Clustering Tutorial

Stephen Coshatt[1] and Dr. WenZhan Song[1]

The University of Georgia[1]

Sensor Data Science and AI
Spring 2025

UNIVERSITY OF
GEORGIA

Outline

Table of Contents

Overview

Clustering is a type of unsupervised machine learning.

Clustering divides unlabeled data into groups, called clusters, based on similarities among data.

Assumes data that is similar is clustered together.

Clustering originated in data mining, but has found use in other applications.

Sample Clustering Output

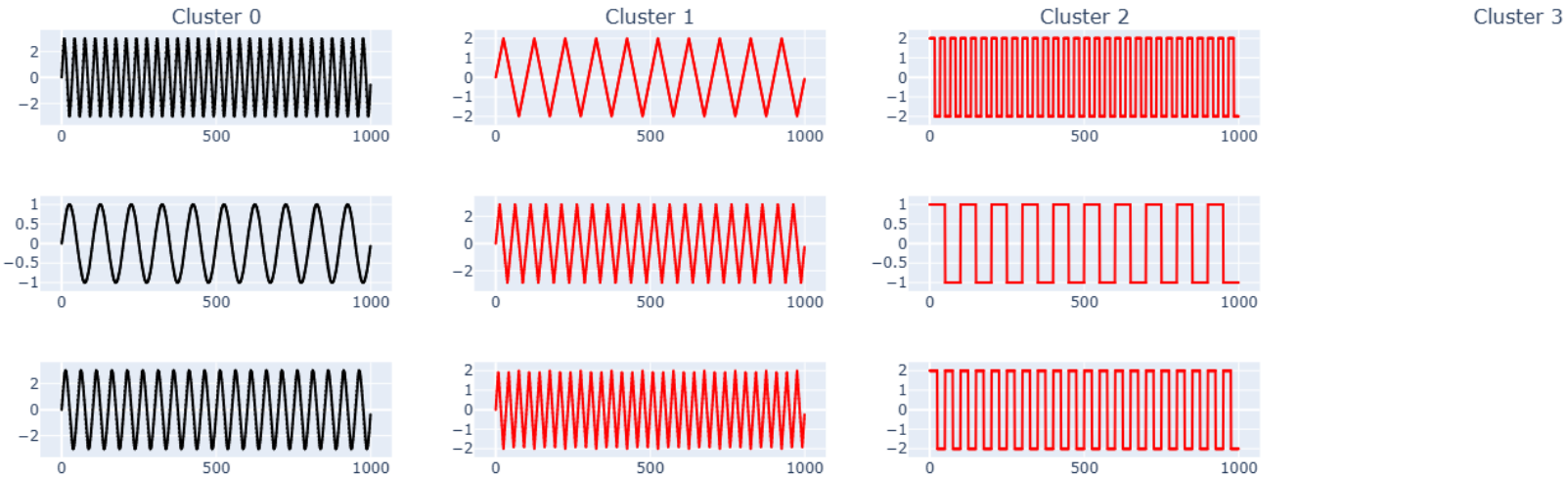


Figure: Clustering Output Sample

Table of Contents

Hierarchical Clustering

1. **Dendrograms**

   Data is organized into a tree like structure know as a **dendrogram**.

2. **Two Types**

   Agglomerative - aka "bottom up" or "AGNES (Agglomerative Nesting)". Every data point starts in its own cluster and are iteratively combined.

   Divisive - aka "top down" or "DIANA (Divise Analysis)". Every data point starts and single cluster and they are iteratively split into smaller clusters.

