UNIVERSITÀ DEGLI STUDI DI SALERNO
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA
BIP LAB
Carmen Bisogni, PhD Student

C.A.S.A. Course

CONTEXT AWARE SECURITY ANALYTICS IN COMPUTER VISION
Lesson 3

Unsupervised learning: Why?

- Unsupervised machine learning finds all kind of unknown patterns in data.
- Unsupervised methods help you to find features which can be useful for categorization.
- It is taken place in real time, so all the input data to be analyzed and labeled in the presence of learners.
- It is easier to get unlabeled data from a computer than labeled data, which needs manual intervention.

Unsupervised Learning: data

The unsupervised learning model can organize the data in different ways.

- **Clustering:** the deep learning model looks for training data that are similar to each other and groups them together.
- **Anomaly detection:** unsupervised learning can be used to flag outliers in a dataset.
- **Association:** By looking at a couple key attributes of a data point, an unsupervised learning model can predict the other attributes with which they're commonly associated.
- **Autoencoders:** Autoencoders take input data, compress it into a code, then try to recreate the input data from that summarized code.

Unsupervised vs supervised learning

Supervised vs. Unsupervised Machine Learning

Parameters	Supervised machine learning technique	Unsupervised machine learning technique
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data which is not labelled
Computational Complexity	Supervised learning is a simpler method.	Unsupervised learning is computationally complex
Accuracy	Highly accurate and trustworthy method.	Less accurate and trustworthy method.

Unsupervised Learning: disadvantages

- You cannot get precise information regarding data sorting, and the output as data used in unsupervised learning is labeled and not known
- Less accuracy of the results is because the input data is not known and not labeled by people in advance. This means that the machine requires to do this itself.
- The spectral classes do not always correspond to informational classes.
- The user needs to spend time interpreting and label the classes which follow that classification.
- Spectral properties of classes can also change over time so you can't have the same class information while moving from one image to another.

Unsupervised Learning algorithms: Density estimation

Density estimation is the act of estimating a continuous density field from a discretely sampled set of points drawn from that density field.

Data: $D = \{D_1, D_2, .., D_n\}$

$D_i = \mathbf{x}_i$ a vector of attribute values

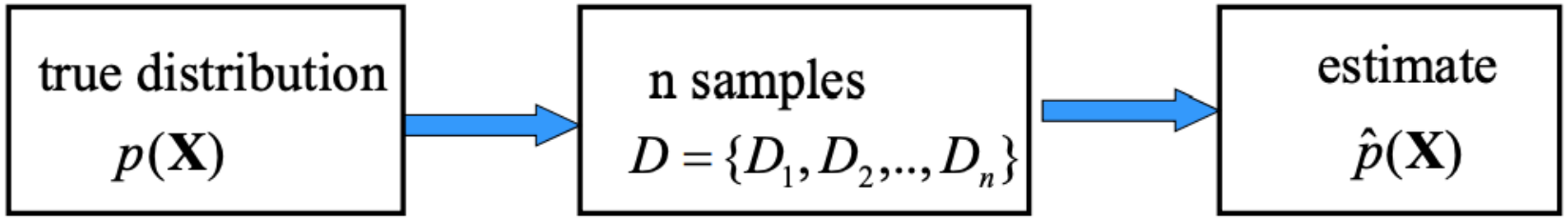
Attributes: $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$ continue or discrete value.

Density estimation attempts to learn the underlying probability distribution:

$$p(\mathbf{X}) = p(X_1, X_2, \ldots, X_d)$$

Unsupervised Learning algorithms: Density estimation

We have only few values of a real distribution and we try to estimate the distribution.



Assumptions:

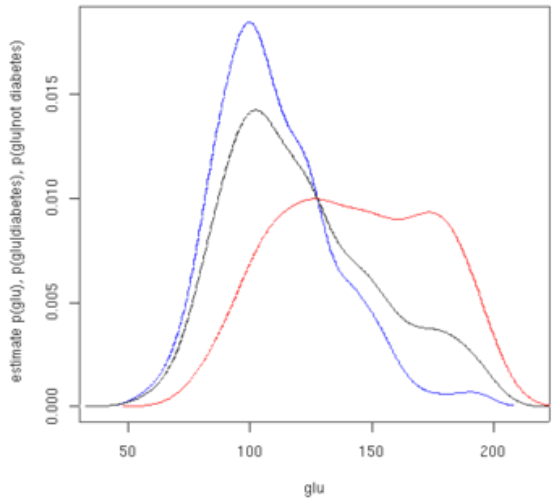
- samples are independent
- samples are identically distributed

Unsupervised Learning algorithms: Density estimation

Estimation can be parametric or non-parametric.

Parametric:
We will use a set of parameters that describe the data. We will search for $P(X|\vartheta)$, where ϑ are the parameters.

Non Parametric:
All the samples D are the parameters of the distribution. An example is nearest-neighbor.



Unsupervised Learning algorithms: Parametric Density estimation

Input:

- A set of variables $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$
- A model of the distribution over variables in X with parameters ϑ, $p'(\mathrm{X},\vartheta)$.
- Data $D = \{D_1, D_2, .., D_n\}$

Objective: find parameters ϑ such that p(X, ϑ) fits data D the best.

Parameters estimation can be performed by:

- Maximum likelihood
- Bayesian Parameters estimation
- Maximum a posteriori probability
- Expected value of the parameters

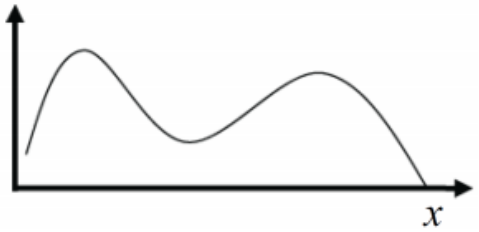
Unsupervised Learning algorithms: Parametric Density estimation

A simple example:
We have a coin that can be biased, we have two possible value, head or tail.
Let's consider a sequence of observed values D, where 1 is head and 0 is tail.
We would like to estimate the probability ϑ of a head from data.
Data: H H T T H H T H T H T T T H T H H H H T H H H H T
Head=15, Tail=10.

If we use frequencies of occurrences to do the estimate, ϑ=15/25≅ 0.6

Unsupervised Learning algorithms: Non-Parametric Density estimation

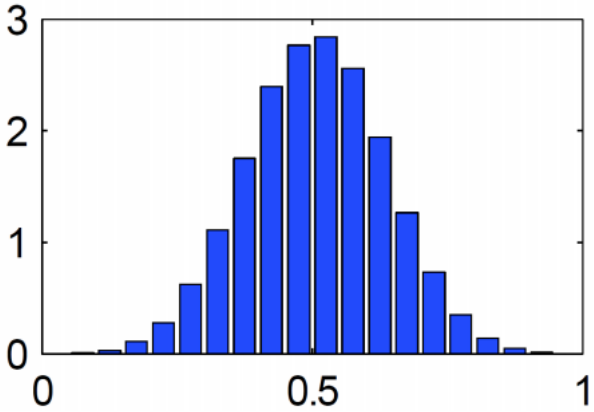
Often the functional form of the distribution is unknown, such as:



We can estimate density from:

- Histograms: Partition the data space into distinct bins with widths Δ_i and count the number of observations, n_i , in each bin (among N observations in total).

$$p_i = \frac{n_i}{N\,\Delta_i}$$



Usually, all Δ_i are the same. It can be adopted for every dimensionality

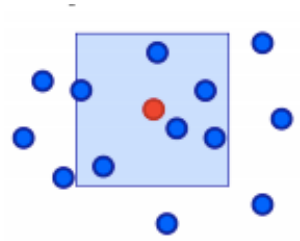
Advantages:Very simple but general. No need to store the training data once histogram is built.

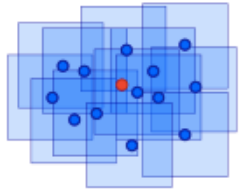
Problems: High-dimensional feature spaces. Discontinuities at bin edges. Choose of bin size.

Unsupervised Learning algorithms: Non-Parametric Density estimation

- Kernel density estimation: Similar to the histogram, we can compute the relative frequency of observations falling into a small region. Parzen window or Gaussian Kernel.

