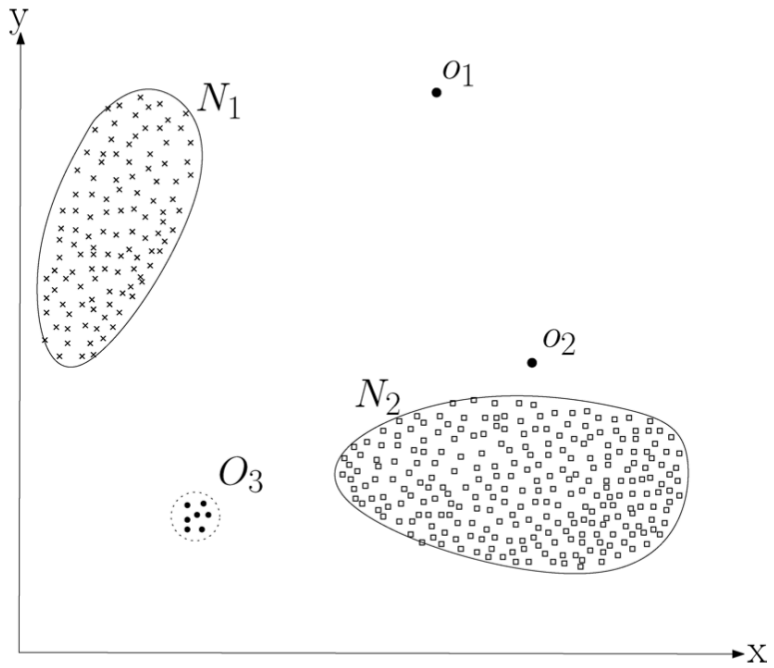
What are Anomalies?

Anomalies are patterns in data that do not conform to a well defined notion of **normal behavior**.



Source: Chandola et al., 2009

Fig. 1. A simple example of anomalies in a two-dimensional data set.

Figure 1 illustrates anomalies in a simple two-dimensional data set. The data has two **normal regions**, N1 and N2, since **most observations** lie in these two regions. Points that are *"sufficiently far away from these regions,"* for example, points o1 and o2, and points in region O3, are anomalies.

Real life relevance

Anomalies may occur in the data for a variety of reasons, such as

malicious activity, for example, credit card fraud, cyber-intrusion …,

but all of the reasons have the common characteristic that they are interesting to the analyst. This **real life relevance** of anomalies is a key feature of anomaly detection.

Noise

Anomaly detection is related to, but distinct from *noise removal* and *noise accommodation*[1], both of which deal with **unwanted noise** in the data.

Noise is a phenomenon in data that is not of interest to the analyst, but acts as a hindrance to data analysis.

Novelties

Novelty detection aims at detecting **previously unobserved** (emergent, novel) patterns in the data. The distinction between novel patterns and anomalies is that the novel patterns are typically incorporated into the normal model after being detected.

[1] Noise accommodation refers to immunizing a statistical model estimation against anomalous observations.

Challenges

An anomaly is defined as a pattern that does not conform to expected normal behavior. An anomaly detection approach, therefore, is to define **a region (or regions)** representing normal behavior and declare any observation in the data that does not belong to this normal region to be outliers. But several factors make this apparently simple approach very challenging:

- Defining normal regions that encompass every possible normal behavior is very difficult. In addition, the **boundary** between normal and anomalous behavior is often not precise.
- When anomalies are the result of malicious actions, the malicious adversaries often adapt themselves to make the anomalous observations appear normal, thereby making the task of defining normal behavior more difficult.

Challenges

An anomaly is defined as a pattern that does not conform to expected normal behavior. An anomaly detection approach, therefore, is to define **a region (or regions)** representing normal behavior and declare any observation in the data that does not belong to this normal region to be outliers. But several factors make this apparently simple approach very challenging:

- Defining normal regions that encompass every possible normal behavior is very difficult. In addition, the **boundary** between normal and anomalous behavior is often not precise.
- When anomalies are the result of malicious actions, the malicious adversaries often adapt themselves to make the anomalous observations appear normal, thereby making the task of defining normal behavior more difficult.
- In many domains normal behavior keeps evolving and a current notion of normal behavior might not be sufficiently representative in the future.
- The exact notion of an anomaly is different for different application domains.
- Availability of **labeled data** for training/validation of models used by anomaly detection techniques is usually a major issue. Real world data is often lacking *ground truth*.

Challenges

An anomaly is defined as a pattern that does not conform to expected normal behavior. An anomaly detection approach, therefore, is to define **a region (or regions)** representing normal behavior and declare any observation in the data that does not belong to this normal region to be outliers. But several factors make this apparently simple approach very challenging:

- Defining normal regions that encompass every possible normal behavior is very difficult. In addition, the **boundary** between normal and anomalous behavior is often not precise.
- When anomalies are the result of malicious actions, the malicious adversaries often adapt themselves to make the anomalous observations appear normal, thereby making the task of defining normal behavior more difficult.
- In many domains normal behavior keeps evolving and a current notion of normal behavior might not be sufficiently representative in the future.
- The exact notion of an anomaly is different for different application domains.
- Availability of **labeled data** for training/validation of models used by anomaly detection techniques is usually a major issue. Real world data is often lacking *ground truth*.
- Often the data contains noise that tends to be similar to the actual anomalies and hence is difficult to distinguish and remove.

⚐ Note: Many anomaly detection techniques solve a "specific anomaly detection problem", depending on various factors like the nature of the data, availability of labeled data, type of anomalies to be detected, etc.

Techniques and application domains covered by various related survey articles:

Table I. Comparison of our Survey to Other Related Survey Articles. 1—Our Survey, 2—Hodge and Austin [2004], 3—Agyemang et al. [2006], 4—Markou and Singh [2003a], 5—Markou and Singh [2003b], 6—Patcha and Park [2007], 7—Beckman and Cook [1983], 8—Bakar et al. [2006]

		1	2	3	4	5	6	7	8
Techniques	Classification Based	√	√	√	√		√		
	Clustering Based	√	√	√			√		
	Nearest Neighbor Based	√	√	√			√		√
	Statistical	√	√	√		√	√	√	√
	Information Theoretic	√							
	Spectral	√							
Applications	Cyber-Intrusion Detection	√					√		
	Fraud Detection	√							
	Medical Anomaly Detection	√							
	Industrial Damage Detection	√							
	Image Processing	√							
	Textual Anomaly Detection	√							
	Sensor Networks	√							

Anomaly Detection Techniques

Three broad categories of anomaly detection techniques exist:

- **Unsupervised anomaly detection** techniques detect anomalies in an unlabeled test data set, under the assumption that the majority of the instances in the data set are normal, by looking for instances that seem to fit least to the remainder of the data set.
- **Supervised anomaly detection** techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection).
- **Semi-supervised anomaly detection** techniques construct a model representing normal behavior from a given normal training data set, and then testing the likelihood of a test instance to be generated by the learnt model.

Likelihood and maximum likelihood estimation
... a method for fitting models to data.

https://www.youtube.com/watch?v=2vh98ful3_M

Complex anomalies

We distinguish ***simple anomalies*** from ***complex anomalies***.

For most application domains, the interesting anomalies are complex in nature, while most of the algorithmic research has focused on simple anomalies.

We differentiate between two types of complex anomalies:

- **contextual** anomalies and
- **collective** anomalies.

Anomaly Detection Problems

This section explores the richness in the problem domain and justifies the need for the **broad spectrum** of anomaly detection techniques.

A key aspect of any anomaly detection technique is the nature of the input data.

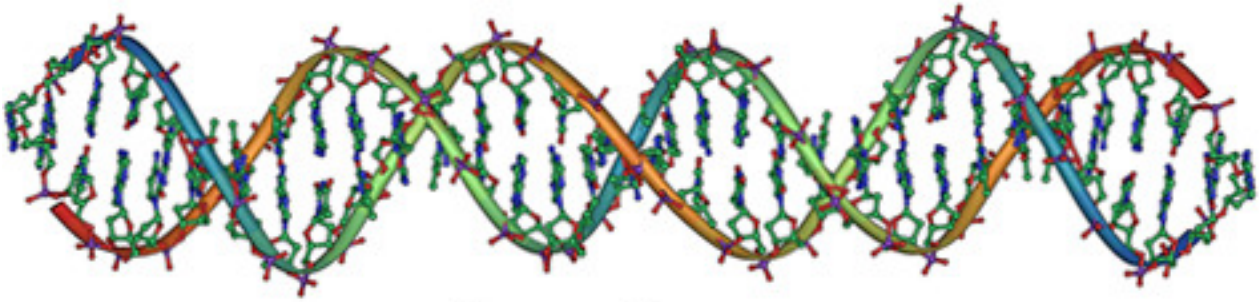
Input data is generally a **collection of data instances**. Each data instance can be described using a set of attributes consisting of only one attribute (univariate) or multiple attributes (multivariate). Attributes can be of different types such as *binary*, *categorical*, or *continuous*.

Input data can also be categorized based on the relationship present among data instances. Most of the existing anomaly detection techniques deal with **record data** or **point data**, in which no relationship is assumed among the data instances.

In general, data instances can be related to each other though. Some examples are sequence data, spatial data, and graph data. In sequence data, the data instances are **linearly ordered**, for example:

- time series data
- genome sequences
- protein sequences.

Genome Sequence



Genome Sequence

AGATAACTGGGCCCCTGCGCTCAGGAGGCCTTCACCCTCTGCTCTGGGTAAAGGTAGTAGA

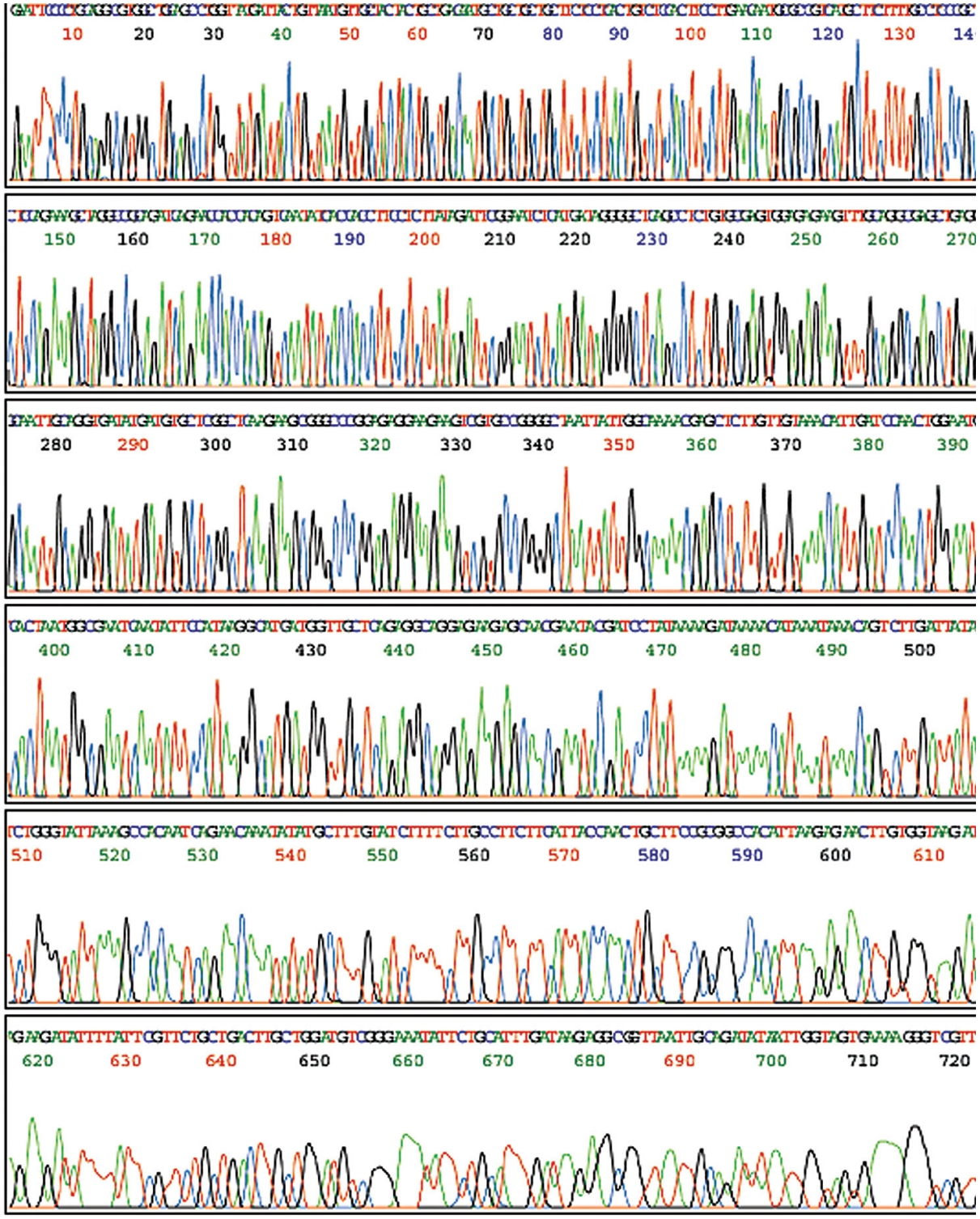
Fragment Reads

AGATAACTGGGCCCCTGCGCTCAGGAGGCCTTCACC
CTGGGCCCCTGCGCTCAGGAGGCCTTCACCCTCTGC
CCCCTGCGCTCAGGAGGCCTTCACCCTCTGCTCTGG
TGCGCTCAGGAGGCCTTCACCCTCTGCTCTGGGTAA
CTCAGGAGGCCTTCACCCTCTGCTCTGGGTAAAGGT
AGGCCTTCACCCTCTGCTCTGGGTAAAGGTAGTAGA

