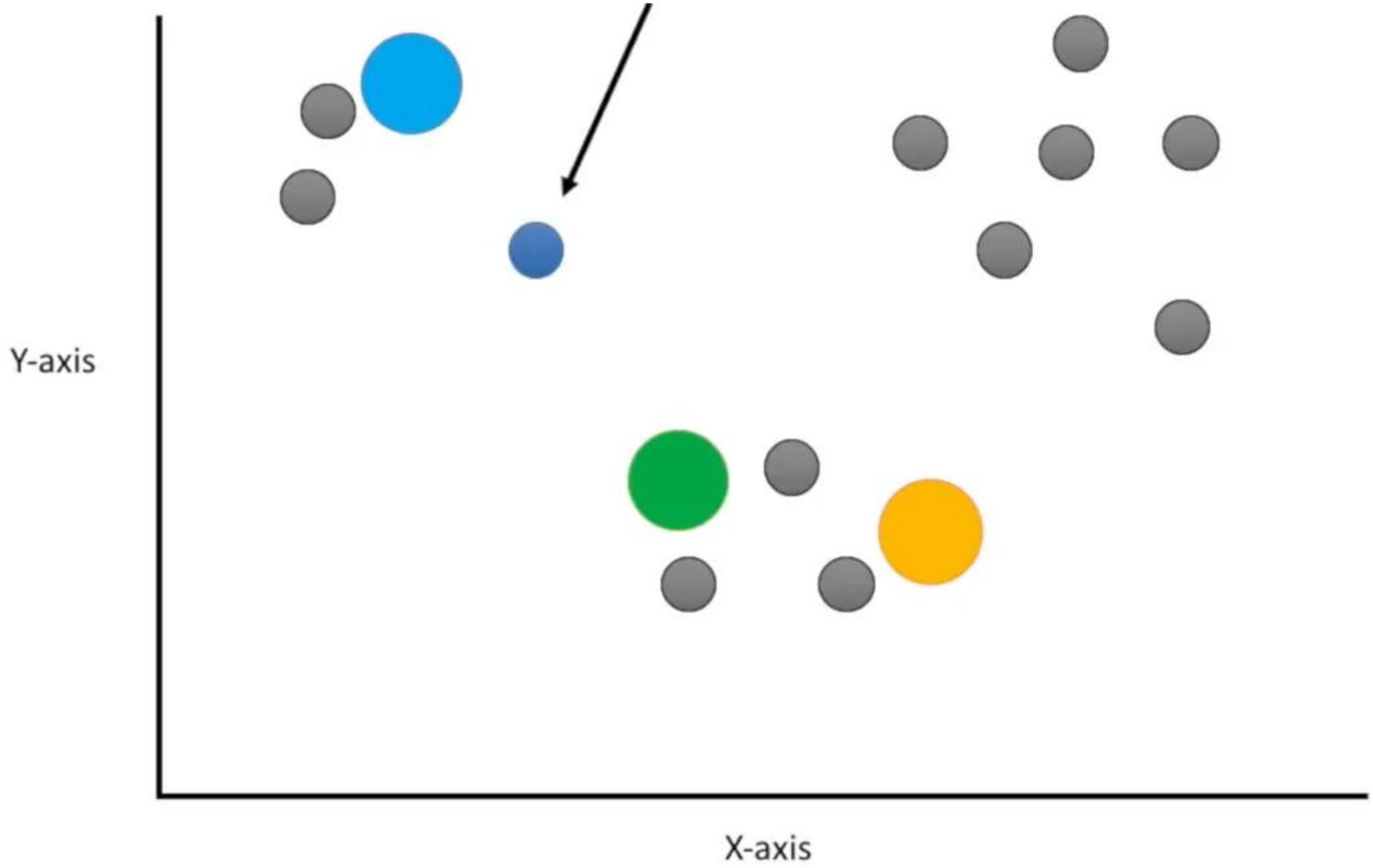
K-means Algorithm: finding cluster centers



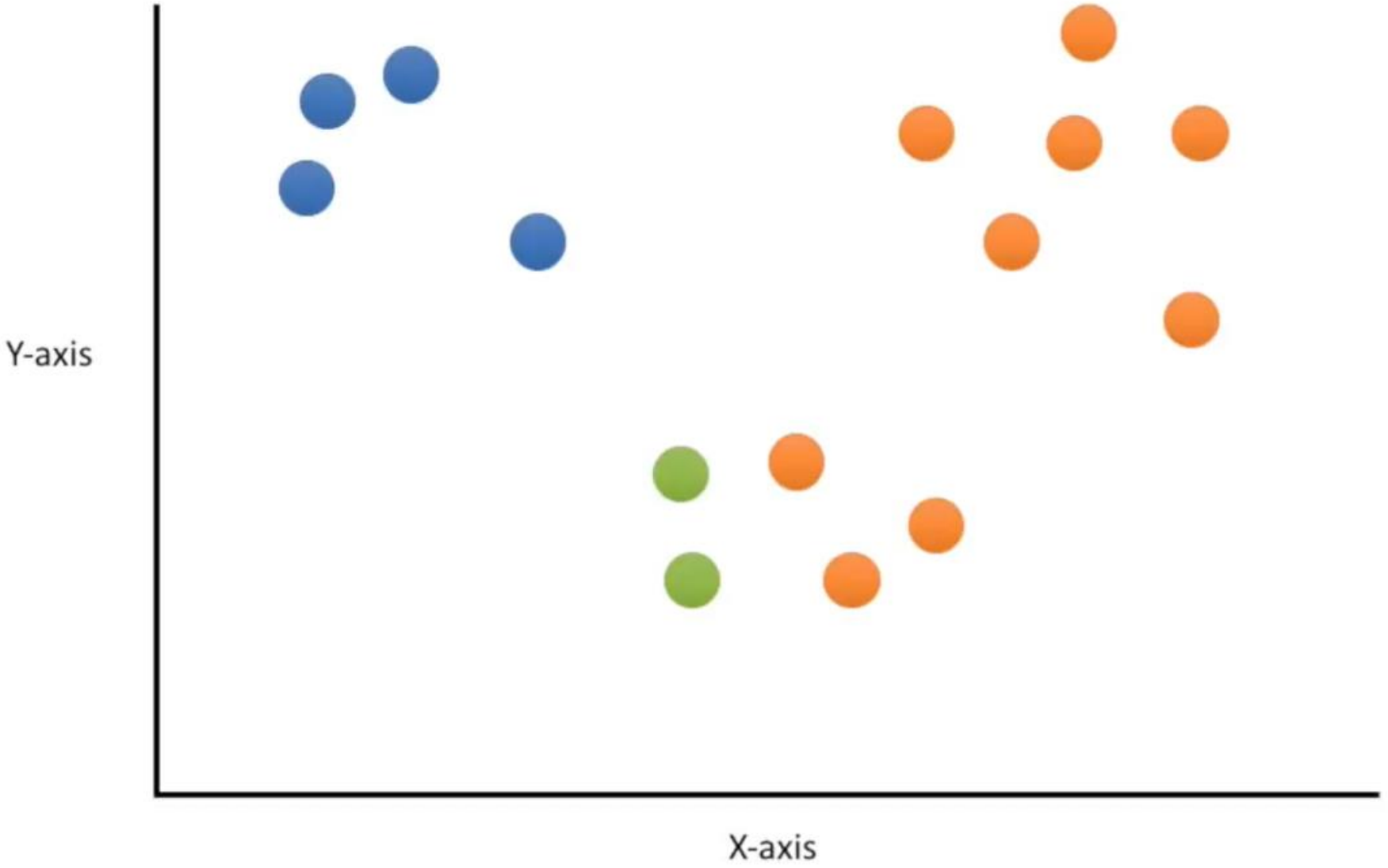
K-means Algorithm: finding cluster centers



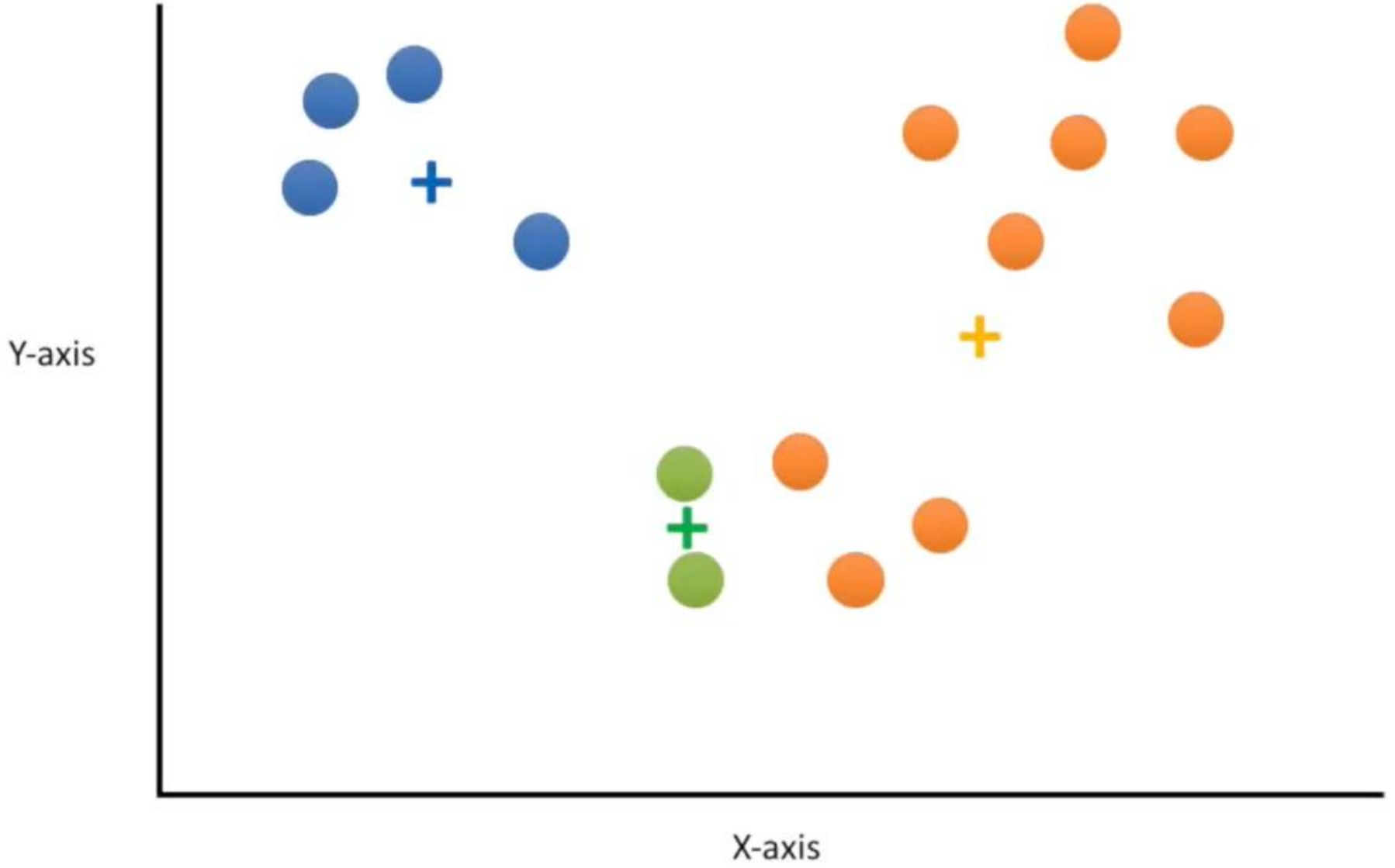
K-means Algorithm: finding cluster centers



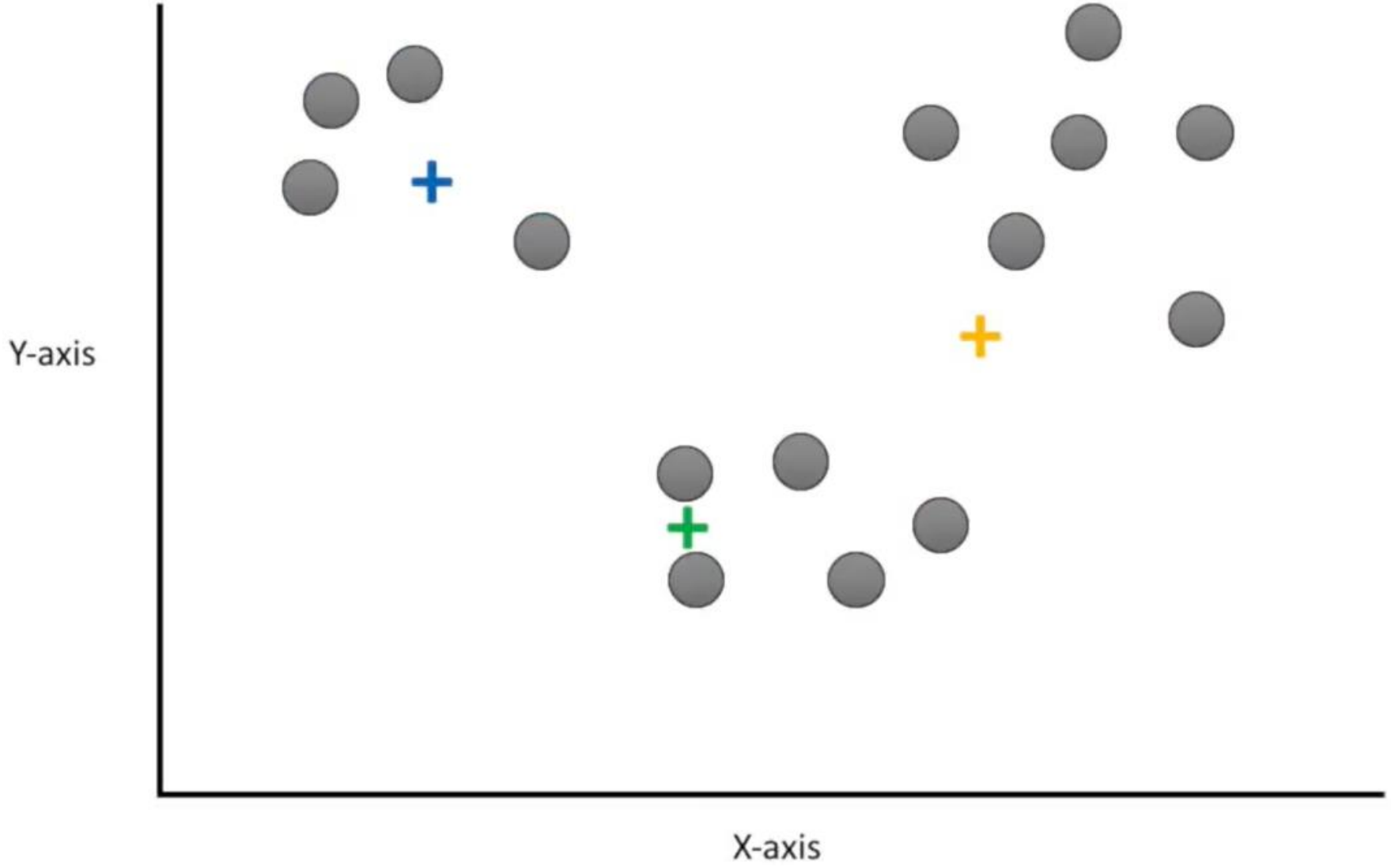
K-means Algorithm: finding cluster centers



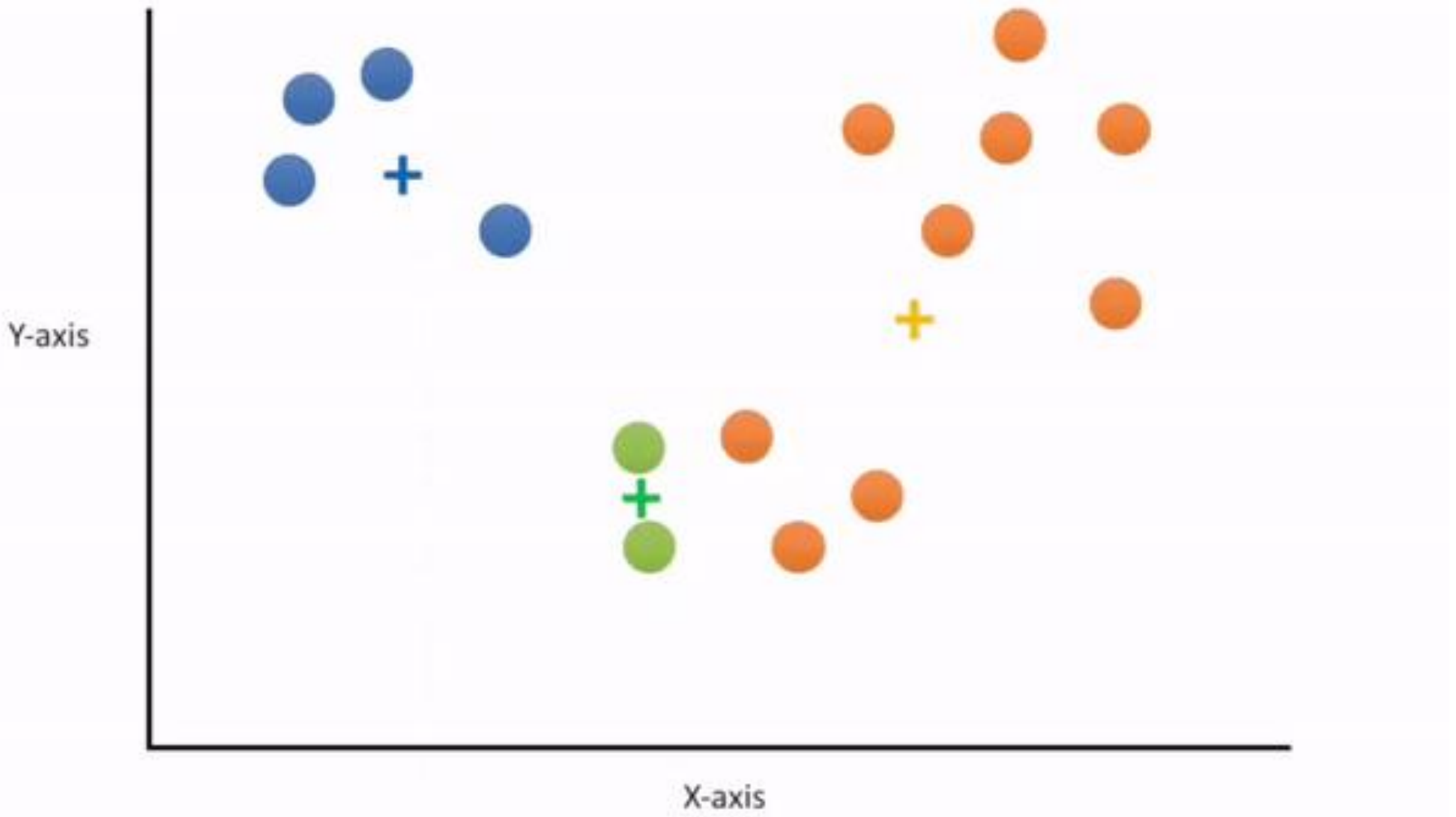
K-means Algorithm: finding cluster centers



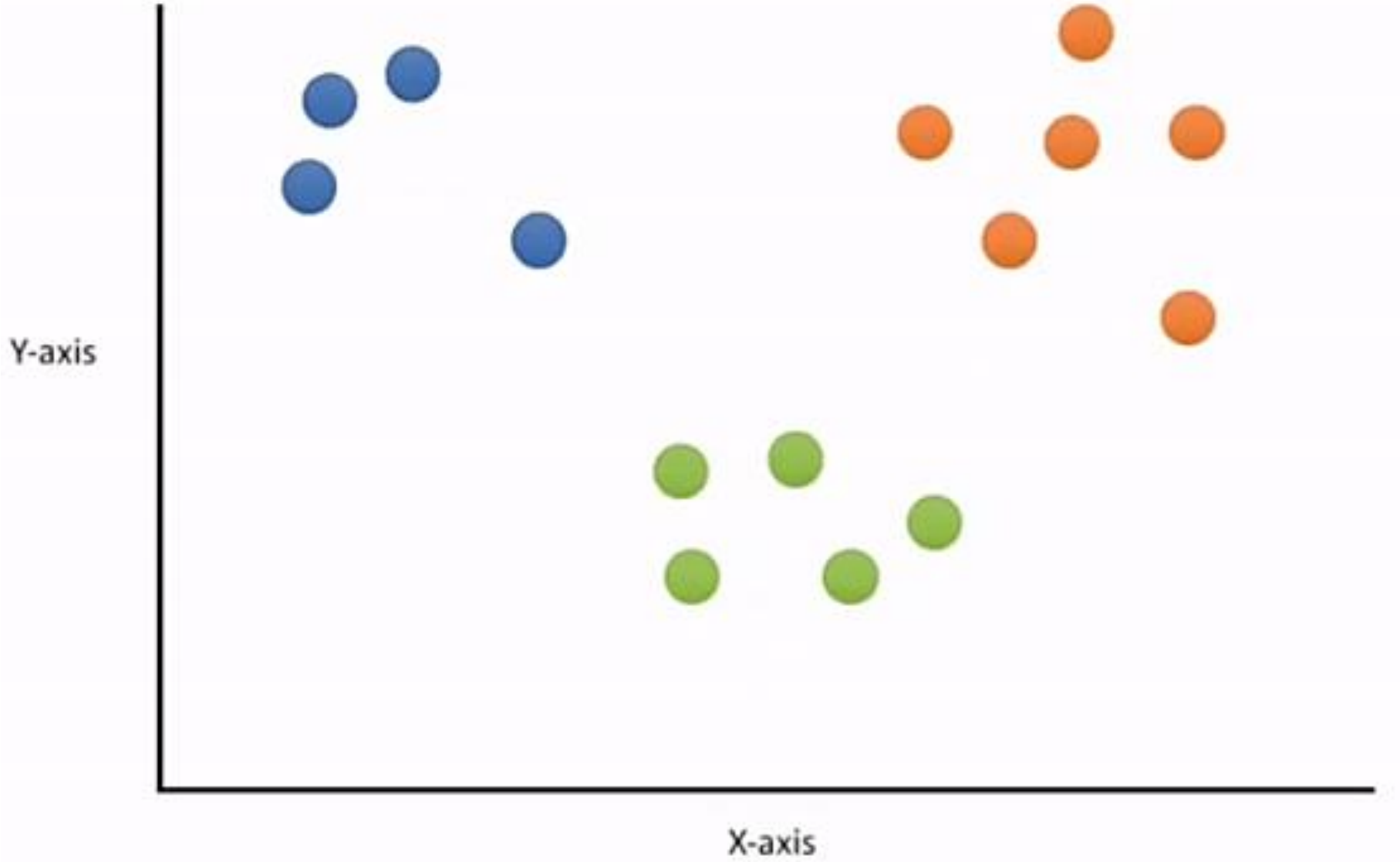
K-means Algorithm: finding cluster centers



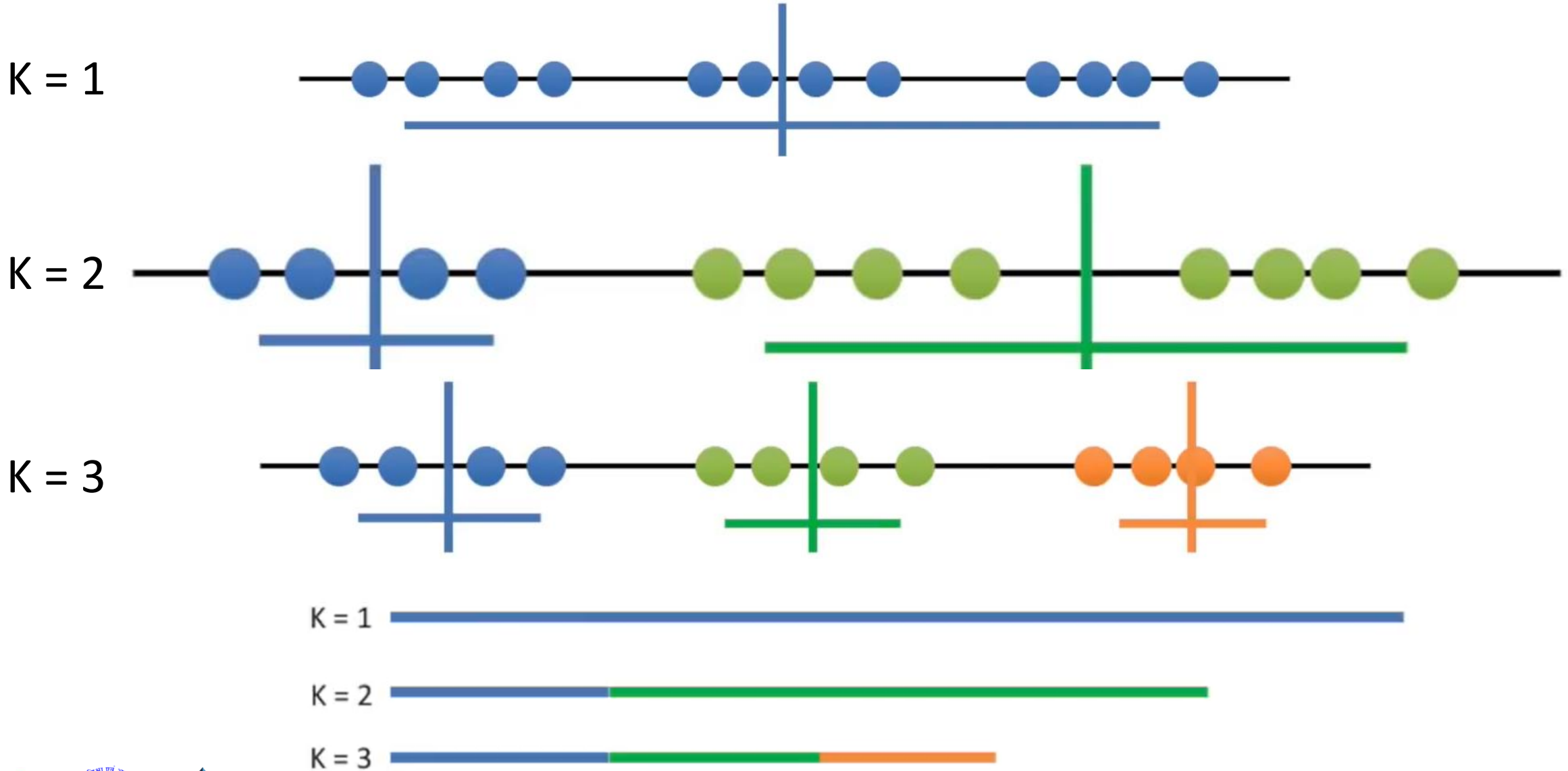
K-means Algorithm: finding cluster centers



K-means Algorithm: finding cluster centers

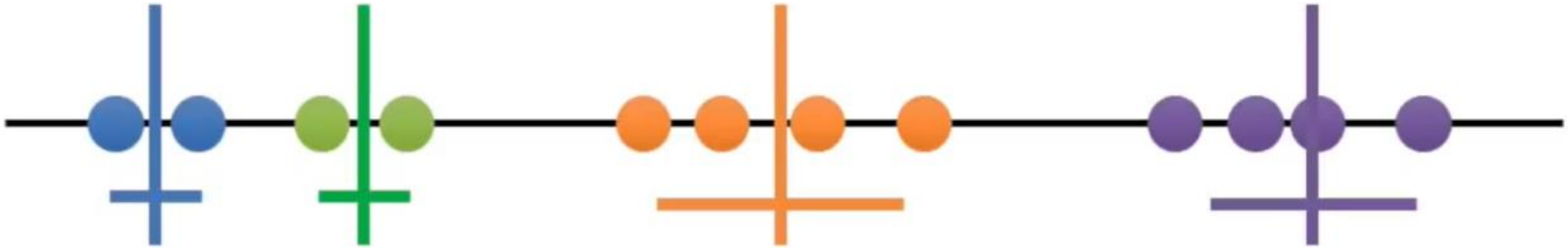


K-means Algorithm: finding value of k



K-means Algorithm: finding value of k

K = 4

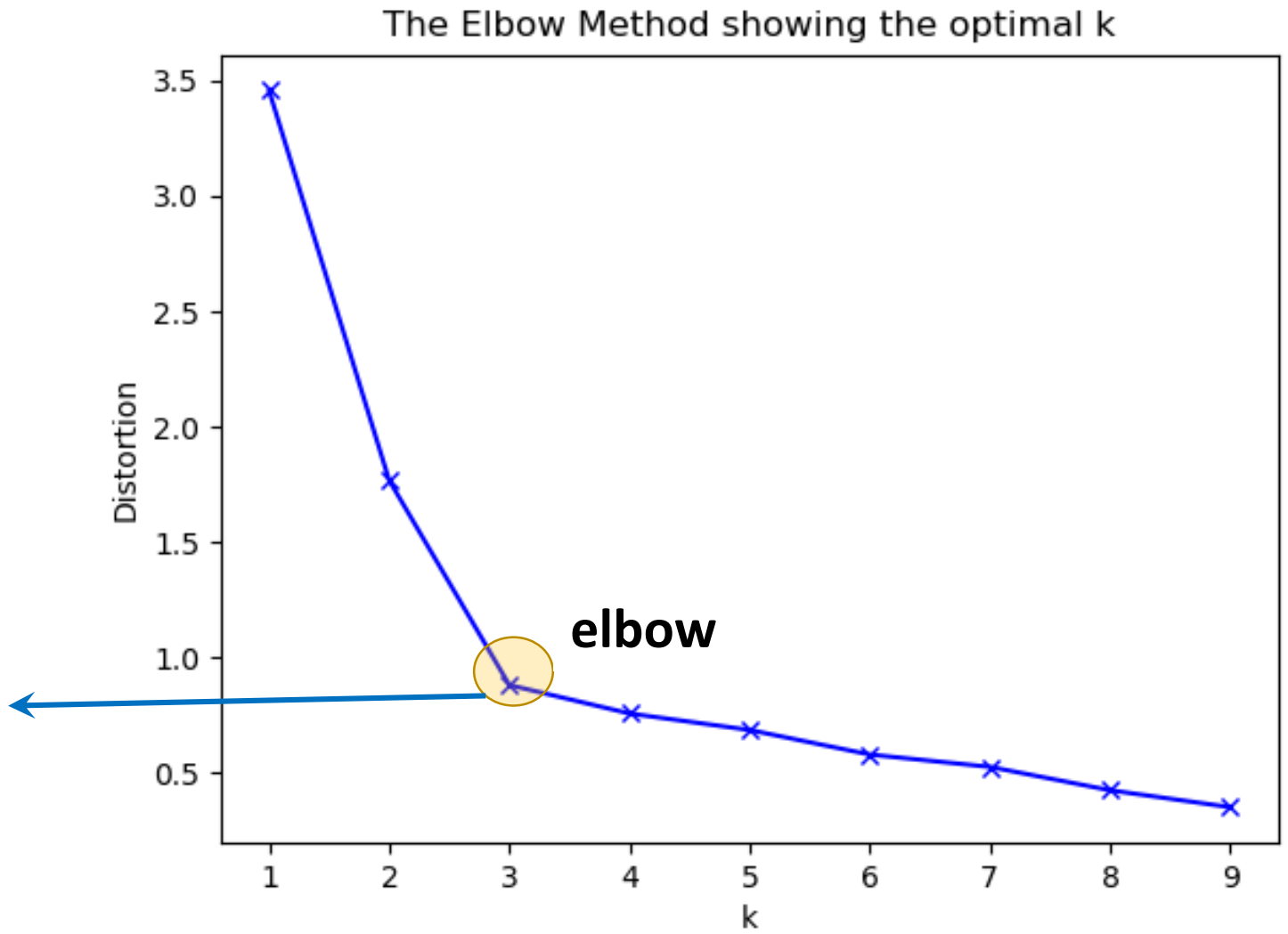


Each time we add a new cluster, the total variation within each cluster is smaller than before. And when there is only one point per cluster, the variation = 0.