Dendrogram

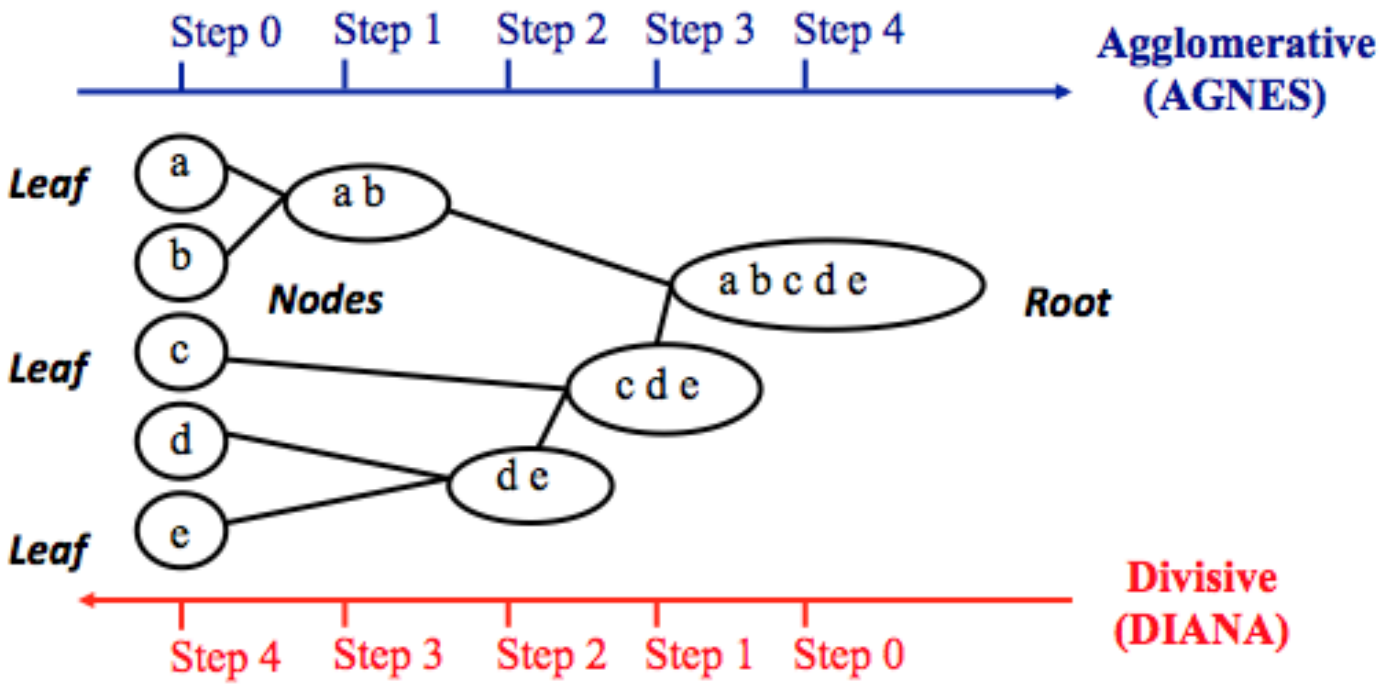


Figure: A Dendrogram

Agglomerative Clustering Parameters

1. **Number of Clusters**

   n_clusters: In SKLearn, if you know the number of clusters, you may set this variable to that number, but **distance_threshold** should be set to "None".

2. **Distance Threshold**

   distance_threshold: The linkage distance threshold at or above which clusters will not be merged.

   In SKLearn, if you do not know the number of clusters or simply want the algorithm to decide the number, you may set this variable, but **n_clusters** should be set to "None".

Agglomerative Clustering Parameters

1. **Metric**

   metric: The distance metric used for distance calculations. Euclidean, Manhattan, etc.

2. **Linkage**

   linkage: The affinity based approach used for calculating distances between clusters.

Linkage Types

1. **Ward**

   'ward': Minimizes the variance of the clusters being merged.

2. **Complete**

   'complete' or 'maximum': uses the maximum distances between all observations of the two sets.

3. **Average**

   'average': uses the average of the distances of each observation of the two sets.

4. **Single**

   'single': uses the minimum of the distances between all observations of the two sets.

Agglomerative Clustering Pros & Cons

1. **Pros**

   A fixed number of clusters is not required

   Easy to implement

2. **Cons**

   Time consuming: $O(n^2 logN)$

   Algorithm cannot back track.