CYBERSECURITY

Genome Sequence Trace

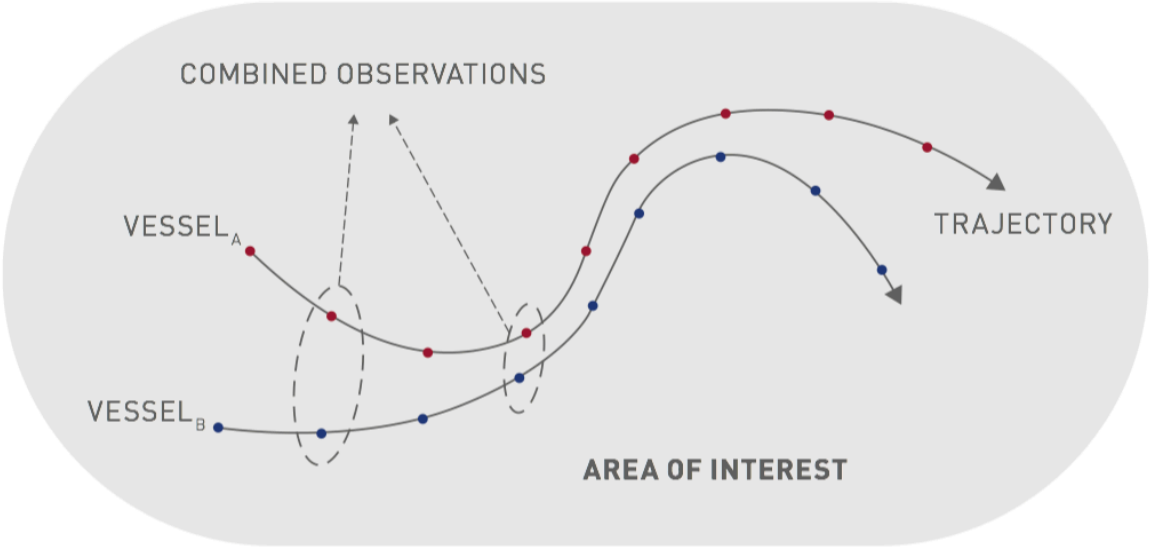
Source: Office of Biological and Environmental Research of the U.S. Department of Energy Office of Science.



Spatial Data and Graph Data

In **spatial data**, each data instance is related to its neighboring instances, for example, vehicular traffic data, and ecological data. When the spatial data has a temporal (sequential) component it is referred to as spatio-temporal data, for example, climate data.

Example: Ship trajectory derived from a sequence of AIS data points (time series)



In **graph data**, data instances are represented as vertices in a graph and are connected to other vertices with edges. Later in this section we will discuss situations where such relationships among data instances become relevant for anomaly detection.

Type of anomaly

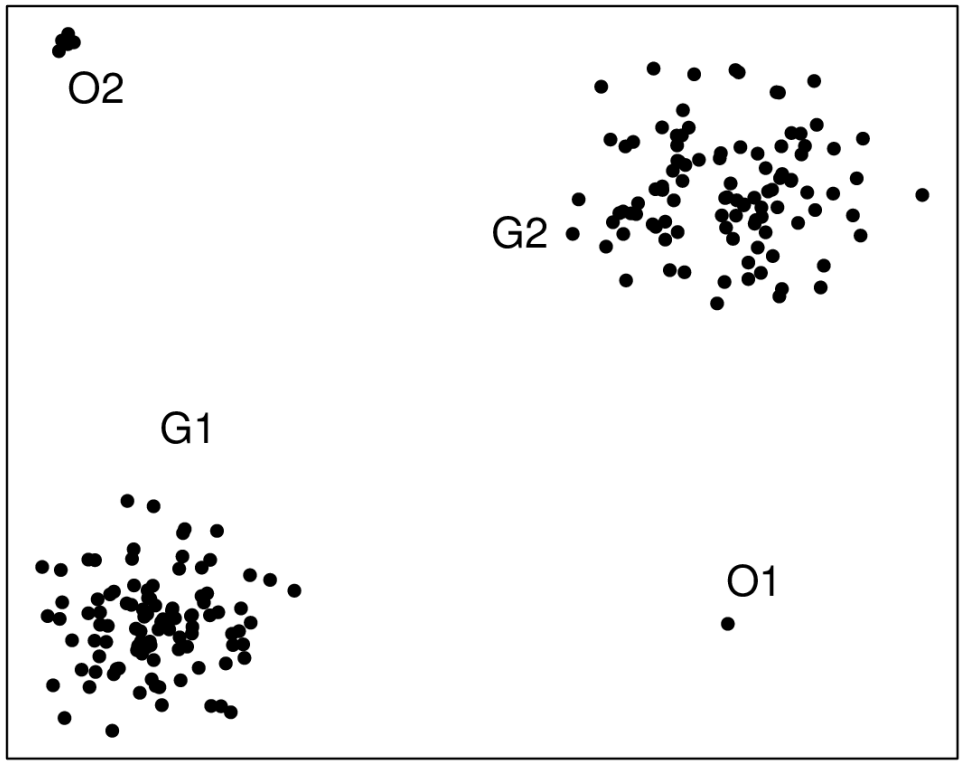
Anomalies can generally be classified into the following three categories:

1. Point Anomalies
2. Contextual Anomalies
3. Collective Anomalies

The differences will be characterized by means of three illustrative examples as follows.

Point Anomalies

If an **individual data instance** can be considered anomalous with respect to the rest of the data, then the instance is termed a point anomaly. This is the simplest type of anomaly and is the focus of the majority of research on anomaly detection.



Example of two-dimensional outliers. Point labeled *O1* and points labeled *O2* deviate significantly from regions labeled *G1* and *G2*. (Source: Wikipedia)

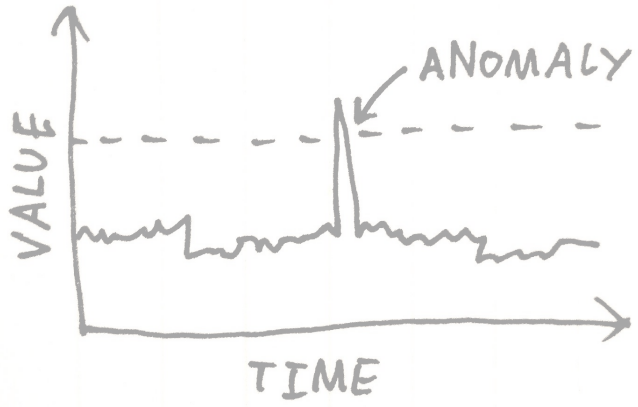
Point Anomaly: Example

As a real-life example, consider **credit card fraud detection**. Let the data set correspond to an individual's credit card transactions.

For simplicity, let us assume that the data is defined using only one *feature*: amount spent.

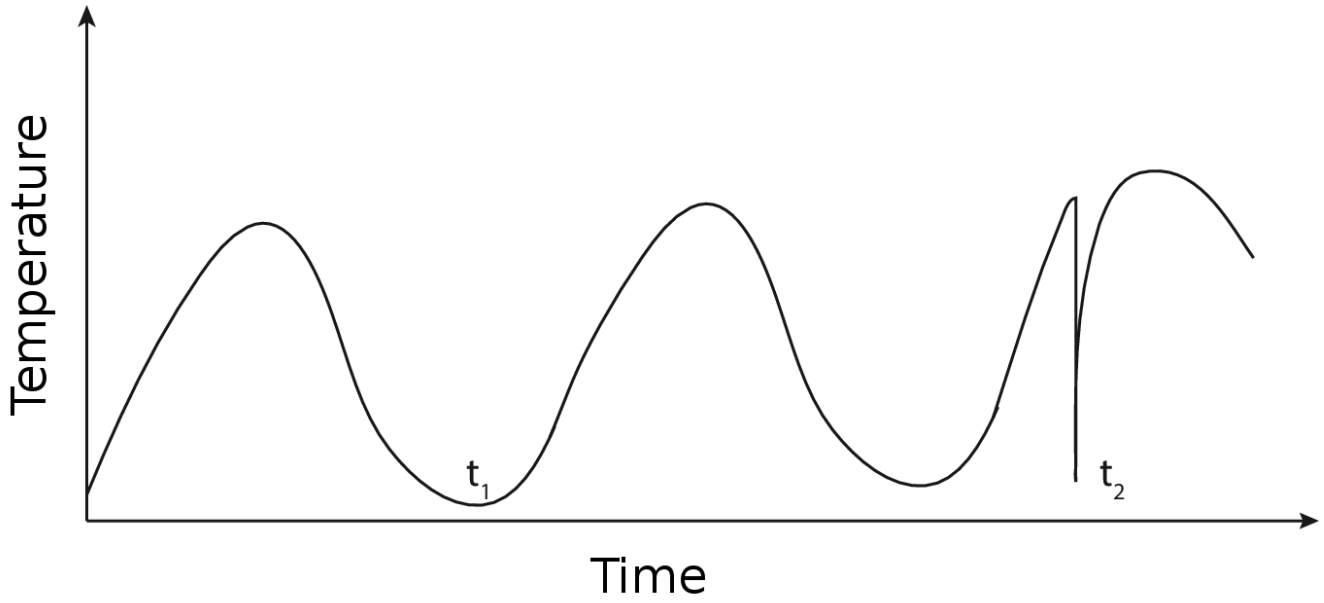
A transaction for which the amount spent is very high (or very low) compared to the normal range of expenditure for that person will be a point anomaly.

What other real world examples of point anomalies do you see?



Contextual Anomalies

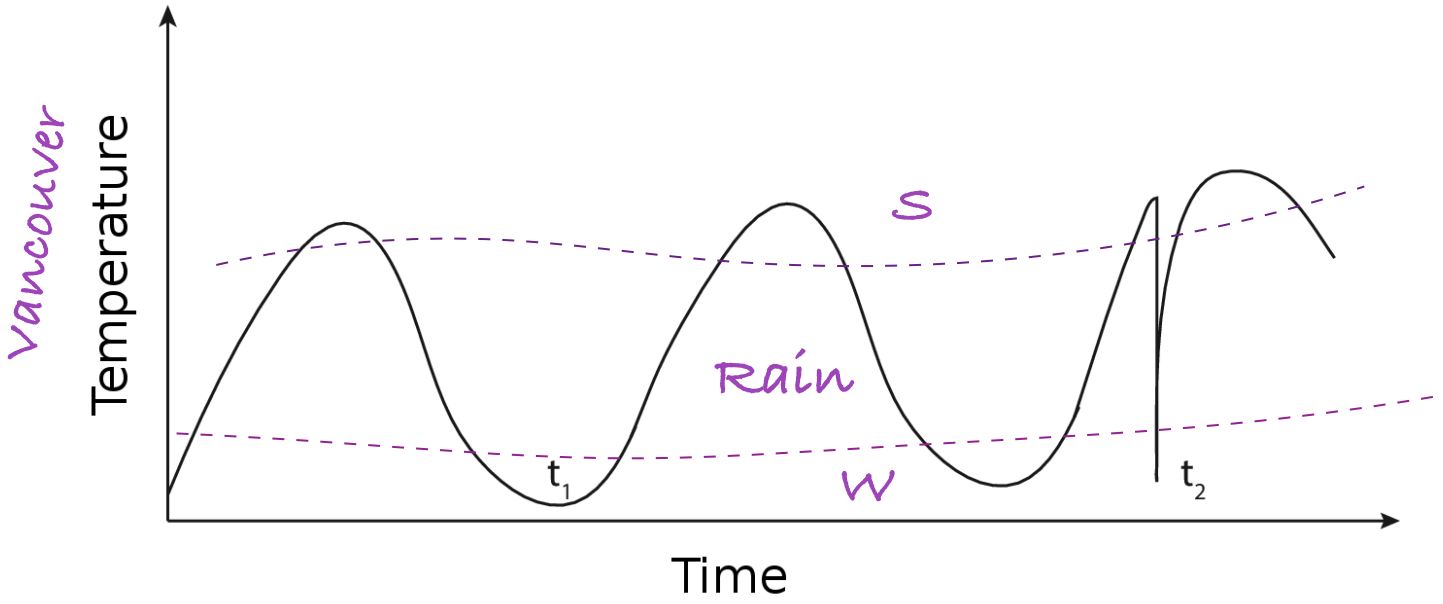
If a data instance is anomalous in a **specific context**, but not otherwise, then it is considered a contextual anomaly (also referred to as *conditional anomaly*).