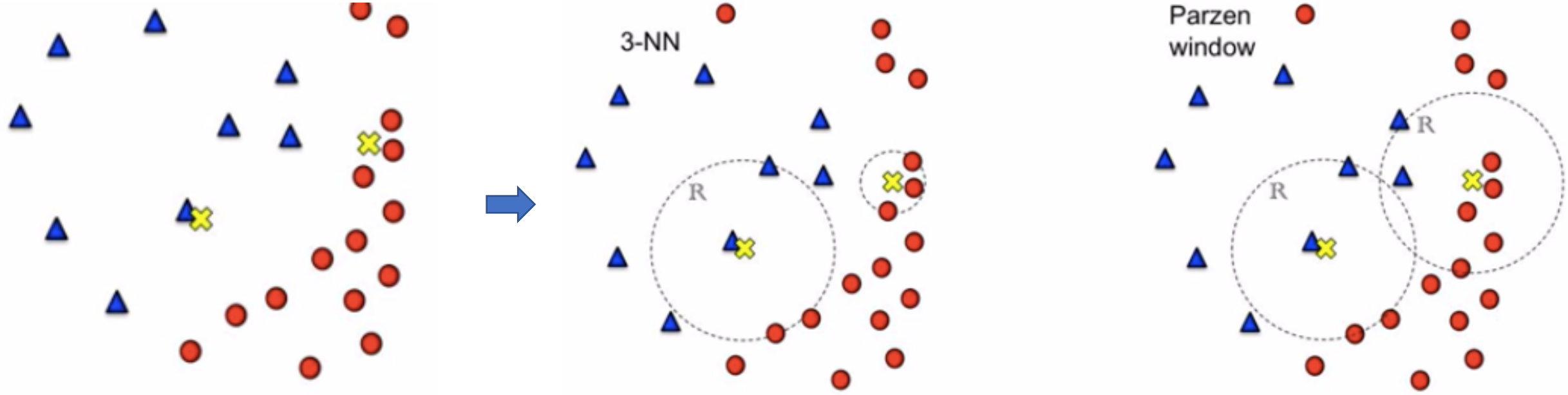
Parzen window:
Place a kernel window k at location x and count how many data points fall inside it
or
Place a kernel window k around each data point and sum up their influences at location x.





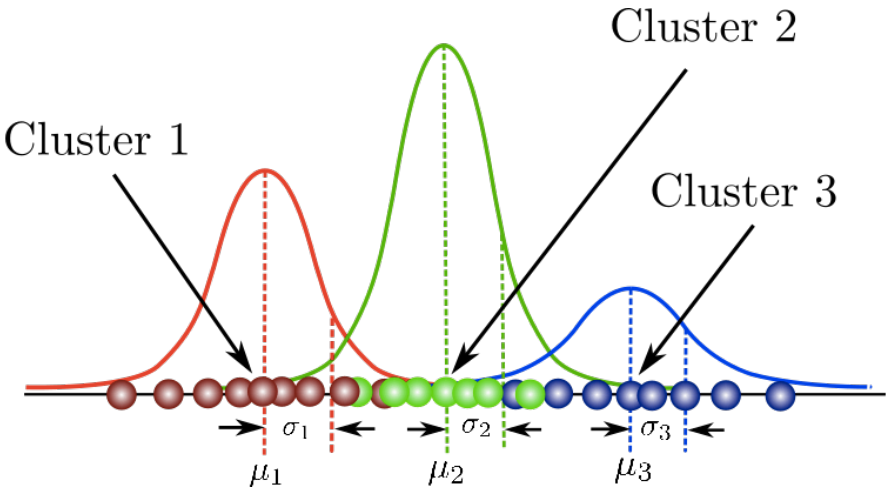
Unsupervised Learning algorithms: Non-Parametric Density estimation

- K-Nearest Neighbor Density Estimation, similar to parzen windows.



Unsupervised Learning algorithms: Non-Parametric Density estimation

- Mixure of Gaussians: A Gaussian Mixture is a function that is comprised of several Gaussians, each identified by $k \in \{1,\ldots, K\}$, where K is the number of clusters of our dataset. Each Gaussian k in the mixture is comprised of the following parameters:
  - -A mean μ that defines its centre.
  - -A covariance Σ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
  - -A mixing probability π that defines how big or small the Gaussian function will be.



Unsupervised Learning algorithms: Non-Parametric Density estimation

Gaussian Mixture Models are a very powerful tool and are widely used in diverse tasks that involve data clustering. A popular applications in Computer Vision is model distributions of pixel colors.

- Each pixel is one data point in, e.g., RGB space
- Learn a MoG to represent the class-conditional densities
- Use the learned models to classify other pixels

(a) input image (b) user input (c) inferred segmentation

Unsupervised Learning algorithms: Principal Component Analysis

There are numerous real-world use cases, where the number of features available, which may potentially be used to train a model, is very large.
Over fitting is one of the results of the curse of dimensionality.
Principal component analysis (PCA) is an unsupervised technique used to preprocess and reduce the dimensionality of high-dimensional datasets while preserving the original structure and relationships inherent to the original dataset so that machine learning models can still learn from them and be used to make accurate predictions.

Unsupervised Learning algorithms: Principal Component Analysis

Let’s start with an example:
We have 300 rows representing 300 user and 3000 columns representing 3000 films.

In this case, each movie is a different feature (or dimension) and each different user is a different instance (or observation). Not all users would have rated all the movies, so there will be a significant number of missing values. Such a dataset, and the matrix used to represent it, is described as sparse.

	Movie #1 Toy Story	Movie #2 Monsters Inc.	Movie #3 Saw	Movie #4 Ring	Movie #5 Hitch
User #1	4	5	1	NULL	4
User #2	5	NULL	1	1	NULL
User #3	5	4	3	NULL	3
User #4	5	4	1	1	NULL
User #5	5	5	NULL	NULL	3

Unsupervised Learning algorithms: Principal Component Analysis

These issues would pose a problem for machine learning algorithms, both in terms of computational complexity and the likelihood of over fitting.
To solve this problem, take a closer look at the previous sample table.
We could say, for example, that User #1 is representative of all fans of computer-animated children's films, and so we could recommend to User #2 the other movies that User #1 has historically rated highly.
At a high level, this is what PCA does—it identifies typical representations, called principal components, within a high-dimensional dataset so that the dimensions of the original dataset can be reduced while preserving its underlying structure and still be representative in lower dimensions!

Unsupervised Learning algorithms: Principal Component Analysis

Steps:

- We standardize the original high-dimensional dataset.
- We take the standardized data and compute a covariance matrix that provides a means to measure how all our features relate to each other.
- We find its eigenvectors and corresponding eigenvalues. Eigenvectors represent the principal components, eigenvalues represent how much variance there is in the data in that direction.
- The eigenvectors are then sorted in descending order based on their corresponding eigenvalues.
- A new matrix is then constructed with top k eigenvectors, thereby reducing the original n-dimensional dataset into reduced k dimensions.

Unsupervised Learning algorithms: Principal Component Analysis

Theory:

- Covariance refers to a measure of how strong the correlation between two or more random variables is and is calculated for variables x and y over i dimensions:

$$cov(x, y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}$$

A covariance matrix is a symmetric square matrix where the general element (i, j) is the covariance, cov(i, j).

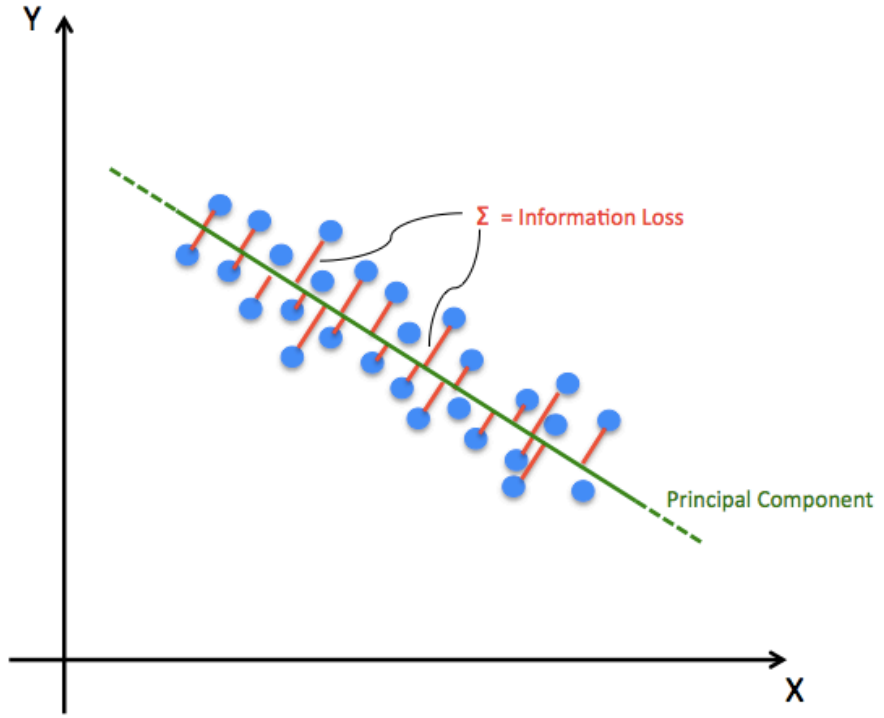
- eigenvectors are a special set of vectors whose direction remains unchanged when a linear transformation is applied to it, and only changes by a scalar factor. Let be A a matrix such that Ax=b, λ is an eigenvalue of A if Ax= λx. x is the eigenvector.