Elbow Plot: find the value of k

This is called an “elbow plot”, and you can pick “k” by finding the “**elbow**” in the plot

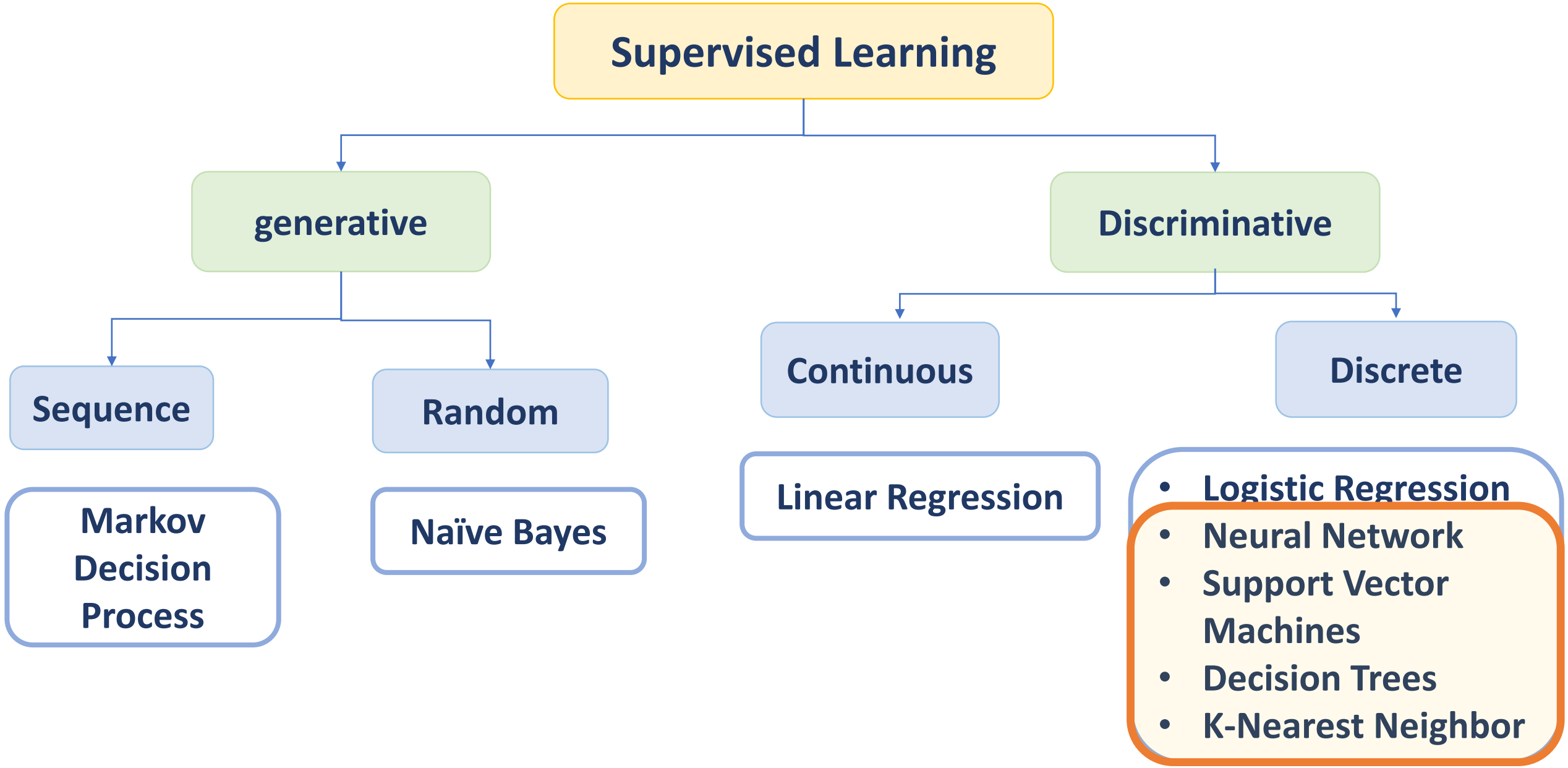
There is a huge reduction in variation with K=3, but after that, the variation doesn’t go down as quickly.



Research work: K-means

M. Jianliang, S. Haikun and B. Ling, "The Application on Intrusion Detection Based on **K-means Cluster Algorithm**," 2009 International Forum on Information Technology and Applications, Chengdu, 2009, pp. 150-152.

Internet security has been one of the most important problems in the world. Anomaly detection is the basic method to defend new attack in intrusion detection. Network intrusion detection is the process of monitoring the events occurring in a computing system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality. A wide variety of data mining techniques have been applied to intrusion detections. In data mining, clustering is the most important unsupervised learning process used to find the **structures or patterns in a collection of unlabeled data.** We use the K-means algorithm to cluster and analyze the data in this paper. Computer simulations show that this method can detect **unknown intrusions efficiently in the real network connections**.


Supervised Learning
generative
Discriminative
Sequence
Random
Continuous
Discrete
Markov Decision Process
Naïve Bayes
Linear Regression
Logistic Regression
Neural Network
Support Vector Machines
Decision Trees
K-Nearest Neighbor

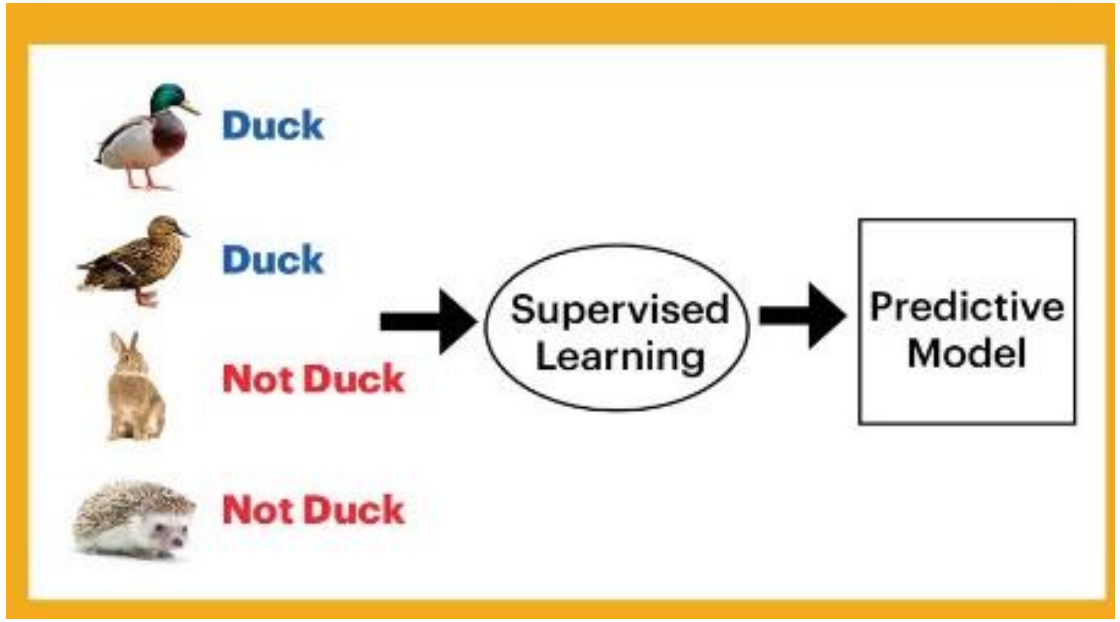
Discriminative vs Generative

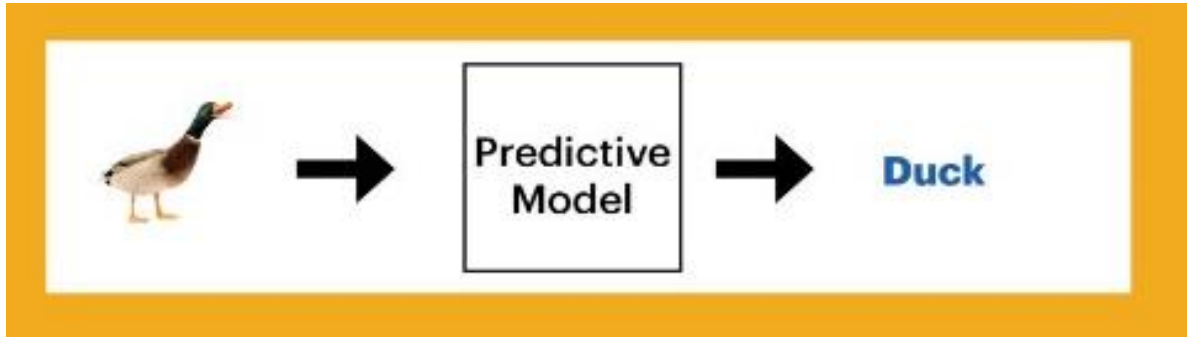
Iron Man

Hulk

Supervised learning

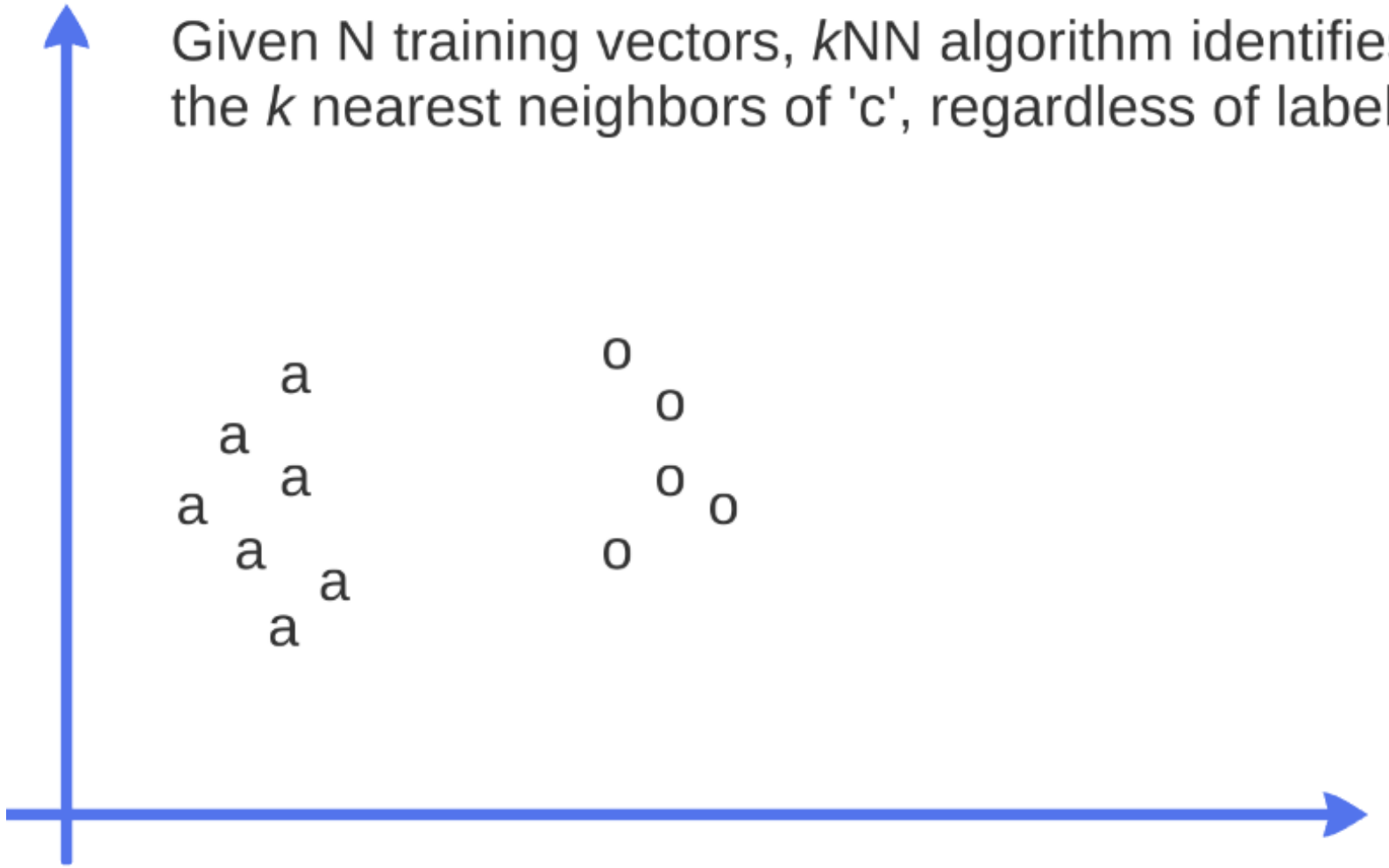
- A model is prepared through a training process in which it is required **to make predictions** and is corrected when those predictions are wrong.
- The training process continues until the model achieves a desired level of accuracy on the training data.



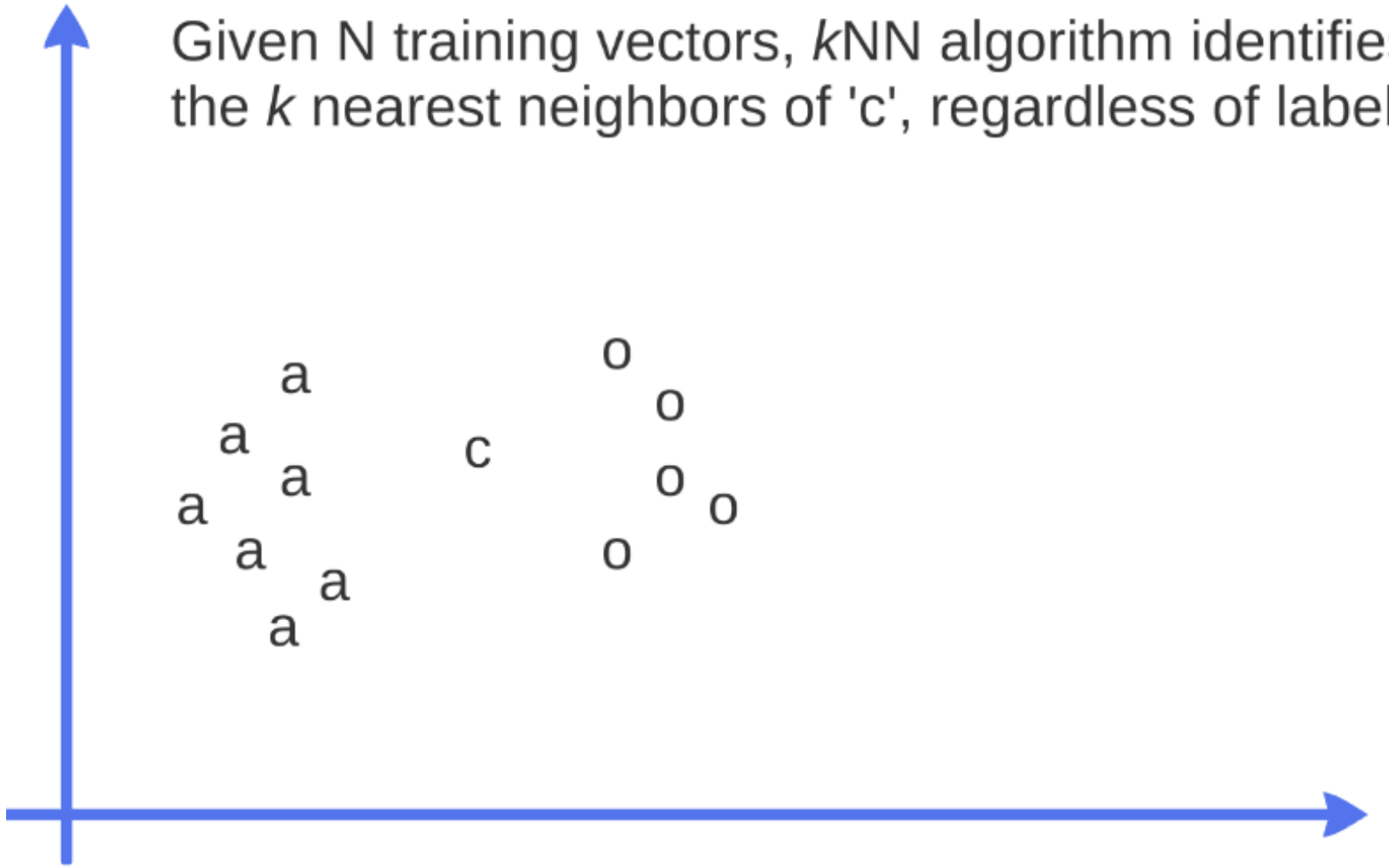


Courtesy: Western Digital https://www.datanami.com/2018/09/05/how-to-build-a-better-machine-learning-pipeline/