An example of a contextual outlier. Points labeled t1 and t2 represent the same temperature value. Though, point t2 is considered an outlier, while point t1 is not. (Source: Wikipedia)

Contextual Anomalies

If a data instance is anomalous in a **specific context**, but not otherwise, then it is considered a contextual anomaly (also referred to as *conditional anomaly*).



An example of a contextual outlier. Points labeled t1 and t2 represent the same temperature

The notion of a context is induced by the structure in the data set and has to be specified as a part of the problem formulation. Each data instance is defined using the following two sets of attributes:

- *Contextual attributes* are used to determine the context (or neighborhood) for that instance.

  For example, in spatial data sets, the longitude and latitude of a location are the contextual attributes. In time-series data, time is a contextual attribute that determines the position of an instance on the entire sequence.

- *Behavioral attributes* define the noncontextual characteristics of an instance.

  For example, in a spatial data set describing the average rainfall across the entire world, the specific amount of rainfall at any given location is a behavioral attribute.

Anomalous behavior is determined using the values for the behavioral attributes within a specific context. A data instance might be a contextual anomaly in a given context, but another data instance with identical values of its behavioral attributes could be considered normal in a different context.

Note: This property is key in identifying contextual and behavioral attributes for a contextual anomaly detection technique.

The choice of applying a contextual anomaly detection technique is determined by the meaningfulness of the contextual anomalies in the target application domain.

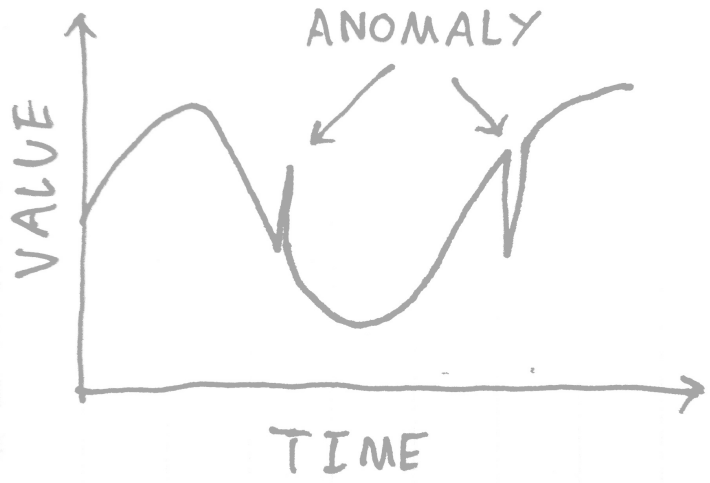
Another key factor is the availability of contextual attributes.

Contextual Anomaly: Example

Consider credit card fraud detection again. A contextual attribute in the credit card domain can be the **time of purchase**.

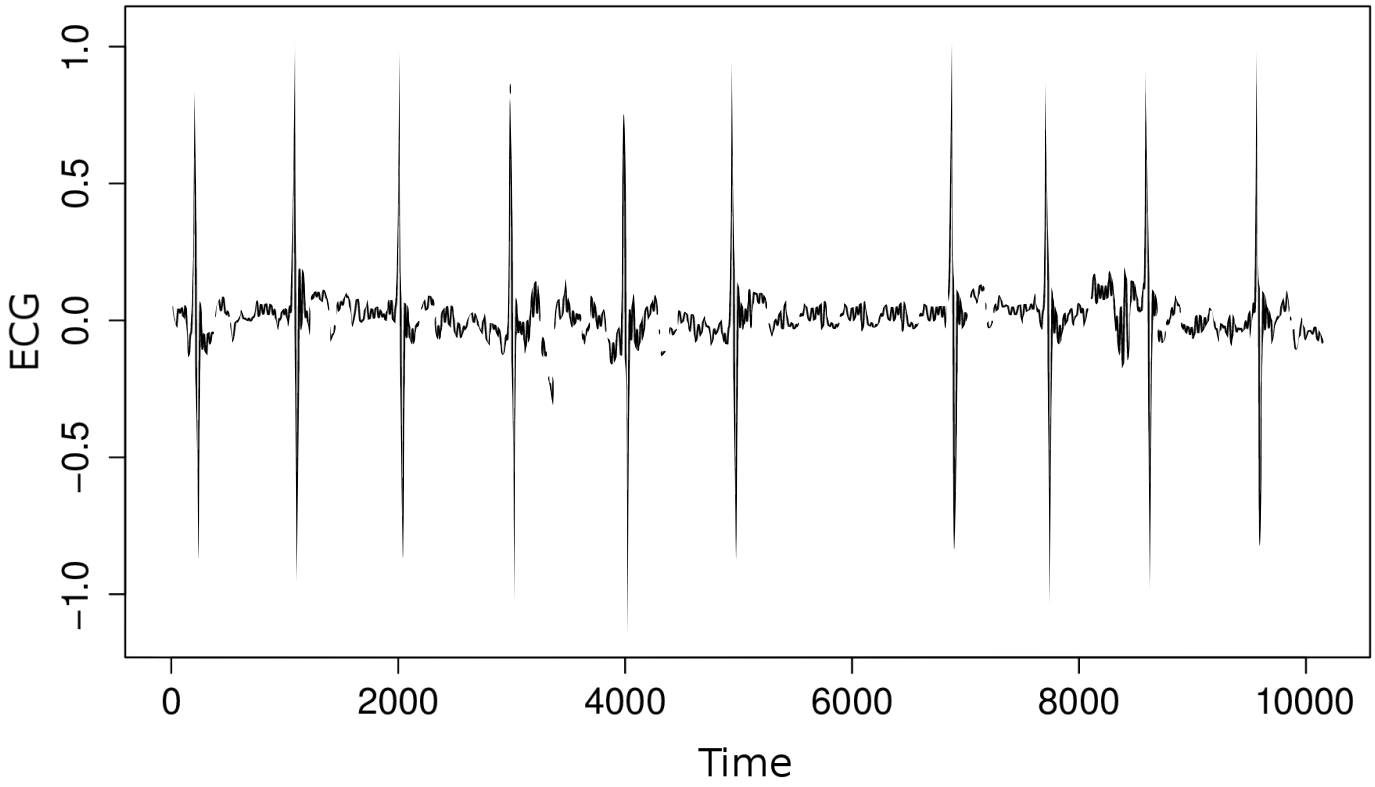
Suppose an individual usually has a weekly shopping bill of $100, except during the Christmas week, when it reaches $1,000.

A new purchase of $1,000 in a week in July will be considered a contextual anomaly, since it does not conform to the normal behavior of the individual in the context of time even though the same amount spent during Christmas week will be considered normal.



Collective Anomaly

If a **collection of related data instances** is anomalous with respect to the entire data set, it is termed a collective anomaly. The individual data instances in a collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous.



An example of the collective outlier in the human electrocardiogram. (Source: Wikipedia)

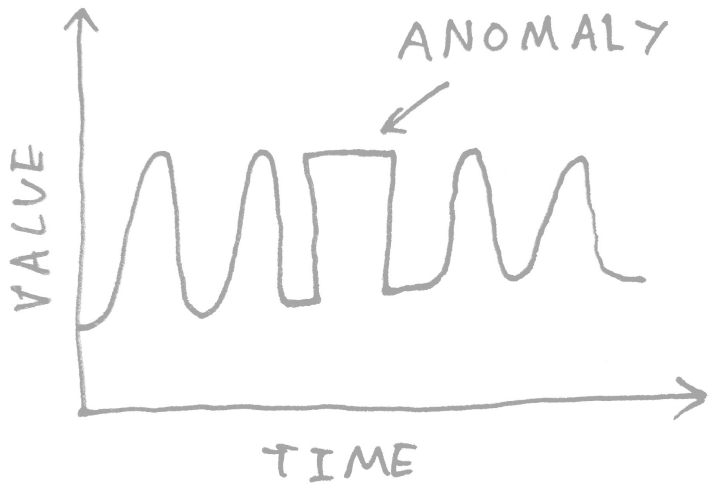
Collective Anomaly: Example

Consider a sequence of actions occurring in a computer as shown below:

... http-web, buffer-overflow, http-web, http-web, smtp-mail, ftp, http-web, ssh, smtp-mail, http-web, **ssh**, **buffer-overflow**, **ftp**, http-web, ftp, smtp-mail, http-web . . .

The highlighted sequence of events (**ssh**, **buffer-overflow**, **ftp** [!!}) correspond to a typical Web-based attack by a remote machine followed by copying of data from the host computer to a remote destination via ftp.

It should be noted that this collection of events is an anomaly, but the individual events are not anomalies when they occur in other locations in the sequence.



Buffer Overflow

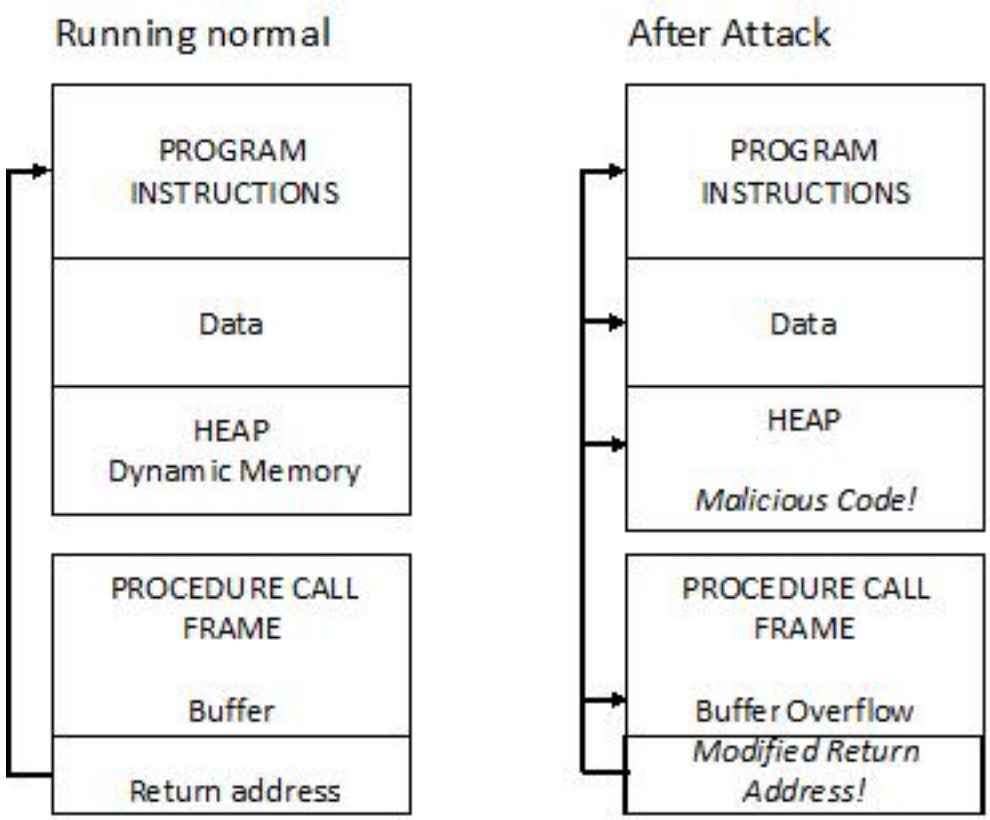
In information security and programming, a buffer overflow (or overrun), is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations.

- Buffers are areas of memory set aside to hold data, often while moving it from one section of a program to another, or between programs.

- Buffer overflows can be triggered by malformed inputs; if one assumes all inputs will be smaller than a certain size and the buffer is created to be that size, then an anomalous transaction that produces more data could cause it to write past the end of the buffer. If this overwrites adjacent data or executable code, this may cause erratic program behavior, including memory access errors, incorrect results, and crashes.