Unsupervised Learning algorithms: Principal Component Analysis

The amount of information loss is the sum of the distance of each point to the principal component line.

The more variance of the data along a principal component, the higher the principal component is ranked. Each principal component is orthogonal to each other, without overlapping, so they are a sort of independent features.

A successful use of PCA is seen in facial recognition problem (reducing number of pixels).



Unsupervised Learning algorithms: Principal Component Analysis

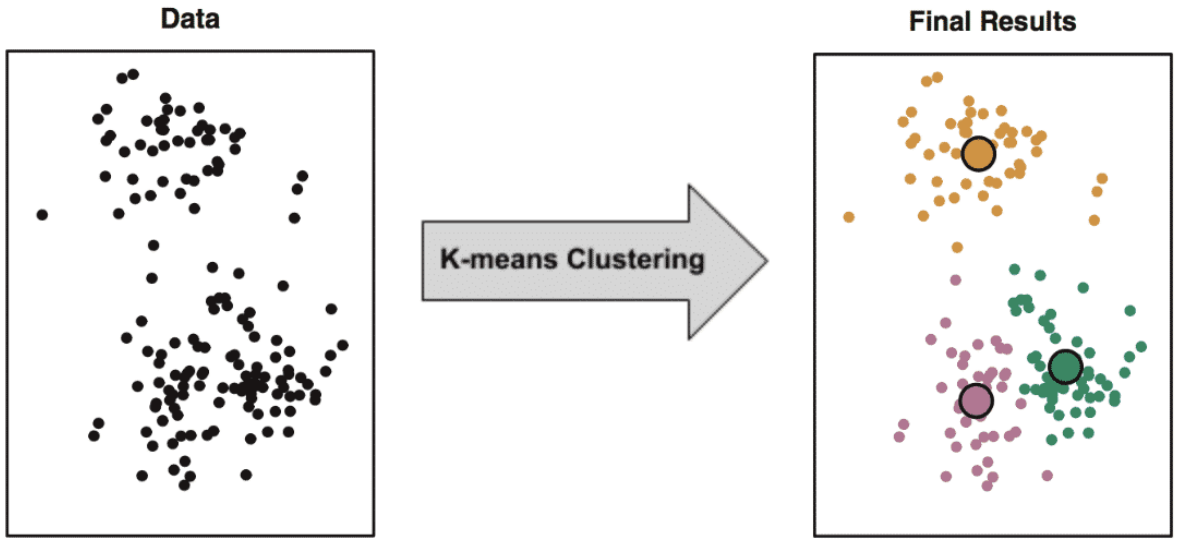
A genetic example:

https://www.youtube.com/watch?v=HMOI_lkzW08

Unsupervised Learning algorithms: K-means clustering

«The objective of K-means is simple: group similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number (k) of clusters in a dataset.»

You'll define a target number k, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster.



Unsupervised Learning algorithms: K-means clustering

Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.
In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.
The ‘means’ in the K-means refers to averaging of the data; that is, finding the centroid.

The K-means algorithm in data mining starts with a first group of randomly selected centroids and then performs iterative calculations to optimize the positions of the centroids until:

- The centroids have stabilized.
- The defined number of iterations has been achieved.

Unsupervised Learning algorithms: K-means clustering

Theory:
Given a set of observations (x1, x2, ..., xn), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (≤ n)
sets S = {S1, S2, ..., Sk} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

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

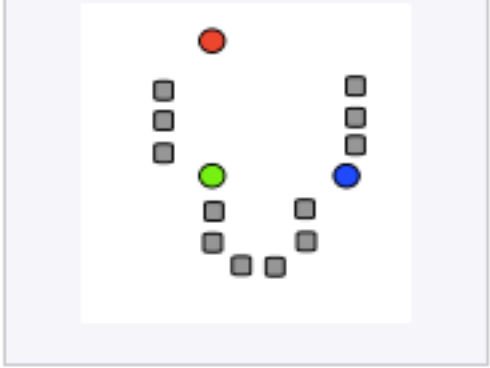
where μi is the mean of points in Si.

Unsupervised Learning algorithms: K-means clustering

Theory:

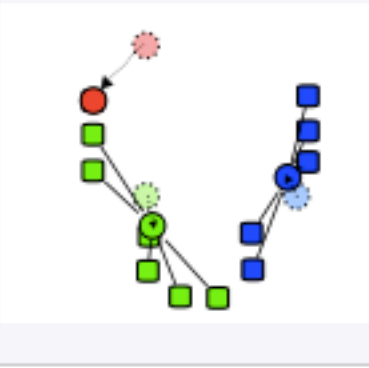
- **Initialization**: Commonly used initialization methods are Forgy and Random Partition. The Forgy method randomly chooses k observations from the dataset and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial mean to be the centroid of the cluster's randomly assigned points.
- **Complexity:** Finding the optimal solution to the k-means clustering problem for observations in d dimensions is NP-hard in general Euclidean space or for a general number of clusters. But, if k and dimension are fixed, the problem can be exactly solved in time $O\left(n^{dk+1}\right)$.

Unsupervised Learning algorithms: K-means clustering



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.



3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

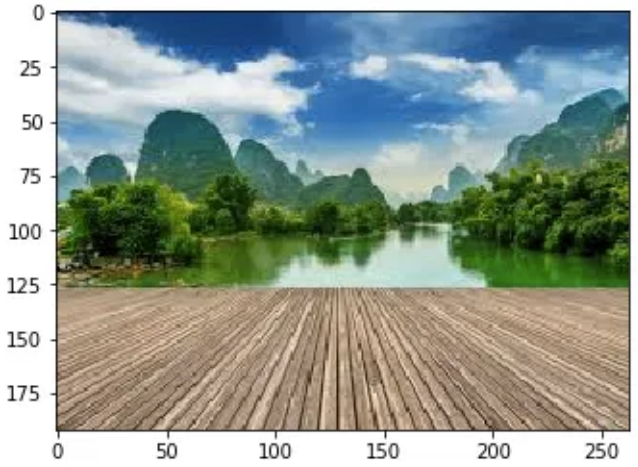
Unsupervised Learning algorithms: K-means clustering

In features learning:

https://www.youtube.com/watch?time_continue=2&v=7Qv0cmJ6FsI

Unsupervised Learning algorithms: K-means clustering

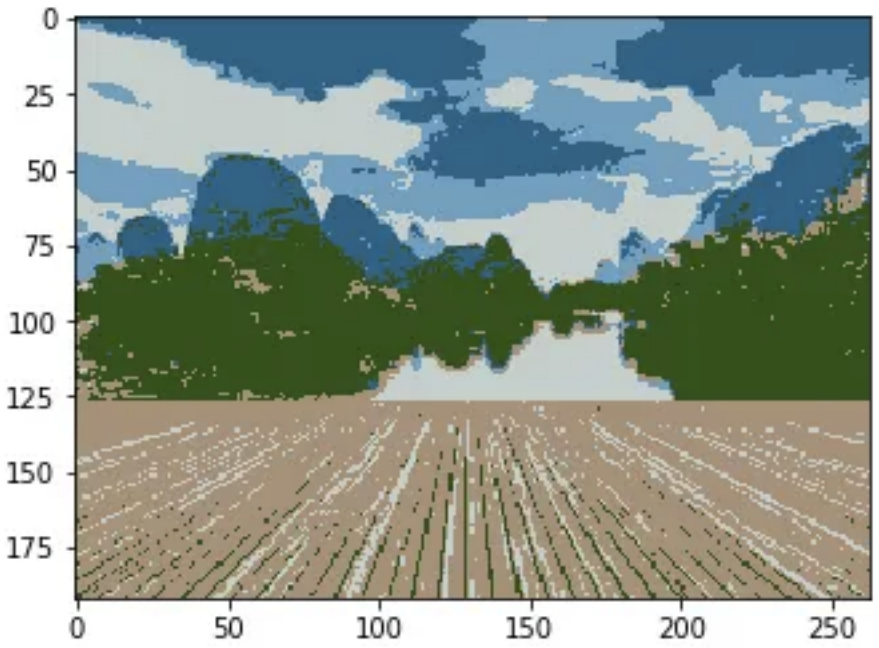
An Example: image segmentation.
The key advantage of using k-means algorithm is that it is simple and easy to understand. We are assigning the points to the clusters which are closest to them.
Let's put our learning to the test and check how well k-means segments the objects in an image.



It's a 3-dimensional image. For clustering the image using k-means, we first need to convert it into a 2-dimensional array whose shape will be (length*width, channels). Next, fit the k-means algorithm on this reshaped array and obtain the clusters. For example, choose 5 clusters.

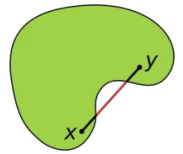
Unsupervised Learning algorithms: K-means clustering

The result will be this.



k-means works really well when we have a small dataset. It can segment the objects in the image and give impressive results. But the algorithm hits a roadblock when applied on a large dataset (more number of images).