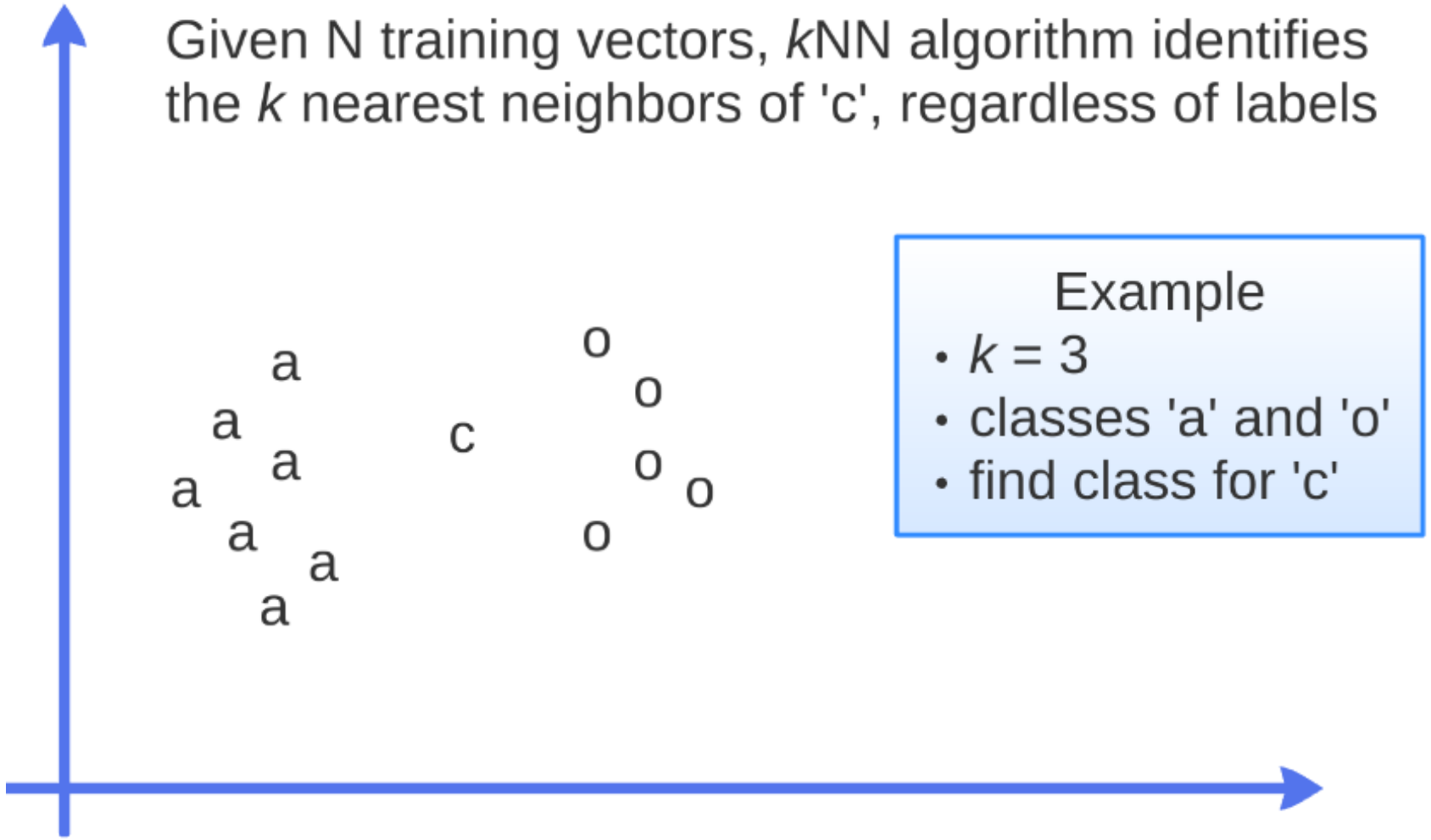
K-Nearest Neighbor (KNN) algorithm



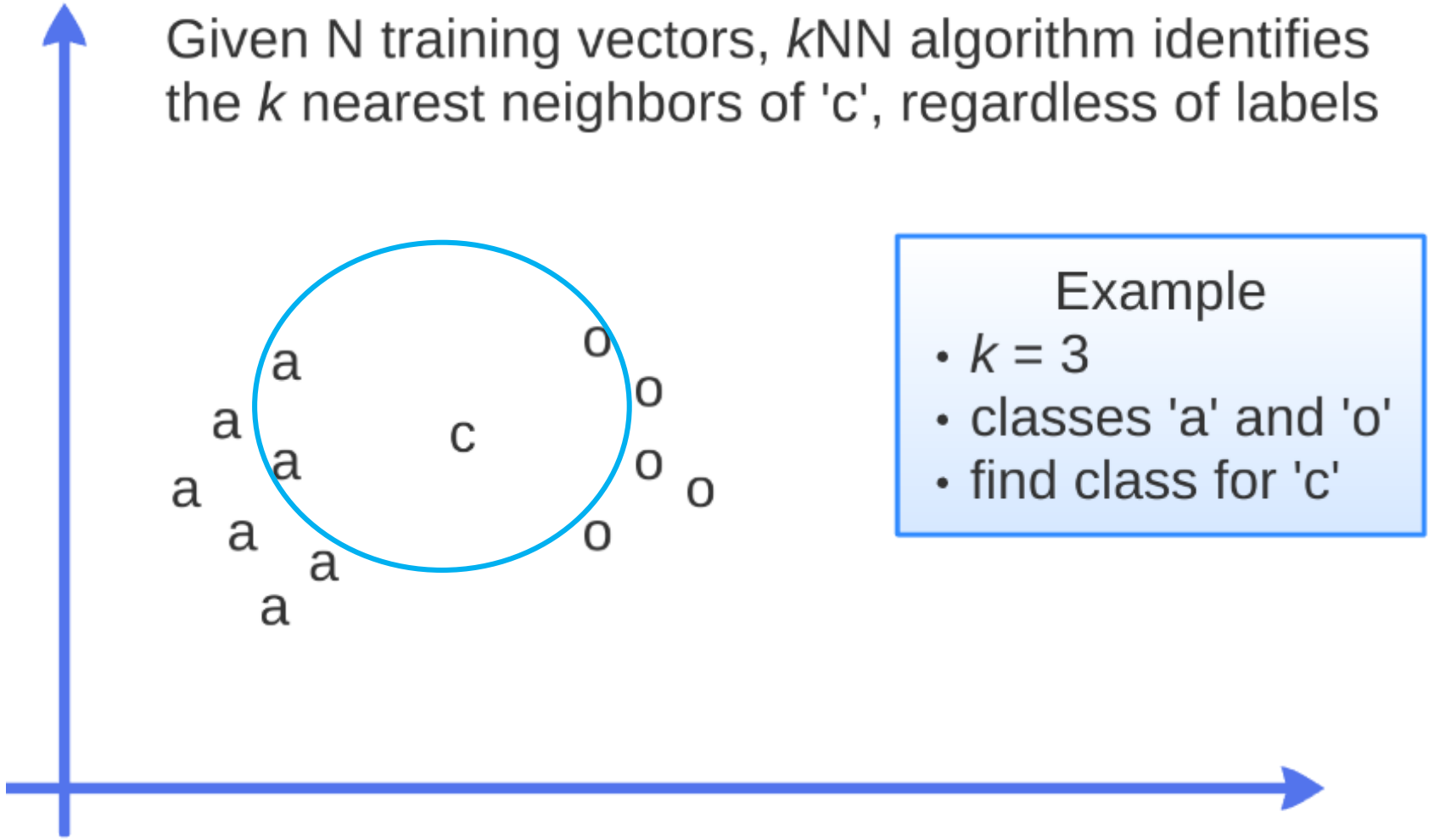
K-Nearest Neighbor (KNN) algorithm



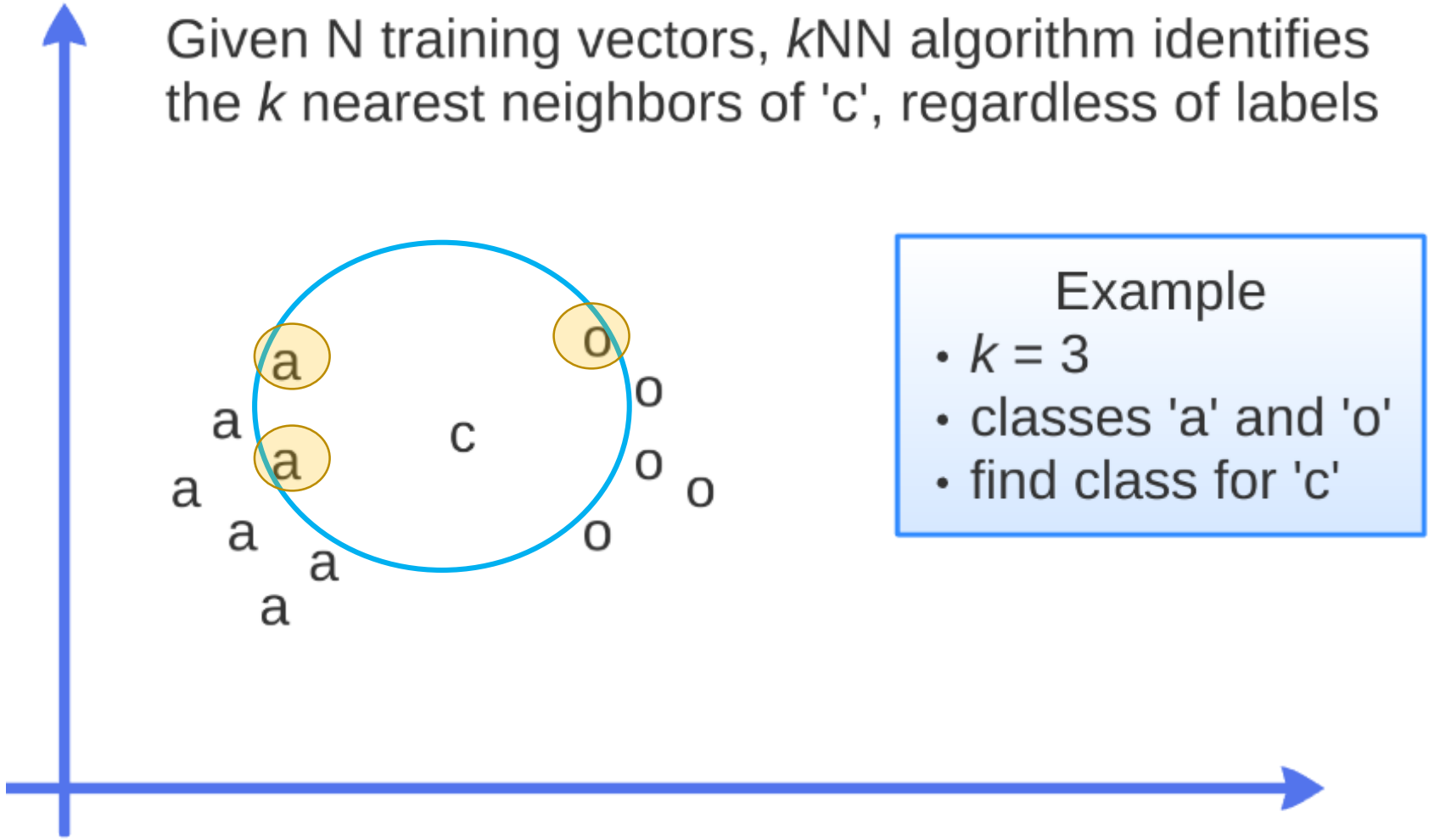
K-Nearest Neighbor (KNN) algorithm



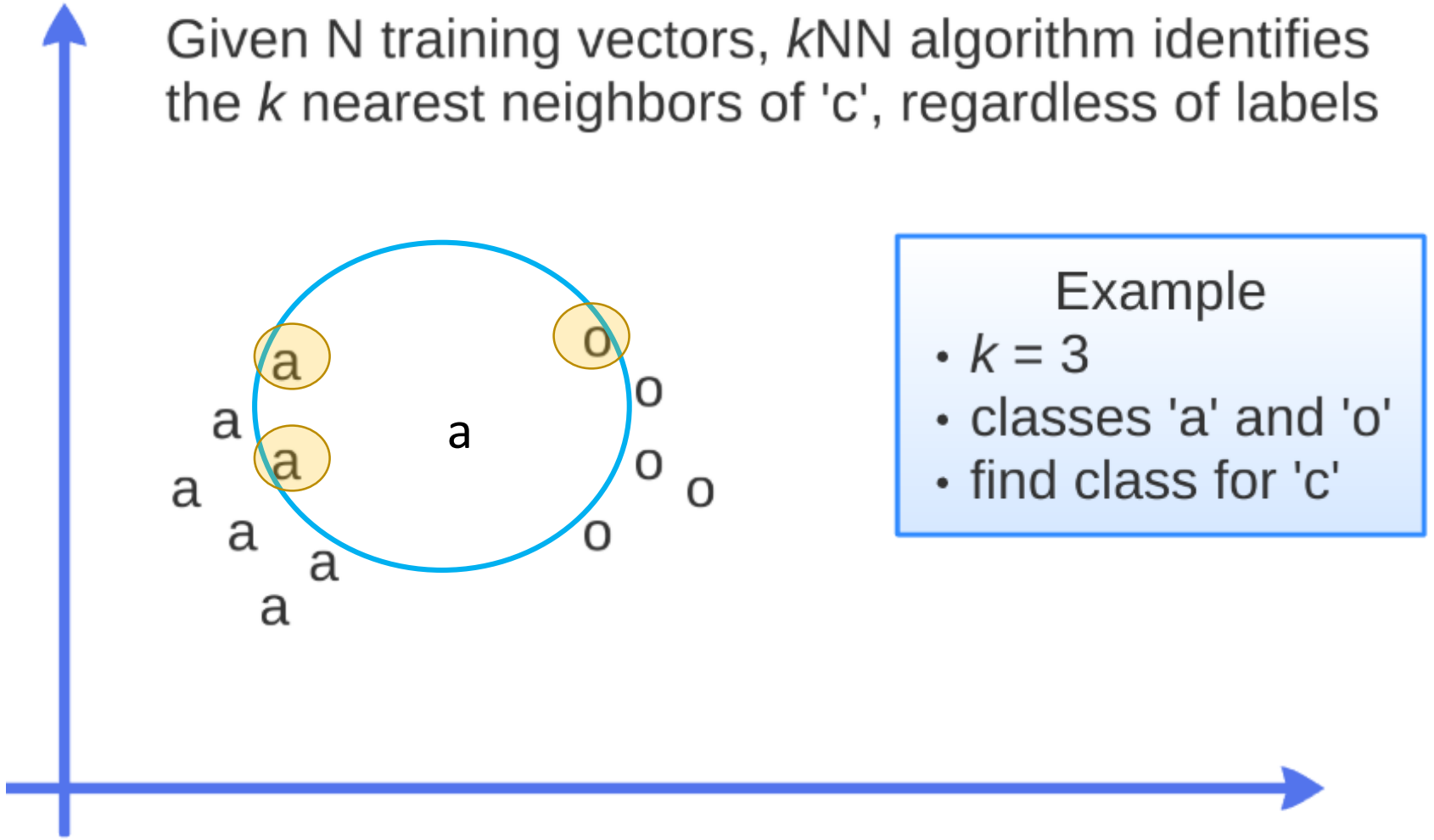
K-Nearest Neighbor (KNN) algorithm



K-Nearest Neighbor (KNN) algorithm

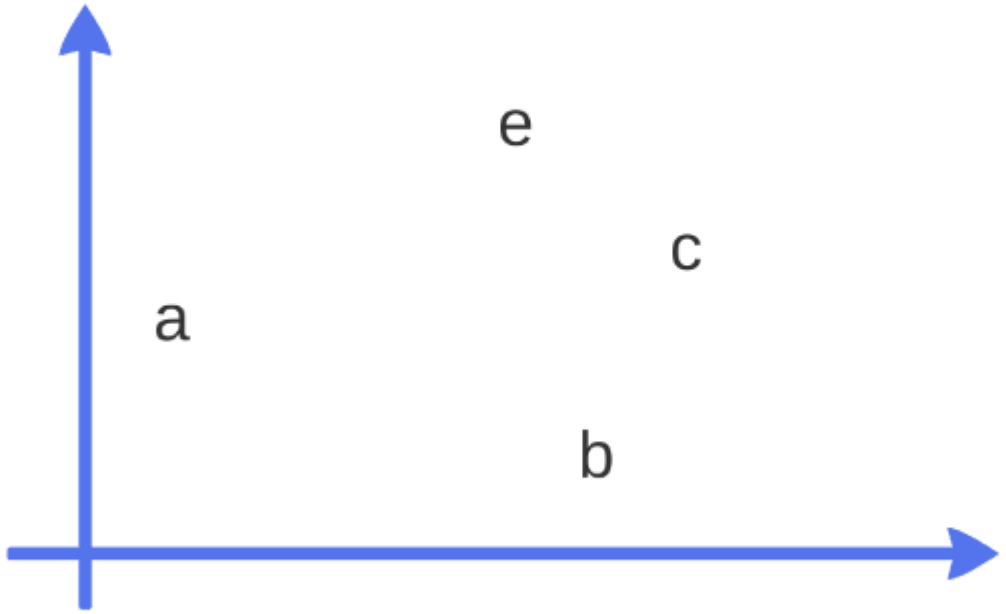


K-Nearest Neighbor (KNN) algorithm



K-Nearest Neighbor (KNN) algorithm (K = 1)

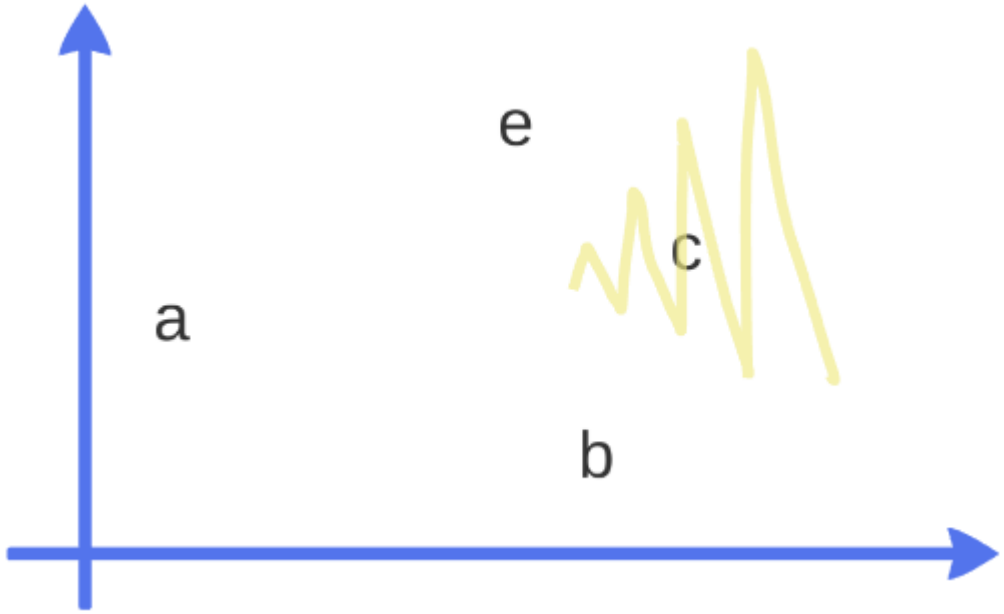
When k = 1, each training vector defines a region in space, defining a *Voronoi* partition of the space



$$R_i = \{x : d(x, x_i) < d(x, x_j), i \neq j\}$$

K-Nearest Neighbor (KNN) algorithm (K = 1)

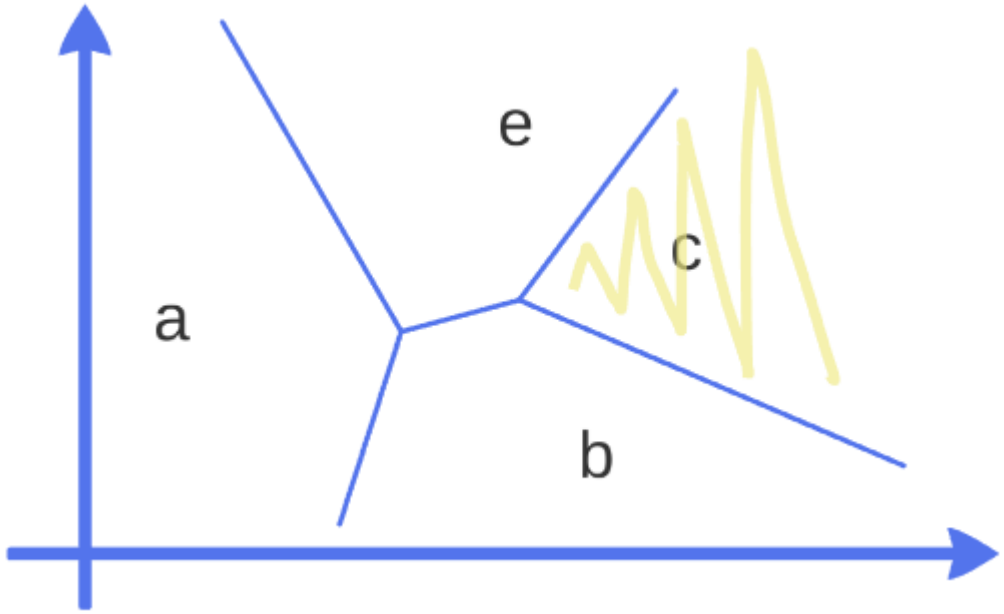
When k = 1, each training vector defines a region in space, defining a *Voronoi* partition of the space



$$R_i = \{x : d(x, x_i) < d(x, x_j), i \neq j\}$$

K-Nearest Neighbor (KNN) algorithm (K = 1)

When k = 1, each training vector defines a region in space, defining a *Voronoi* partition of the space



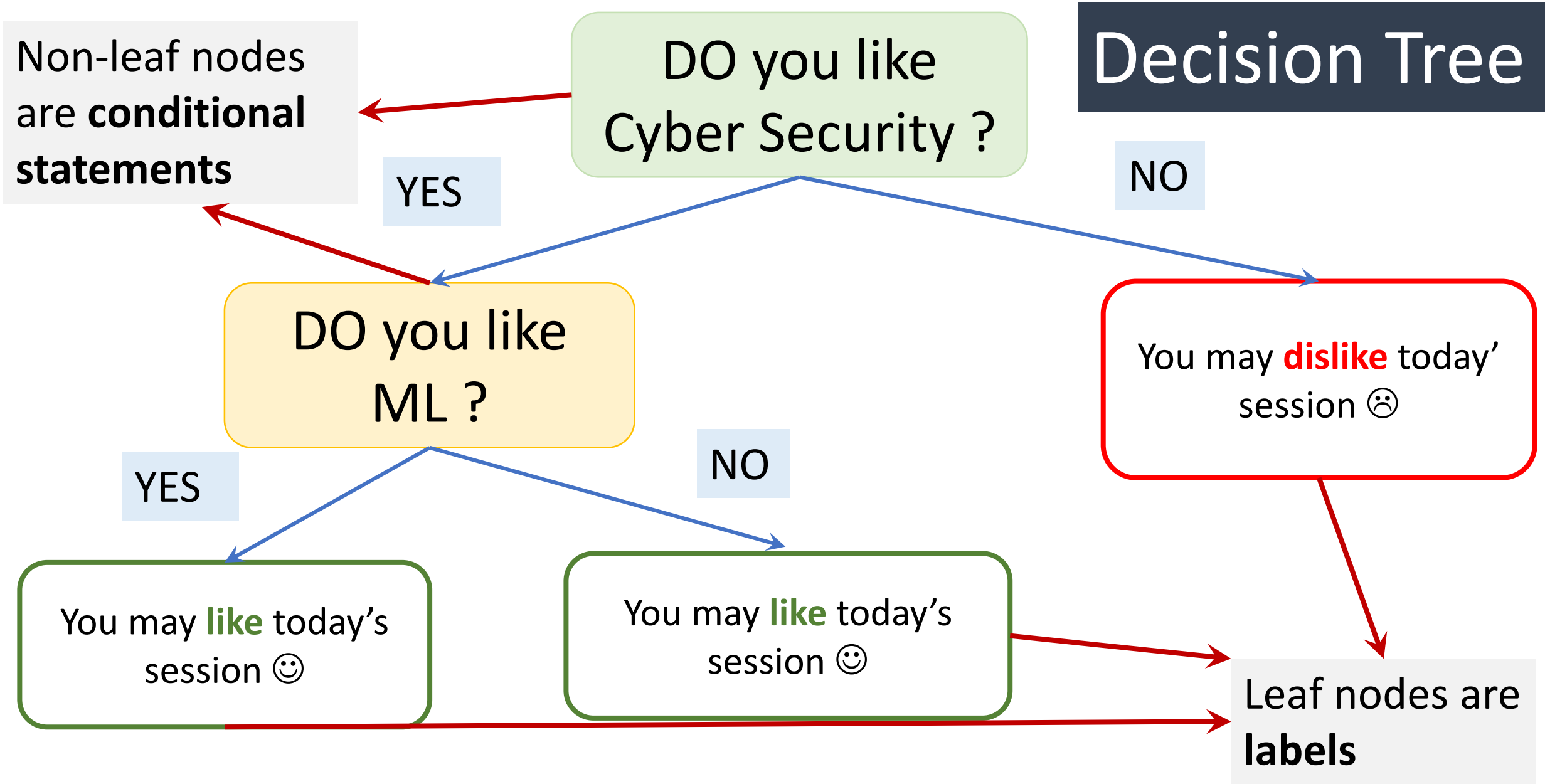
$$R_i = \{x : d(x, x_i) < d(x, x_j), i \neq j\}$$

Research work: K-NN

Wenchao Li, Ping Yi, Yue Wu, Li Pan, and Jianhua Li, "A New Intrusion Detection System Based on **KNN Classification Algorithm in Wireless Sensor Network**“ , Journal of Electrical and Computer Engineering, Volume 2014, Article ID 240217, 8 pages, http://dx.doi.org/10.1155/2014/240217

The Internet of Things has broad application in military field, commerce, environmental monitoring, and many other fields. However, the open nature of the information media and the poor deployment environment have brought great risks to the security of wireless sensor networks, seriously restricting the application of wireless sensor network. Internet of Things composed of wireless sensor network faces security threats mainly from Dos attack, replay attack, integrity attack, false routing information attack, and flooding attack. In this paper, we proposed a new intrusion detection system based on -nearest neighbor (K-nearest neighbor, referred to as KNN below) classification algorithm in wireless sensor network. This system can separate abnormal nodes from normal nodes by observing their abnormal behaviors, and we analyze parameter selection and error rate of the intrusion detection system. The paper elaborates on the design and implementation of the detection system. This system has achieved efficient, rapid intrusion detection by improving the wireless ad hoc on-demand distance vector routing protocol (Ad hoc On-Demand Distance the Vector Routing, AODV). Finally, the test results show that: the system has high detection accuracy and speed, in accordance with the requirement of wireless sensor network intrusion detection

Decision Tree



Research work: Decision Tree

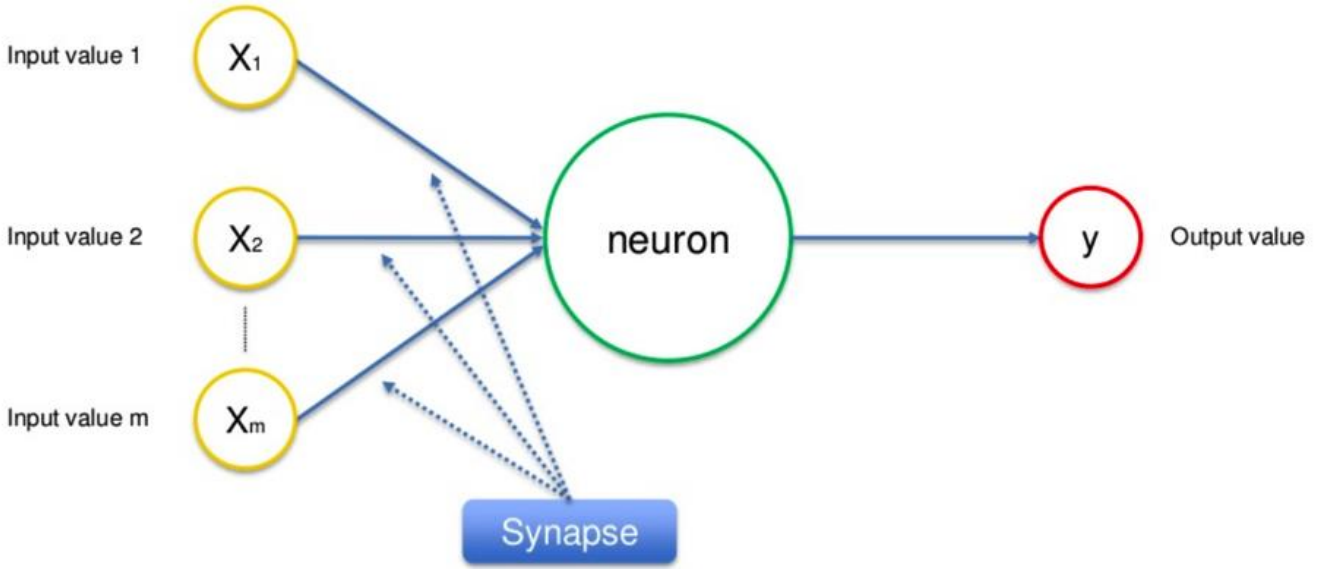
Kruegel C., Toth T. (2003) Using **Decision Trees to Improve Signature-Based Intrusion Detection.** In: Vigna G., Kruegel C., Jonsson E. (eds) Recent Advances in Intrusion Detection. RAID 2003. Lecture Notes in Computer Science, vol 2820. Springer, Berlin, Heidelberg

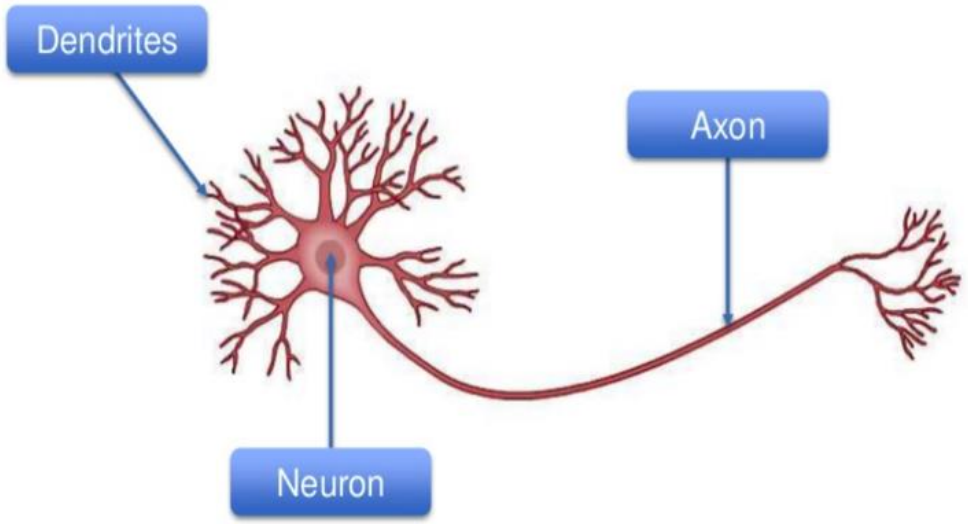
Most deployed intrusion detection systems (IDSs) follow a signature-based approach where attacks are identified by matching each input event against predefined signatures that model malicious activity. This matching process accounts for the most resource intensive task of an IDS. Many systems perform the matching by comparing each input event to all rules sequentially. This is far from being optimal. Although sometimes ad-hoc optimizations are utilized, no general solution to this problem has been proposed so far.

This paper describes an approach where machine learning clustering techniques are applied to improve the matching process. Given a set of signatures (each dictating a number of constraints the input data must fulfill to trigger it) an algorithm generates a **decision tree that is used to find malicious events using as few redundant comparisons as possible.**

This general idea has been applied to a network-based IDS. In particular, a system has been implemented that replaces the detection engine of **Snort**. Experimental evaluation shows that the **speed of the detection process has been significantly improved,** even compared to Snort's recently released, fully revised detection engine.

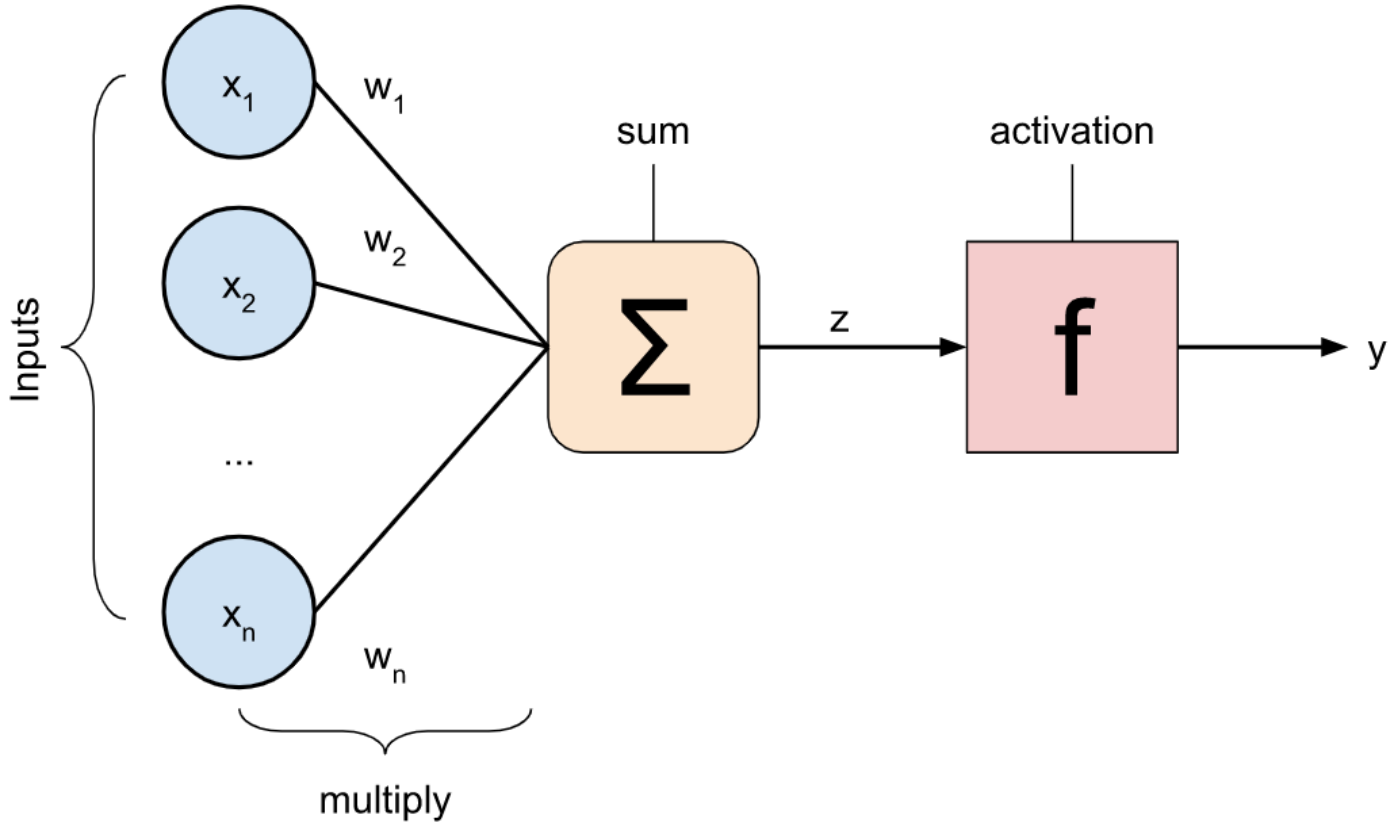
Neuron





Nerve cell

Neuron structure

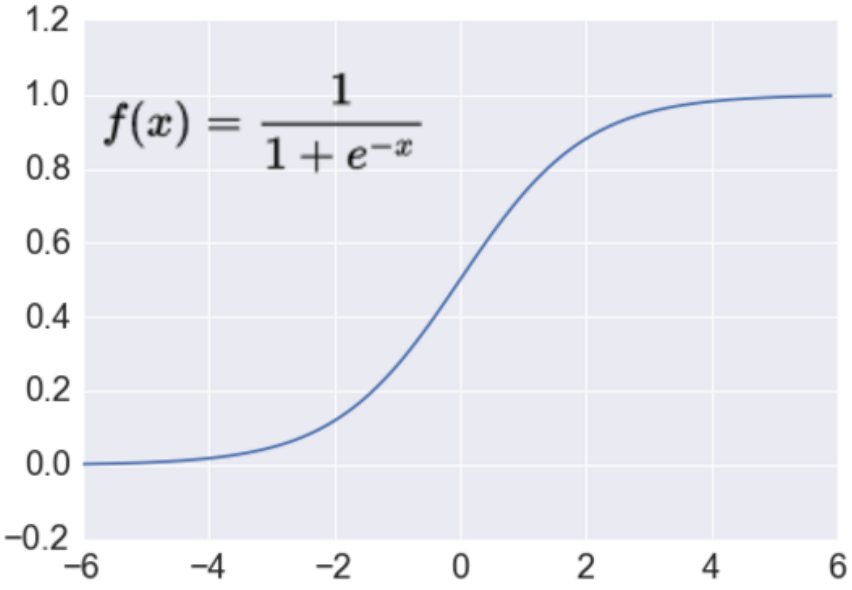


$$y = f(z) = f(\sum_i w_i x_i) = f(w^T x)$$

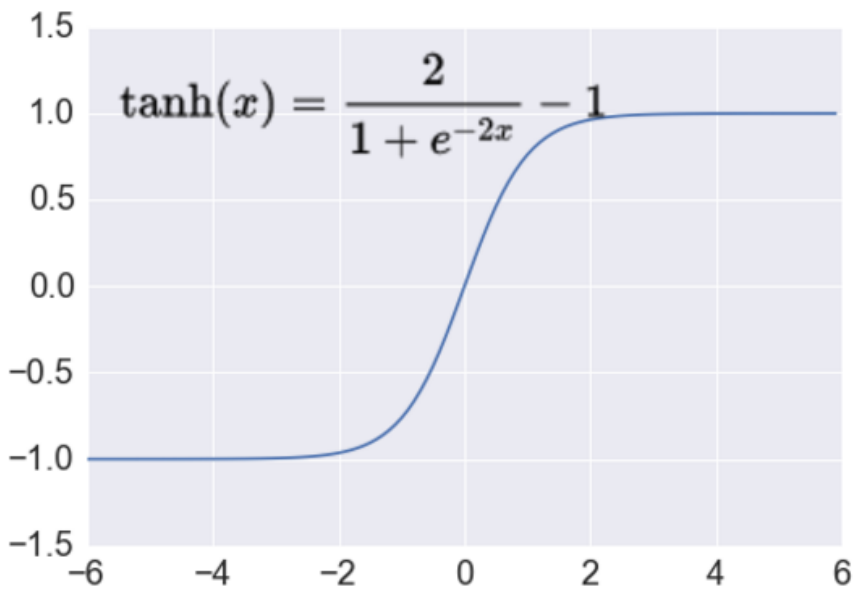
https://www.doc.ic.ac.uk/~nuric/teaching/imperial-college-machine-learning-neural-networks.html