- Exploiting the behavior of a buffer overflow is a **well-known security exploit**. On many systems, the memory layout of a program, or the system as a whole, is well defined. By sending in data designed to cause a buffer overflow, it is possible to write into areas known to hold executable code and **replace it with malicious code**, or to selectively overwrite data pertaining to the program's state, therefore causing behavior that was not intended by the original programmer.

- Buffers are widespread in operating system (OS) code, so it is possible to make attacks that perform **privilege escalation** and gain unlimited access to the computer's resources.



Attacker plants code that overflows buffer and corrupts the return address. Instead of returning to the appropriate calling procedure, the modified return address returns control to malicious code, located elsewhere in process memory.

Recurring Anomaly Detection System (Source: NASA)

Concluding observations on anomalies

It should be noted that while point anomalies can occur in any data set,

- collective anomalies can occur only in data sets in which data instances are related;
- contextual anomalies depend on the availability of context attributes in the data.

⚐ Note: A point anomaly or a collective anomaly can also be a contextual anomaly if analyzed with respect to a context. Thus a point anomaly detection problem or collective anomaly detection problem can be **transformed** to a contextual anomaly detection problem by incorporating the context information.

Data labels

The labels associated with **a data instance** denote whether that instance is *normal* or *anomalous*.

Obtaining labeled data that is accurate as well as representative of all types of behaviors, is often **prohibitively expensive**. Labeling is often done **manually** by a human expert and hence substantial effort is required to obtain the labeled training data set.

Typically, getting a labeled set of anomalous data instances that covers **all possible types** (!) of anomalous behavior is more difficult than getting labels for normal behavior.

Anomalous behavior is often **dynamic in nature**, for example, new types of anomalies might arise, for which there is no labeled training data. In certain cases, such as air traffic safety, ``anomalous instances would translate to catastrophic events, and hence are very rare" ...

Recall: Based on the extent to which the labels are available, anomaly detection techniques can operate in one of the following three modes:

- Supervised Anomaly Detection
- Semisupervised Anomaly Detection
- Unsupervised Anomaly Detection

Supervised Anomaly Detection

Techniques trained in supervised mode assume the availability of a training data set that has labeled instances for normal as well as anomaly classes.

A typical approach in such cases is to **build a predictive model** for "normal vs. anomaly" classes. Any unseen data instance is compared against the model to determine which class it belongs to.

There are two major issues that arise in supervised anomaly detection.

1. Anomalous instances are **far fewer** compared to the normal instances in the training data.

2. Obtaining accurate and representative labels, especially for the anomaly class is usually challenging.

Semisupervised Anomaly Detection

Techniques that operate in a semisupervised mode, assume that ``the training data has labeled instances **only for the normal class**" … or is essentially comprised of normal instances!

Since they do not require labels for the anomaly class, they are more widely applicable than supervised techniques. E.g., in spacecraft fault detection, an anomaly scenario may signify an accident, which is not easy to model. … Boeing Vancouver Labs for Data Analytics

> 7 Minutes of Terror
> https://www.youtube.com/watch?v=Ki_Af_09Q9s

The typical approach used in such techniques is to build a model for the class corresponding to normal behavior, and use the model to **identify anomalies** in the test data.

Rare exception: A limited set of anomaly detection techniques exists that assumes availability of **only the anomaly instances** for training. Such techniques are not commonly used, primarily because it is difficult to obtain a training data set that covers every possible anomalous behavior that can occur in the data.

Unsupervised Anomaly Detection

Techniques that operate in unsupervised mode do not require training data, and thus are most widely applicable.

The techniques in this category make the implicit assumption that normal instances are **far more frequent** than anomalies in the test data. If this assumption is ***not true*** then such techniques suffer from high false alarm rate.

⚐ Compromise: Many semisupervised techniques can be adapted to operate in an unsupervised mode by using a sample of the unlabeled data set as training data. Such adaptation **assumes** that the test data contains "very few anomalies" **and** the model learned during training is robust to these few anomalies.

Anomaly detection output

An important aspect for any anomaly detection technique is the way in which detected **anomalies are reported**. Typically, the outputs produced by anomaly detection techniques are one of the following two types:

A. Scoring techniques assign an **anomaly score** to each instance in the test data depending on the *degree* to which that instance is considered an anomaly.

   The output of such techniques is a **ranked list of anomalies**.

   An analyst may choose to either analyze the **top few anomalies** or use a cutoff threshold to select the anomalies. … *example from maritime intelligence!*

B. Assign a binary ***label*** (normal or anomalous) to each test instance.

Scoring-based anomaly detection techniques allow the analyst to use a ***domain-specific threshold*** to select the most relevant anomalies.

Techniques that provide binary labels to the test instances do not directly allow the analysts to make such a choice, though this can be controlled indirectly through **parameter choices** within each technique.

Precision and Recall

In pattern recognition, information retrieval and binary classification, precision is the fraction of **relevant instances** among the **retrieved instances**, while recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over total relevant instances. Both precision and recall are therefore based on an understanding and measure of **relevance**.

Precision and Recall

In pattern recognition, information retrieval and binary classification, precision is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over total relevant instances. Both precision and recall are therefore based on an understanding and measure of **relevance**.

Example

A computer program for recognizing **dogs in photographs** identifies 8 "dogs" in a picture containing 12 dogs and some cats. Of the 8 dogs identified, 5 actually are dogs (true positives), while the rest are cats (false positives).

Thus, the program's precision is 5/8, while its recall is 5/12.

- Precision can be seen as a measure of *exactness or quality*.
- Recall is a measure of *completeness or quantity*.

Precision and Recall

In pattern recognition, information retrieval and binary classification, precision is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over total relevant instances. Both precision and recall are therefore based on an understanding and measure of **relevance**.

Example

A computer program for recognizing **dogs in photographs** identifies 8 dogs in a picture containing 12 dogs and some cats. Of the 8 dogs identified, 5 actually are dogs (true positives), while the rest are cats (false positives).

The program's precision is 5/8, while its recall is 5/12.

- Precision can be seen as a measure of *exactness or quality*.
- Recall is a measure of *completeness or quantity*.

Intuitively, high precision means that an algorithm returns substantially more relevant results than irrelevant ones, while high recall means that it returns most of the relevant results.

Often, there is an **inverse relationship** between precision and recall, where it is possible to increase one at the cost of reducing the other.

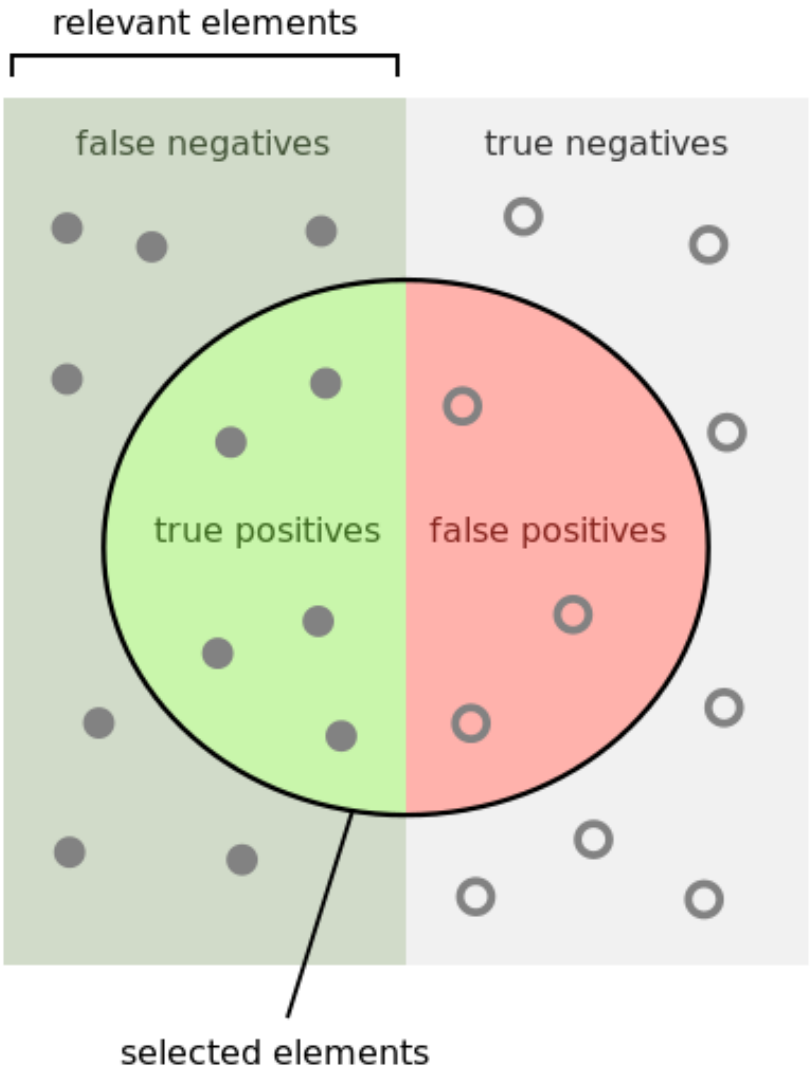
Classification context

For classification tasks, the terms true positives, *true negatives*, false positives, and *false negatives* compare the results of the ***classifier under test*** with ***trusted external judgments***.

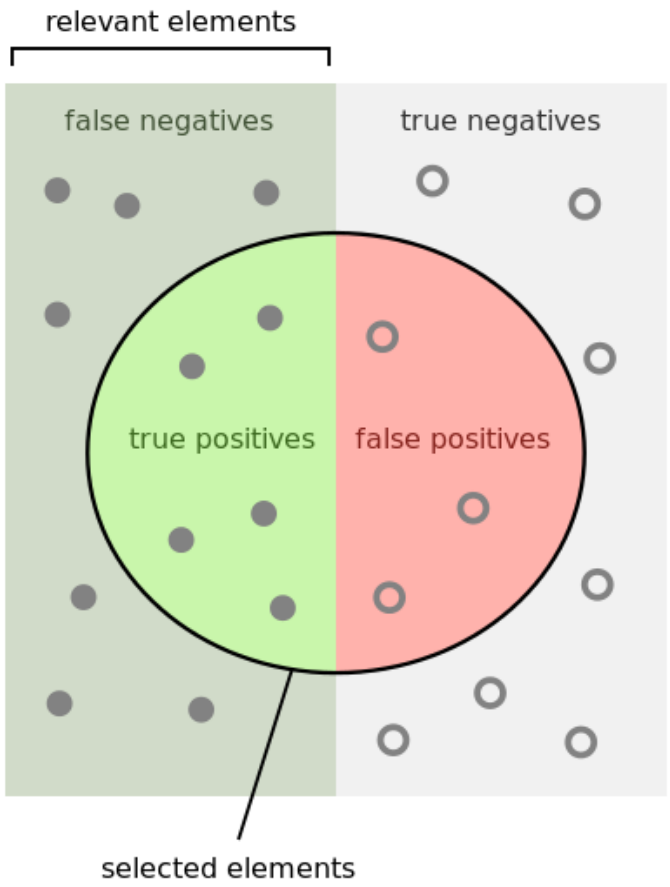
The terms `positive' and `negative' refer to the **classifier's prediction** (a.k.a. the *expectation*), and the terms *true* and *false* refer to whether that prediction corresponds to the **external judgment** (a.k.a. the *observation*).

Alternative terminology:

- true positive (TP) – eqv. with hit
- true negative (TN) – eqv. with correct rejection
- false positive (FP) – eqv. with false alarm
- false negative (FN) – eqv. with miss



Source: Wikipedia



How many selected items are relevant?

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

How many relevant items are selected?

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

F-measure

A measure that combines precision and recall is the ***harmonic mean*** of precision and recall, known as the traditional F-measure (or balanced F-score):

$$F = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

This measure is also known as the F_1 measure, because recall and precision are **evenly weighted**. It is approximately the average of the two when they are close, and is more generally the harmonic mean, which, for the case of two numbers, coincides with the square of the geometric mean divided by the arithmetic mean.

There are several reasons that the F-score can be criticized in particular circumstances due to its bias as an evaluation metric.

Two other commonly used F measures are the F_2 measure, which weights recall higher than precision, and the $F_{0.5}$ measure, which puts more emphasis on precision than recall.

Harmonic mean

The **harmonic mean** can be expressed as the reciprocal of the arithmetic mean of the reciprocals of the given set of observations.