Disdvantages

1. **Can create overly complex models that do not generalize well (overfitting)**
2. **Requires little data preperation**
3. **Cost of using a tree is logarithmic**
4. **Can handle numeric and categorical data**
5. **Can handle multi-output problems**
6. **Can be validated using statistical tests**

Feature Agglomeration

Feature agglomeration is a dimensionality reduction technique based on agglomerative clustering. It essentially looks for the most important features in a dataset by grouping similar correlated features into clusters.

Reference: Feature agglomeration in Scikit Learn

Feature Agglomeration Steps

1. "Initially, each feature is treated as a separate cluster, if 'n' number of features are present then 'n' clusters will be formed."

2. "After the above step distance between each feature clusters are measured by the affinity parameter which we mention while initializing the feature agglomeration."

3. "In this step the grouping of the clusters happen, the closest clusters according to the distance are merged together into a single cluster and this step is repeated until only 'n_clusters' amount of clusters are left."

4. "After each merge the pairwise distances are updated with the new clusters formed and the existing clusters."

Reference: Feature agglomeration in Scikit Learn

Feature Agglomeration Steps

5. "The steps are repeated until the required number of clusters are achieved."

6. "Once the stage of remaining 'n_clusters' is reached each cluster gets represented by a feature. The value of 'pooling' parameter determines the way in which the new features are going to be represented, for example if we set the value of pooling parameter of be 'mean' it will represent the mean value."

7. "At last the original features are transformed by replacing the existing features with the new features that are formed by feature agglomeration."

Reference: Feature agglomeration in Scikit Learn

Feature Agglomeration Parameters

n_cluster: number of clusters

affinity: distance metric for the distance calculations. Euclidean, Manhattan, etc.

linkage: defines the approach used for calculating distance between clusters and is based on affinity.

1. **Ward:** minimizes variance between clusters
2. **Complete:** calculates the maximu value between all data points in two clusters
3. **Average:** calculates the average value of distances between two clusters
4. **Single:** calculates the minimum value of distances between two clusters

pooling_func: method to combine the values of agglomerated features into the final cluster. Mean, median, etc.

Reference: Feature agglomeration in Scikit Learn

Feature Agglomeration Pros & Cons

Advantages:

1. "As a main objective of feature agglomeration, it helps reducing the number of extra features in the dataset grouping similar features together and therefore reducing the complexity of data exploration."
2. "Grouping of similar function may reduce the noise in the data and help in understanding better relationship in data."
3. "Reduction of extra features may also improve the model performance."

Cons:

1. "Feature agglomeration may reduce the descriptive information of the data present originally."
2. "If the parameters of feature agglomeration not chosen wisely may result into loss of informative details about the data."
3. "In some cases feature agglomeration might not provide significant benefits as compared to the original data which was present previously."

Reference: Feature agglomeration in Scikit Learn

Table of Contents

K Means

K means algorithm is an iterative algorithm that tries to partition the dataset into **K** predefined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum.

K Means Pseudo Code

Assign initial values for eachμ (from $\mu = 1$ till $\mu = k$); Repeat

1. Assign each point in the input data to the μ that is closest to it in value
2. Calculate the new mean for each μ
3. if all μ values are unchanged break out of loop;

The algorithm starts by placing k different averages (i.e. means) whose values are either initialized randomly or set to real data points on the plane. Next, the algorithm goes through the data points one-by-one, measuring the distance between each point and the centroids. The algorithm then groups the the data point with the closest centroid (i.e. closest in distance). This grouping forms the clusters.

Reference: How Does k-Means Clustering in Machine Learning Work?

K-Means: When not to Use

When K-Means may not perform well:

1. Clusters are non-spherical
2. Clusters have different sizes
3. Data has outliers
4. Clusters are non-linearly separable
5. Clusters have overlap
6. Cluster centroids have poor initialization

Reference: Understanding K-Means Clustering and Kernel Methods

K Means Pros

1. **Simple:** It is easy to implement k-means and identify unknown groups of data from complex data sets.
2. **Flexible:** K-means algorithm can easily adjust to the changes. If there are any problems, adjusting the cluster segment will allow changes to easily occur on the algorithm.
3. **Suitable in a large dataset:** K-means is suitable for a large number of datasets and it's computed much faster than the smaller dataset.
4. **Efficient:** The algorithm used is good at segmenting the large data set. Its efficiency depends on the shape of the clusters. K-means work well in hyper-spherical clusters.
5. **Easy to interpret:** The results are easy to interpret. It generates cluster descriptions in a form minimized to ease understanding of the data.