Activation functions

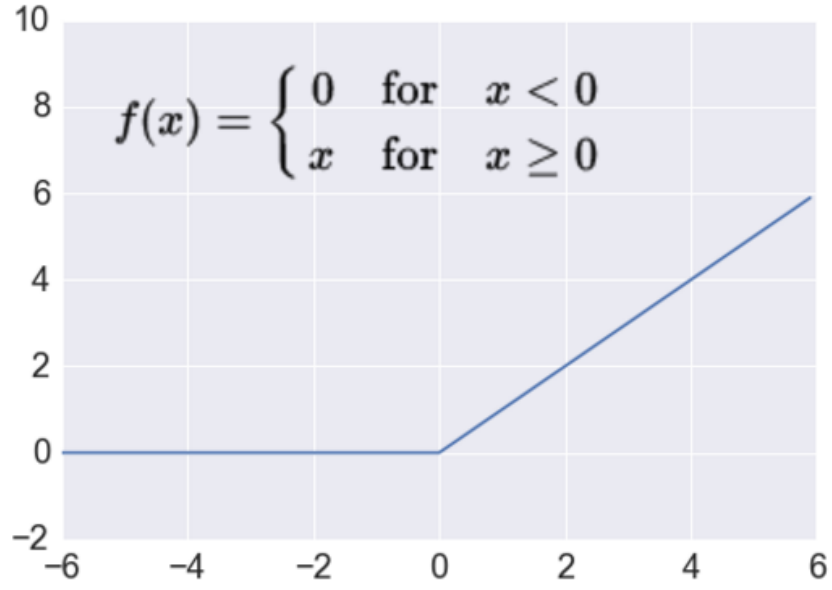
Sigmoid



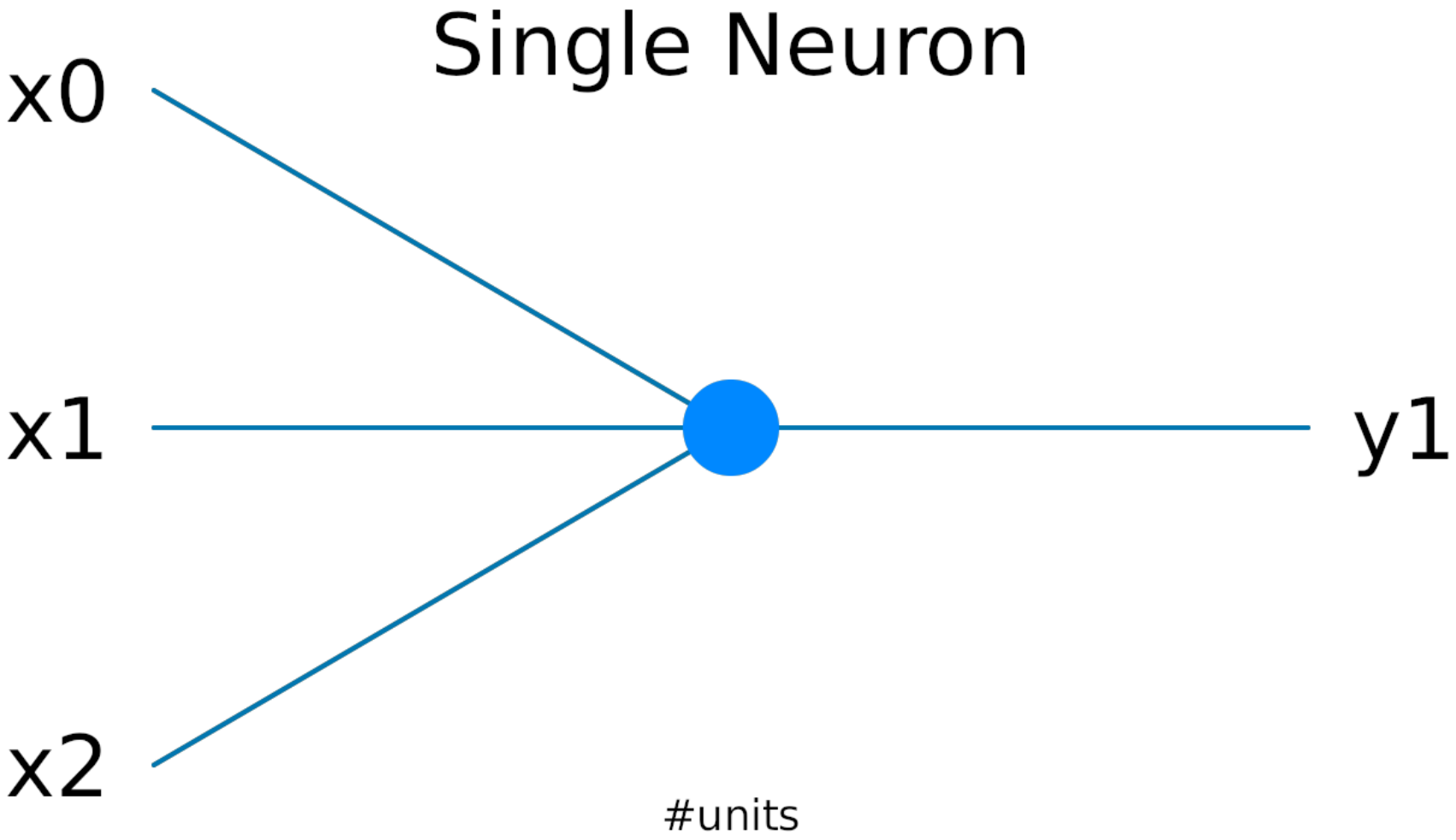
TanH



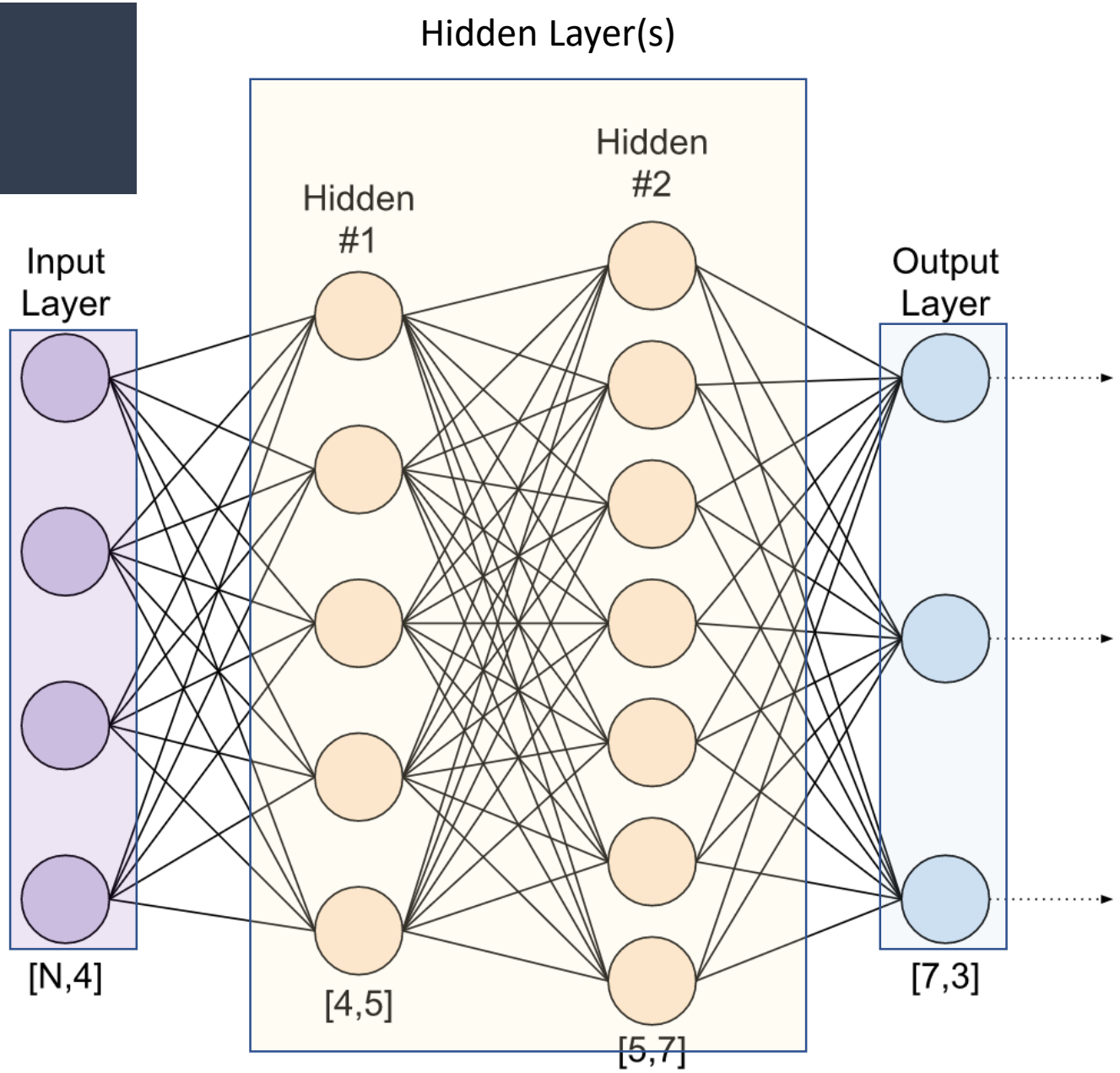
ReLU



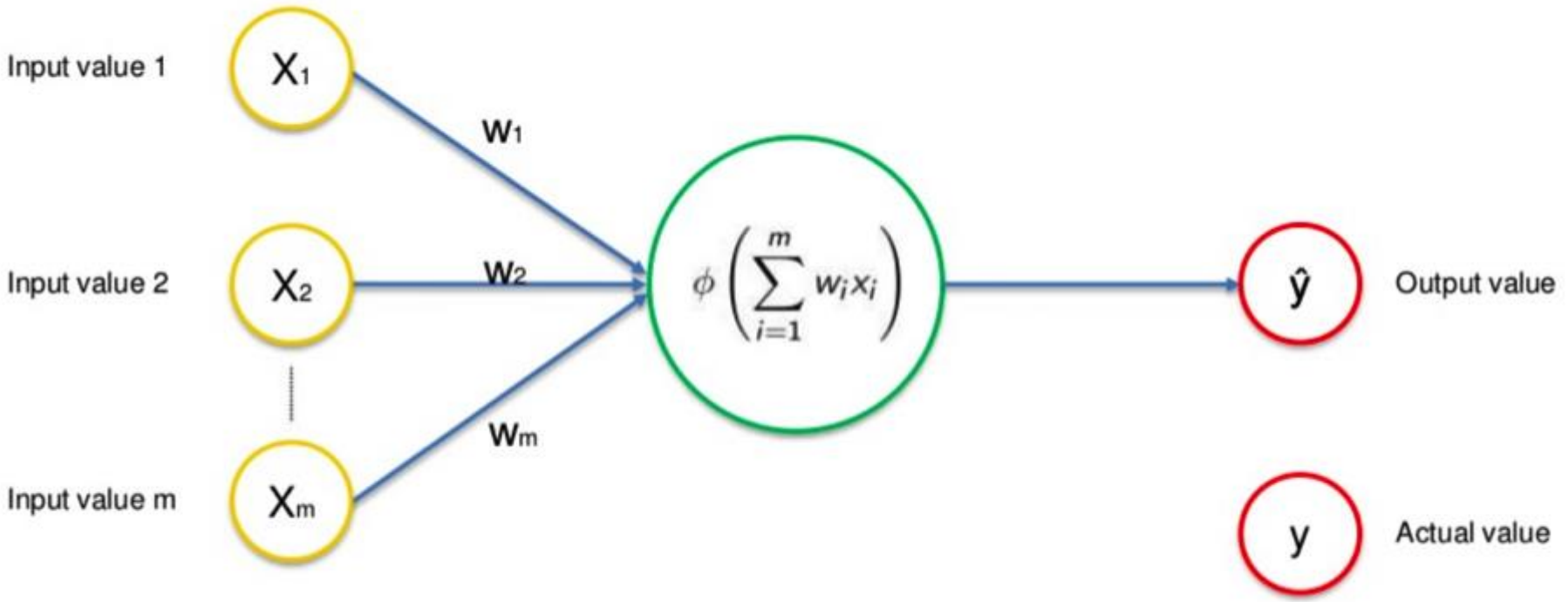
Neural Network: layer



Neural Network Structure

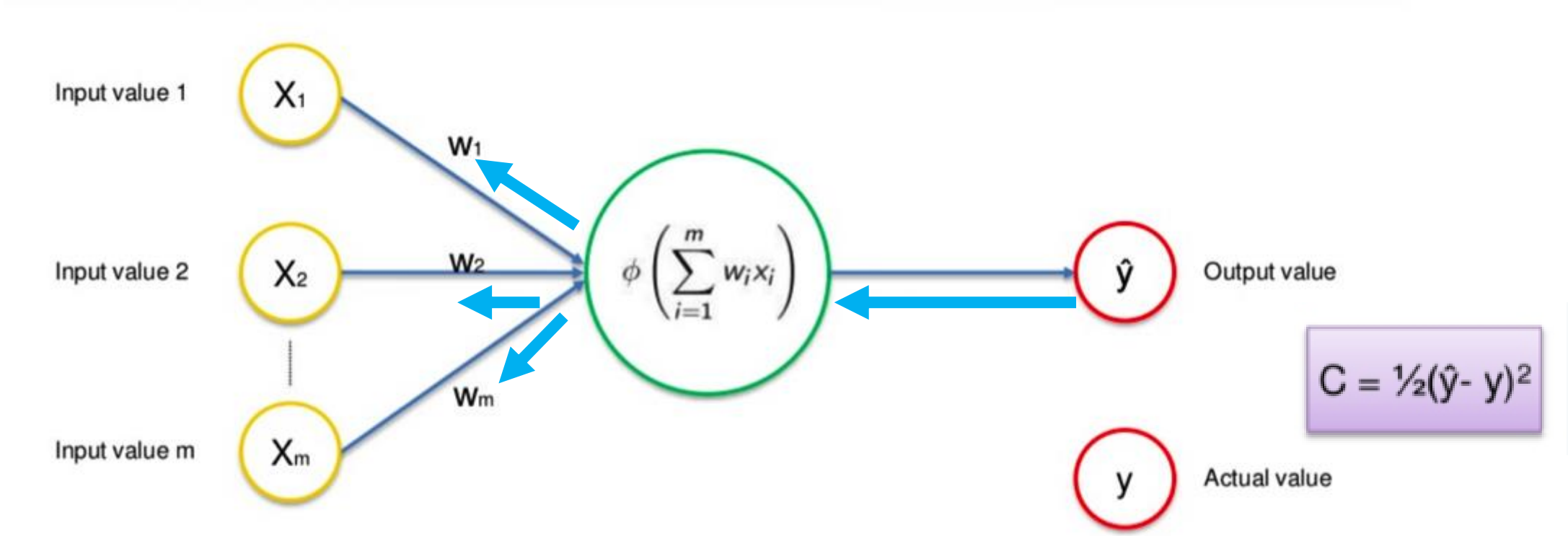


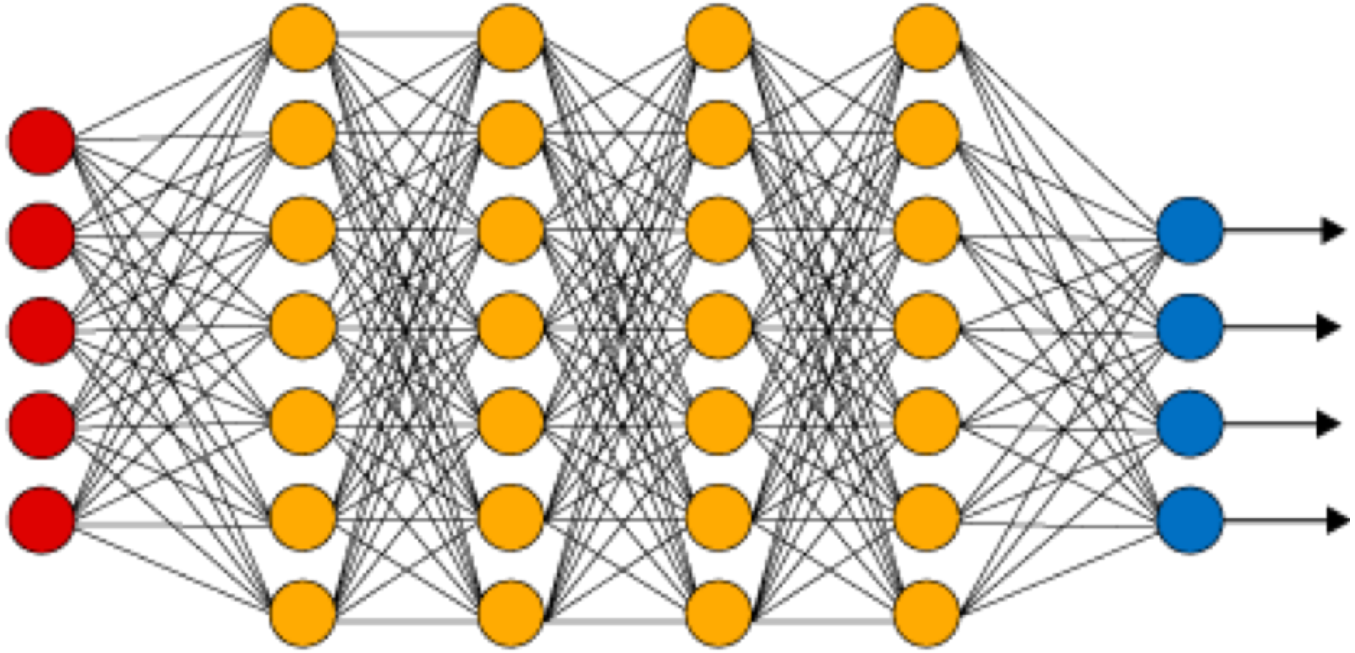
Neural Network: neuron function



https://www.superdatascience.com/blogs/artificial-neural-networks-how-do-neural-networks-learn

Neural Network: back propagation





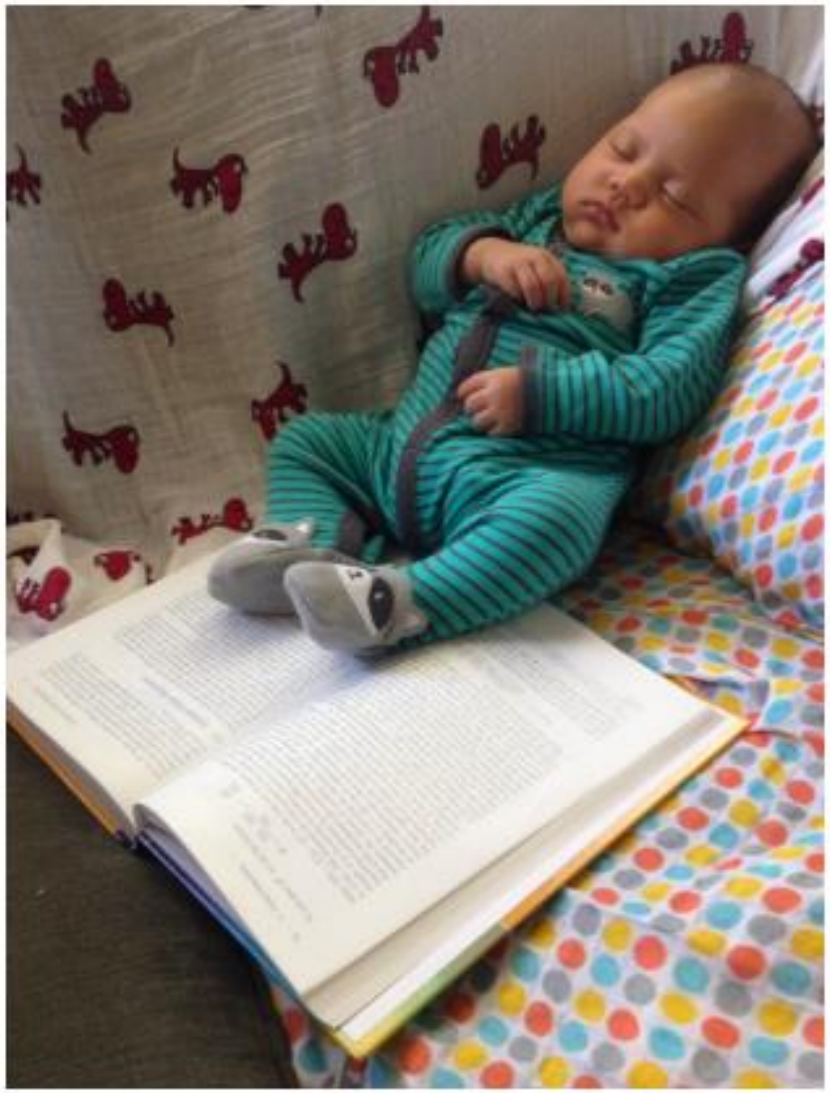
Deep learning

Research work: Deep Learning

N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A **Deep Learning Approach to Network Intrusion Detection**," in *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41-50, Feb. 2018.

Network intrusion detection systems (NIDSs) play a crucial role in defending computer networks. However, there are concerns regarding the feasibility and sustainability of current approaches when faced with the demands of modern networks. More specifically, these concerns relate to the increasing levels of required human interaction and the decreasing levels of detection accuracy. This paper presents a novel deep learning technique for intrusion detection, which addresses these concerns. We detail our proposed **nonsymmetric deep autoencoder (NDAE) for unsupervised feature learning.** Furthermore, we also propose our novel deep learning classification model constructed using **stacked NDAEs**. Our proposed classifier has been implemented in graphics processing unit (GPU)-enabled TensorFlow and evaluated using the benchmark **KDD Cup '99** and **NSL-KDD** datasets. Promising results have been obtained from our model thus far, demonstrating improvements over existing approaches and the strong potential for use in modern NIDSs.

Support Vector Machines (SVM)



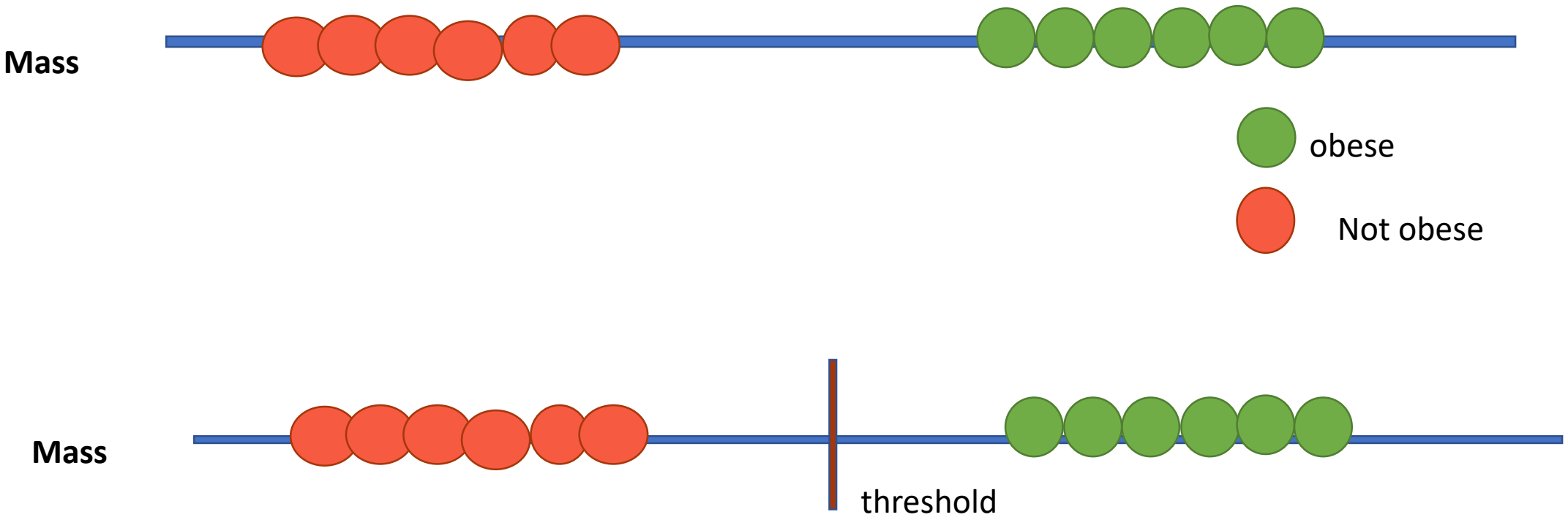
Linear Regression?

I covered that last year.

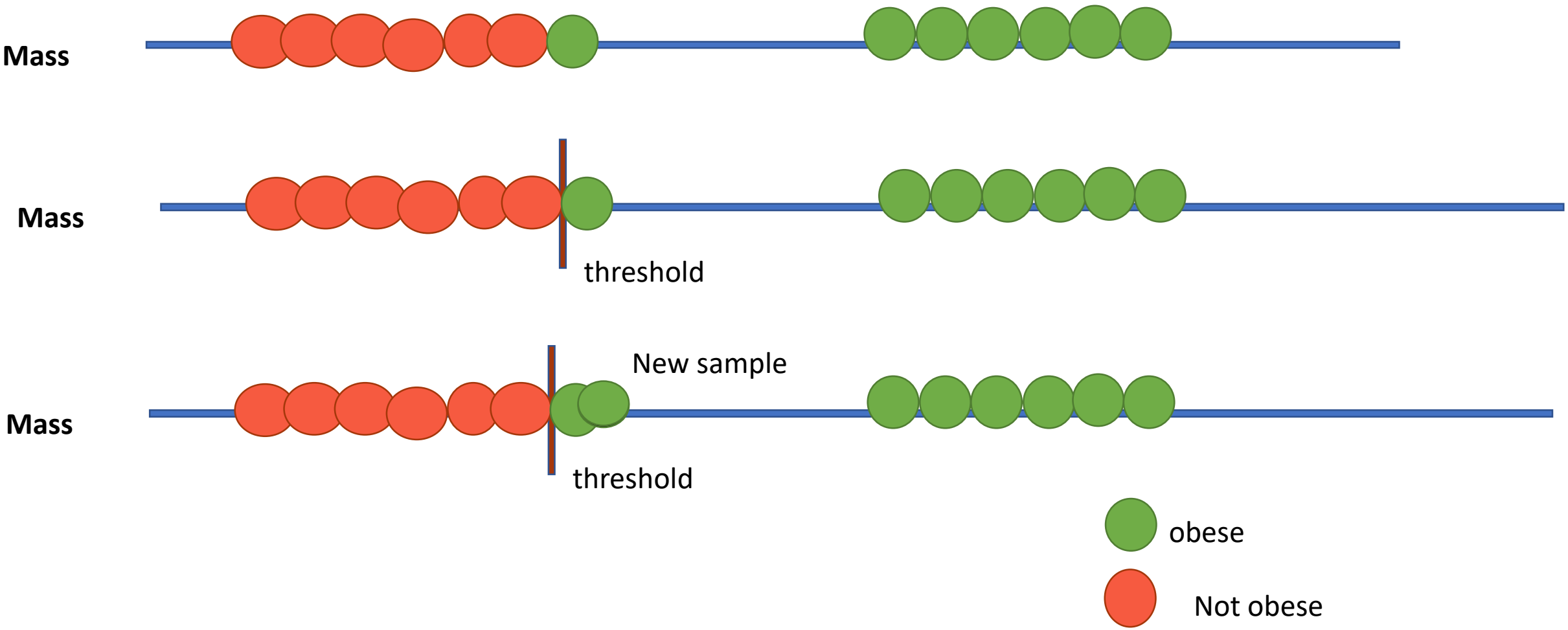
Wake me up when we get to Support Vector Machines!