Example

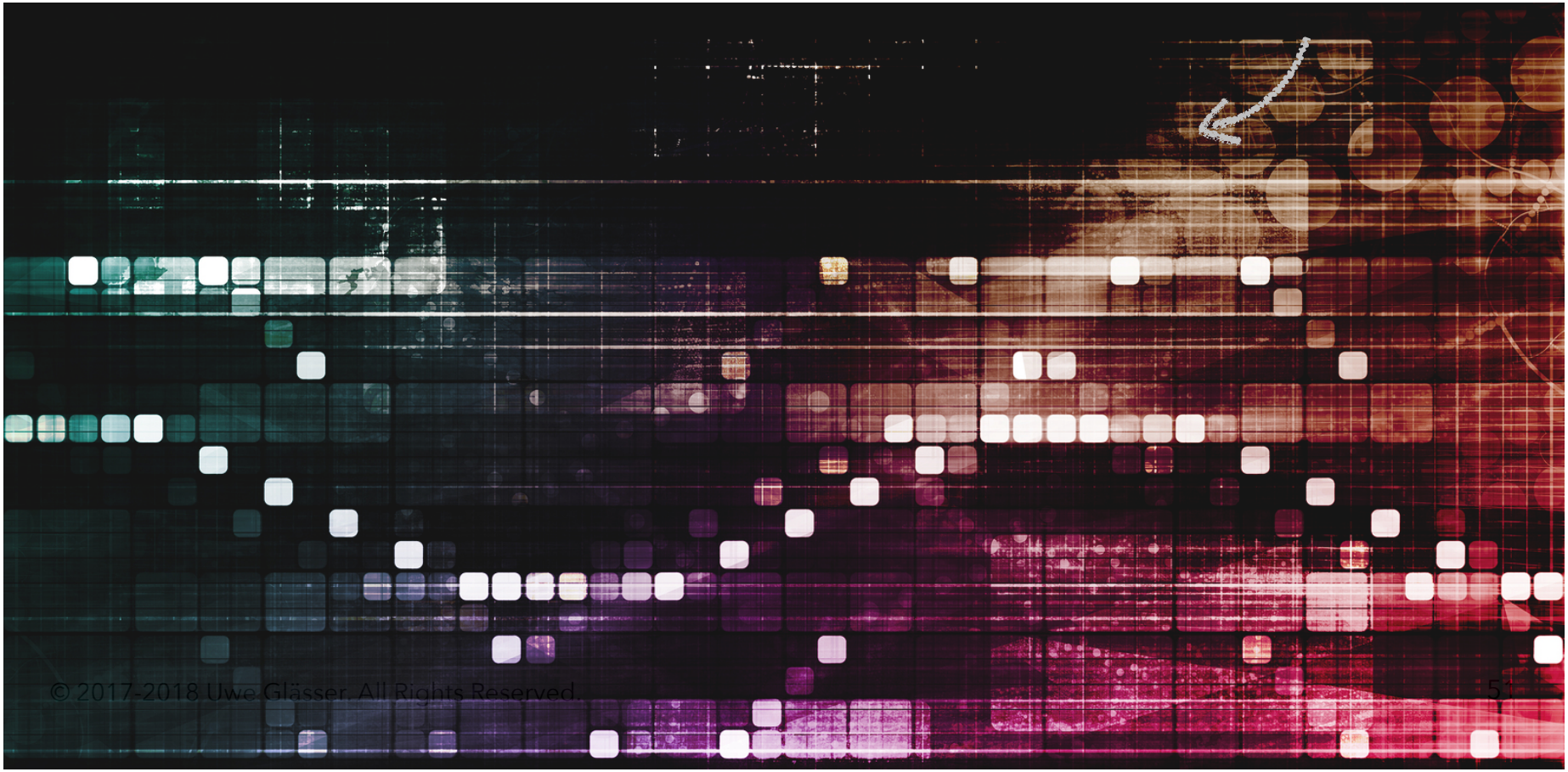
The harmonic mean of the three values1, 4, and 4 is

$$\left(\frac{1^{-1} + 4^{-1} + 4^{-1}}{3} \right)^{-1} = \frac{3}{\frac{1}{1} + \frac{1}{4} + \frac{1}{4}} = \frac{3}{1.5} = 2\,.$$

Applications of Anomaly Detection

Application domain specific characteristics to be addressed:

- nature of the data
- the notion of anomaly
- challenges associated with detecting anomalies
- existing anomaly detection techniques



Intrusion Detection

Intrusion detection refers to detection of suspicious and malicious activity—like break-ins, penetrations, and other forms of computer and computer network abuse.

An intrusion is different from the normal behavior of the system (... really always?), and hence anomaly detection techniques are applicable in the intrusion detection domain.

The key challenge for anomaly detection in this domain is the huge volume of data.

- Anomaly detection techniques need to be computationally efficient to handle large inputs. The data typically comes in a streaming fashion, thereby requiring online analysis (processing data in real time).
- Another issue that arises because of the large sized input is the false alarm rate. When data amounts to millions of data objects, **a few percent of false alarms** can make analysis overwhelming for an analyst.

Labeled data for normal behavior is usually available, while labels for intrusions are not. Thus, semisupervised and unsupervised anomaly detection techniques are preferred in this domain.
Note that `labeled data' often means historic data from mostly normal behavior.

Intrusion Detection System

An IDS is a device and/or software application that monitors a network or system for **malicious activity** or **policy violations**.

Any detected activity or violation is reported either to an administrator or collected centrally using **security information and event management** (SIEM) software products and services, combining security information management (SIM) and security event management (SEM) to provide real-time analysis of security alerts.

A wide spectrum of IDS extends from antivirus software to hierarchical systems that monitor the traffic of an entire backbone network.

Intrusion Detection System

An IDS is a device and/or software application that monitors a network or system for **malicious activity** or **policy violations**.

Any detected activity or violation is reported either to an administrator or collected centrally using **security information and event management** (SIEM) software products and services, combining security information management (SIM) and security event management (SEM) to provide real-time analysis of security alerts.

A wide spectrum of IDS extends from antivirus software to hierarchical systems that monitor the traffic of an entire backbone network.

Classification by where detection takes place (network or host):

- **host-based IDS** (HIDS) – e.g., a system that monitors important operating system files
- **network IDS** (NIDS) – e.g., a system that analyzes incoming network traffic

Classification by the detection method that is employed:

- **signature-based detection** – recognizing bad patterns, such as malware
- **anomaly-based detection** – detecting deviations from a model of "good" traffic (often relying on machine learning).

IDS that can respond to intrusions are typically referred to as **intrusion prevention system** (IPS).

Comparison with firewalls

In computing, a **firewall** is a network security system that monitors and controls the incoming and outgoing network traffic based on predetermined **security rules**. A firewall typically establishes a barrier between a trusted, secure internal network and an outside network, such as the Internet, that is considered not to be secure or trusted.

An IDS/ IPS differs from a firewall in that a firewall looks outward for intrusions in order to stop them from happening.

Firewalls limit access between networks to prevent intrusions and **do not** signal an attack from inside the network.

An IDS evaluates a suspected intrusion once it has taken place and signals an alarm. An IDS also watches for attacks that **originate from within a system**. This is traditionally achieved by examining network communications, identifying heuristics and patterns (often known as signatures) of common computer attacks, and taking action to alert operators.

A system that **terminates connections** is called an intrusion prevention system.

Sensor Networks

Anomalies in data collected from a sensor network has become **an important topic of research** from the data analysis perspective since sensor data collected from various (wireless) sensors has several unique characteristics.

Anomaly detection in sensor networks can capture *sensor fault* detection or *intrusion* detection or both. A sensor network may be comprised of sensors that collect **different types of data**, such as binary, discrete, continuous, audio, video, etc. The data is generated in a **streaming mode**. The environment in which the various sensors are deployed, and the communication channel, often induce **noise** and **missing values** in the collected data.

Anomaly detection in sensor networks poses unique challenges. The anomaly detection techniques are required to **operate online** (… *in real time!*) Due to resource constraints, the anomaly detection techniques need to be lightweight.

Another challenge is that data is collected in a distributed fashion, and hence a distributed data mining approach is required to analyze it.

Classification Based Anomaly Detection

Classification is used to learn a model (*classifier*) from a set of labeled data instances (*training*) and then classify a test instance into one of the classes using the learned model (*testing*).

Classification-based **anomaly detection** operates in a similar two-phase fashion. The training phase learns a classifier using the available labeled training data. The testing phase classifies a test instance as normal or anomalous, using the classifier.

General assumption: A classifier that can distinguish between normal and anomalous classes can be **learned in the given *feature space***.

Classification Based Anomaly Detection

Classification is used to learn a model (*classifier*) from a set of labeled data instances (*training*) and then classify a test instance into one of the classes using the learned model (*testing*).

Classification-based **anomaly detection** operates in a similar two-phase fashion. The training phase learns a classifier using the available labeled training data. The testing phase classifies a test instance as normal or anomalous, using the classifier.

General assumption: A classifier that can distinguish between normal and anomalous classes can be **learned in the given *feature space***.

Classification-based anomaly detection can be grouped (based on the labels available for the training phase) into two broad categories: ***multi-class*** and ***one-class*** anomaly detection techniques.

- Multi-class anomaly detection assumes that the training data contains labeled instances belonging to multiple normal classes, where the classifier distinguishes between each normal class and the rest of the data.
  - A test instance is considered anomalous if it is **not** classified as normal by any of the classifiers. Some techniques in this subcategory associate a confidence score with the prediction made by the classifier.
  - If none of the classifiers are confident in classifying the test instance as normal, the instance is declared to be anomalous.

- One-class classification based anomaly detection techniques assume that all training instances have only one class label. Such techniques learn a ***discriminative boundary*** around the normal instances using a *one-class classification algorithm*. Any test instance that does not fall within the learned boundary is declared as anomalous.