It looks at all the samples at every iteration, so the time taken is too high. Hence, it's also too expensive to implement. And since k-means is a distance-based algorithm, it is only applicable to convex datasets and is not suitable for clustering non-convex clusters.

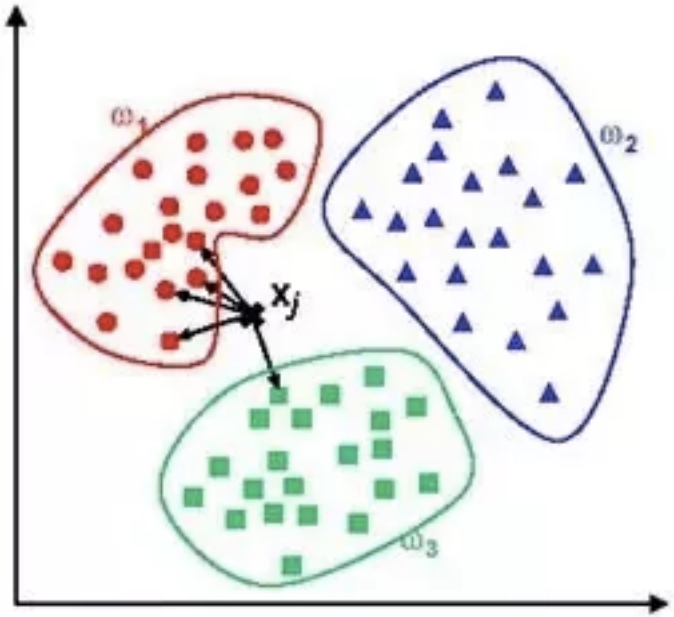


K-means clustering appears similar to K-Nearest Neighbor… why?

K-means vs KNN

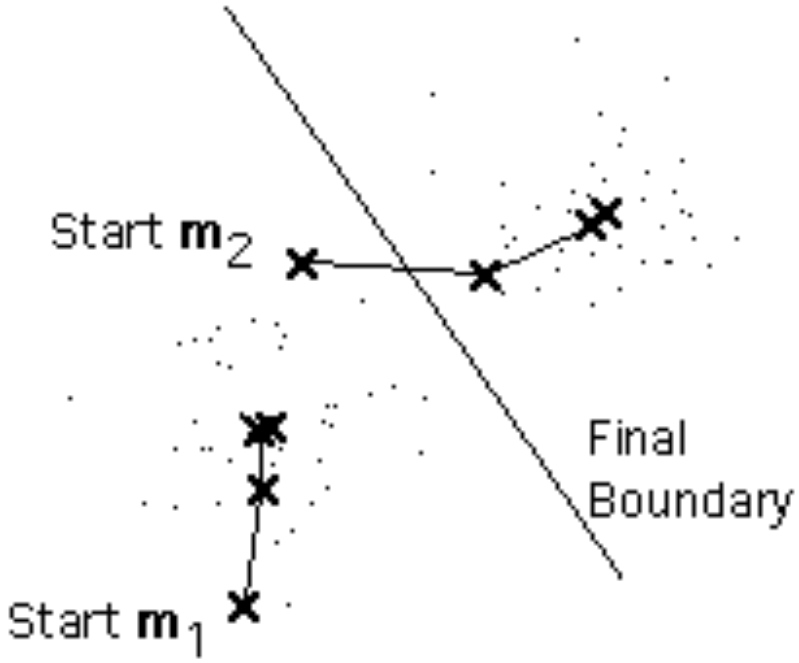
We can say that KNN is a classification technique and K-means is a clustering technique. But…why?

- In a KNN algorithm, a test sample is given as the class of majority of its nearest neighbours. In plain words, if you are similar to your neighbours, then you are one of them. Or if apple looks more similar to banana, orange, and melon (fruits) than monkey, cat and rat (animals), then most likely apple is a fruit. Below is an example, we have three classes and the goal is to find a class label for the unknown example xj. In this case we use the Euclidean distance and a value of k=5 neighbors. Of the 5 closest neighbors, 4 belong to $\omega 1$ and 1 belongs to $\omega 3$, so xj is assigned to $\omega 1$, the predominant class.



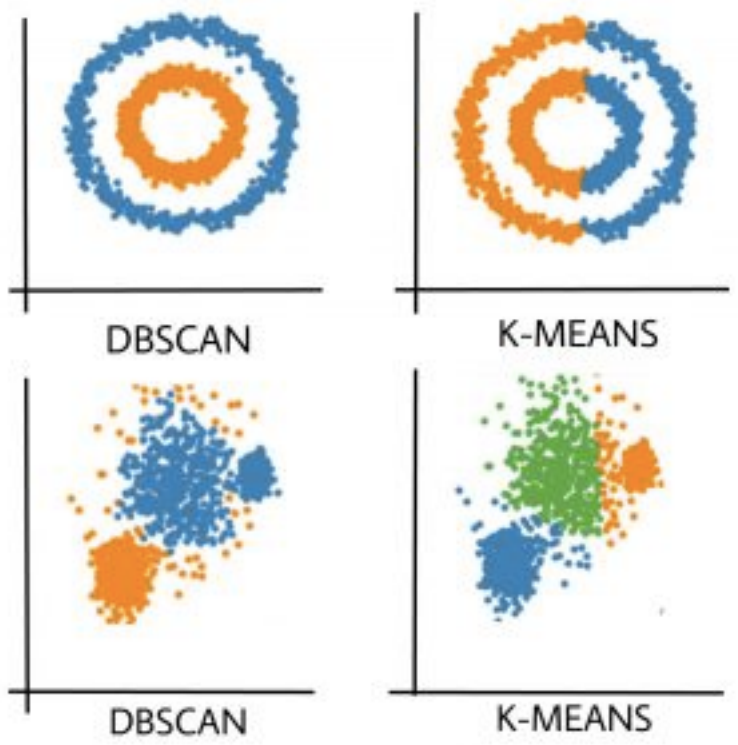
K-means vs KNN

- The situation with K-means is that given some data you cluster them in K-groups or clusters. K-means belongs to the family of moving centroid algorithms, i.e. at every iteration the center (or centroid) of the cluster moves slightly to minimize the objective function. you start with an initial guess for means for two clusters $m1$ and $m2$, use these means to group the objects, then update these means then regroup and so on until either the means $m1$ and $m2$ stop to move (or change) or some threshold is reached (e.g. number of iterations).



Unsupervised Learning algorithms: DBScan

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm.
It is a **density-based clustering non-parametric** algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms.

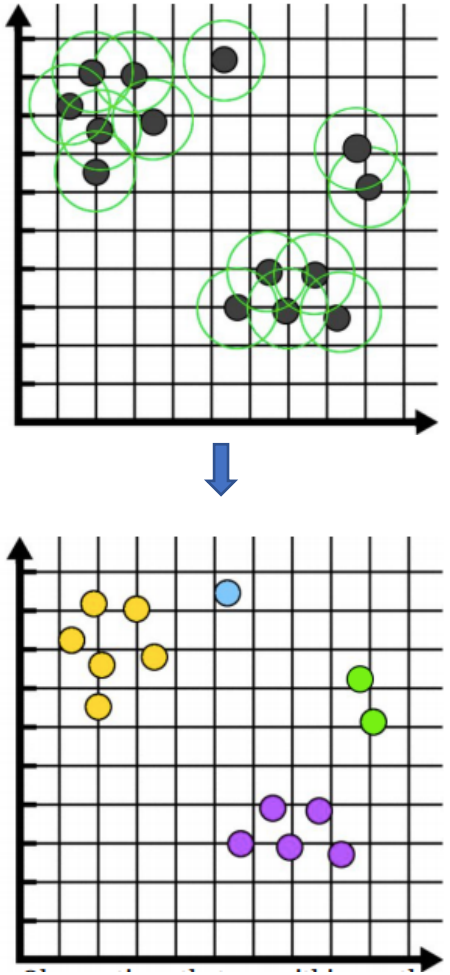


Unsupervised Learning algorithms: DBScan

Steps:

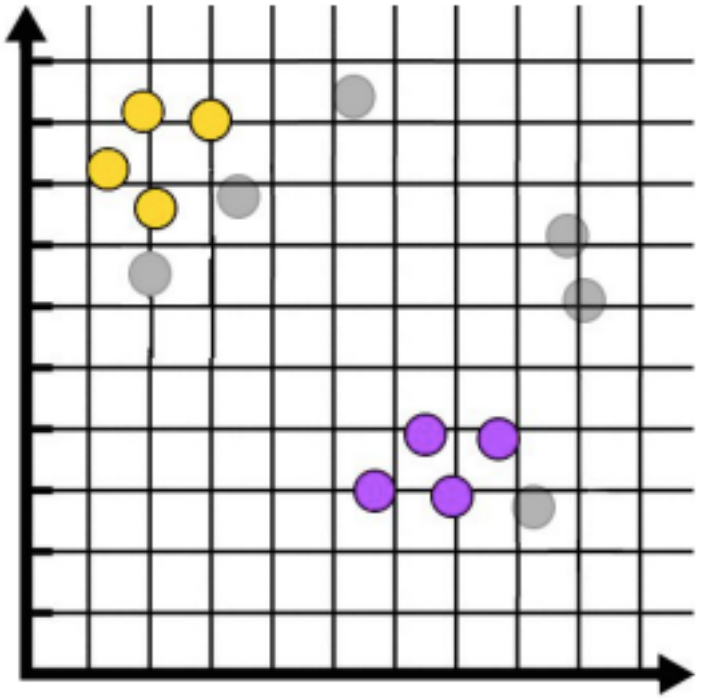
- DBSCAN starts by identifying the neighboring observations of each observation within some radius (a hyperparameter).
- Any data point that is within the data point of radius of another data point are in the same cluster…but… There is something pretty odd with the clusters on the right. There are three observations which are noise and we end up creating two clusters entirely from these bad observations themselves.