Noah Mackey

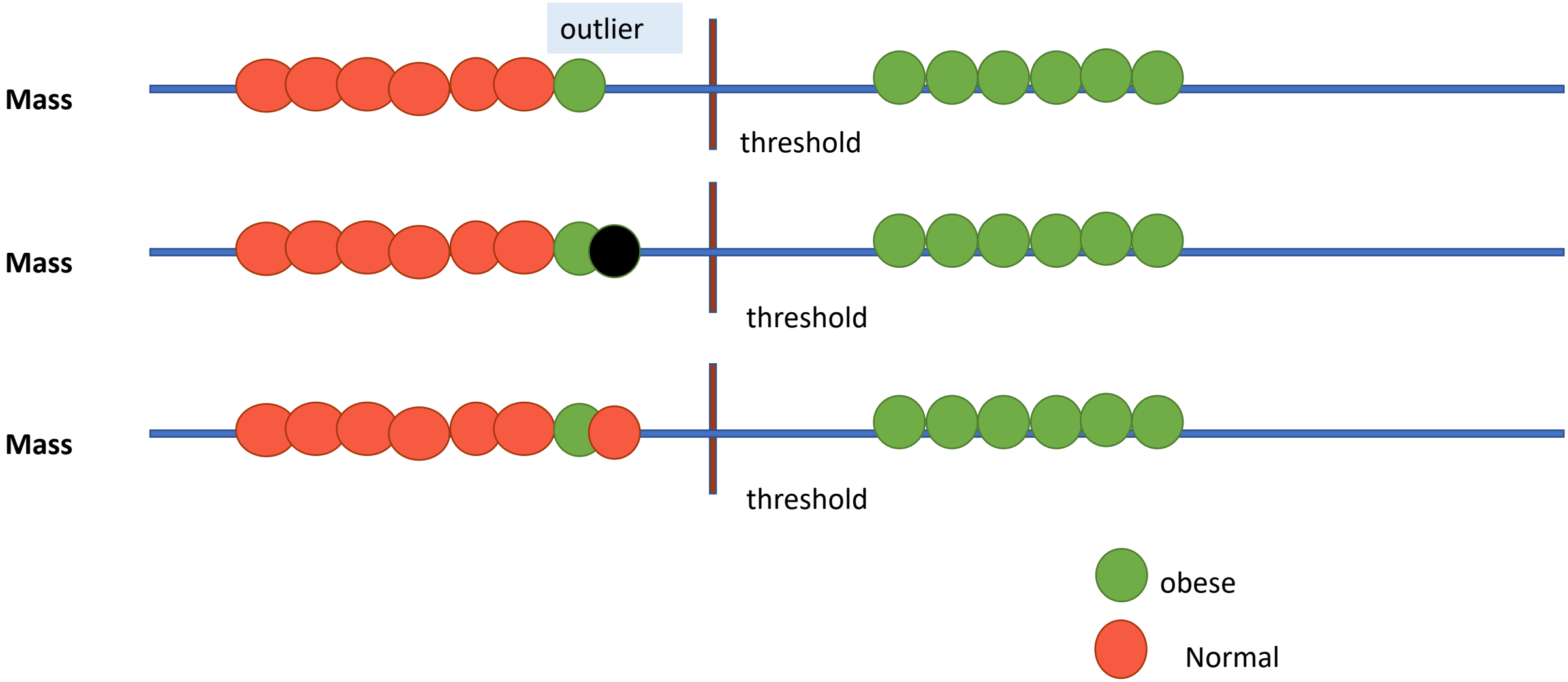
Support Vector Machines (SVM)



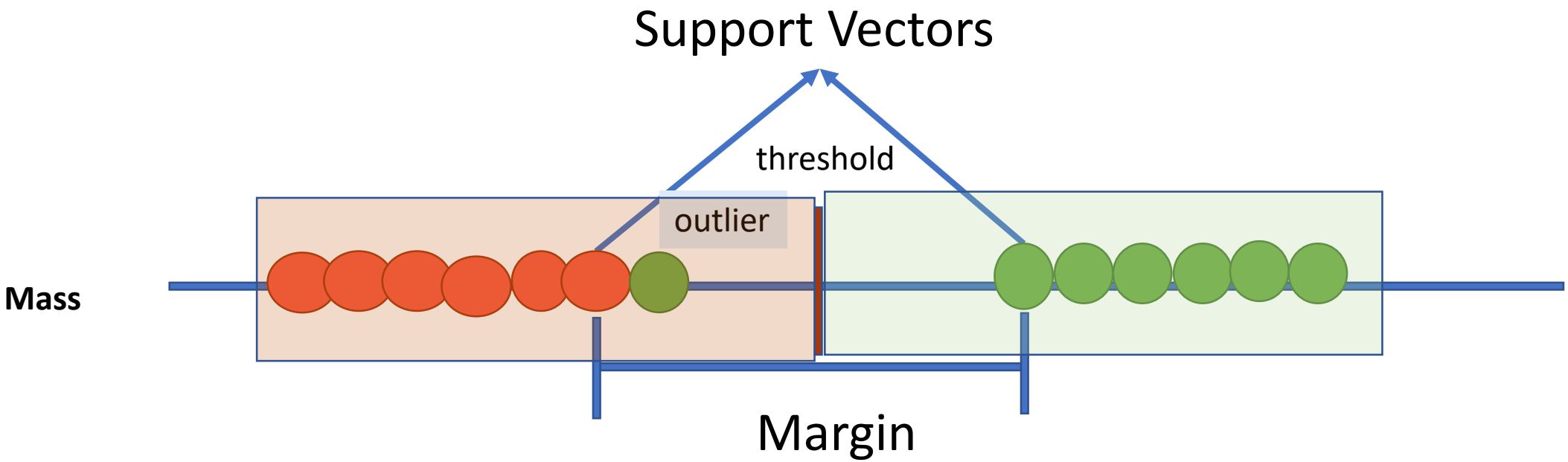
Support Vector Machines (SVM)



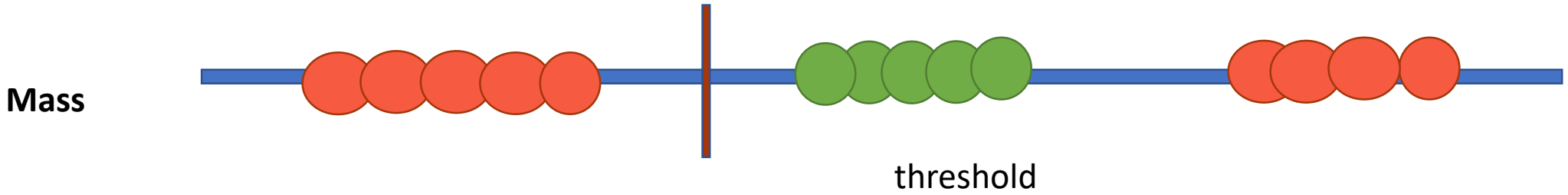
Support Vector Machines (SVM): handling outlier



Support Vector Machines (SVM): support vectors



Support Vector Machines (SVM)

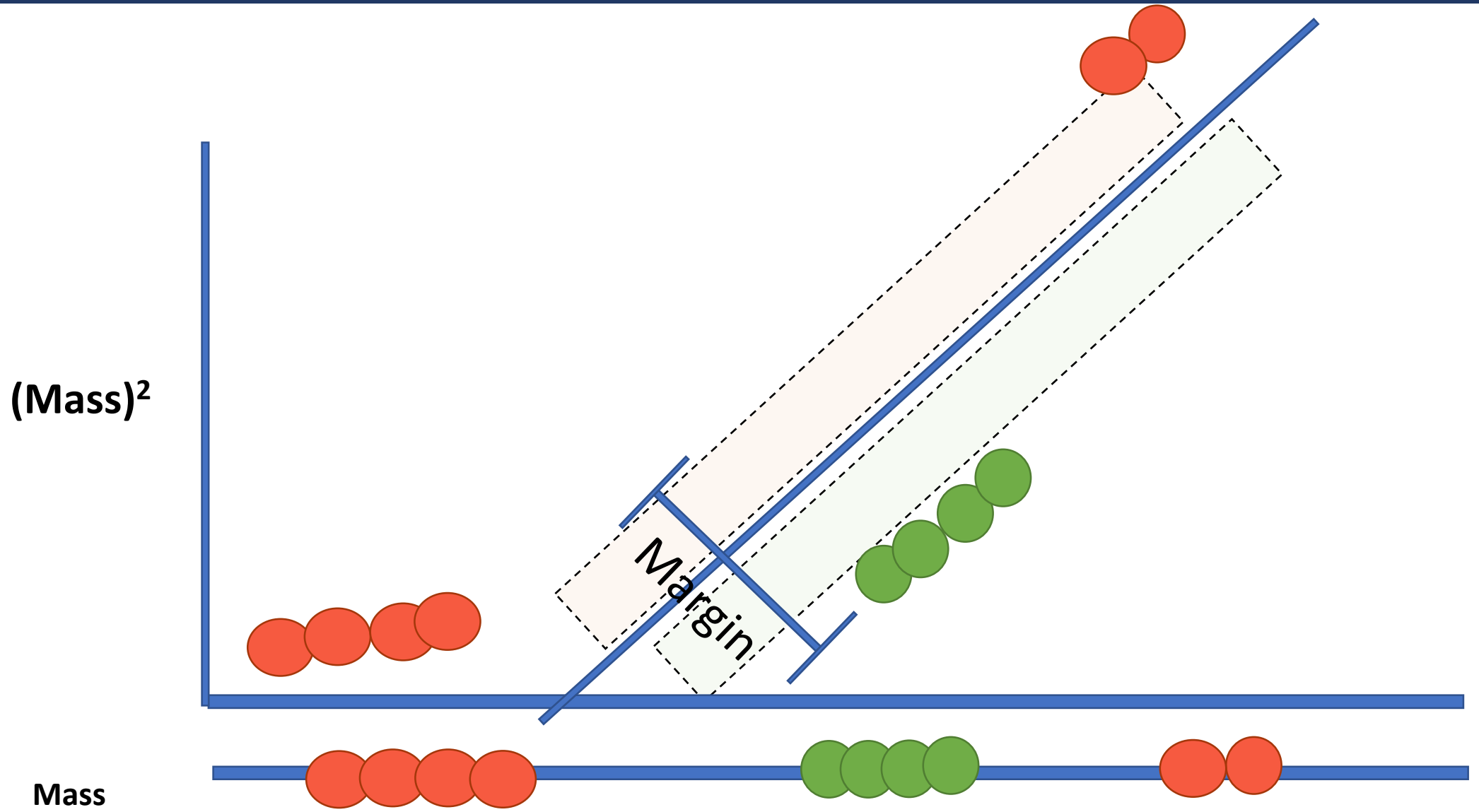


No proper threshold

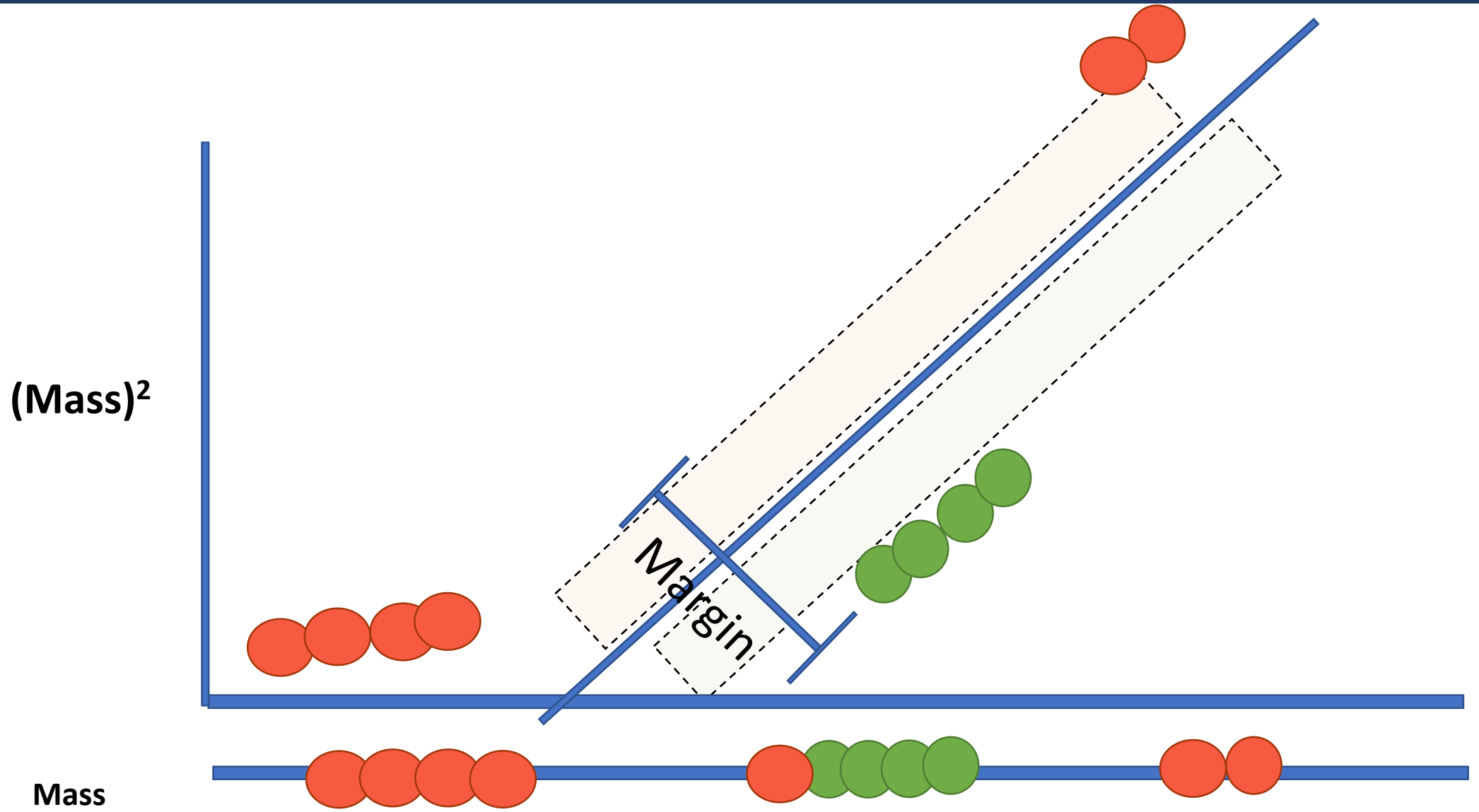
Fit BMI

Not fit BMI

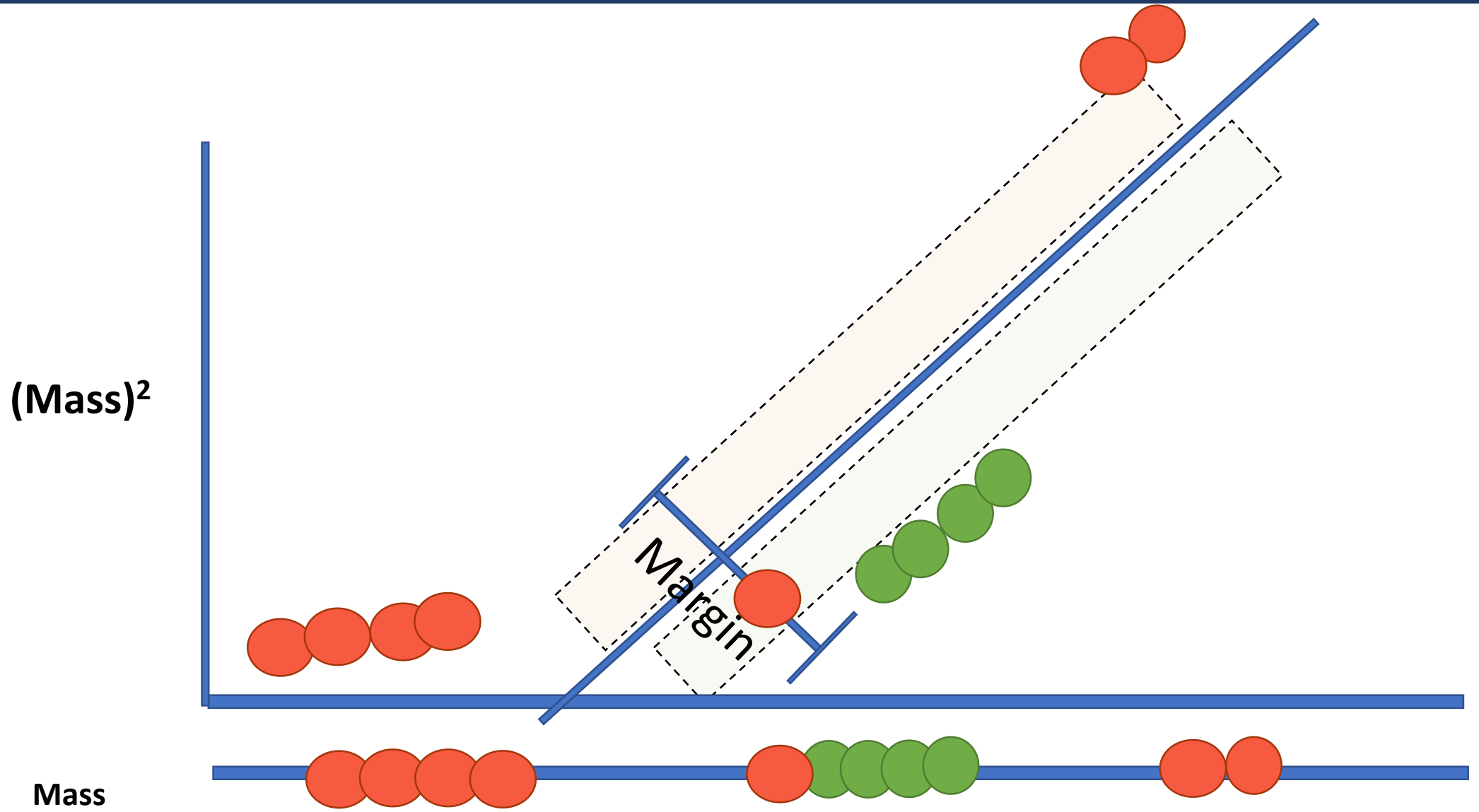
Support Vector Machines (SVM)



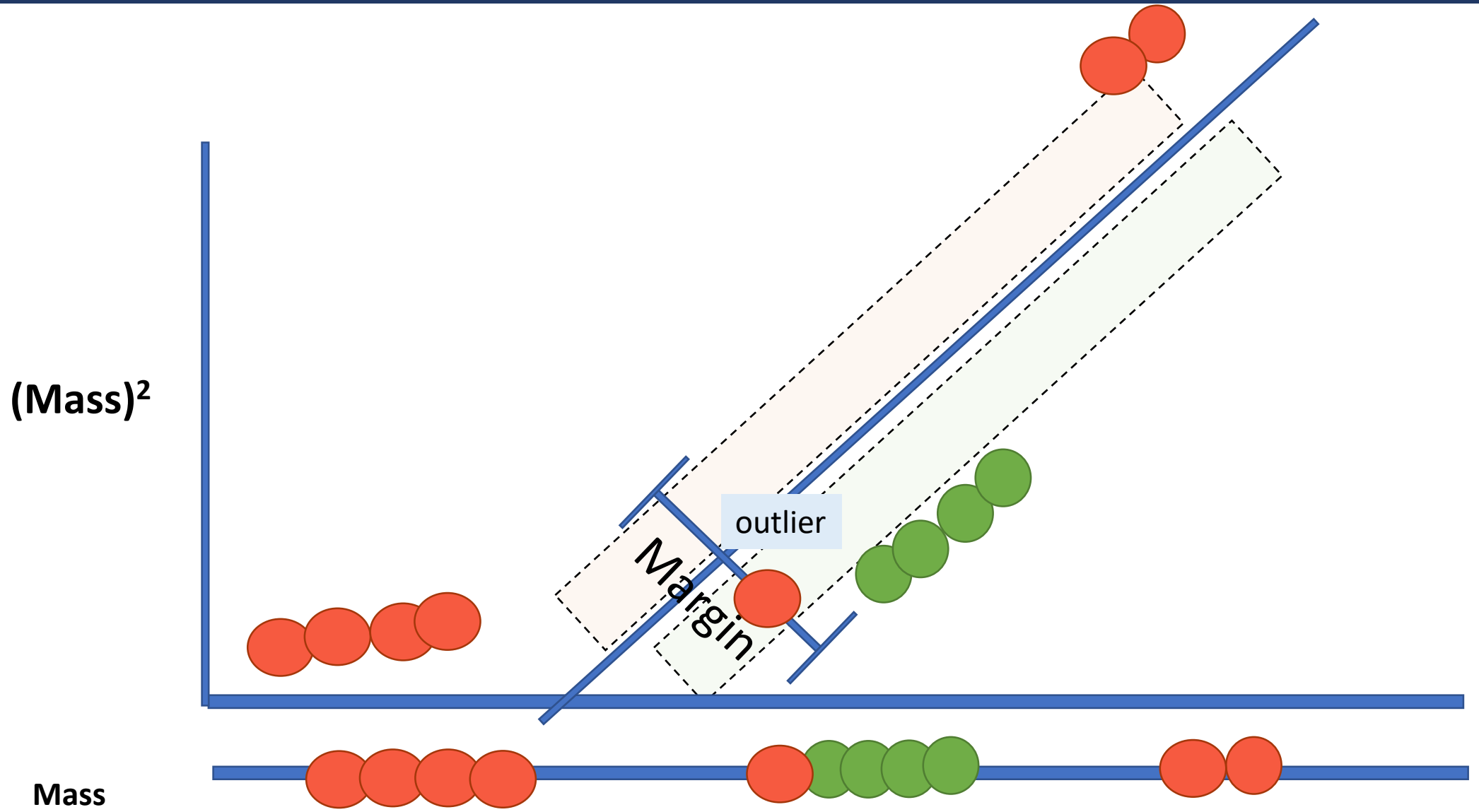
Support Vector Machines (SVM)



Support Vector Machines (SVM)



Support Vector Machines (SVM)

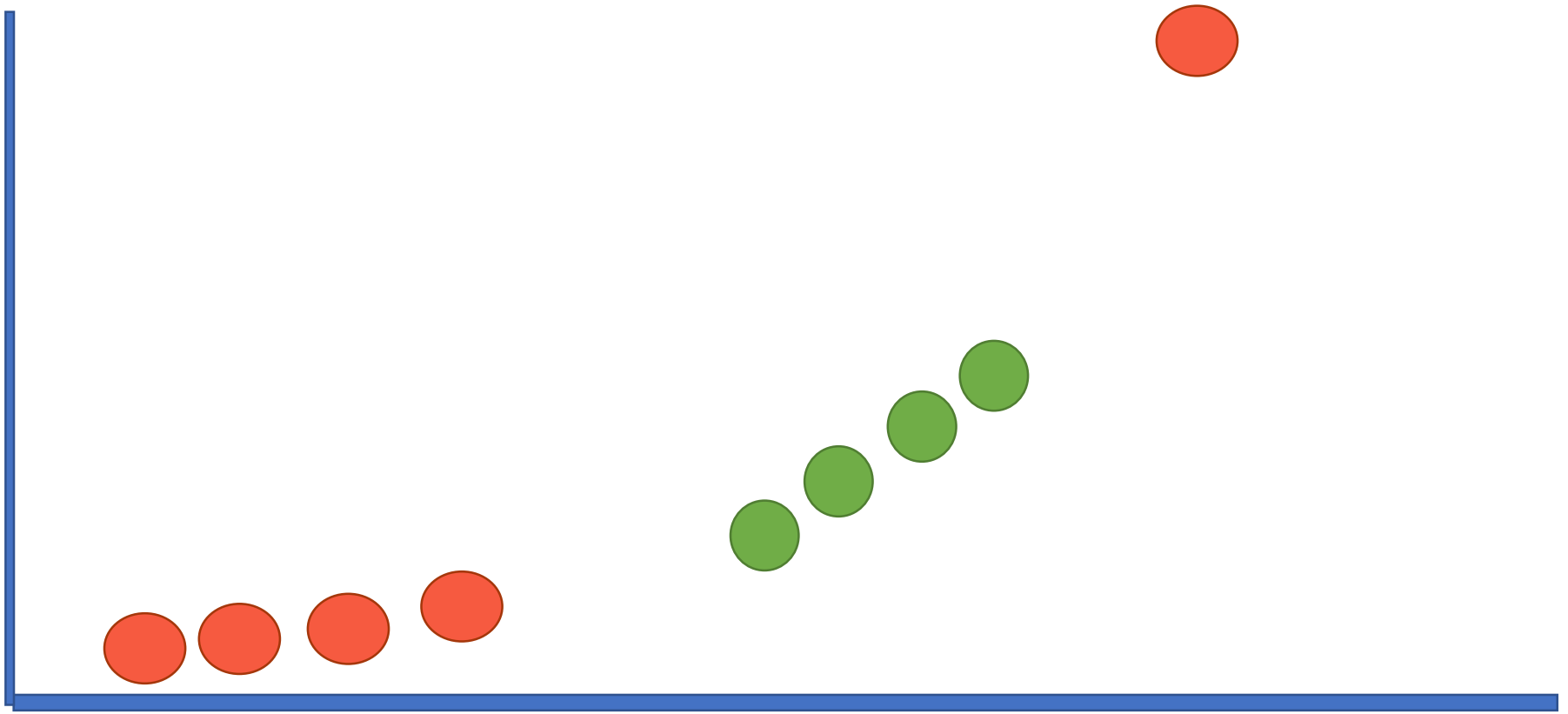


Support Vector Machines (SVM)

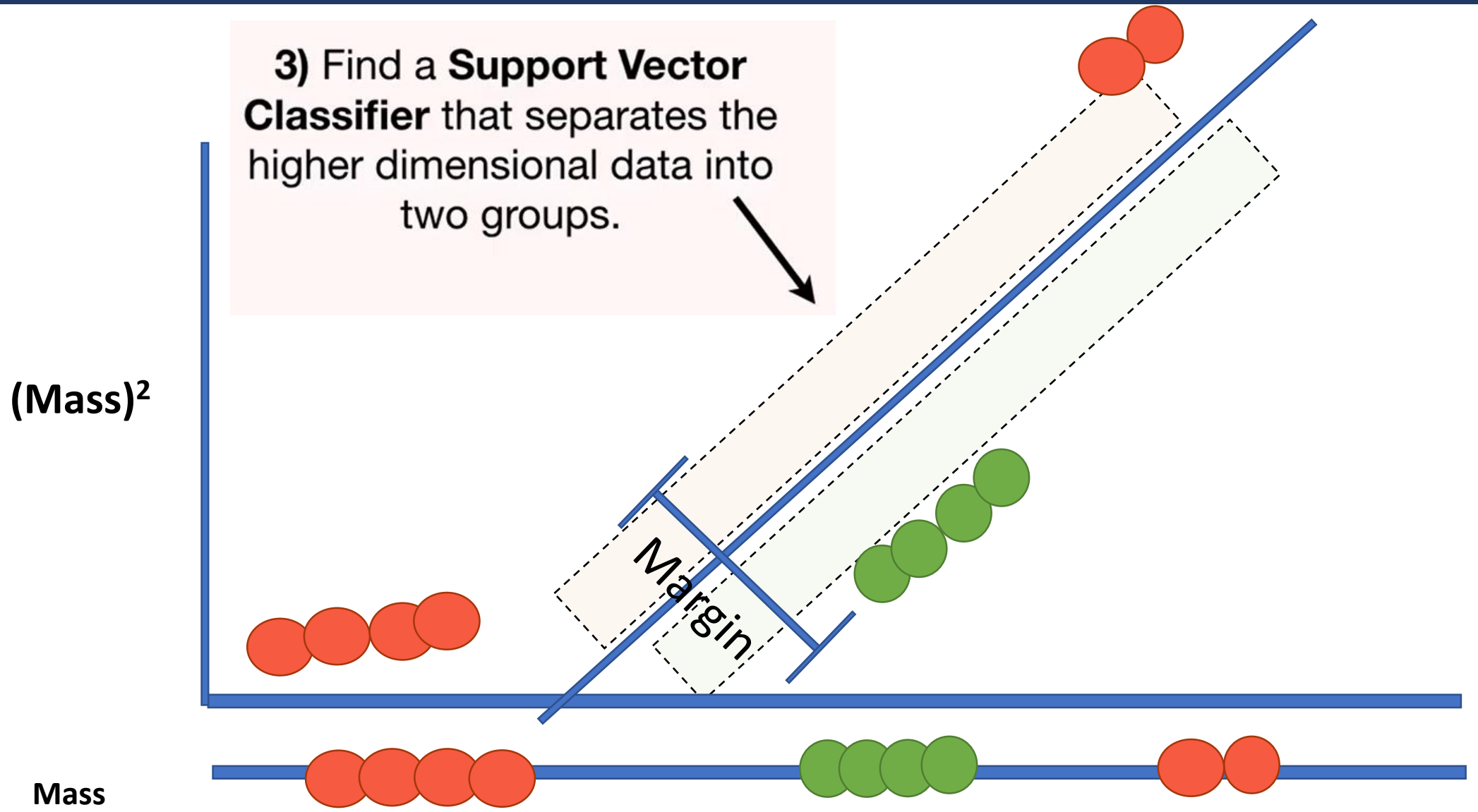
1) Start with data in a relatively low dimension…

Support Vector Machines (SVM)

2) Move the data into a higher dimension…



Support Vector Machines (SVM)

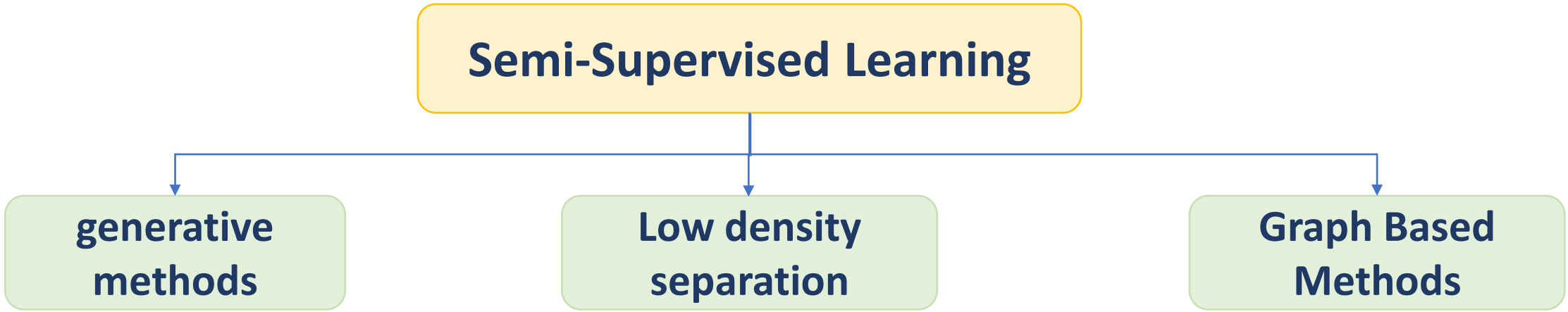


Research work: SVM

R. R. Reddy, Y. Ramadevi and K. V. N. Sunitha, "**Effective discriminant function for intrusion detection using SVM,"** 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, 2016, pp. 1148-1153

Pinpointing the intrusion from the available huge intrusion data is demanding. Intrusion detection is treated as a data analysis problem. The process of finding accurate intrusion is typical in real world scenarios. For improving accurate identification of intrusions data mining approaches are adopted and proved automatic analysis with improved performance. These techniques enhancing the detection rate of the intrusions which is very effective. Discriminant function is very critical in separating the normal and anomaly behavior accurately. The **support vector machine based classification algorithm is used to classify the intrusions accurately by using the discriminant function**. The effective discriminant function will be accurately identifies the data into intrusion and anomaly. The evaluation of the discriminant is important in the evaluation of the intrusion detection system. Performance of intrusion detection system depends on the choice of the discriminant function.

- Model is prepared based on a desired prediction problem along with **learning the structures** to organize the data as well as **make predictions**.
- Same goal as supervised learning, but in addition a set of unlabelled points is available.



The first attempts to deal with unlabelled data correspond to this area that is also denoted as Cluster-then-Label methods.

Find a decision boundary which lies in low density regions

This represents the SSC problem as a graph min-cut problem.