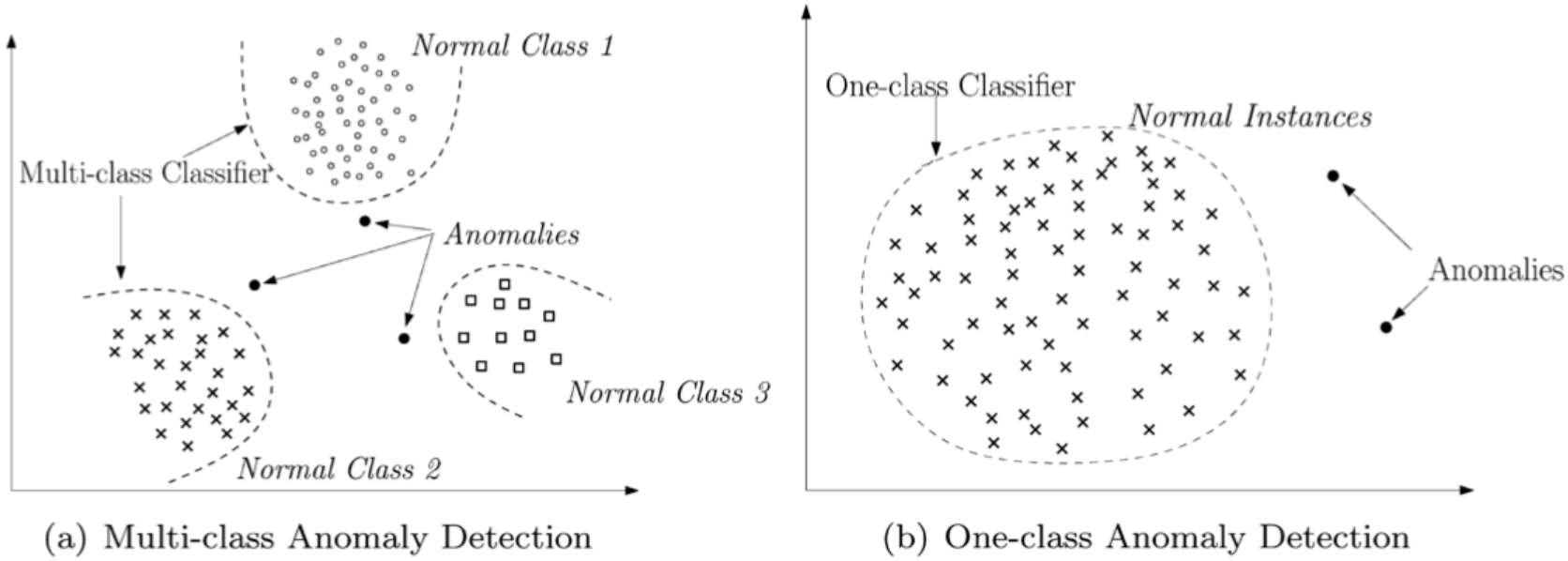
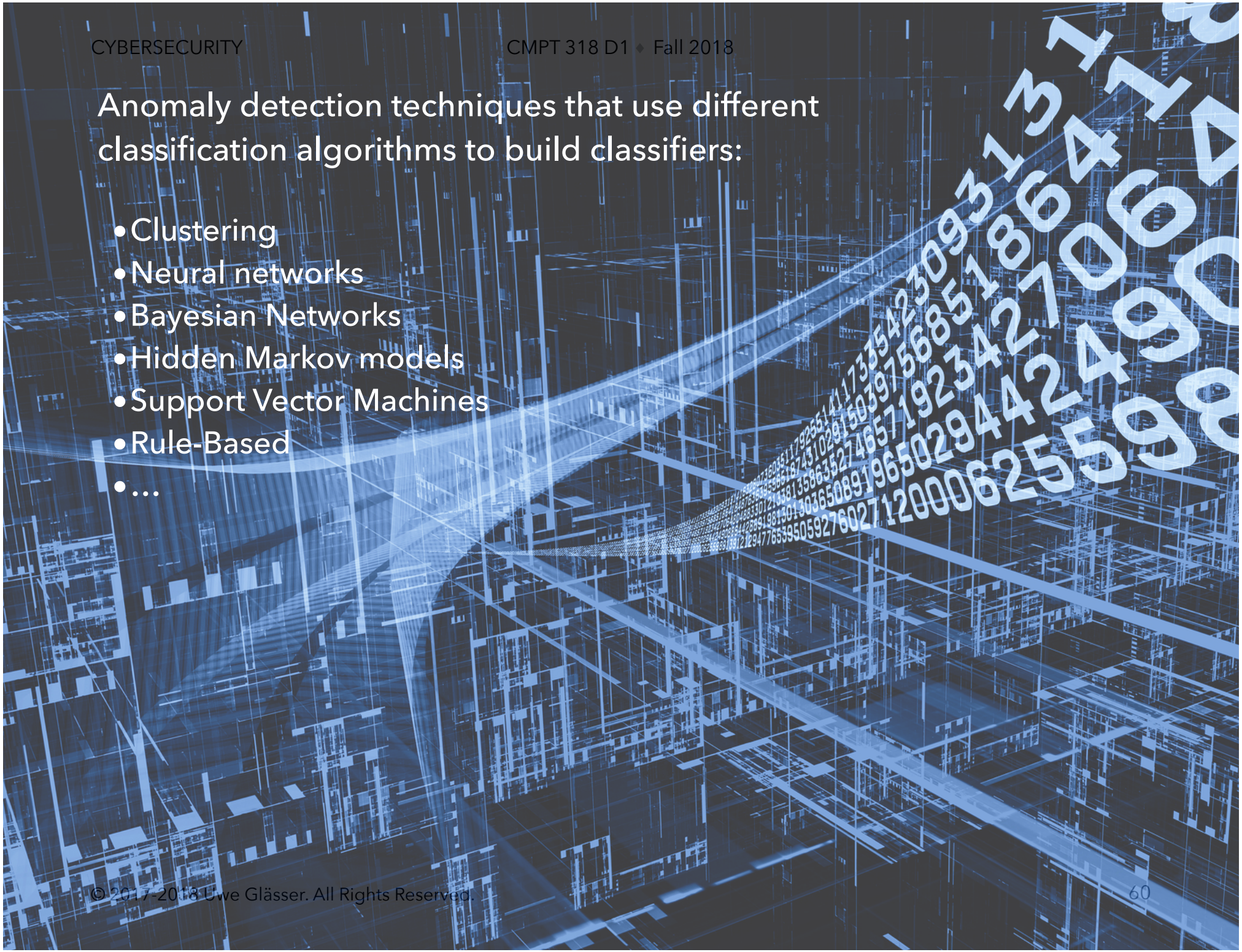


Fig. 6. Using classification for anomaly detection.

Source: Chandola et al., 2009

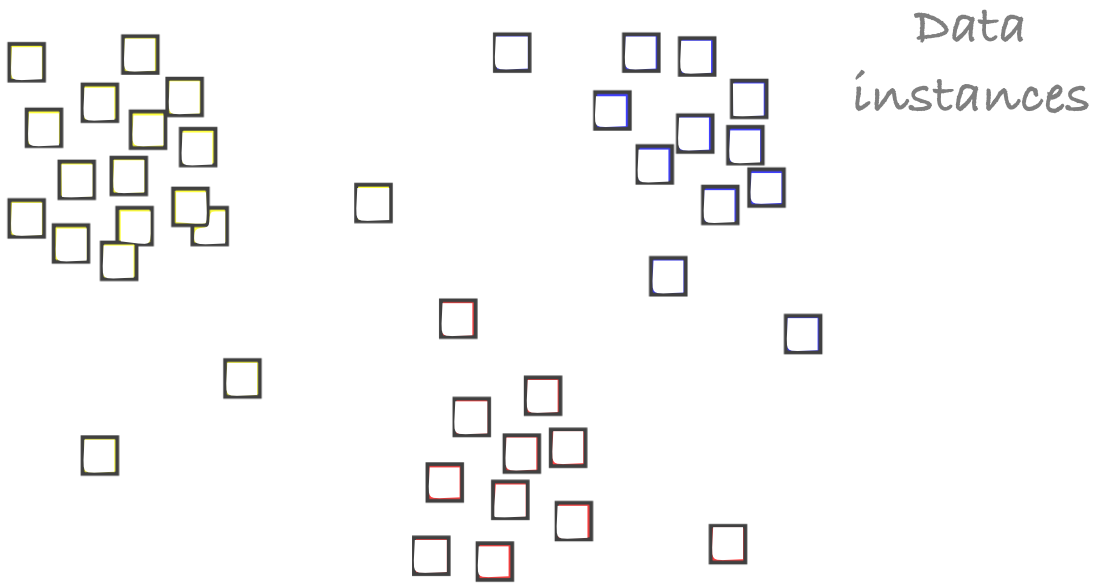
Anomaly detection techniques that use different classification algorithms to build classifiers:

- Clustering
- Neural networks
- Bayesian Networks
- Hidden Markov models
- Support Vector Machines
- Rule-Based
- …



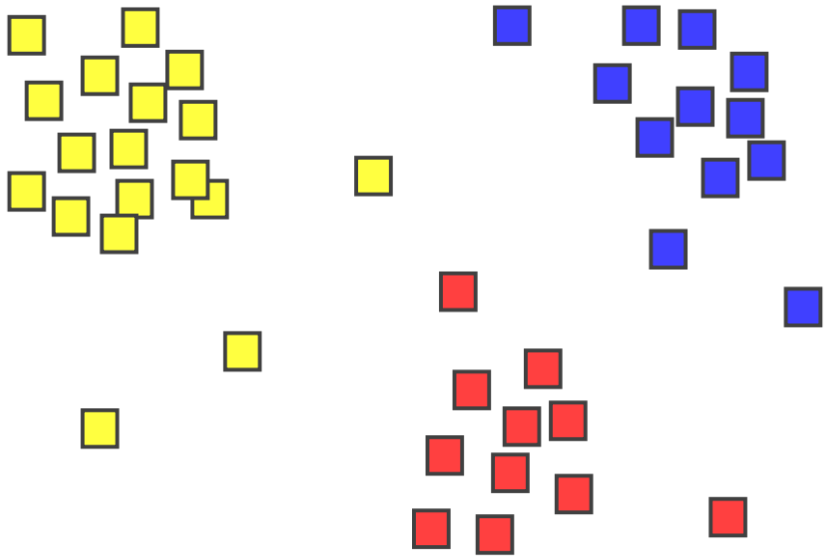
Clustering Based Anomaly Detection

Idea: Cluster analysis groups "similar data instances" into clusters; it is primarily an unsupervised technique. Even though clustering and anomaly detection appear to be fundamentally different from each other, several clustering-based anomaly detection techniques have been developed.



Clustering Based Anomaly Detection

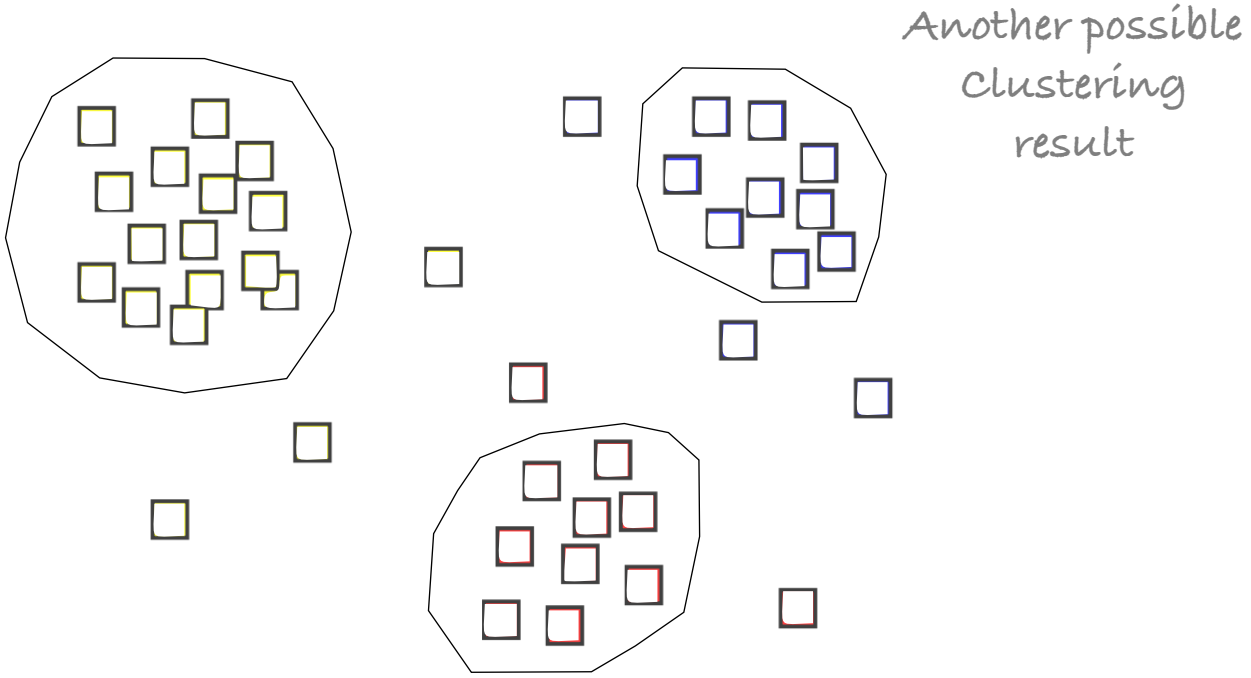
Idea: Cluster analysis groups "similar data instances" into clusters; it is primarily an unsupervised technique. Even though clustering and anomaly detection appear to be fundamentally different from each other, several clustering-based anomaly detection techniques have been developed.

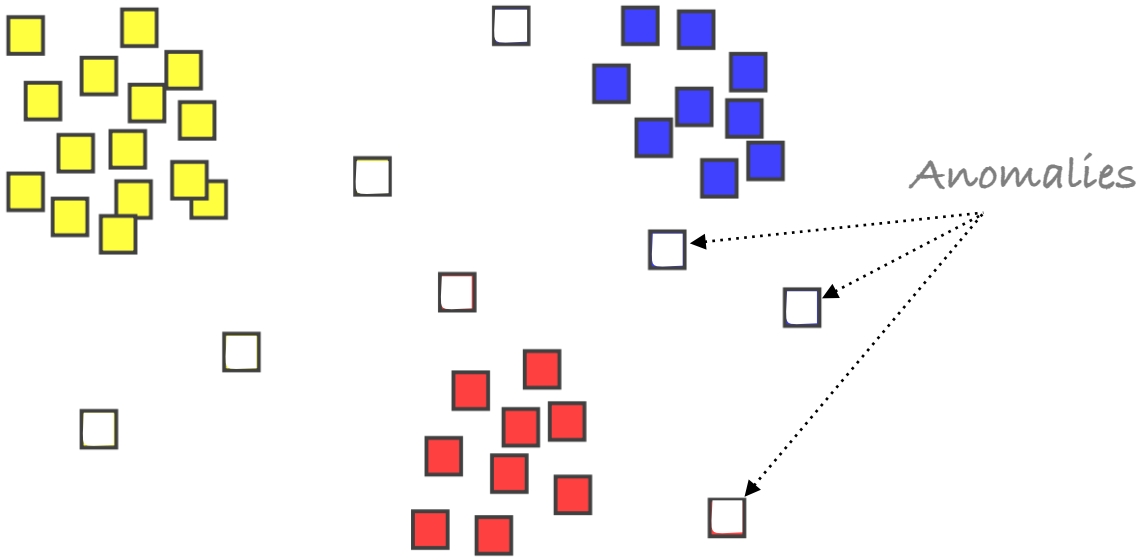


Custer analysis produces colouring of data instances into three clusters. (Source: Wikipedia)

Clustering Based Anomaly Detection

Idea: Cluster analysis groups "similar data instances" into clusters; it is primarily an unsupervised technique. Even though clustering and anomaly detection appear to be fundamentally different from each other, several clustering-based anomaly detection techniques have been developed.