What should we do to avoid this?



Unsupervised Learning algorithms: DBScan

Maybe we can only consider points that contain a minimum number of samples within their radii.
The other data points can be neglected and be considered as noise.
Let's set this this value to two; meaning that there must be at least 2 observations within the radius of a data point to be accounted-for.



These colored points have at least 2 data points within their neighbourhood. Grey points do not have 2 data points in their raidus and are called noise.

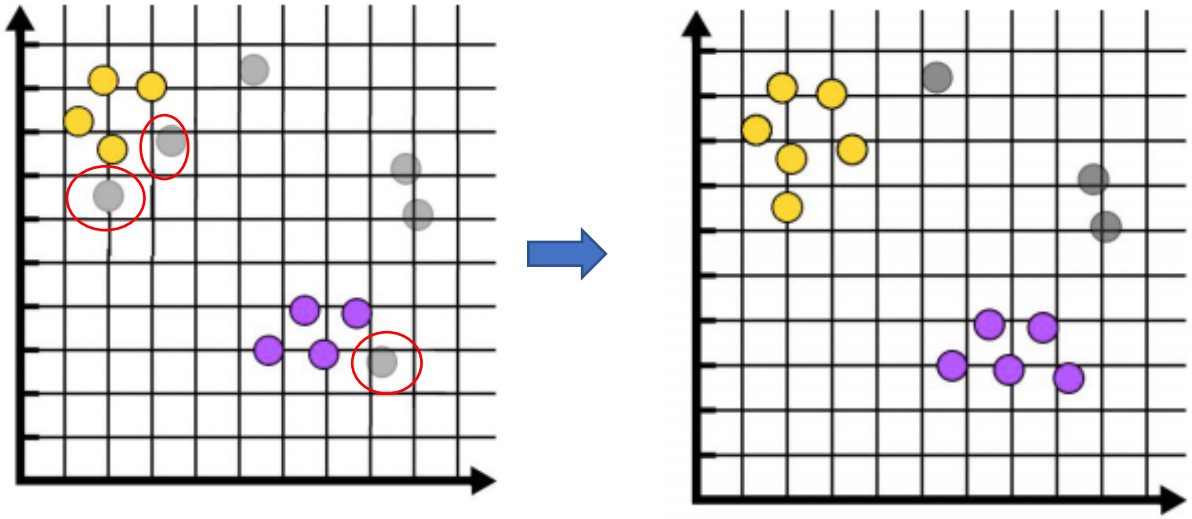
Unsupervised Learning algorithms: DBScan

But we lost also points that are not noise! These points are edge points. To avoid losing it, we should consider noise to be:

- observations which don't satisfy the minimum neighbour requirement

AND

- observation that are not within the radius of a core observation.

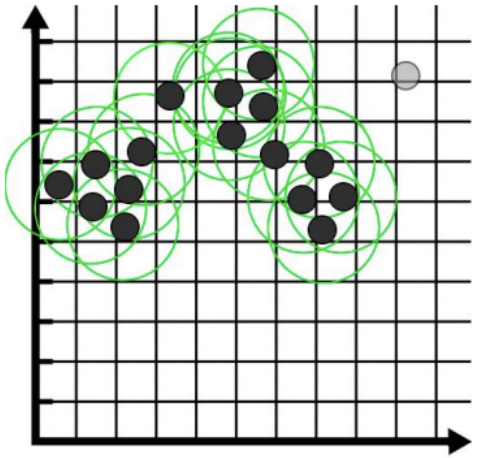


Unsupervised Learning algorithms: DBScan

How to choose hyperparameters?
We have to manually choose these parameters using domain-specific knowledge related to the problem at hand Interpret whether the resulting clusters makes logical sense.
Choosing the hyperparameters is still an active topic in the literature.

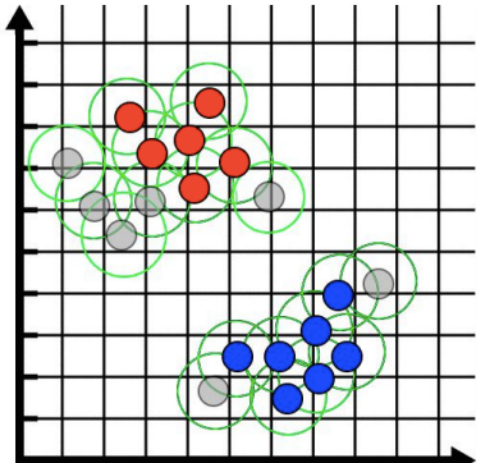
We may have radius too high or too small.

Too high: all samples belong to the same cluster.



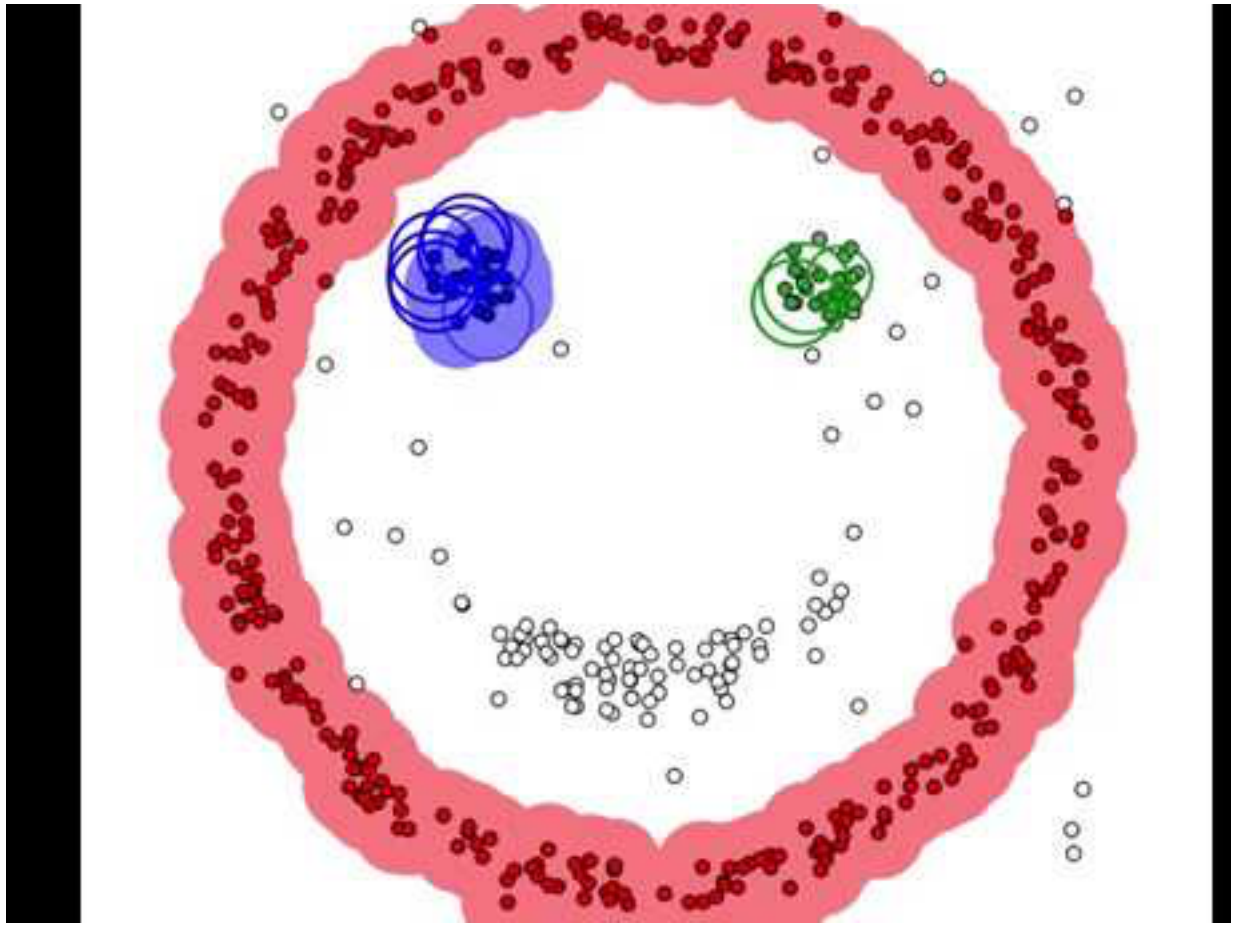
We may have minimum sample too high or too small.