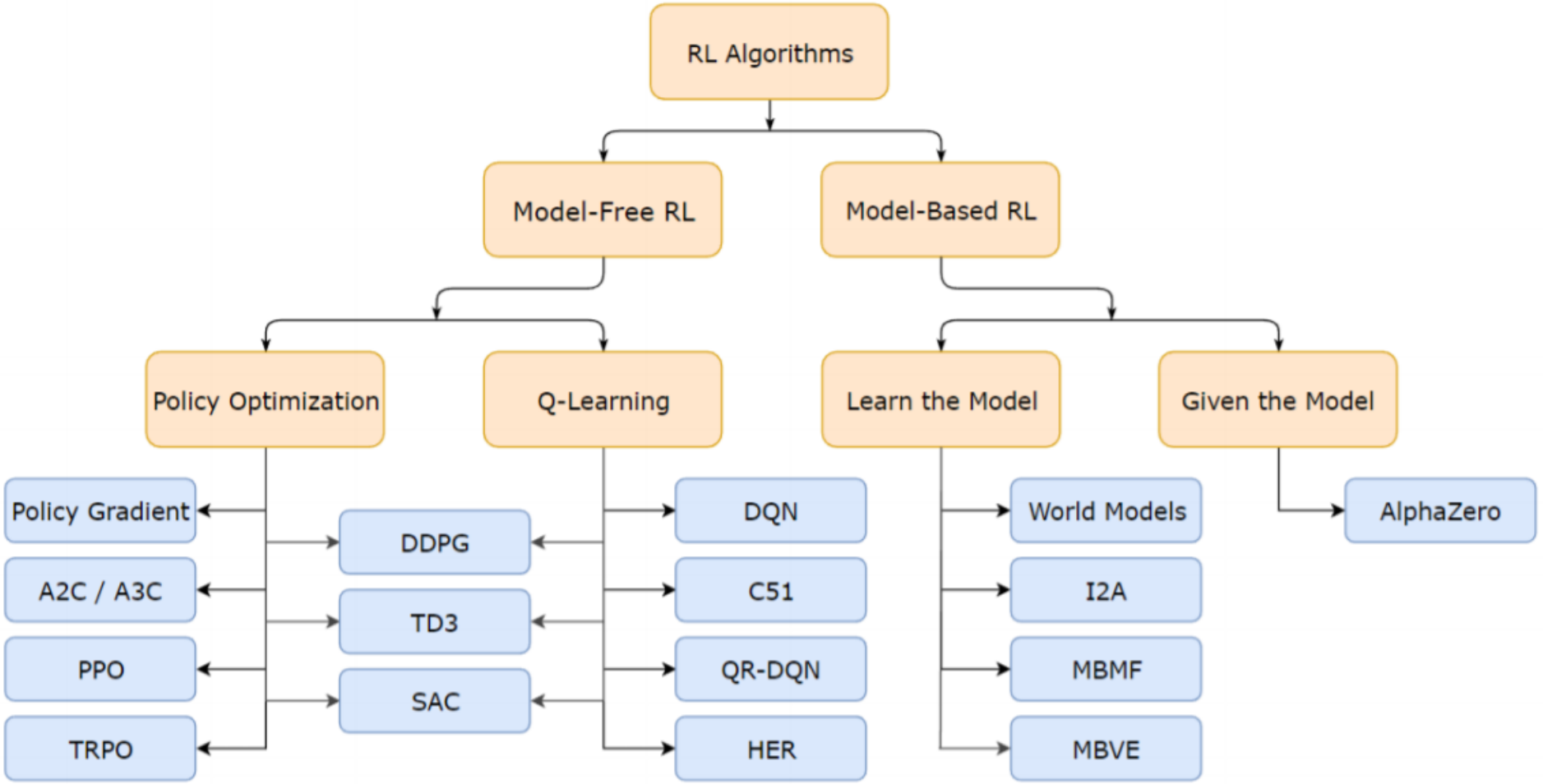
What is Reinforcement Learning ?

Reinforcement learning or RL for short is the science of decision making or the optimal way of making decisions.

Agent takes **actions**, receives observations from the environment that consists of a **reward** for his action and information of his new state. That reward informs the agent of how **good** or **bad** was the taken action, and the observation tells him what is his next **state** in the environment.

Source: Madhu Sanjeevi (medium)

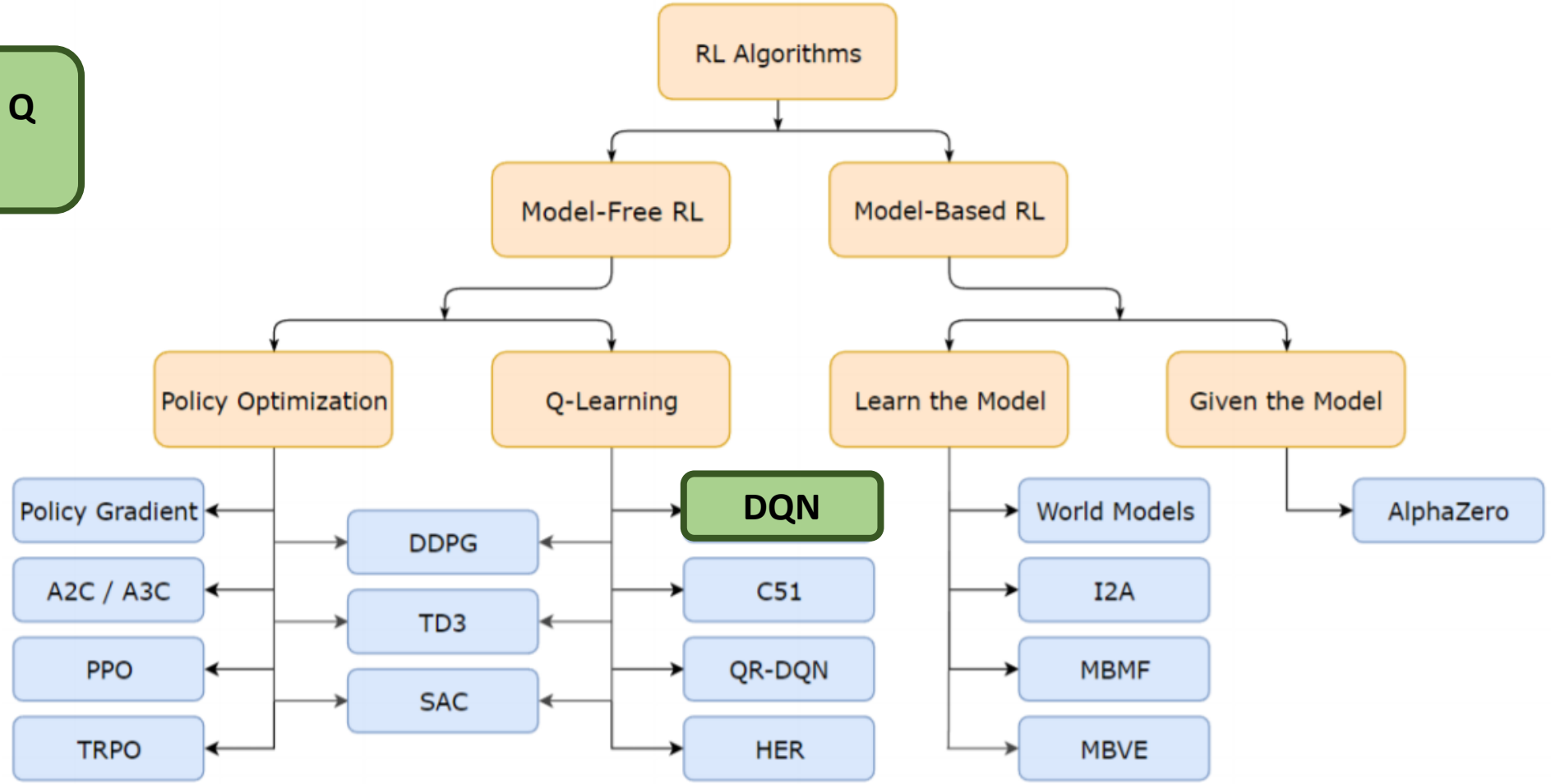
Taxonomy of RL



Source: https://deeplearning.mit.edu

Taxonomy of RL

DQN: Deep Q Network



Source: https://deeplearning.mit.edu

Timeline of RL in security and DQN

RL in Security ...

2000

Cannady J. et. al.

"Next generation intrusion detection: Autonomous reinforcement learning of network attacks "

2005

X. Xu.

"A reinforcement learning approach for host-based intrusion detection using sequences of system calls",

2008

A. Servin.

"Multi-agent Reinforcement Learning for Intrusion Detection,"

2015

K. Malialis

"Distributed reinforcement learning for adaptive and robust network intrusion response,"

Evolution of DQN..

1950s

R. Bellman

Introduced the concept of the term "optimal control" he has also introduced the concept of dynamic programming, Bellman-Ford algorithm,(1)

2005

Martin Riedmiller

Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method

2015

V Minh.

"Human-level control through deep reinforcement learning"

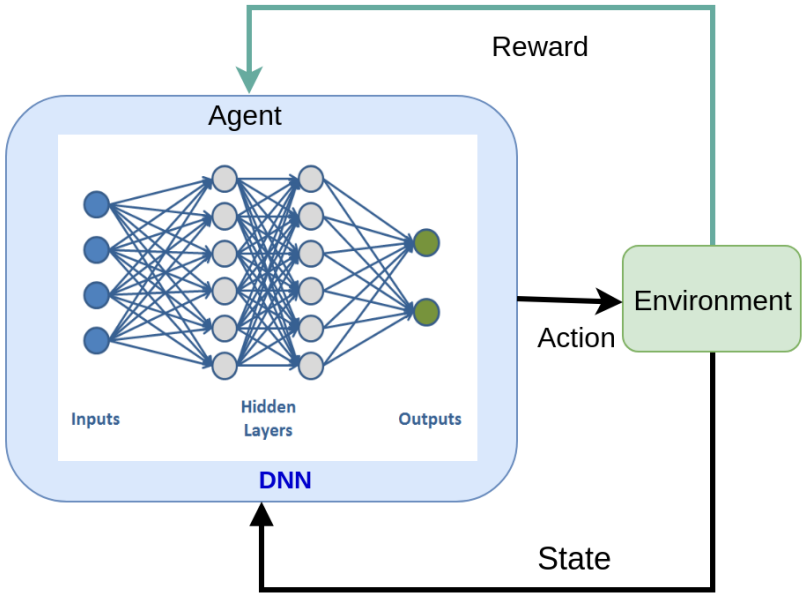
DQN for Atari games

LETTER

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]



Volodymyr Mnih,
research scientist at Google DeepMind

Algorithm for Q-value calculation

Algorithm:

Start with $Q_0(s, a)$ for all s, a.

Get initial state s

For k = 1, 2, ... till convergence

Sample action a, get next state s'

If s' is terminal:

$$\text{target} = R(s, a, s')$$

Sample new initial state s'

else:

$$\text{target} = R(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$Q_{k+1}(s, a) \leftarrow (1 - \alpha) Q_k(s, a) + \alpha [\text{target}]$$

$$s \leftarrow s'$$

Source: Deep RL Bootcamp (UC Berkeley)

Q-value calculation (given state)

Game Board:

Current state (*s*): 0 0 0
0 1 0

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	0.4	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Source:Shaked Zychlinski (https://towardsdatascience.com)

Q-value calculation (q-value of action)

Game Board:

Current state (s): 0 0 0 / 0 1 0

Selected action (a): →

Q Table: $\gamma = 0.95$

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	0.4	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Source:Shaked Zychlinski (https://towardsdatascience.com)

Q-value calculation (upgrading the table)

Game Board:

Current state (*s*): **0 0 0 / 0 1 0**

Selected action (*a*): →

Reward (*r*): **0**

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	0.4	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Source:Shaked Zychlinski (https://towardsdatascience.com)

Q-value calculation (next state a/c to action)

Game Board:

Current state (s): 0 0 0 / 0 1 0

Selected action (a): ⇨

Reward (r): 0

Next state (s'): 0 0 0 / 0 0 1

Q Table: $\gamma = 0.95$

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	0.4	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Source:Shaked Zychlinski (https://towardsdatascience.com)

Q-value calculation (q-value of best action)

Game Board:

Current state (*s*): 0 0 0 / 0 1 0

Selected action (*a*): ⇨

Reward (*r*): 0

Next state (*s'*): 0 0 0 / 0 0 1

max Q(s'): 1.0

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	0.4	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Source:Shaked Zychlinski (https://towardsdatascience.com)

Q-value calculation (finding the new q-value)

Game Board:

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	0.4	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Current state (*s*): **0 0 0 / 0 1 0**

Selected action (*a*): →

Reward (*r*): **0**

Next state (*s'*): **0 0 0 / 0 0 1**

max Q(s'): **1.0**

→ New Q(s,a) = r + γ * maxQ(s') = 0 + 0.95 * 1 = **0.95**

Source:Shaked Zychlinski (https://towardsdatascience.com)

Q-value calculation (upgrading the table)

Game Board:

Q Table: $\gamma = 0.95$

	000 100	000 010	000 001	100 000	010 000	001 000
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	**0.95**	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Current state (s): **0 0 0 / 0 1 0**

Selected action (a): →

Reward (r): **0**

Next state (s'): **0 0 0 / 0 0 1**

max Q(s'): **1.0**

→ New Q(s,a) = r + γ * maxQ(s') = 0 + 0.95 * 1 = **0.95**

Source:Shaked Zychlinski (https://towardsdatascience.com)

Q-valued RL vs DQN

$$Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha\,[\text{target}]$$

Table Q-valued RL

Algorithm:

Start with $Q_0(s,a)$ for all s, a.

Get initial state s

For k = 1, 2, ... till convergence

Sample action a, get next state s'

If s' is terminal:

$$\text{target} = R(s,a,s')$$

Sample new initial state s'

else:

$$\text{target} = R(s,a,s') + \gamma \max_{a'} Q_k(s',a')$$

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \mathbb{E}_{s' \sim P(s'|s,a)} \left[(Q_\theta(s,a) - \text{target}(s'))^2 \right] \Big|_{\theta=\theta_k}$$

$$s \leftarrow s'$$

Q-value based RL + DL =DQN

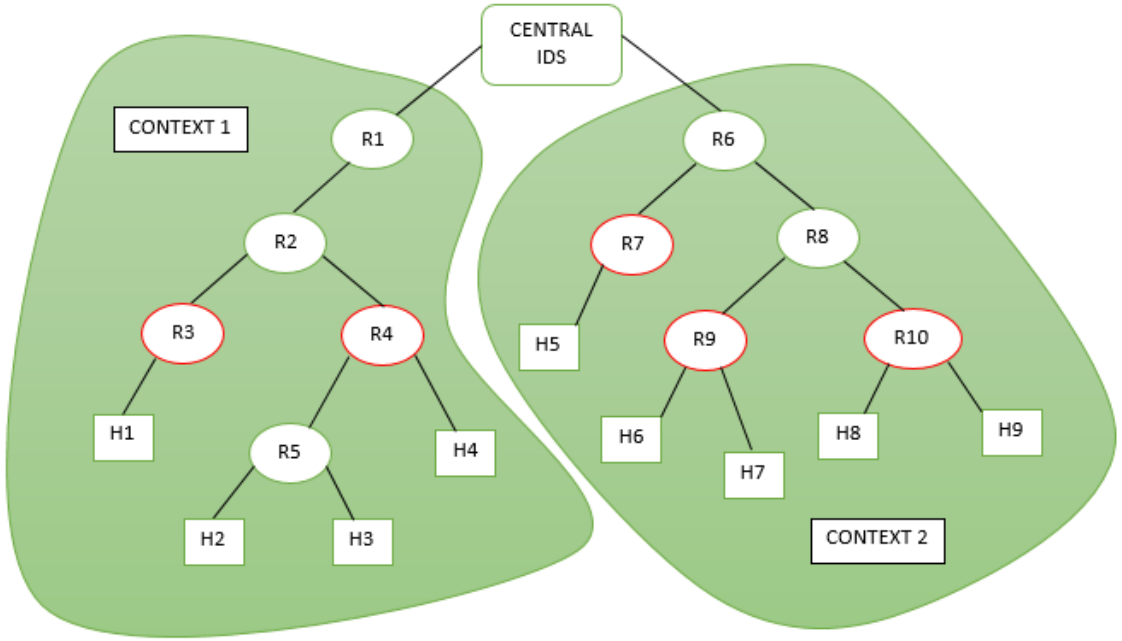
Source: Deep RL Bootcamp (UC Berkeley)

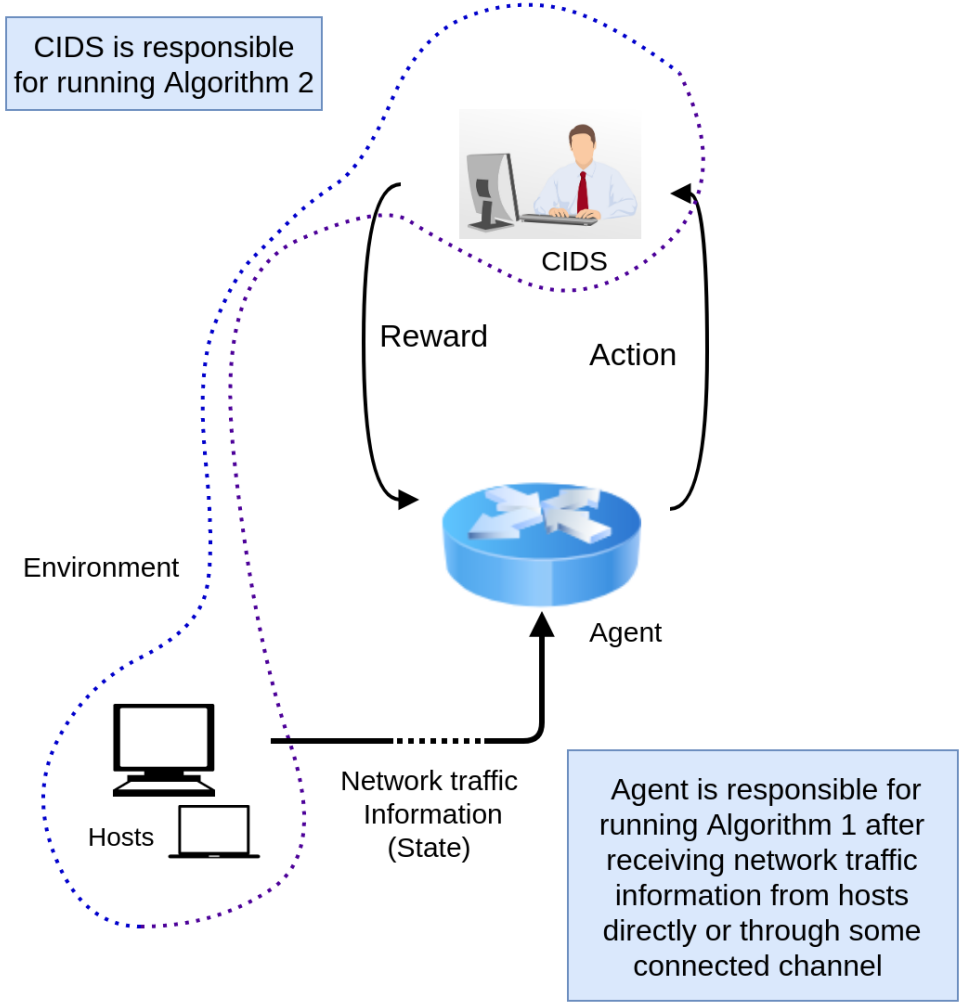
Research work: Deep RL

K Sethi, ES Rupesh, **R Kumar**, P Bera, YV Madhav, "**A context-aware robust intrusion detection system: a reinforcement learning-based approach**“ International Journal of Information Security, 1-22

Detection and prevention of intrusions in enterprise networks and systems is an important, but challenging problem due to extensive growth and usage of networks that are constantly facing novel attacks. An intrusion detection system (IDS) monitors the network traffic and system-level applications to detect malicious activities in the network. However, most of the existing IDSs are incapable of providing higher accuracy and less false positive rate (FPR). Therefore, there is a need for adaptive techniques to detect network intrusions that maintain a balance between accuracy and FPR. In this paper, we present **a context-adaptive IDS that uses multiple independent deep reinforcement learning agents distributed across the network for accurate detection and classification of new and complex attacks.** We have done extensive experimentation using three benchmark datasets including **NSL-KDD, UNSW-NB15 and AWID** on our model that shows better accuracy and less FPR compared to the state-of-the-art systems. Further, we analyzed the robustness of our model **against adversarial attack** and observed only a small decrease in accuracy as compared to the existing models. To further improve the robustness of the system, we implemented the concept of **denoising autoencoder (DAE)**. Also, we have shown the usability of our system in real-life application with changes in the attack pattern

Network Deployment of model

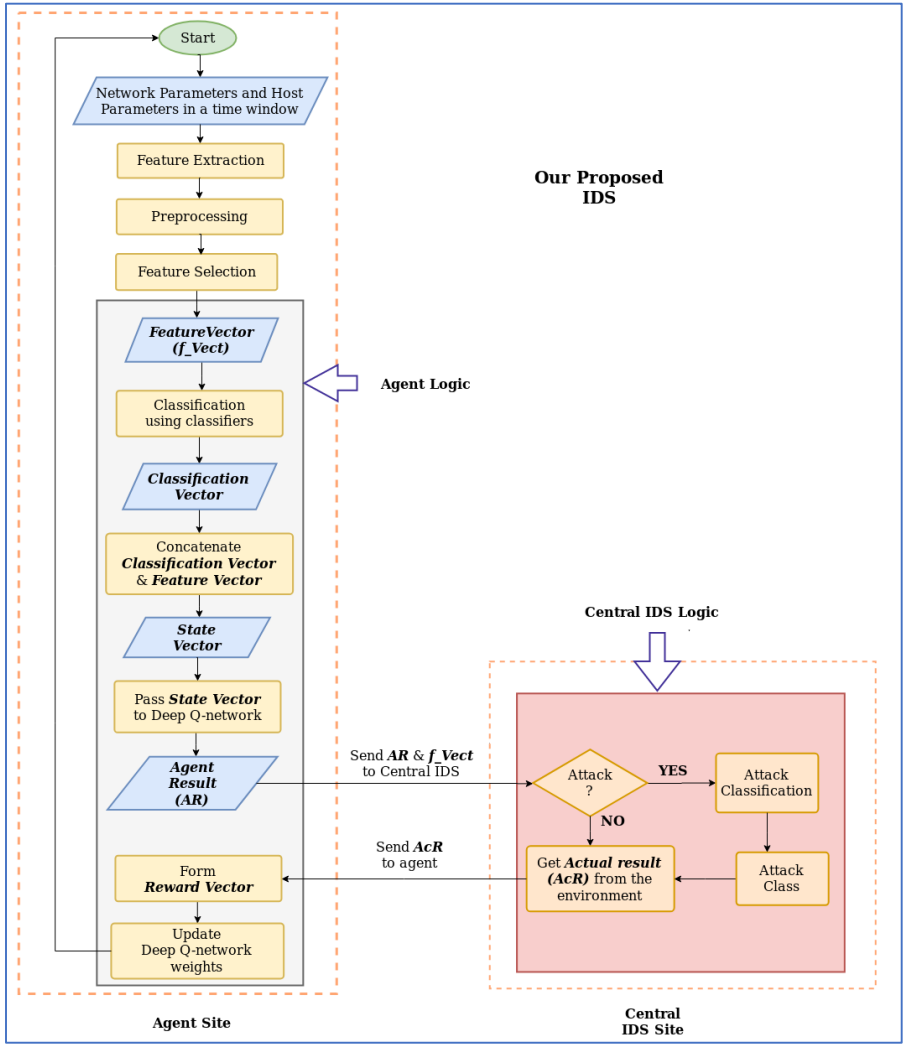




Algorithm and Workflow

Algorithm 1: Agent Logic

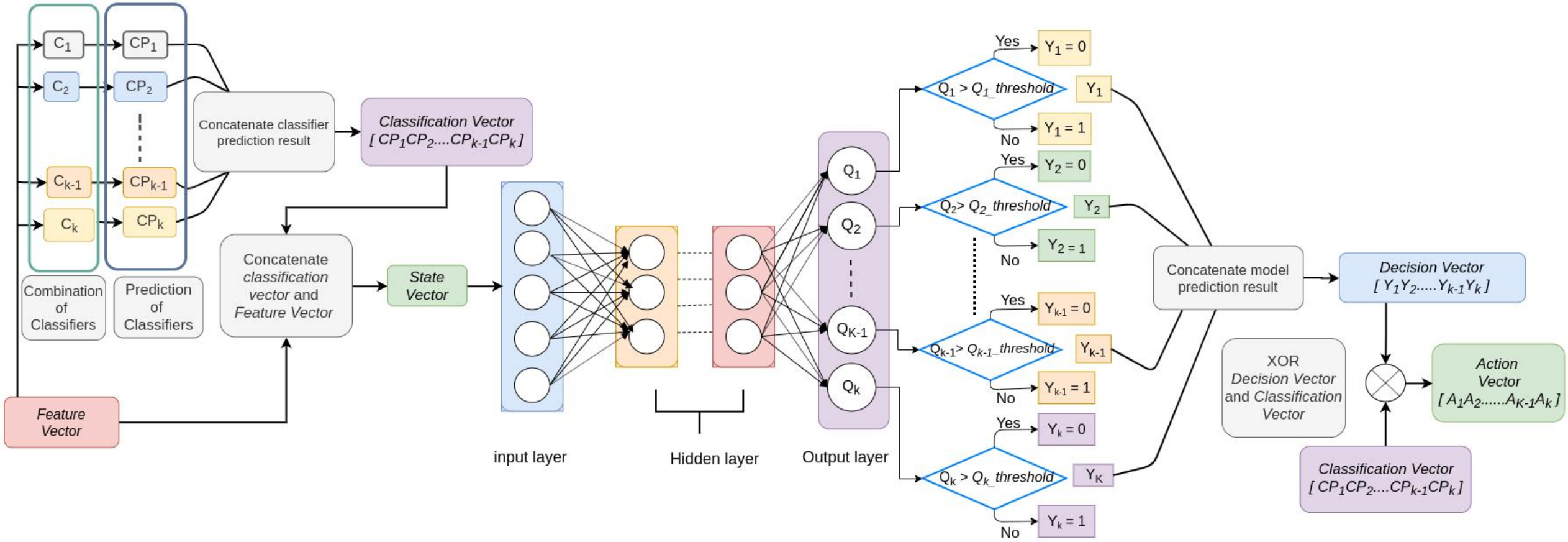
Initialize memory M;
Get the *feature_set* using the network parameters;
for *each feature_vector in feature_set* **do**
    Input the *feature_vector* to different classifiers C_1,C_2...C_n and store the output as *confidence_vector*;
    Run thresholding on *confidence_vector* and generate *classification_vector*;
    Form the *state_vector* using *feature_vector* and *classification_vector*;
    Input the *state_vector* to the Deep Q-network and store the output as *Q_value_vector*;
    Run thresholding on *Q_value_vector* and generate *decision_vector*;
    calculate the *action_vector*;
    Get the *actual_result* from the central IDS;
    Initialize *reward_vector*;
    for *i in range(len(sign_vector))* **do**
        calculate *reward_vector[i]*;
        Update Q values using the following equation:
        $Q_{\text{new}}(s,a,r,w) = reward\_vector[i] + \gamma * Q_{\text{old}}(s,a,r,w)$;
    end
    Store the transition in memory M;
    Update Deep Q-network weights (w) and train the model classifiers using newly added transitions;
    for *j in range(n)* **do**
        Update Deep Q-network weights (w) using random sample of transitions from M;
    end
end



Algorithm 1: Central IDS Logic

Get *agent_result* and *feature_vector* from the agents;
Group the agents according to context;
for *each context* **do**
    for *each agent* **do**
        evaluated the status of the attack from agent_result;
        if *(status == "attack")* **then**
            pre-process feature vector for attack classification;
            Input processed feature vector to classifier and get the attack type
            attack_type = output of classifier;
        else
            attack_type = "*normal*";
        end
        Get the *actual_result* from the environment;
        Send the *actual_result* to the agent for use in calculation of rewards;
    end
end