Clustering-based anomaly detection techniques can be organized into **three categories**:

- First category

  Assumption: Normal data instances belong to a cluster in the data, while anomalies do not belong to any cluster.

  Disadvantage: such techniques are not optimized to **find anomalies**, since the main aim of the underlying clustering algorithm is to find clusters.

- Second category

  Assumption: Normal data instances lie close to their closest cluster centroid[2], while anomalies are far away from their closest cluster centroid.

- Third category

  Assumption: Normal data instances belong to large and dense clusters, while anomalies either belong to small or sparse clusters.

[2] Centre of gravity (mass) of a two-dimensional planar closed surface or a three-dimensional solid

DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is a *density-based clustering* algorithm proposed by Ester et al., 1996.

Given a **set of data points** in some space, the algorithm:

- groups together points that are closely packed together,
  that is points with many nearby neighbors;
- marks as outliers points that lie alone in low-density regions,
  those whose nearest neighbors are "too far away."

DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.

For the purpose of DBSCAN clustering, the points are classified as (i) *core points*, (ii) *(density-) reachable points* and (iii) *outliers*:

i) A point *p* is a core point if at least ***minPts*** points are within **distance ε** of it, including *p* itself, where *ε* is the maximum radius of the neighborhood from *p*.
Those points are said to be directly reachable from *p*.

By definition, no points are directly reachable from a non-core point.

ii) …

For the purpose of DBSCAN clustering, the points are classified as (i) *core points*, (ii) *(density-) reachable points* and (iii) *outliers*:

i) A point p is a core point if at least ***minPts*** points are within **distance ε** of it, including p itself, where ε is the maximum radius of the neighborhood from p.
Those points are said to be directly reachable from p.

By definition, no points are directly reachable from a non-core point.

ii) A point q is (density-) reachable from p if there is a path $p_1, ..., p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i and all the points on the path must be core points, with the exception of the end point q.

iii) All points not reachable from any other point are outliers (or noise).

For the purpose of DBSCAN clustering, the points are classified as (i) *core points*, (ii) *(density-) reachable points* and (iii) *outliers*:

i) A point p is a core point if at least ***minPts*** points are within **distance ε** of it, including p itself, where ε is the maximum radius of the neighborhood from p.
Those points are said to be directly reachable from p.

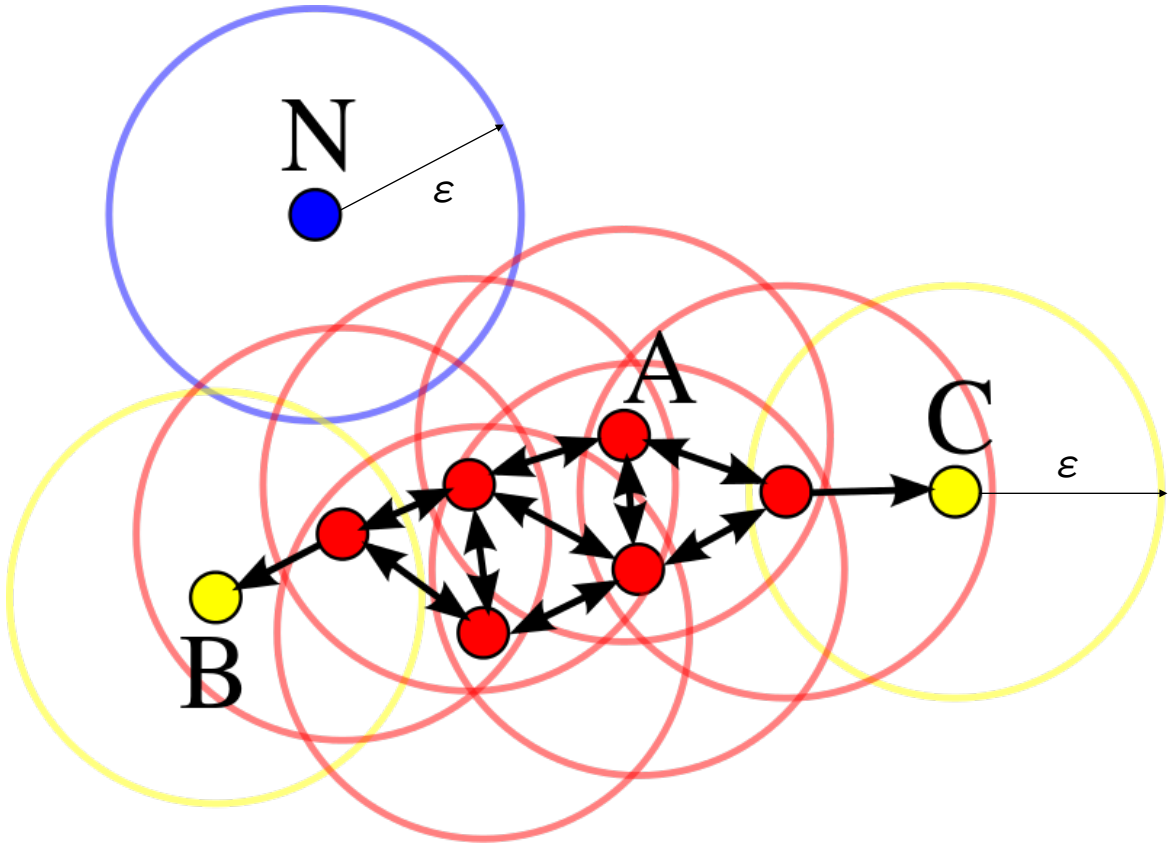
By definition, no points are directly reachable from a non-core point.

ii) A point q is (density-) reachable from p if there is a path $p_1, ..., p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i and all the points on the path must be core points, with the exception of the end point q.

iii) All points not reachable from any other point are outliers (or noise).

A core point p forms a **cluster** together with all points (core or non-core) reachable from it.

Each cluster contains at least one core point; non-core points can be part of a cluster, but they form its "edge", since they cannot be used to reach more points.



Assume *minPts* = 4. Point A and the other red points are core points, because the area surrounding these points in an ε radius contains at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is an outlier as it is neither a core point nor directly-reachable.

Reachability is **not a symmetric relation** since, by definition, no point may be reachable from a non-core point, regardless of distance (so a non-core point may be reachable, but nothing can be reached from it). Therefore, a further notion of connectedness is needed to formally define the extent of the clusters found by DBSCAN.

iv) Two points *p* and *q* are *density-connected* if there is a point *o* such that both *p* and *q* are density-reachable from *o*. Density-connectedness is **symmetric**.

Each cluster then satisfies the following two properties:

1. All points within the cluster are mutually **density-connected**.
2. If a point is density-reachable from any point of the cluster, it is part of the cluster as well.

Algorithm

DBSCAN requires **two parameters**: ε (*eps*) and the minimum number of points required to form a dense region (*minPts*).

Intuitive description

Starting with an arbitrary starting point *p* that has not been visited, this point's ε-neighborhood is retrieved. If it contains sufficiently many points, **a cluster is started**. Otherwise, *p* is labeled as **noise**.

- Note that this point might later be found in a sufficiently sized ε-environment of a different point and hence be made part of a cluster.

If a point is found to be a dense part of a cluster, its ε-neighborhood is also part of that cluster. Hence, all points that are found within the ε-neighborhood are added, as is their own ε-neighborhood when they are also dense.

This process continues until the density-connected cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a new cluster or noise.

The algorithm described in pseudocode (following the original published nomenclature):

```
DBSCAN(D, eps, MinPts) {
   C = 0
   for each point P in dataset D {
      if P is visited
         continue next point
      mark P as visited
      NeighborPts = regionQuery(P, eps)
      if sizeof(NeighborPts) < MinPts
         mark P as NOISE
      else {
         C = next cluster
         expandCluster(P, NeighborPts, C, eps, MinPts)
      }
   }
}
```

```
expandCluster(P, NeighborPts, C, eps, MinPts) {
   add P to cluster C
   for each point P' in NeighborPts {
      if P' is not visited {
         mark P' as visited
         NeighborPts' = regionQuery(P', eps)
         if sizeof(NeighborPts') >= MinPts
            NeighborPts = NeighborPts joined with NeighborPts'
      }
      if P' is not yet member of any cluster
         add P' to cluster C
   }
}



regionQuery(P, eps)
   return all points within P's eps-neighborhood (including P)
```

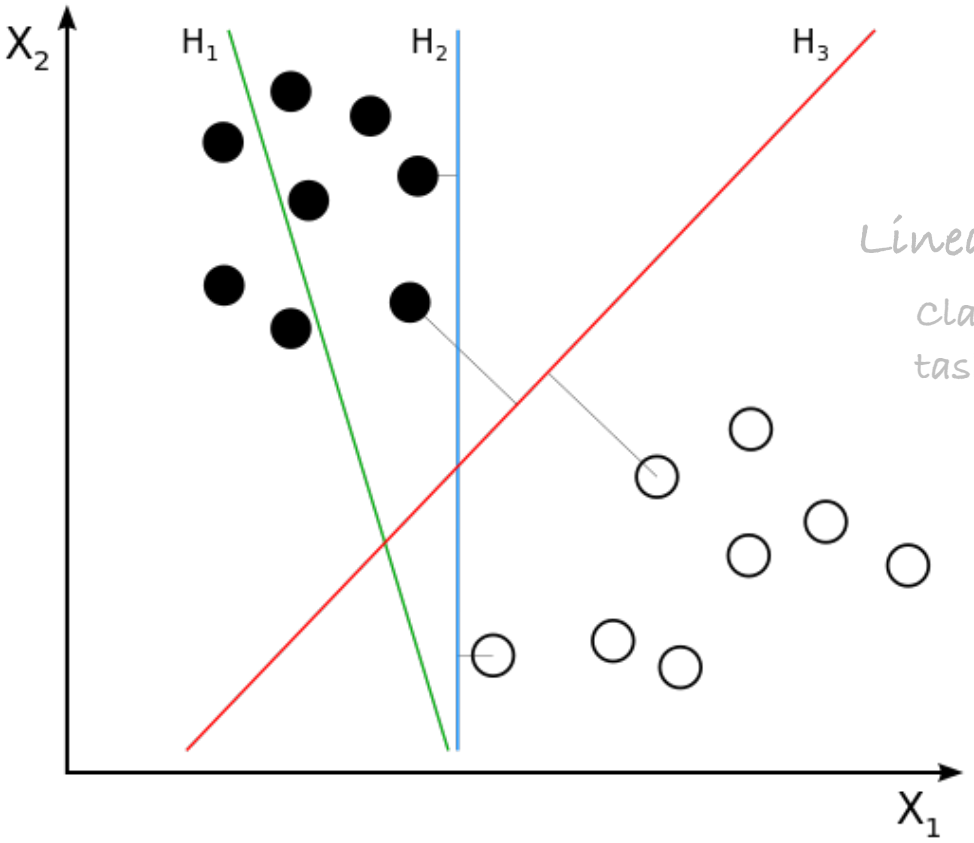
Properties

- DBSCAN does not require to specify the #(clusters) in the data a priori, as opposed to *k*-means.
- DBSCAN is not entirely deterministic: border points that are reachable from **more than one cluster** can be part of either cluster, depending on the order the data is processed.

  For most data sets and domains, this situation fortunately does not arise often and has little impact on the clustering result: both on core points and noise points, DBSCAN is deterministic.

  DBSCAN* is a variation that treats border points as **noise**, and this way achieves a fully deterministic result as well as a more consistent statistical interpretation of density-connected components.

  While ***minPts*** is intuitively the minimum cluster size, in some cases DBSCAN can produce smaller clusters. A cluster consists of at least one core point. As other points may be border points to more than one cluster, there is no guarantee that at least *minPts* points are included in every cluster.
- DBSCAN cannot cluster data sets well with large differences in densities, since the *minPts*-ε combination cannot then be chosen appropriately for all clusters.
- DBSCAN can find non-linearly separable clusters.



Linear separability

Classifying data is a common task in machine learning.

Source: Zack Weinberg, https://commons.wikimedia.org/w/index.php?curid=22877598

Each data point is viewed as a **p-dimensional vector**, and we want to know whether we can separate such points with a **(p − 1)-dimensional hyperplane**. This is called a linear classifier.