Too high: Good observation classified as noise.



Unsupervised Learning algorithms: DBScan

An example.



https://www.youtube.com/watch?v=h53WMIImUuc

How to handle noise in general?

Unsupervised Learning algorithms: Outlier detection

“Observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism” — Hawkins(1980)

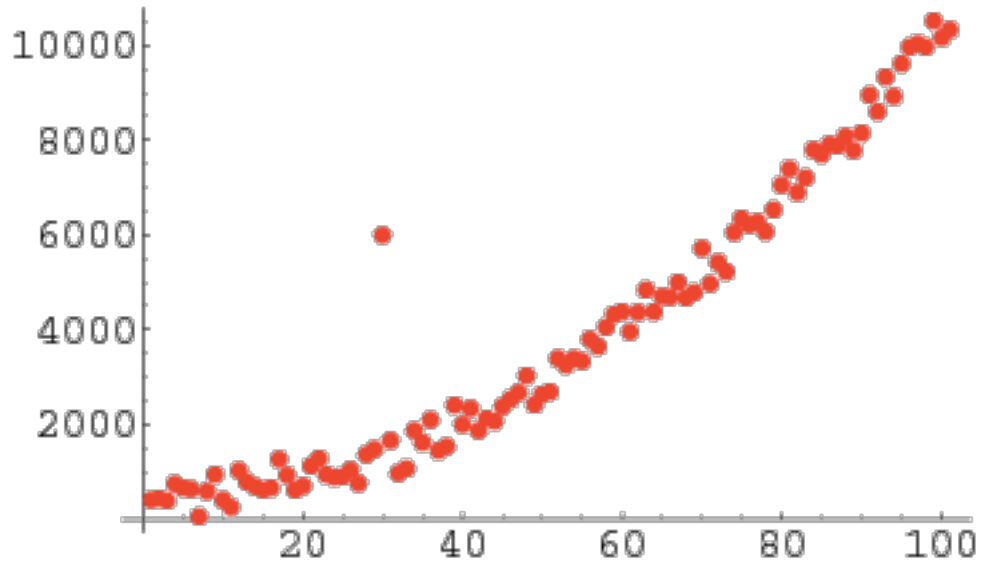
An outlier is an observation that diverges from an overall pattern on a sample.
Outliers can be of two kinds: **univariate** and **multivariate**. Univariate outliers can be found when looking at a distribution of values in a single feature space. Multivariate outliers can be found in a n-dimensional space.
We can find in data:

- **Point outliers:** are single data points that lay far from the rest of the distribution.
- **Contextual outliers:** can be noise in data.
- **Collective outliers:** can be subsets of novelties in data.

Unsupervised Learning algorithms: Outlier detection

Causes of outliers on a dataset:

- Data entry errors (human errors)
- Measurement errors (instrument errors)
- Experimental errors (data extraction errors)
- Intentional (dummy outliers made to test detection methods)
- Data processing errors (data manipulation errors)
- Sampling errors (extracting or mixing data errors)
- Natural (not an error, novelties in data)



Unsupervised Learning algorithms: Outlier detection methods

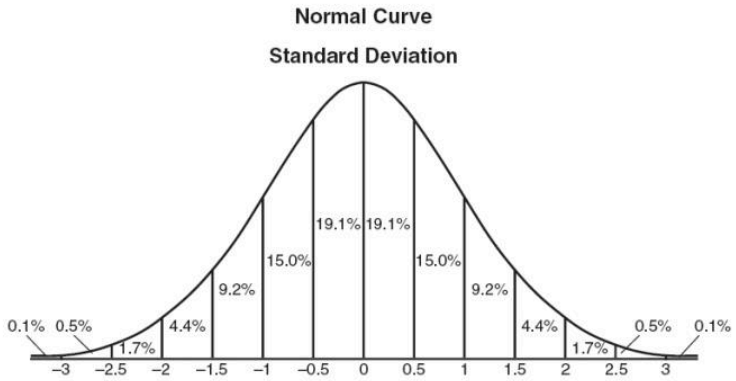
Z-Score

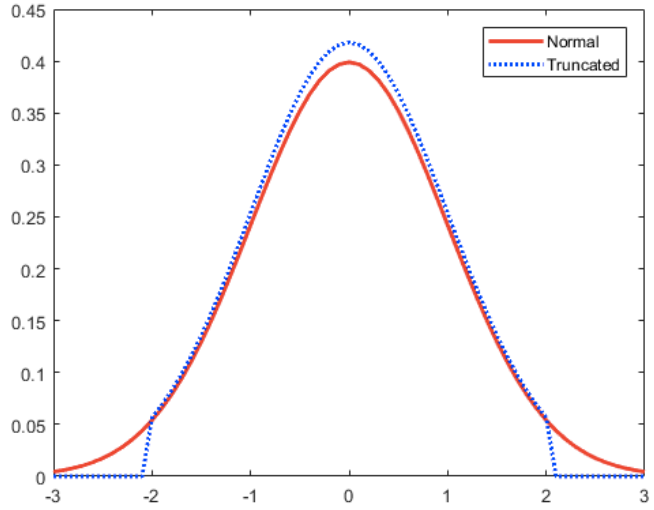
The z-score or standard score of an observation is a metric that indicates how many standard deviations a data point is from the sample's mean, assuming a gaussian distribution. This makes z-score a parametric method.

The z-score of any data point can be calculated with the following expression:

$$z = \frac{x - \mu}{\sigma}$$

By 'tagging' or removing the data points that lay beyond a given threshold we are classifying data into outliers and not outliers.



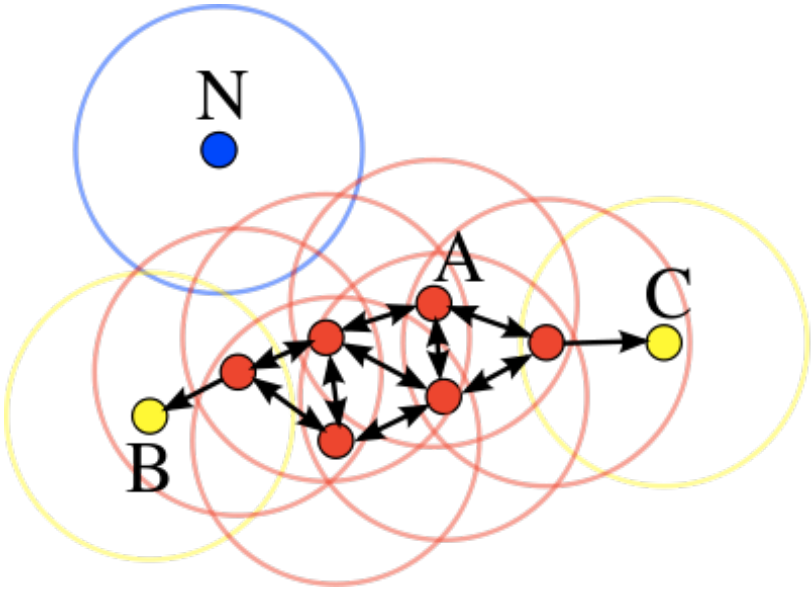


Unsupervised Learning algorithms: Outlier detection methods

DBScan

Dbscan then defines different classes of points:

- Core point: A is a core point if its neighborhood (defined by ε) contains at least the same number or more points than the parameter MinPts.
- Border point: C is a border point that lies in a cluster and its neighborhood does not contain more points than MinPts, but it is still ‘density reachable’ by other points in the cluster.
- Outlier: N is an outlier point that lies in no cluster and it is not ‘density reachable’ nor ‘density connected’ to any other point. Thus this point will have “his own cluster”.



Unsupervised Learning algorithms: Outlier detection methods

Isolation forests

Isolation forests are an effective method for detecting outliers or novelties in data. It is a relatively novel method based on binary decision trees.