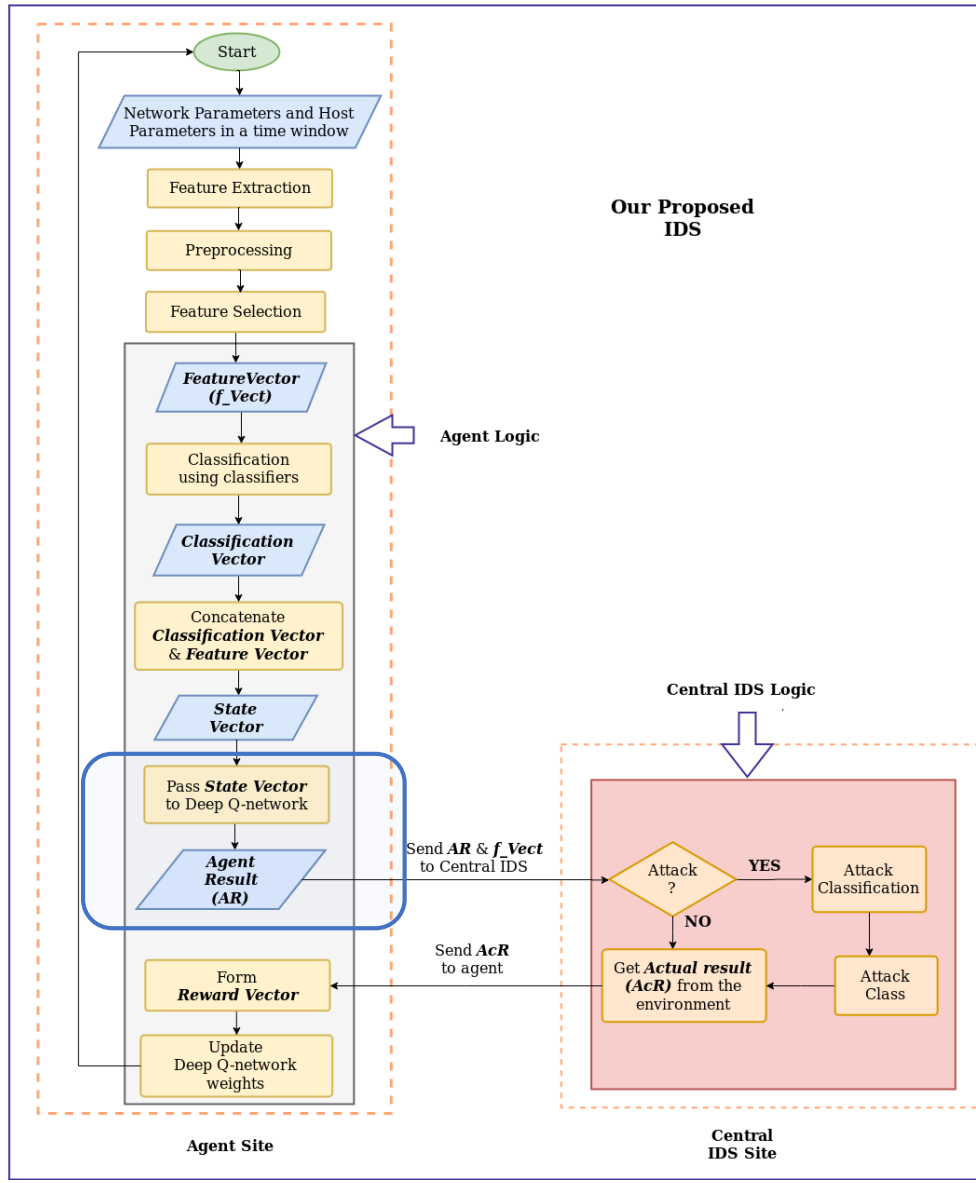
Agent Result Calculation

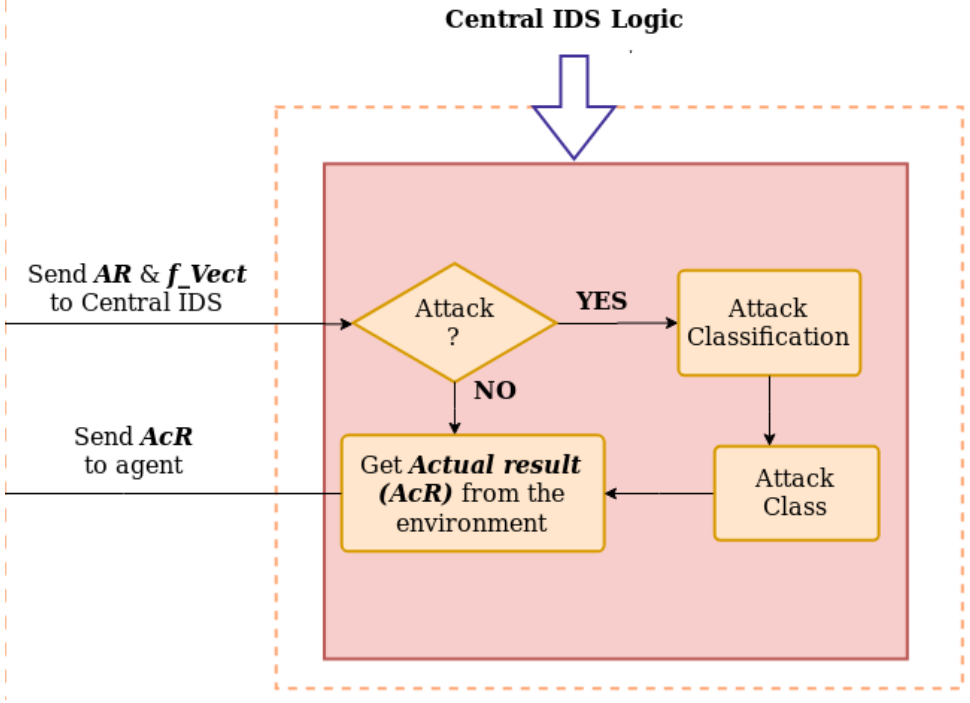


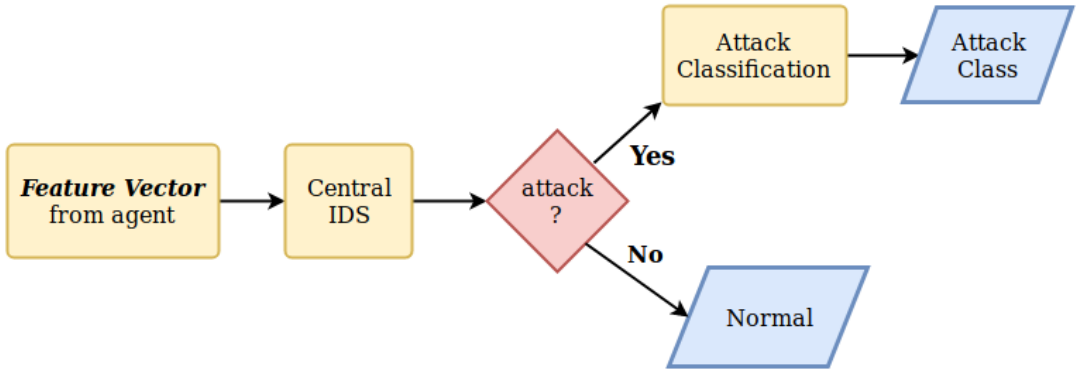
Algorithm 1: Agent Logic

```
Initialize memory M;
Get the feature_set using the network parameters;
for each feature_vector in feature_set do
    Input the feature_vector to different classifiers C1,C2...Cn and store
     the output as confidence_vector;
    Run thresholding on confidence_vector and generate
     classification_vector;
    Form the state_vector using feature_vector and
     classification_vector;
    Input the state_vector to the Deep Q-network and store the output
     as Q_value_vector;
    Run thresholding on Q_value_vector and generate decision_vector;
    calculate the action_vector;
    Get the actual_result from the central IDS;
    Initialize reward_vector;
    for i in range(len(sign_vector)) do
        calculate reward_vector[i];
        Update Q values using the following equation:
        Q_new(s, a, r, w) = reward_vector[i] + γ * Q_old(s, a, r, w);
    end
    Store the transition in memory M;
    Update Deep Q-network weights (w) and train the model classifiers
     using newly added transitions;
    for j in range(n) do
        Update Deep Q-network weights (w) using random sample of
         transitions from M;
    end
end
```

Agent result Calculation







Environment

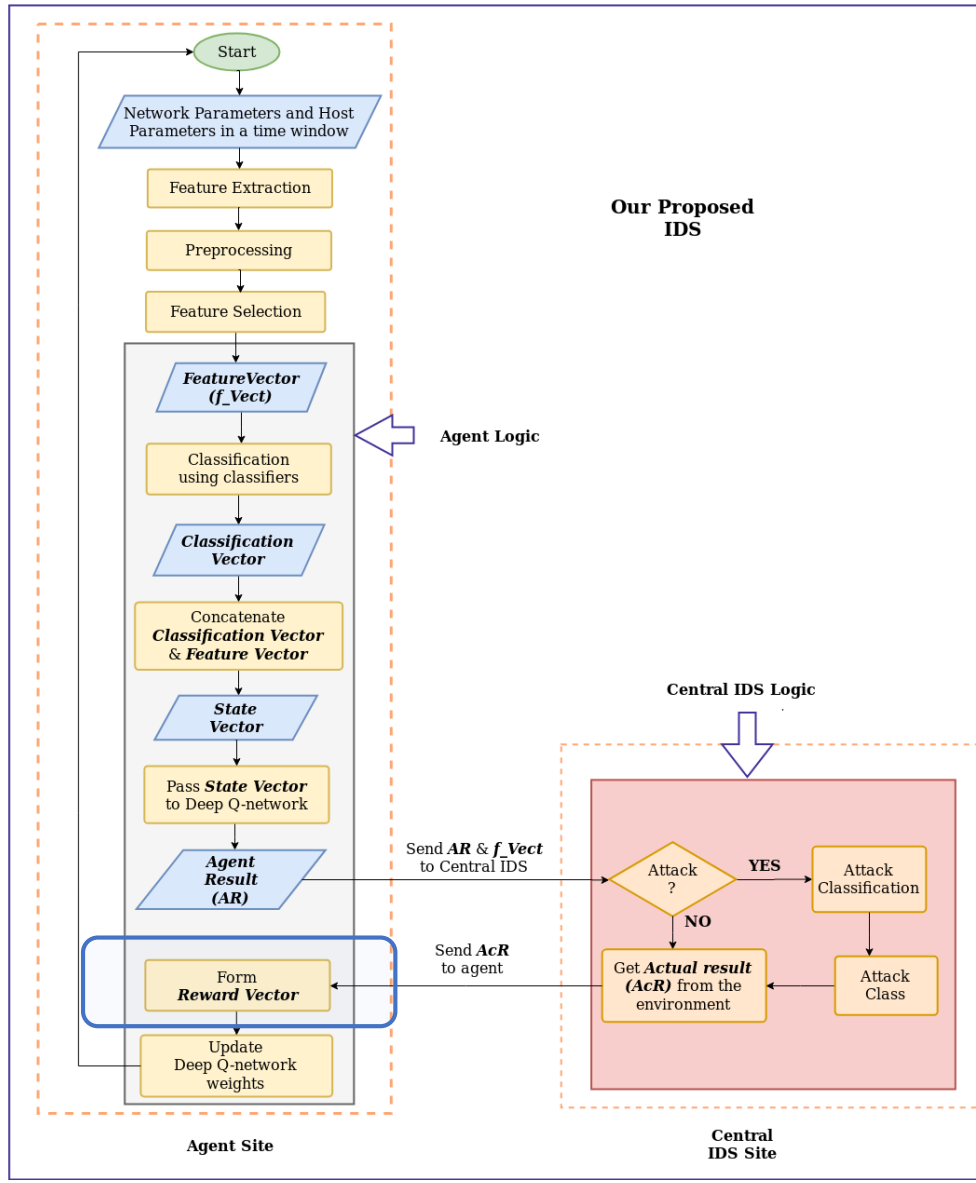
Algorithm 1: Central IDS Logic

```
Get agent_result and feature_vector from the agents;
Group the agents according to context;
for each context do
    for each agent do
        evaluated the status of the attack from agent_result;
        if (status == "attack") then
            pre-process feature vector for attack classification;
            Input processed feature vector to classifier and get the attack
              type
            attack_type = output of classifier;
        else
            attack_type = "normal";
        end
        Get the actual_result from the environment;
        Send the actual_result to the agent for use in calculation of
          rewards;
    end
end
```

Algorithm 1: Agent Logic

```
Initialize memory M;
Get the feature_set using the network parameters;
for each feature_vector in feature_set do
    Input the feature_vector to different classifiers C1,C2...Cn and store
     the output as confidence_vector;
    Run thresholding on confidence_vector and generate
     classification_vector;
    Form the state_vector using feature_vector and
     classification_vector;
    Input the state_vector to the Deep Q-network and store the output
     as Q_value_vector;
    Run thresholding on Q_value_vector and generate decision_vector;
    calculate the action_vector;
    Get the actual_result from the central IDS;
    Initialize reward_vector;
    for i in range(len(sign_vector)) do
        calculate reward_vector[i];
        Update Q values using the following equation:
        Q_new(s, a, r, w) = reward_vector[i] + γ * Q_old(s, a, r, w);
    end
    Store the transition in memory M;
    Update Deep Q-network weights (w) and train the model classifiers
     using newly added transitions;
    for j in range(n) do
        Update Deep Q-network weights (w) using random sample of
         transitions from M;
    end
end
```

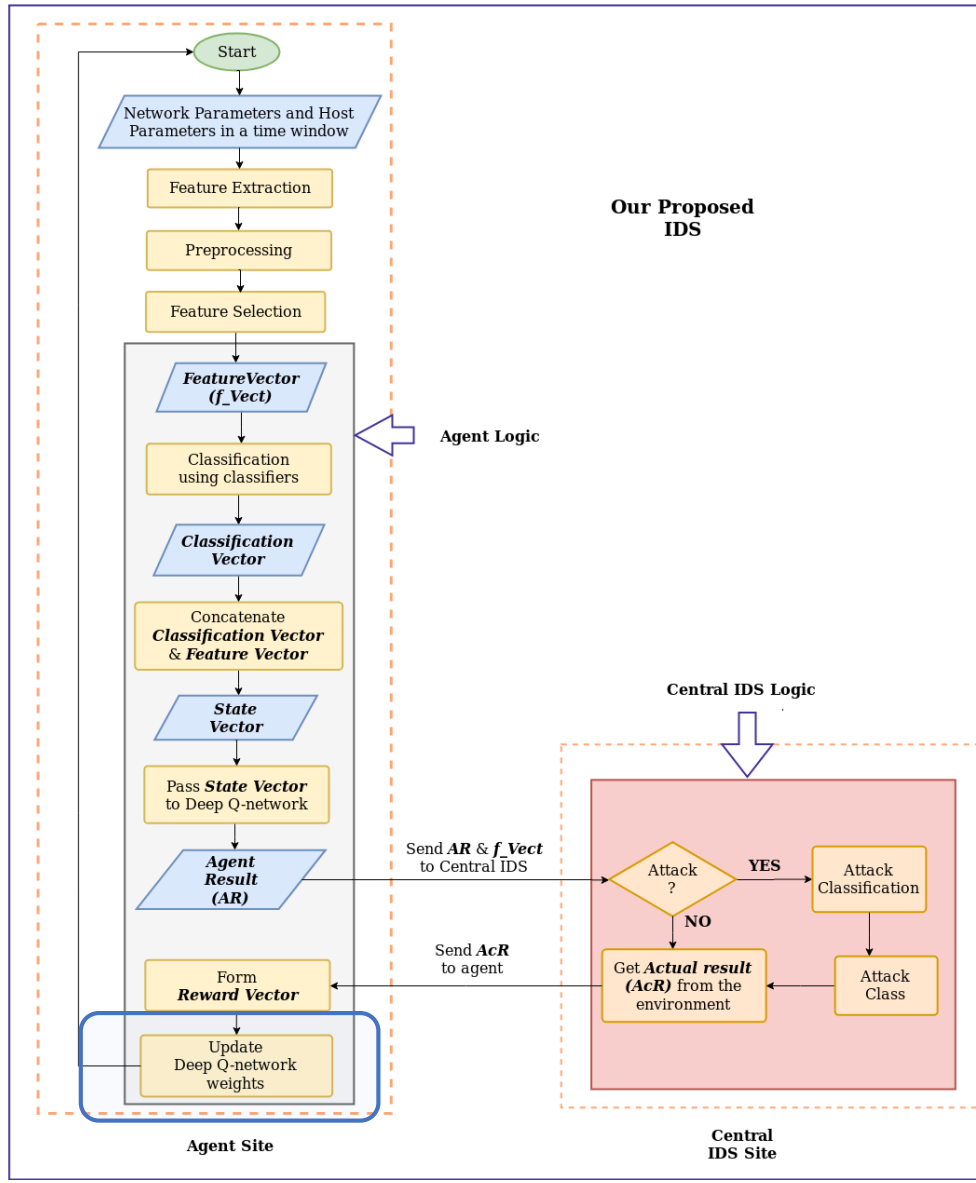
Reward Calculation



Algorithm 1: Agent Logic

```
Initialize memory M;
Get the feature_set using the network parameters;
for each feature_vector in feature_set do
    Input the feature_vector to different classifiers C1,C2...Cn and store
     the output as confidence_vector;
    Run thresholding on confidence_vector and generate
     classification_vector;
    Form the state_vector using feature_vector and
     classification_vector;
    Input the state_vector to the Deep Q-network and store the output
     as Q_value_vector;
    Run thresholding on Q_value_vector and generate decision_vector;
    calculate the action_vector;
    Get the actual_result from the central IDS;
    Initialize reward_vector;
    for i in range(len(sign_vector)) do
        calculate reward_vector[i];
        Update Q values using the following equation:
        Q_new(s, a, r, w) = reward_vector[i] + γ * Q_old(s, a, r, w);
    end
    Store the transition in memory M;
    Update Deep Q-network weights (w) and train the model classifiers
     using newly added transitions;
    for j in range(n) do
        Update Deep Q-network weights (w) using random sample of
         transitions from M;
    end
end
```

Experience Replay



Performance of individual classifier

Table : Individual classifier performance

Classifier	NSL-KDD			UNSW-NB15			AWID		
	Accuracy (%)	FPR (%)	AUC	Accuracy (%)	FPR (%)	AUC	Accuracy (%)	FPR (%)	AUC
RF	77.95	2.66	0.8054	82.12	12.32	0.79	95.38	3.1	0.792
ADB	75.65	7.009	0.7763	85.61	10.60	0.843	94.16	7.8	0.705
GNB	66	1.688	0.7056	70.62	20.3	0.69	82.07	12.03	0.672
KNN	78.67	2.801	0.8087	78.47	24.3	0.76	92.20	4.2	0.7
QDA	69.71	1.59	0.7299	68.18	8.4	0.592	95.40	1.6	0.709

Table : High accuracy and low FPR classifiers on NSL-KDD, AWID, and UNSW-NB15 dataset

Datasets	High-Accuracy classifiers	Low-FPR classifiers
NSL-KDD	RF,ADB,KNN	QDA,GNB
UNSW-NB15	RF,ADB	QDA,RF
AWID	RF,ADB,QDA,KNN	QDA,RF

Performance of model with diff. combination

Table 7: Performance of our system with classifiers.
C1: <RF, ADB, GNB>; C2: <RF, ADB, QDA>; C3: <RF, QDA, KNN>;

Threshold	NSL-KDD				UNSW-NB15		AWID			
	C1		C2		C2		C2		C3	
	Acc (%)	FPR (%)	Acc (%)	FPR (%)	Acc (%)	FPR (%)	Acc (%)	FPR (%)	Acc (%)	FPR (%)
0.5	76.17	2.33	75.6	1.96	83.09	8.3	95.20	3.4	94.20	2.1
0.6	76.17	2.34	81	2.6	84.20	5.6	95.50	1.9	94.8	1.1
0.7	81	2.61	81.07	2.6	**85.09**	**3.3**	96.02	0.3	95.2	0.4
0.8	81	2.59	81.07	2.61	84.80	2.4	95.80	0.2	94.5	0.3

Table 8: Performance of our system with classifiers.
C4: <RF, KNN, ADB, QDA>; C5: <RF, KNN, ADB, GNB>

Threshold	NSL-KDD				UNSW-NB15		AWID	
	C4		C5		C4		C4	
	Acc (%)	FPR (%)	Acc (%)	FPR (%)	Acc (%)	FPR (%)	Acc (%)	FPR (%)
0.5	77.69	2.2	77.96	2.3	81.49	11.3	94.08	3.30
0.6	81.06	2.59	78.84	2.36	82.80	9.6	95.30	1.20
0.7	81.07	2.6	81.04	2.61	80.03	7.3	**96.12**	**0.35**
0.8	**81.80**	**2.6**	81.04	2.617	79.80	5.4	95.50	0.30

Attack type classification performance for **UNSW-NB15**

Table 10: TPR, FPR and ACC comparison of classifiers for attack type classification on UNSW-NB15 Dataset

Attack Type	RF			GaussianNB			Decision Tree			QDA		
	TPR(%)	FPR (%)	ACC(%)	TPR (%)	FPR (%)	ACC(%)	TPR(%)	FPR (%)	ACC(%)	TPR(%)	FPR (%)	ACC(%)
Generic	96.37	0.8	98.28	95.57	0.09	90.13	95.78	0.13	96.35	94.35	1.21	95.53
Exploit	79.21	0.9	86.91	92.93	1.31	89.45	86.15	0.78	85.52	83.62	1.54	82.51
Fuzzers	93.43	1.2	95.37	89.81	0.42	90.21	91.19	1.03	92.41	90.44	1.62	91.13
DoS	92.12	1.0	92.4	79.19	2.41	89.92	93.52	1.07	94.93	91.82	1.55	93.51
Reconnaissance	89.45	1.4	92.1	87.54	2.56	92.01	87.43	0.89	89.65	89.58	1.87	81.37
Analysis	84.67	1.6	86.4	89.35	2.92	86.37	86.81	1.32	87.56	80.26	1.93	81.57
Backdoor	83.53	1.8	85.72	89.87	1.31	84.93	84.35	1.69	87.21	77.59	2.03	80.23
Shellcode	92.79	1.6	88.01	84.59	0.89	87.39	87.15	1.44	89.20	82.96	1.88	85.92
Worm	65.31	1.9	72.69	69.34	1.73	74.56	69.99	1.79	70.59	69.88	2.97	69.53

Attack type classification performance for **NSL-KDD** and **AWID**

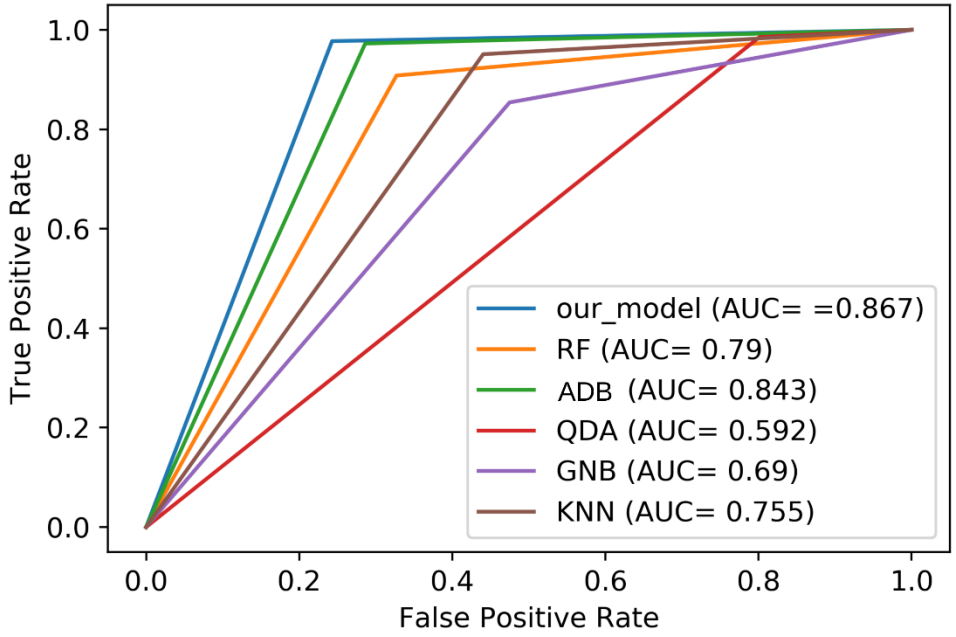
Table 9: TPR, FPR and ACC comparison of classifiers for attack type classification on NSL-KDD Dataset

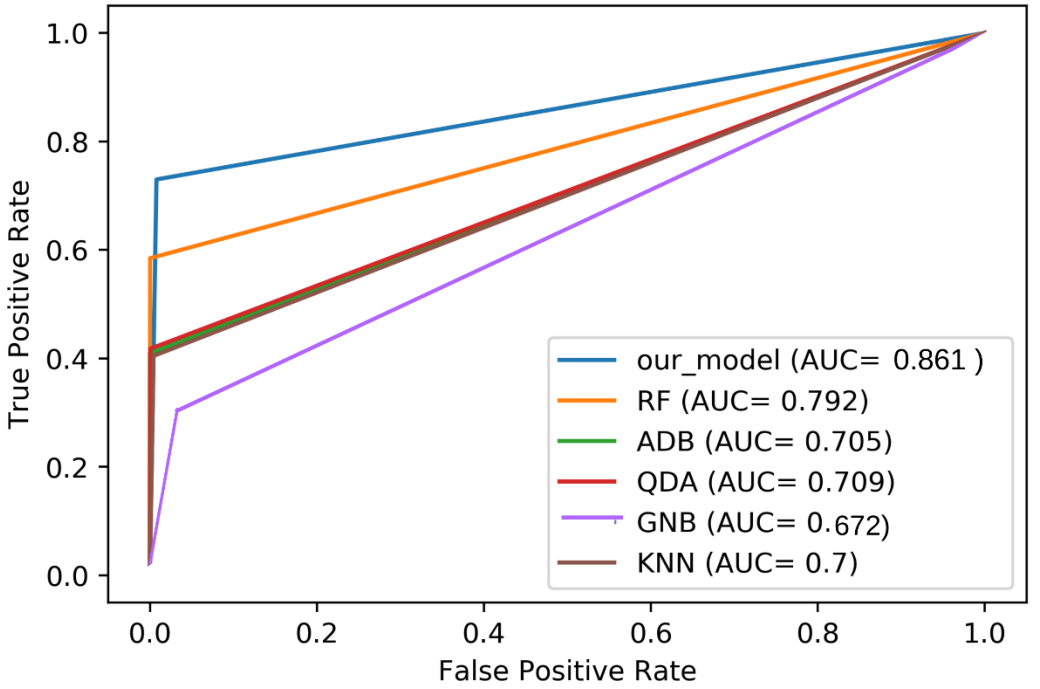
Attack Type	RF			GaussianNB			Decision Tree			QDA		
	TPR(%	FPR (%)	ACC(%)	TPR (%)	FPR (%)	ACC(%)	TPR(%)	FPR (%)	ACC(%)	TPR(%)	FPR (%)	ACC(%)
DOS	99.33	0.29	99.47	93.25	23.72	87.00	98.85	6.58	96.85	91.20	13.04	89.63
Probe	99.27	15.03	86.79	30.19	12.02	80.94	99.81	11.72	89.68	60.66	13.9	82.95
R2L	45.15	0.21	86.55	43.19	0.11	81.07	34.65	0.23	84.00	41.83	5.40	81.82
U2R	40.54	0.46	99.29	78.37	12.35	87.60	40.54	3.93	95.83	64.86	3.43	96.43

Table 11: TPR, FPR and ACC comparison of classifiers for attack type classification on AWID Dataset

Attack Type	RF			GaussianNB			Decision Tree			QDA		
	TPR(%)	FPR (%)	ACC(%)	TPR (%)	FPR (%)	ACC(%)	TPR(%)	FPR (%)	ACC(%)	TPR(%)	FPR (%)	ACC(%)
Flooding	99.91	0.03	99.90	99.47	0.04	98.72	99.2	0.05	98.02	99.1	0.28	96.47
Impersonation	99.99	0.04	99.03	99.91	0.04	99.01	98.57	0.07	98.49	95.18	0.32	99.34
Injection	99.42	0.06	99.6	99.02	0.03	99.24	99.23	0.13	97.52	94.27	0.49	93.69

ROC curve for **AWID** and **UNSW-NB15**





Comparison of proposed IDS with state-of-art work

Reference	model	Adaptive	ENS	Robustness	Dataset	Accuracy (%)	FPR (%)
[57]	ANN	✗	✗	✗	NSL-KDD	81.2	3.23
[58]	ENS	✗	✓	✗	NSL-KDD	80.07	3.04
[59]	SOM	✗	✗	✗	NSL-KDD	75.49	5.77
[60]	ELM	✓	✗	✗	NSL-KDD	78.49	3.08
[61]	DNN	✗	✗	✗	NSL-KDD	75.75	24.69
[65]	ANN	✗	✗	✗	UNSW-NB15	81.34	23.79
[66]	GA-LR	✗	✓	✗	UNSW-NB15	81.42	6.39
[42]	SBN	✗	✓	✗	AWID	95.26	3.48
Our Model	DQN	✓	✓	✓	NSL-KDD	81.80	2.6
					AWID	96.12	0.35
					UNSW-NB15	85.09	3.3

* ANN - Artificial Neural Network, ENS - Ensemble Methods, SOM- Soft Organizing Map, ELM-Extreme Learning Machine, DNN-Deep Neural Network, DQN-Deep Q-Network, SBN-Semi Boosted Nested, GA-LR: Genetic Algorithm-Logistic Regression

Model Evaluation in changing attack pattern

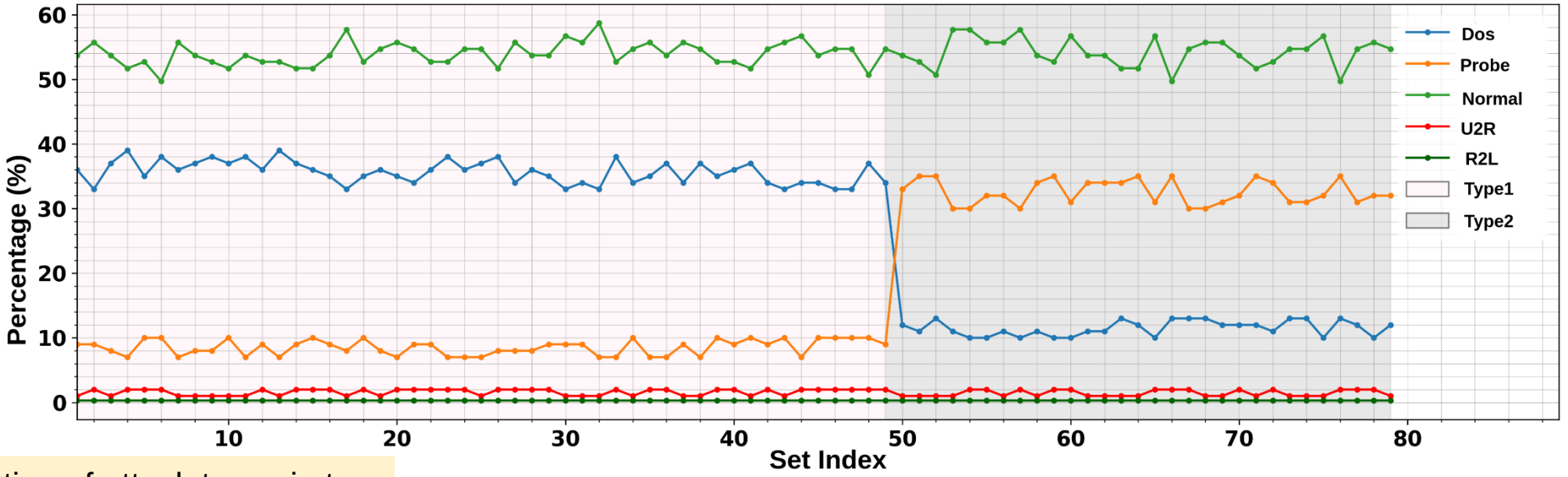
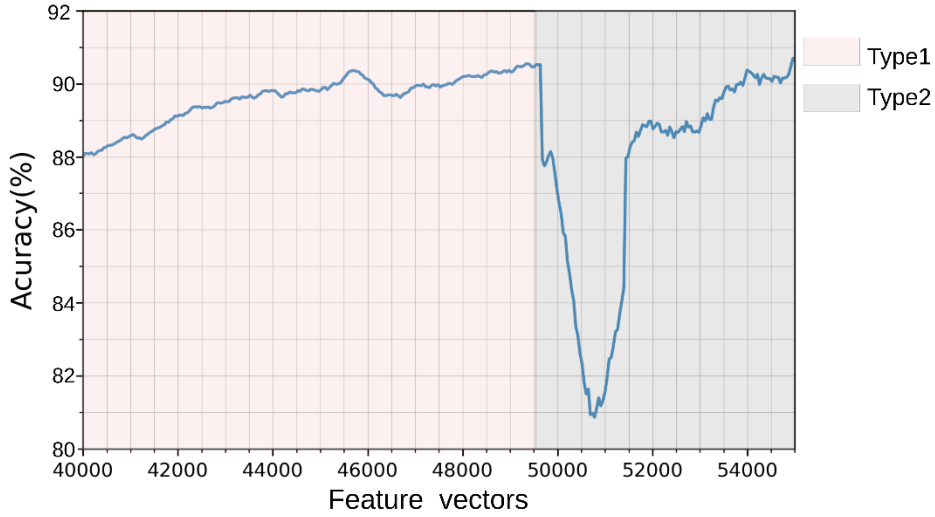
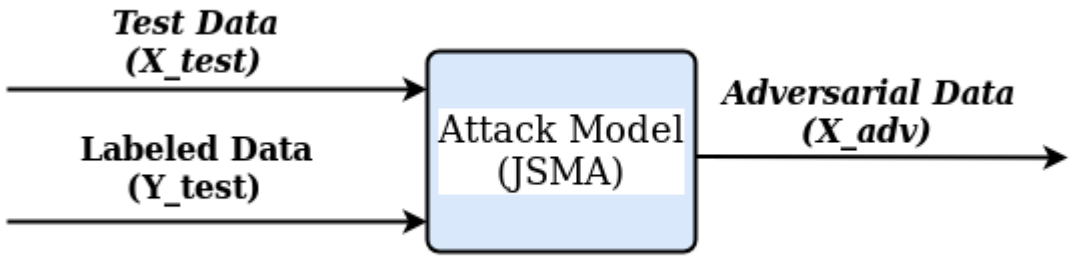


Table: Distribution of attack types in two different attack patterns (% value in 1000 samples)

Attack Type	Type1 (%)	Type2 (%)
DOS	33-39	10-12
Probe	7-10	30-35
U2r	1-2	1-2
R2l	0.3-0.8	0.3 - 0.8
Normal	rest	rest



Adversarial Data generation



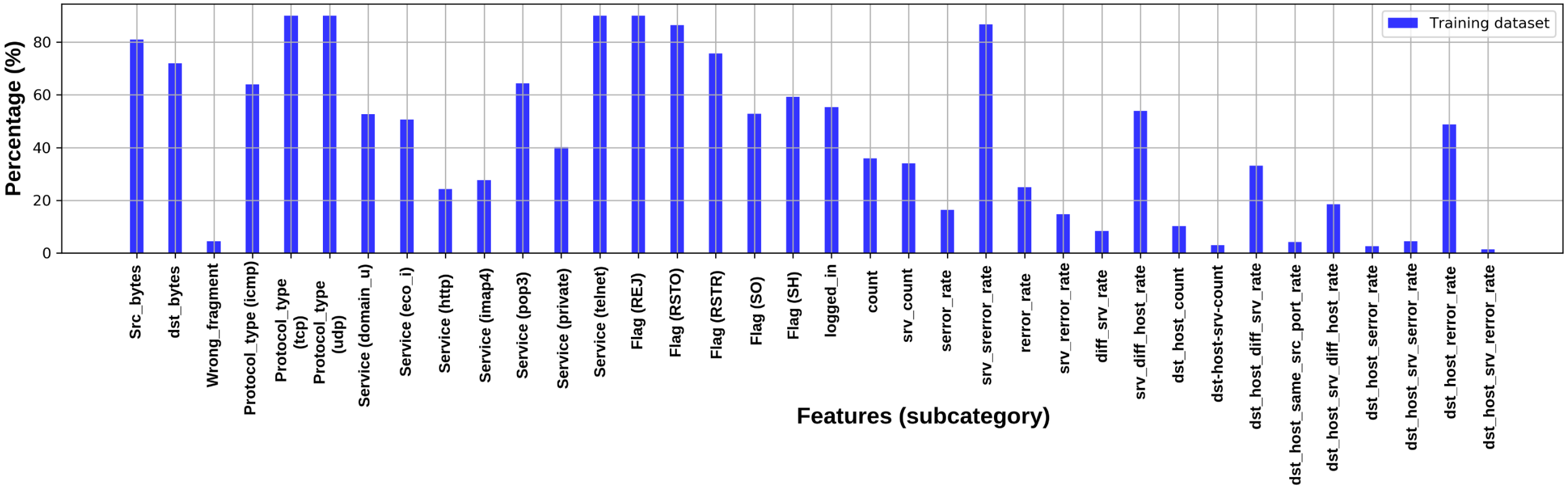
The objective of any adversarial model is to add **small perturbations** that are similar to **normal samples.**

Jacobian-based Saliency Map Attack (JSMA): It is a simple white-box (i.e., the attacker has access to the model's parameters) iterative method to misclassify the target model. This method allows the use of forward derivative of DNN to generate adversarial perturbation.