Example: H1 does not separate the sets. H2 does, but only with a small margin. Finally, H3 separates them with the maximum margin.

Parameter estimation

For DBSCAN, the parameters ε and *minPts* influence the algorithm in specific ways and need to be set by the user. Ideally, the value of ε is given by the problem to solve (e.g. a physical distance), and *minPts* is then the desired minimum cluster size.

- As a rule of thumb, a minimum *minPts* can be derived from the number of dimensions D in the data set, as $minPts \geq D + 1$. The low value of $minPts = 1$ does not make sense, as then every point on its own will already be a cluster. Larger values are usually better for data sets with noise and will yield more significant clusters. The larger the data set, the larger the value of *minPts* should be chosen.

- If ε is chosen much **too small**, a large part of the data will not be clustered; whereas for a **too high** value of ε, clusters will merge and the majority of objects will be in the same cluster. In general, small values of ε are preferable, and as a rule of thumb only a small fraction of points should be within this distance of each other.

Complexity

DBSCAN visits each point of the database, possibly multiple times (e.g., as candidates to different clusters). For practical considerations, however, the time complexity is mostly governed by the number of **regionQuery** invocations.

- **TIME:** DBSCAN executes exactly one **regionQuery** query for each point, and if an indexing structure is used that executes a neighborhood query in O(log n), an overall **average runtime complexity** of O(n log n) is obtained (if parameter ε is chosen in a meaningful way, i.e. such that on average only O(log n) points are returned).

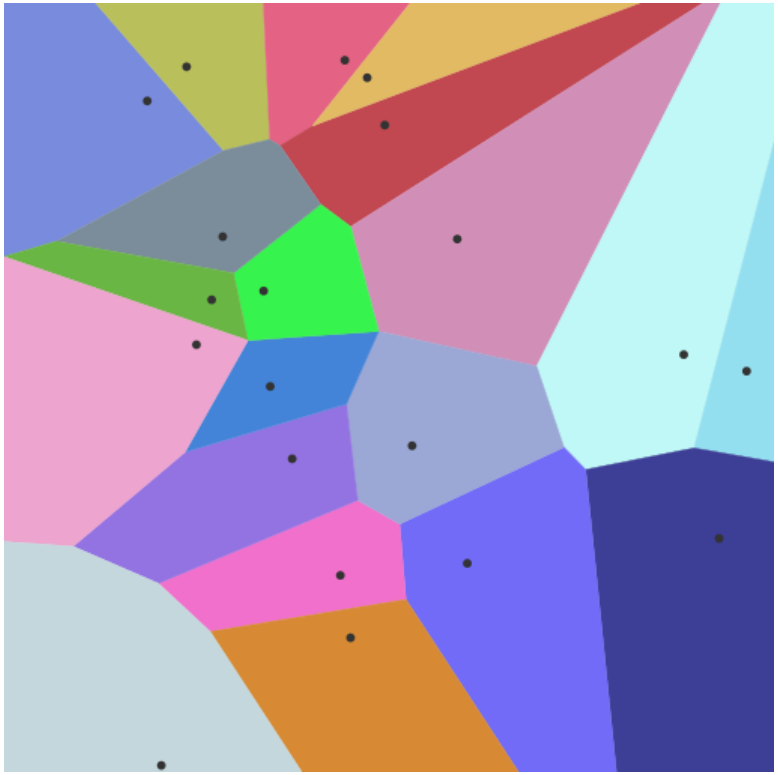
  Without the use of an accelerating index structure, or on degenerated data (e.g. all points within a distance less than ε), the **worst case run time complexity** remains O(n^2).

- **SPACE:** The distance matrix of size (n^2-n)/2 can be materialized to avoid distance re-computations, but this needs O(n^2) memory, whereas a non-matrix based implementation of DBSCAN only needs O(n) memory.

Centroid-based clustering

Clusters are represented by a central vector, or centroid, which **may not** necessarily be a member of the data set.

- When the number of clusters is fixed to *k*, *k-means clustering* aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean.
- **This is an optimization problem**: find the cluster centres and assign the objects to the nearest cluster centre, such that the squared distances from the cluster are minimized, resulting in a partitioning of the data space into **Voronoi cells**.

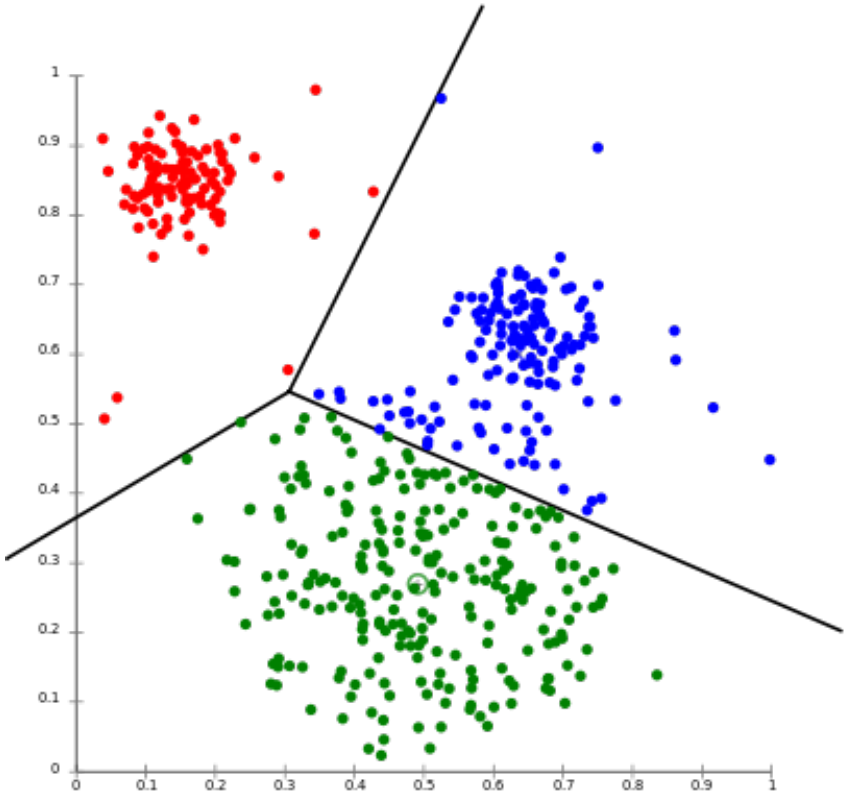


A **Voronoi diagram** is a partitioning of a plane into regions (*Voronoi cells*) based on distance to specific points in the plane (called seeds or generators). For each seed there is a corresponding region comprising all points closer to that seed than to any other.

Centroid-based clustering

Clusters are represented by a central vector, or centroid, which **may not** necessarily be a member of the data set.

- When the number of clusters is fixed to *k*, *k-means clustering* aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean.



Centroid-based clustering

Clusters are represented by a central vector, or centroid, which **may not** necessarily be a member of the data set.

- When the number of clusters is fixed to *k*, *k-means clustering* aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean.
- This is an optimization problem: find the **cluster centres** and assign the objects to the nearest cluster centre, such that the squared distances from the cluster are minimized, resulting in a partitioning of the data space into **Voronoi cells**.

Given a set of *n* observations ($x_1, x_2, \ldots, x_n$), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* ($\leq n$) sets $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares over all *k* clusters.

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

> *The arguments of the minimum (arg min) are the points of the domain of some function at which the function values are minimized.*

Centroid-based clustering

Clusters are represented by a central vector, or centroid, which **may not** necessarily be a member of the data set.

- When the number of clusters is fixed to *k*, *k-means clustering* aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean.
- This is an optimization problem: find the **cluster centres** and assign the objects to the nearest cluster centre, such that the squared distances from the cluster are minimized, resulting in a partitioning of the data space into **Voronoi cells**.

Given a set of *n* observations ($x_1, x_2, \ldots, x_n$), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* ($\leq n$) sets $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the pairwise squared deviations of points in the same cluster.

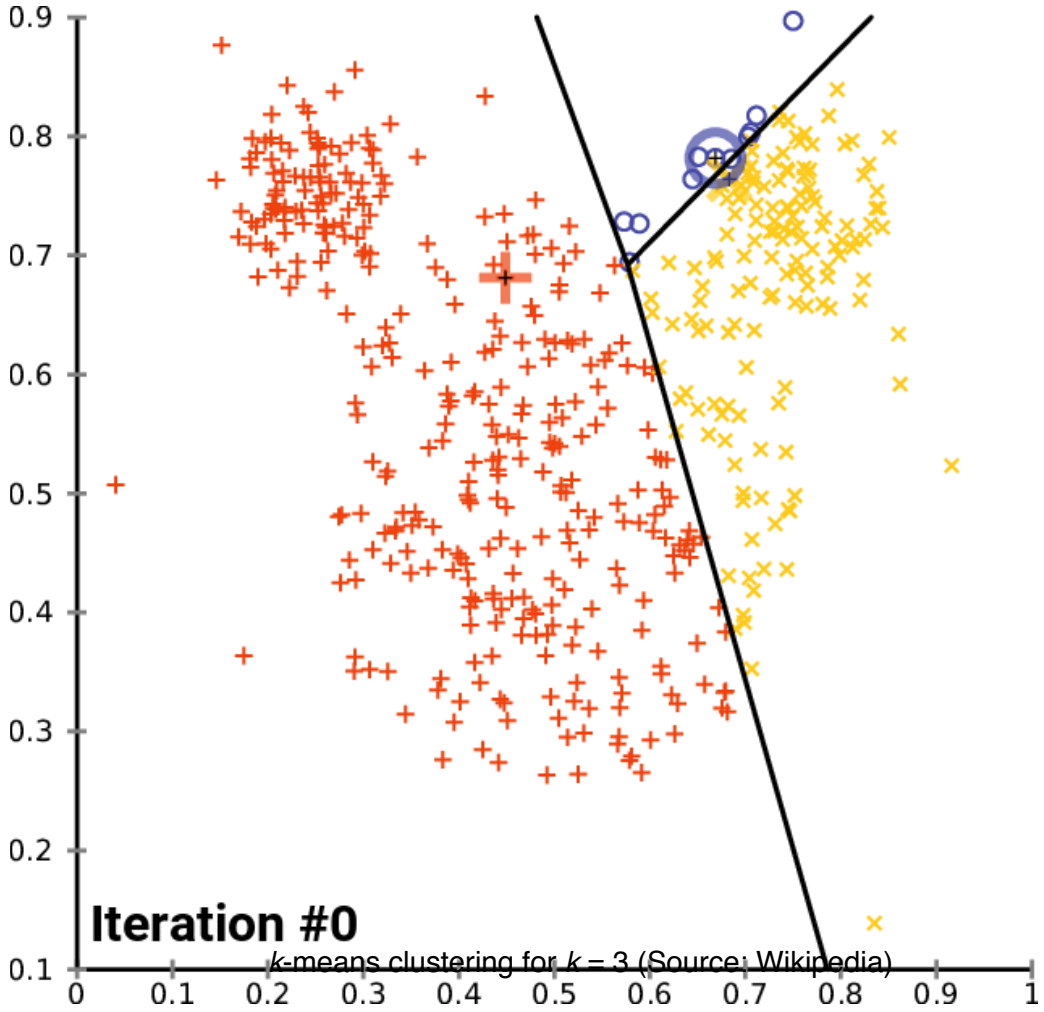
This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y}\in S_i} \|\mathbf{x}-\mathbf{y}\|^2$$

The equivalence can be deduced from identity

$$\sum_{\mathbf{x}\in S_i} \|\mathbf{x}-\boldsymbol{\mu}_i\|^2 = \sum_{\mathbf{x}\neq\mathbf{y}\in S_i} (\mathbf{x}-\boldsymbol{\mu}_i)(\boldsymbol{\mu}_i-\mathbf{y})\,.$$

The optimization problem itself is known to be **NP-hard**, and thus the common approach is to search only for approximate solutions, using efficient heuristic algorithms that are commonly converge quickly to a **local optimum**.



Convergence of *k*-means (Source: Wikipedia)

Sources

Chandola et al., 2009

Chandola, V., Banerjee, A., and Kumar, V. 2009. Anomaly detection: A survey. *ACM Computing Survey 41, 3, Article 15* (July 2009)

Ester et al., 1996

M. Ester, H-P Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, and U. Fayyad (eds.), Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231.

Yaghoubi Shahir et al., 2015

H. Yaghoubi Shahir, U. Glässer, A. Yaghoubi Shahir, and H. Wehn. Maritime Situation Analysis Framework: Vessel Interaction Classification and Anomaly Detection. In Proceedings of the *2015 IEEE International Conference on Big Data*, Santa Clara, CA, USA, IEEE (2015).