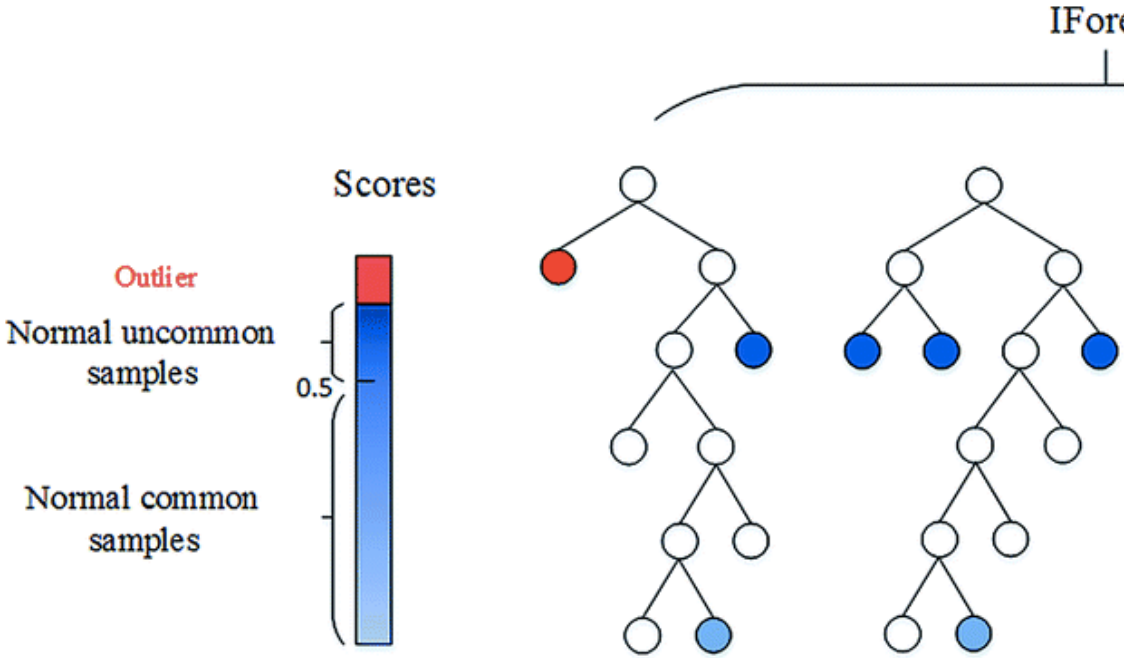
Isolation forest's basic principle is that outliers are few and far from the rest of the observations. For all the observations in the training set, a tree is built. To build the forest a tree ensemble is made averaging all the trees in the forest.

Then for prediction, it compares an observation against that splitting value in a "node".

An outlier score can computed for each observation:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Where $h(x)$ is the path length of the sample x, and $c(n)$ is the 'unsuccessful length search' of a binary tree, n is the number of external nodes.



Unsupervised Learning algorithms: Outlier detection methods Pros&Cons

Z score:

- Pros: It is a very effective method if you can describe the values in the feature space with a gaussian distribution. (Parametric). The implementation is very easy .
- Cons: It is only convenient to use in a low dimensional feature space, in a small to medium sized dataset. Is not recommended when distributions can not be assumed to be parametric.

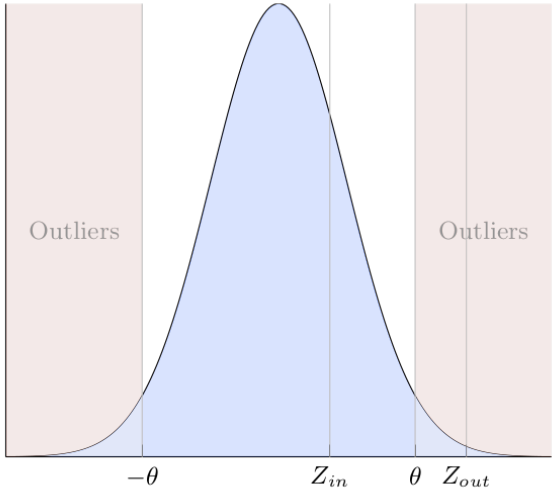
DBScan:

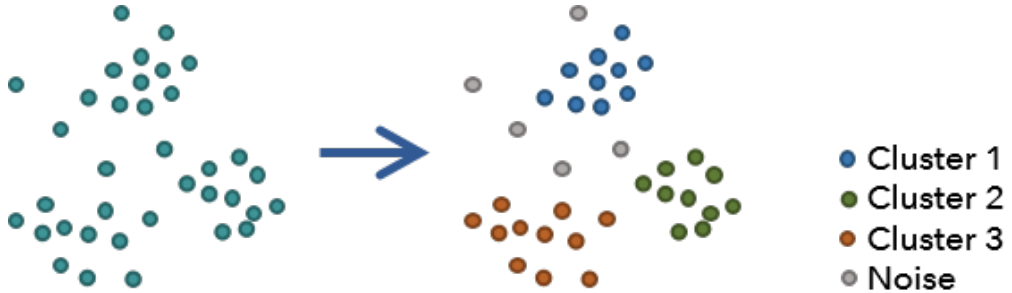
- Pros: It is a super effective method when the distribution of values in the feature space can not be assumed. Works well if the feature space for searching outliers is multidimensional. It's easy to implement. Visualizing the results is easy and the method itself is very intuitive.
- Cons: The values in the feature space need to be scaled accordingly. Selecting the optimal parameters can be difficult. Needs to be re-calibrated each time a new batch of data is analyzed.

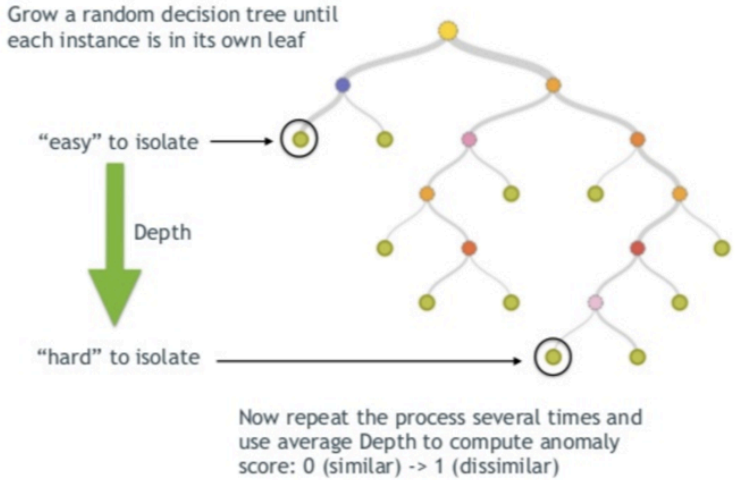
Unsupervised Learning algorithms: Outlier detection methods Pros&Cons

Isolation Forests

- Pros: There is no need of scaling the values in the feature space. It is an effective method when value distributions can not be assumed. It has few parameters, this makes this method fairly robust and easy to optimize. Implementation is easy.
- Cons: Only few kinds of implementation exist. Visualizing results is complicated. If not correctly optimized, training time can be very long and computationally expensive.



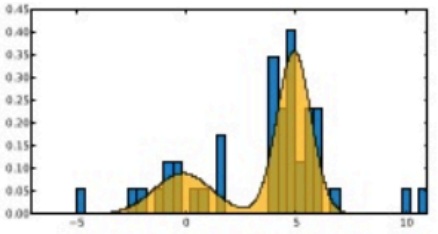




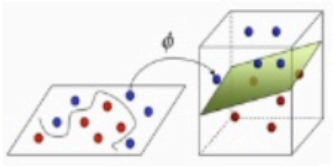
Unsupervised Learning algorithms: Outlier detection methods

In general, each clustering method perform outlier detection.
The important thing is to find them.

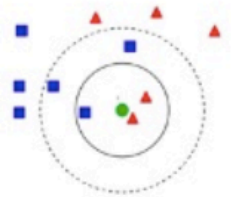
Density based approach (KDE, Gaussian Ellipse, GMM)