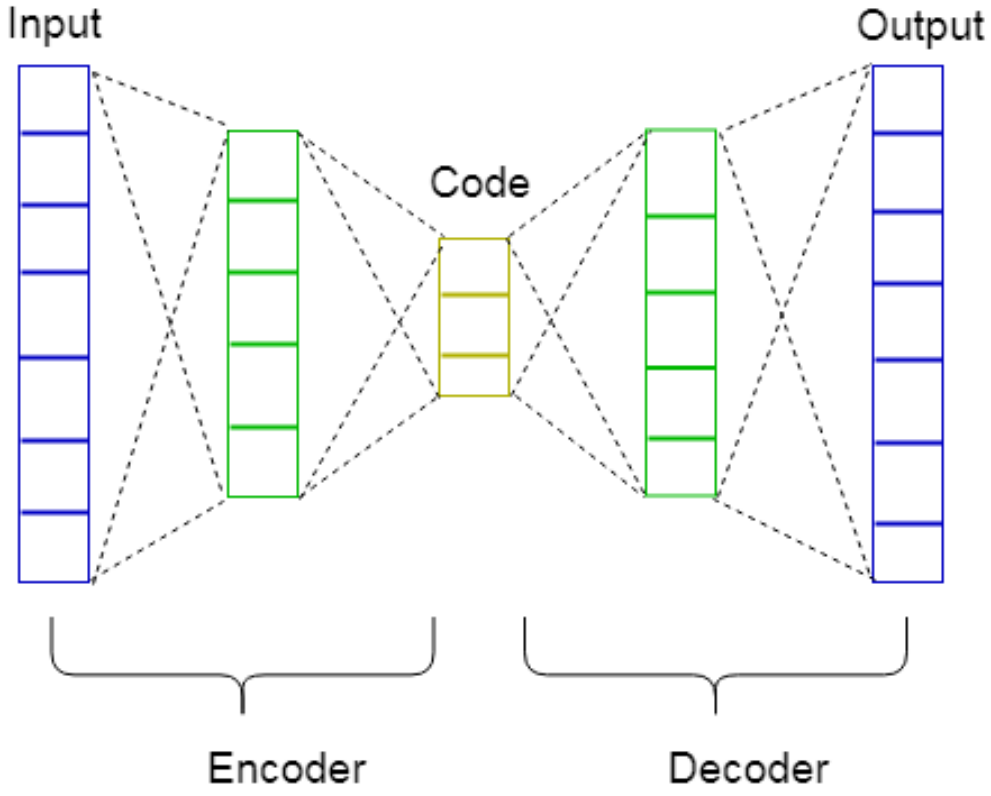
Algorithm 3: Generation of the adversarial sample for binary class

```
no_sample = number of rows in X_test;
Set X_adv = an empty array of size (0,36);
jsma = an instance of SaliencyMapMethod;
Set jsma_parameter as
  theta,gamma,clip_min,clip_max,y_target;
for sampleNo in range(no_sample) do
    sample = X_test[sampleNo]
    Set current_class = argmax(Y_test[sampleNo]);
    if current_class ≠ 0 then
        target_class = [0];
    else
        target_class = [1];
    end
    new_target_class = generate zero matrix of size
      (1,2);
    Set new_target_class[0,target_class] = 1;
    jsma_parameter.y_target = new_target_class;
    adv_x =
      jsma.generate_np(sample, jsma_parameter);
    X_adv = Push(stack,adv_x);
end
```

Adversarial output deviation percentage



Denoising Auto-encoder (DAE)



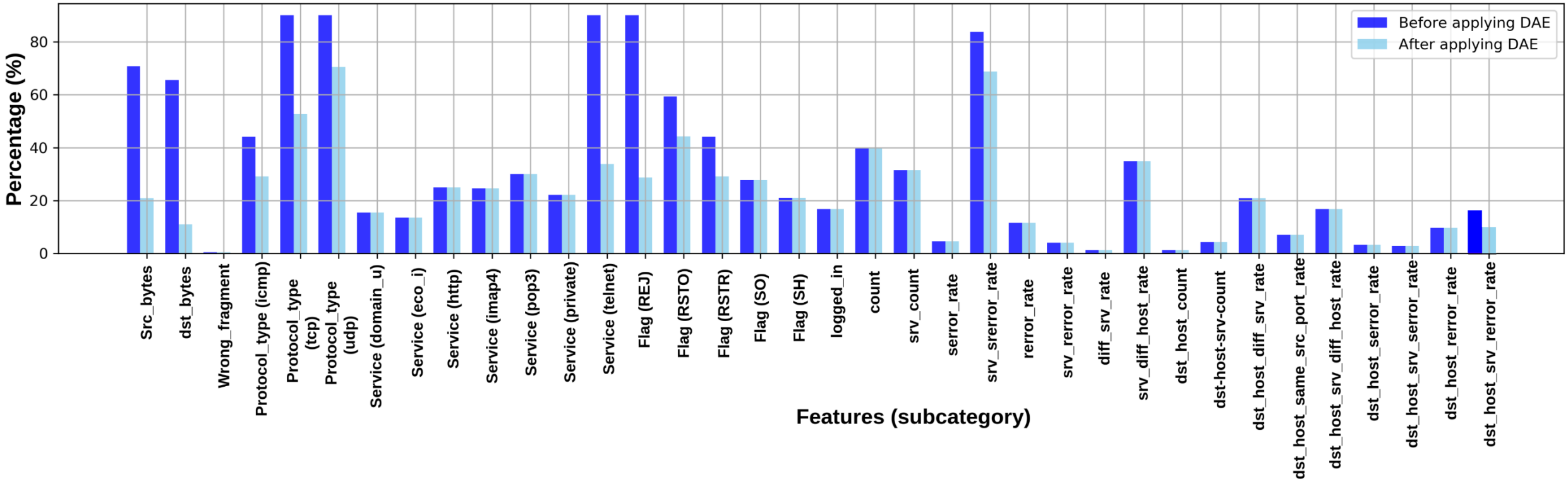
Layered architecture of proposed denoising autoencoder

DAE is a deep neural network which maps n inputs to n outputs.

It takes an **adversarial dataset** (i.e., corrupted dataset) as input and is trained to predict the **unperturbed dataset** (i.e., uncorrupted dataset) as its output. It consists of two components namely encoder and decoder.

Encoder reduces the dimension of the input data to produce code which is a compressed representation of the input data and **decoder** reconstructs the output data from the code.

Denoising output deviation percentage



Comparison of model against adversarial dataset

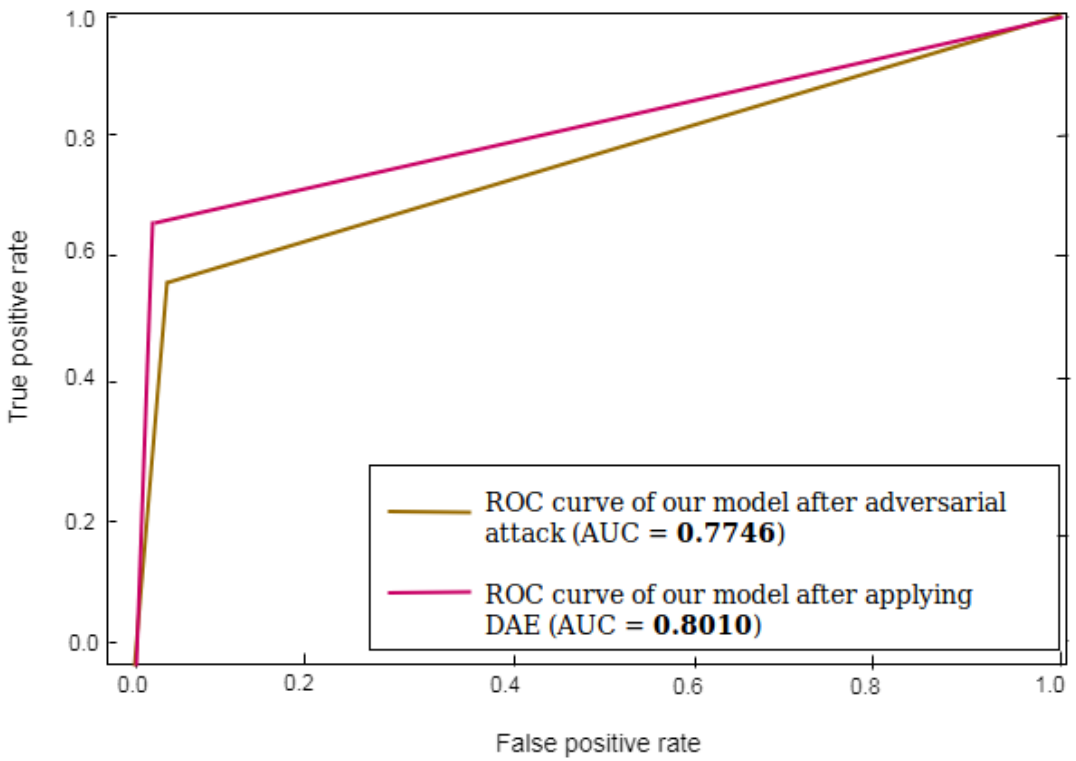


Table: Robustness of our system against adversarial Attack

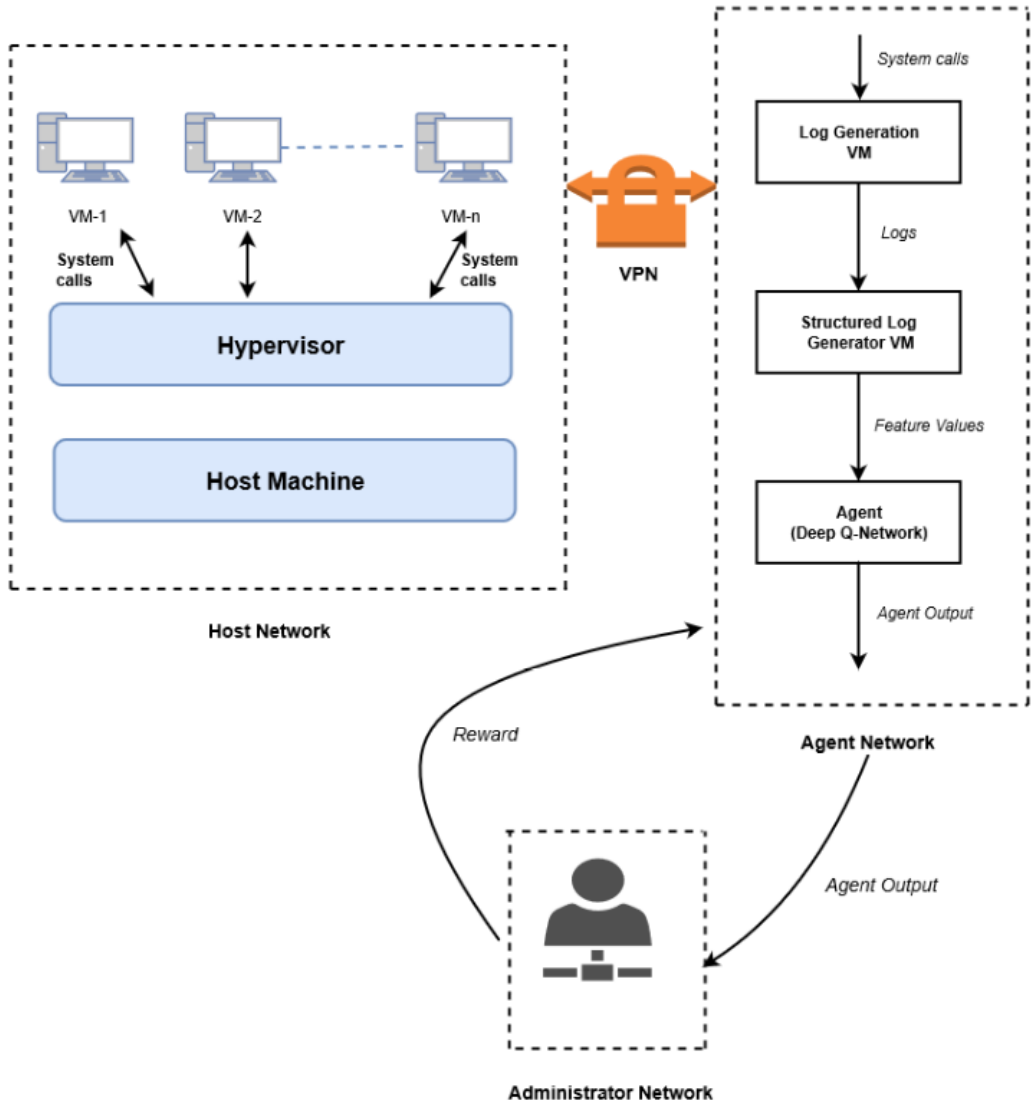
Action	Accuracy	FPR	AUC
Model before adversarial attack	81.80%	2.6%	0.8212
Model after adversarial attack	78.44%	5.8%	0.7746
Model after applying DAE	80.05%	6.8%	0.8010

Table: Comparison of Robustness on NSL-KDD: Our Model vs. [R1]

Reference	Original		After attack	
	Accuracy	FPR	Accuracy	FPR
[62]	81.0%	20.4%	53.3%	20.4%
Our Model	81.80%	2.6%	78.44%	5.8%

[62] K. Yang, J. Liu, C. Zhang and, Y. Fang, "Adversarial Examples Against the Deep Learning Based Network Intrusion Detection Systems," IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, pp. 559-564, (2018). DOI: 10.1109/MILCOM.2018.8599759

Cloud IDS system



Malware is malicious software that is specifically designed to breach a computer systems security policy and disrupt, damage, or gain unauthorized access.

Malware can be classified into different types like virus, trojan horse, adware, spyware, etc. according to the way it exploits the vulnerabilities of the system.

potentially
unwanted
program

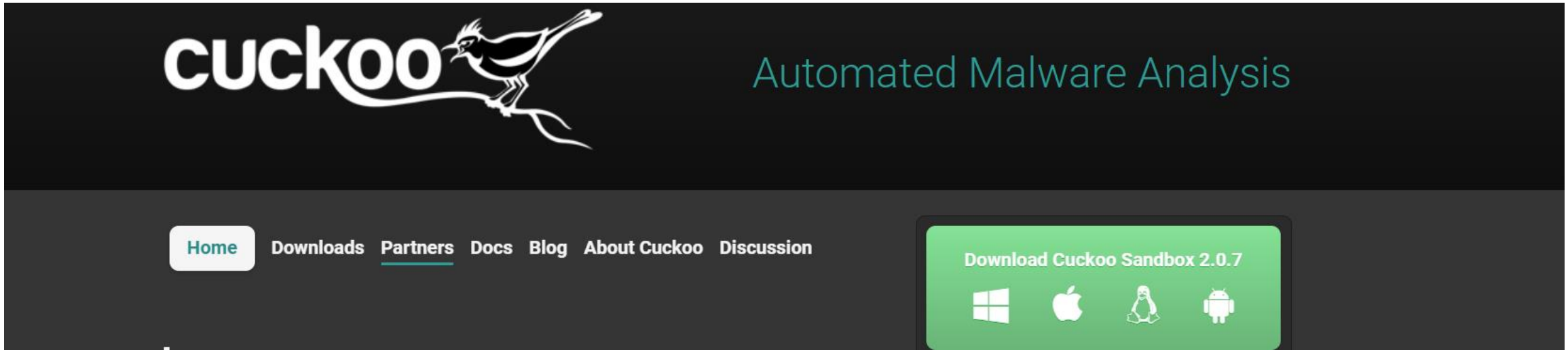
(n.) when you don't read the
Download Agreement

K. Sethi, R. Kumar, P. Bera et. **Al., "A Novel Machine Learning Based Malware Detection and Classification Framework,"** 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)
Oxford, United Kingdom, 2019, pp. 1-4. doi: 10.1109/CyberSecPODS.2019.8885196

As time progresses, new and complex malware types are being generated which causes a serious threat to computer systems. Due to this drastic increase in the number of malware samples, the signature-based malware detection techniques cannot provide accurate results. Different studies have demonstrated the proficiency of machine learning for the detection and classification of malware files. Further, the accuracy of these machine learning models can be improved by using feature selection algorithms to select the most essential features and reducing the size of the dataset which leads to lesser computations. In this paper, we have developed a **machine learning based malware analysis framework for efficient and accurate malware detection and classification. We used Cuckoo sandbox for dynamic analysis which executes malware in an isolated environment and generates an analysis report based on the system activities during execution.** Further, we propose a feature extraction and selection module which extracts features from the report and selects the **most important features for ensuring high accuracy at minimum computation cost**. Then, we employ different machine learning algorithms for accurate detection and fine-grained classification. Experimental results show that we got high detection and classification accuracy in comparison to the state-of-the-art approaches.

What is Cuckoo?
Cuckoo Sandbox is the **leading open source automated malware analysis system**.

You can throw any **suspicious file** at it and in a matter of minutes Cuckoo will provide a **detailed report outlining the behaviour of the file** when executed inside a realistic but isolated environment.



https://cuckoosandbox.org/

Malware Detection operation workflow

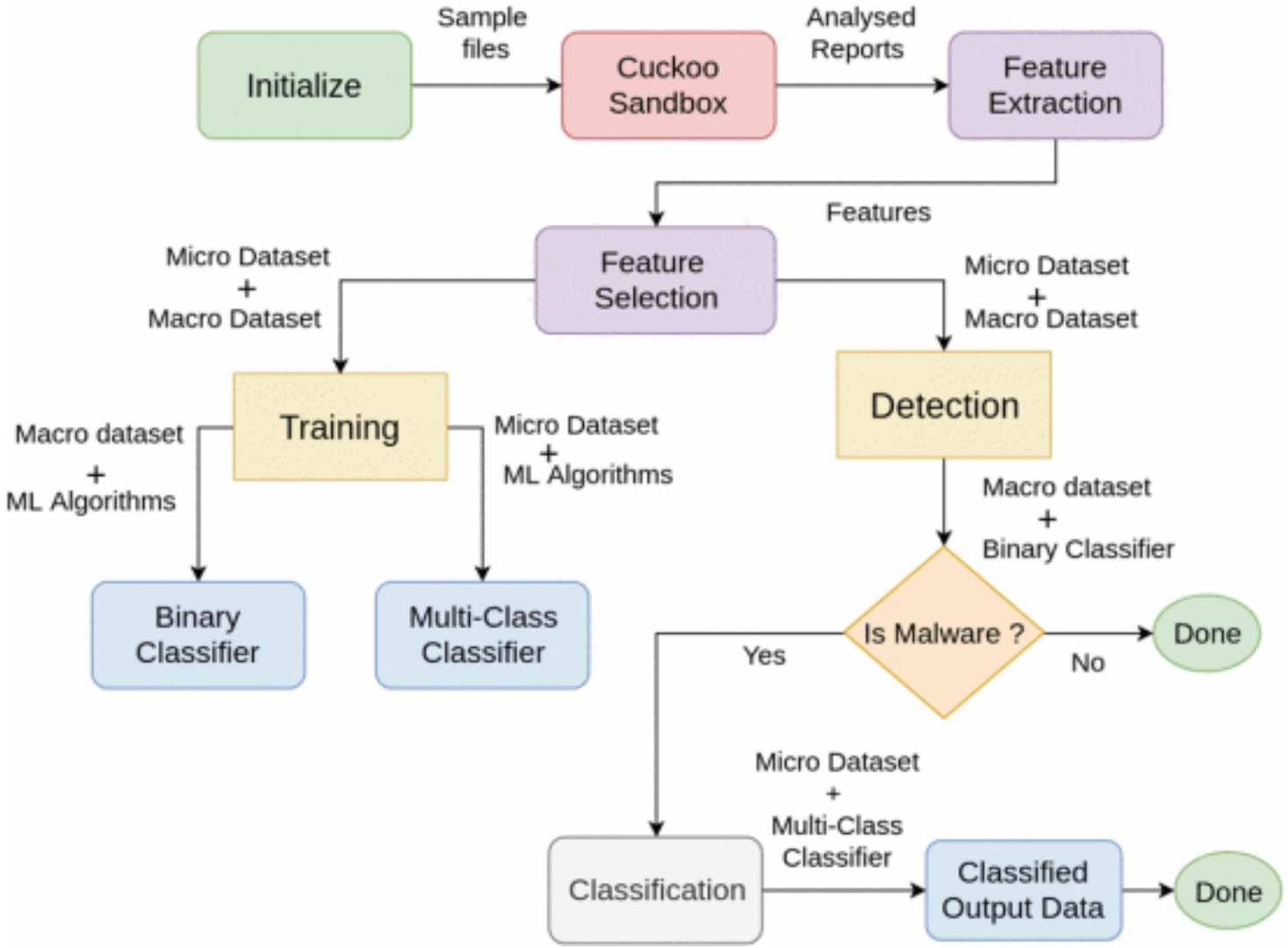


Fig: Operational flow of proposed methodology

Results

Fig: ROC curve for decision tree classifier

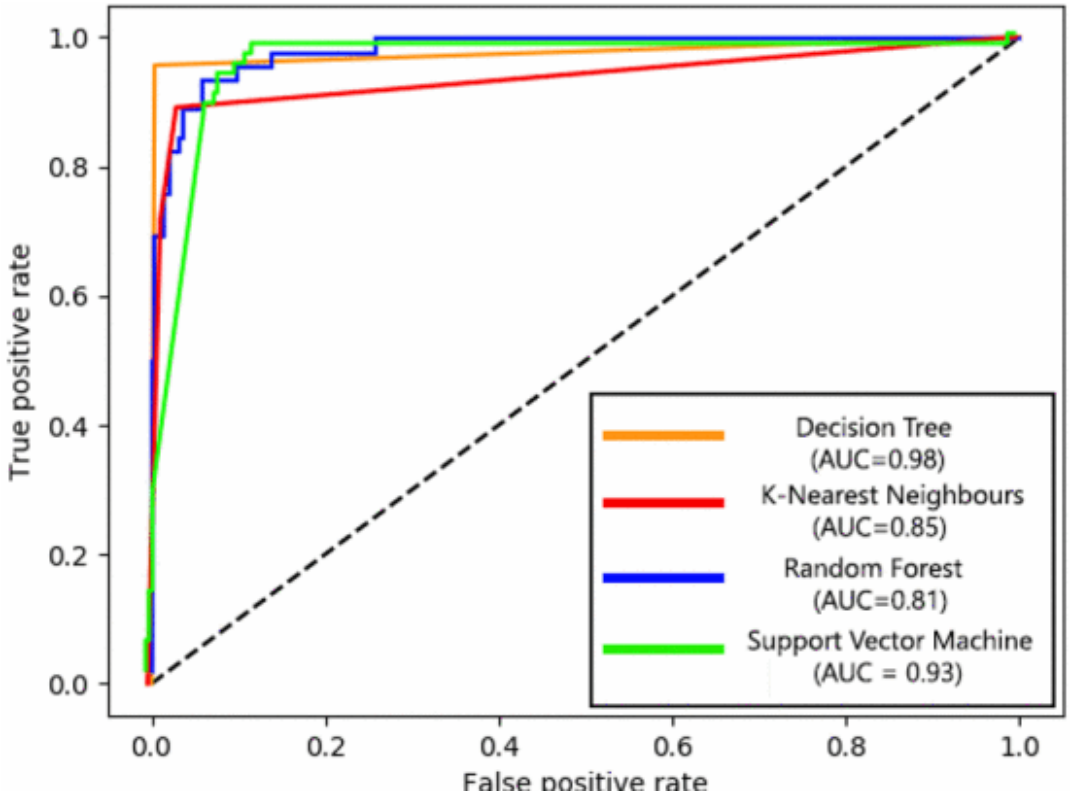


Table : Malware family classification results using our proposed method

Classification Algorithms	Accuracy	Precision	Recall	F-measure
K-Nearest Neighbor	96.46%	0.97	0.96	0.96
Decision Tree	99.11%	0.99	0.99	0.99
Support Vector Machine	86.72%	0.88	0.87	0.87
Random Forest	88.23%	0.90	0.88	0.86

Malware family classification

Table : Malware family classification results using our proposed method

Class	# samples	Accuracy	Precision	Recall	F-measure
Virus	81	90%	1.00	0.90	0.95
Adware	231	100%	1.00	1.00	1.00
Hoax	12	60%	0.85	0.85	0.85
Trojan	255	100%	0.94	1.00	0.97
Spyware	48	77.77%	0.58	0.78	0.67
Worm	27	66.66%	1.00	0.67	0.80
Backdoor	24	62.5%	1.00	0.62	0.77

Demo link

https://colab.research.google.com/drive/1wXvSNejME-oNhzRV9bG-iueZ9aojyz9q

References

Topic	Description
Intrusions	https://www.dnsstuff.com/intrusion-detection-system
Intrusions Detection system	https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-systems-definition-challenges-343
Firewall	https://resources.infosecinstitute.com/network-design-firewall-idsips/#gref
Unsupervised learning	http://is.tuebingen.mpg.de/fileadmin/user_upload/files/publications/taxo_[0].pdf
Generative Vs Discriminative	https://medium.com/@mlengineer/generative-and-discriminative-models-af5637a66a3
Machine Learning Videos	https://statquest.org/video-index/
Neural Network	https://www.doc.ic.ac.uk/~nuric/teaching/imperial-college-machine-learning-neural-networks.html

References

Topic	Description
Taxonomy of ML	https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781783558742/1/ch01lvl1sec12/taxonomy-of-machine-learning-algorithms
Challenges of IDS	https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-systems-definition-challenges-343
Detecting Network Intrusions With Machine Learning Based Anomaly Detection Techniques	https://www.youtube.com/watch?v=c71gt-I8Lik
Introduction to ML	https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Email: Kamalakanta Sethi: ks23[at]iitbbs.ac.in Rahul Kumar:rk36[@]iitbbs.ac.in