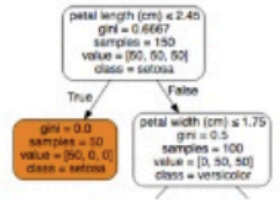
Kernel methods



Nearest neighbors



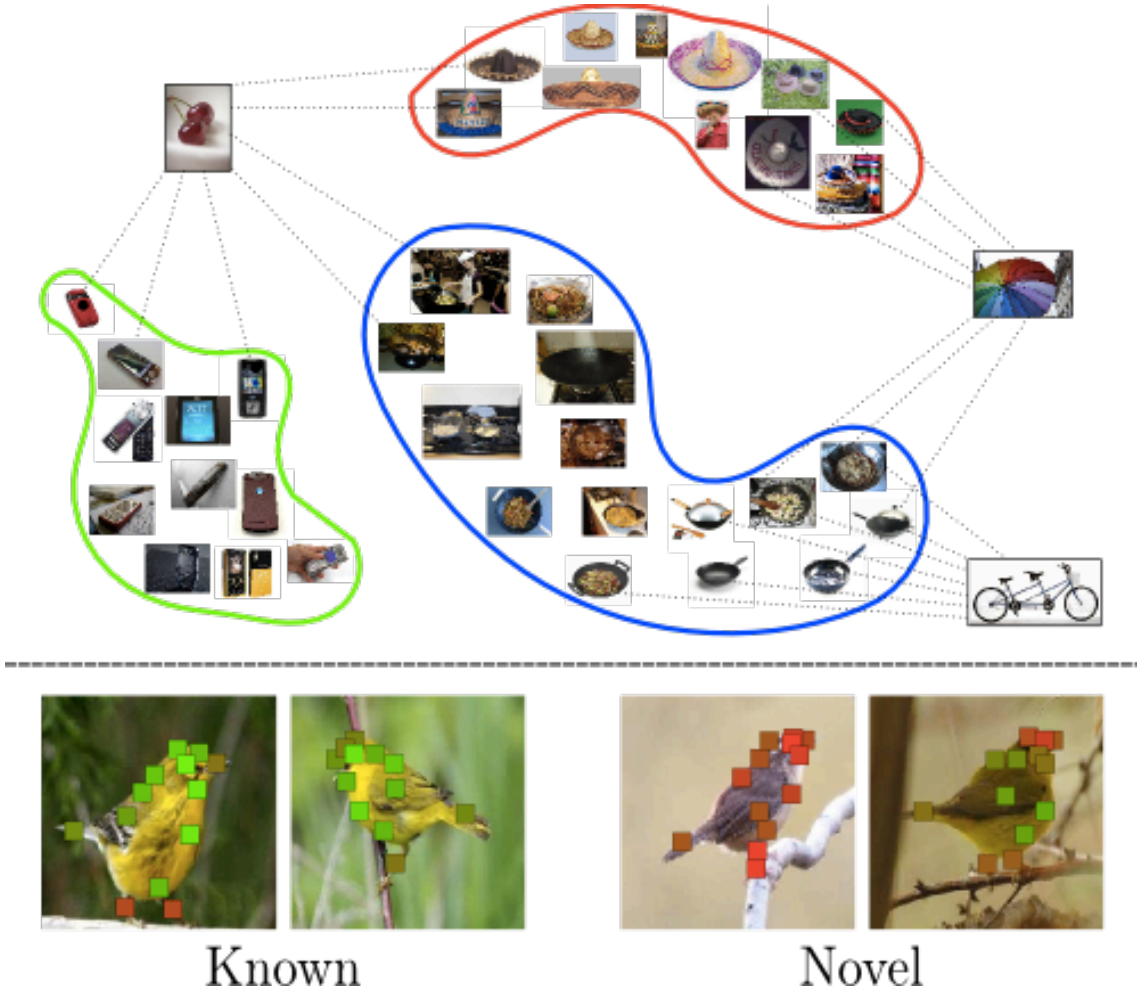
Trees / Partitioning



What happens if outliers are a new kind of data?

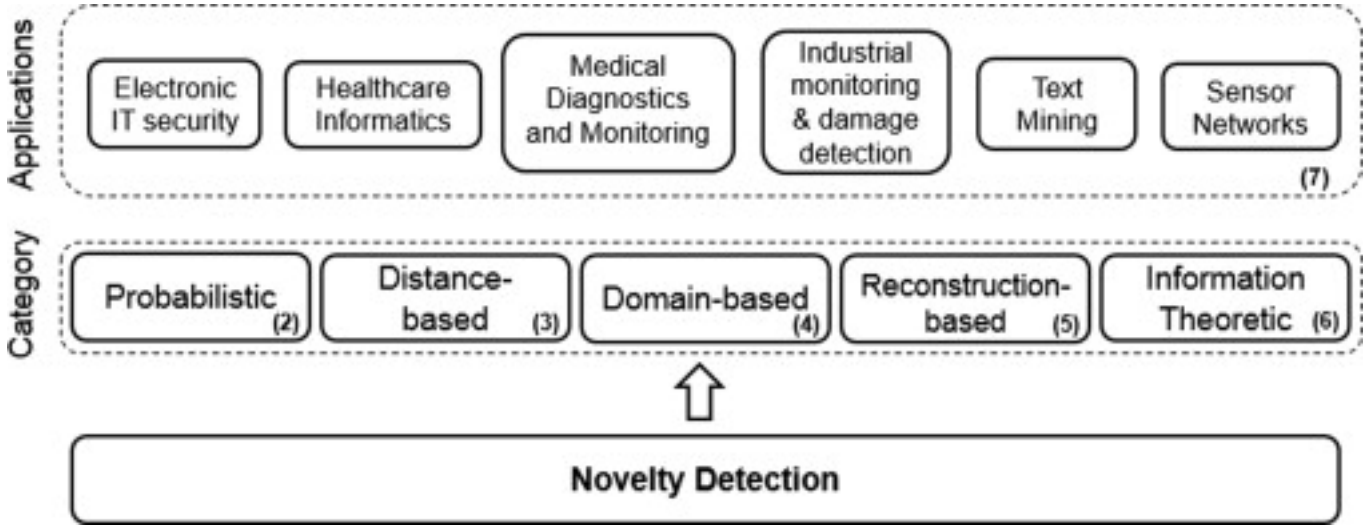
Unsupervised Learning algorithms: Novelty detection

Consider a data set of n observations from the same distribution described by p features.
Consider now that we add one more observation to that data set.
Is the new observation so different from the others that we can doubt it is regular? (i.e. does it come from the same distribution?)
Or on the contrary, is it so similar to the other that we cannot distinguish it from the original observations? This is the question addressed by the novelty detection tools and methods.



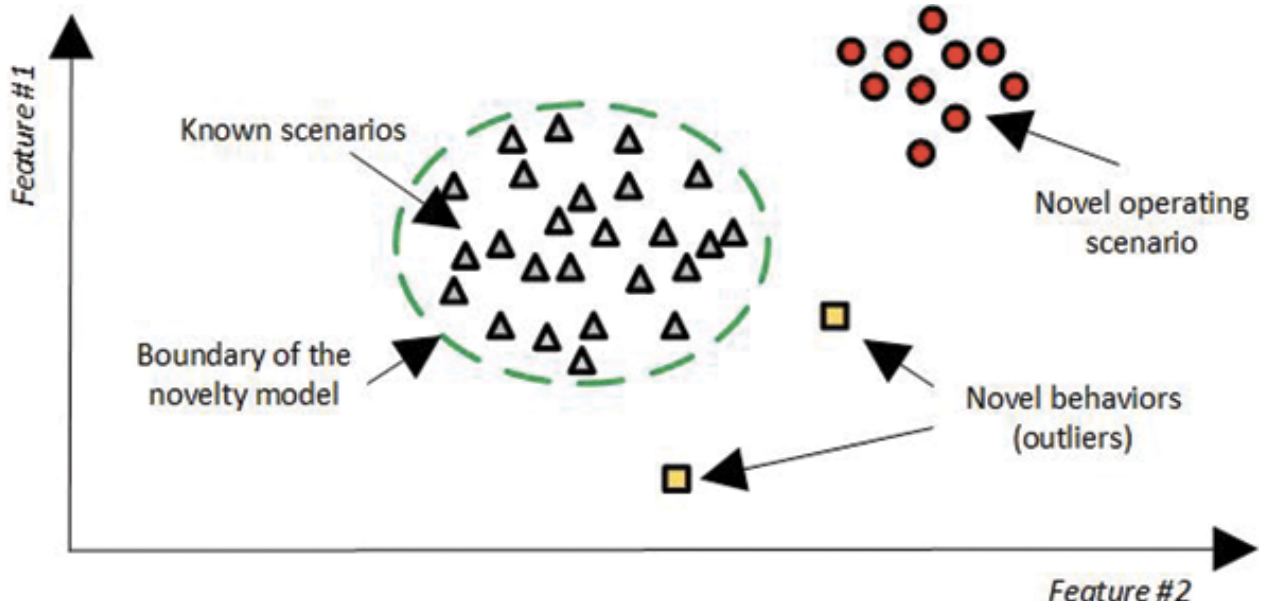
Unsupervised Learning algorithms: Novelty detection

Approaches to novelty detection include both Frequentist and Bayesian approaches, information theory, extreme value statistics, support vector methods, other kernel methods, and neural networks. In general, all of these methods build some model of a training set that is selected to contain no examples (or very few) of the important (i.e., novel) class. Novelty scores z(x) are then assigned to data x, and deviations from normality are detected according to a decision boundary that is usually referred to as the novelty threshold z(x)=k.



Unsupervised Learning algorithms: Outlier vs Novelty detection

Outlier detection is similar to novelty detection in the sense that the goal is to separate a core of regular observations from some polutting ones, called “outliers”. Yet, in the case of outlier detection, we don’t have a clean data set representing the population of regular observations that can be used to train any tool.



Coming soon: Neural Networks!