K Means Cons

1. **No-optimal set of clusters:** K-means doesn't allow development of an optimal set of clusters and for effective results, clusters has to be decided beforehand.
2. **Handle numerical data:** K-means algorithm can be performed on numerical data only.
3. **Lacks consistency:** K-means clustering gives varying results on different runs of an algorithm. A random choice of cluster patterns yields different clustering results resulting in inconsistency.
4. **Sensitivity to scale:** Changing or rescaling the dataset either through normalization or standardization will completely change the final results.

Bisection K Means

1. Bisection k-means is a variation of the original k-means algorithm. It uses divisive hierarchical clustering to progressively pick centroids based on previous clustering rather than creating them all at once.
2. Bisection k-means is more efficient that k-means when the number of clusters is large. This is because it work on a subset of data at each bisection rather than the entire dataset.
3. Note that this variant cannot produce empty clusters.

Reference: Bisecting K-Means

Kernel K Means

Kernel k-means make use of kernels to calculate the distance instead of using the Euclidean distance. It otherwise functions as the standard k-means.

Advantages

1. Algorithm is able to identify the non-linear structures.
2. Algorithm is best suited for real life data set.

Disadvantages

1. Number of cluster centers need to be predefined.
2. Algorithm is complex in nature and time complexity is large.

Reference: kernel k-means clustering algorithm

Other K Means Variants

Mini-Batch K Means

This variant of k-means uses a subset of data, called mini-batches, to reduce computation time. Mini-batches are randomly sampled in each training iteration. Centroid are updated per batch vs per data point of k-means.

Mini-batch k-means is faster, but produces slightly lower quality results.

Time Series K Means

This a a TSLearn variation of k-means optimized for time series data. This is done by using Dynamic Time Warping (DTW) as a metric for similarity, which has shown to be effective for time series data, instead of the Euclidean distance. Otherwise the algorithm works the same as the standard K-Means.

K Shape

K-Shape is a centroid based clustering algorithm used to group time-series data based on their shapes rather than their raw values. It is particularly useful for clustering time-series data where the values themselves might differ, but the patterns or trends over time are similar.

It is an extension of the traditional K-means clustering algorithm but tailored to handle the unique nature of time series data, such as similarities in patterns and temporal relationships.

K Shape Features

1. **Shape-based Similarity:** Unlike traditional clustering methods (like k-means) that focus on Euclidean distance between data points, K-Shape uses a shape-based distance metric. It matches the general trend or shape of time-series data points, even if the values themselves are not directly comparable.
2. **Normalization:** Before comparing time series, K-Shape normalizes them, meaning it eliminates shifts in amplitude and time. This allows it to focus purely on the structure or shape of the data.
3. **Iterative Clustering:** The algorithm works similarly to k-means in that it assigns time-series to clusters and iteratively updates the centroids (representative time-series for each cluster) to minimize the dissimilarity between the cluster's members.
4. **Efficiency:** K-Shape is considered computationally efficient compared to some other shape-based clustering methods, making it particularly useful for large time-series datasets.

Table of Contents

Density-based Clustering

Density-based clustering is a type of unsupervised machine learning algorithm that groups together data points that are closely packed, marking points that lie in low-density regions as outliers. The primary idea behind density-based clustering is to form clusters based on the density of points in a given region, rather than using a fixed number of clusters like in methods such as k-means.

DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clusters by measuring the distance each point is from one another. If enough points are close enough together, then DBSCAN will classify it as a new cluster.
The main concept of DBSCAN algorithm is to locate regions of high density that are separated from one another by regions of low density.

Reference: DBSCAN with Python

DBSACN Parameters & Terms

Parameters:

1. **Epsilon** — The maximum distance a point can be from another point to be considered a neighbor
2. **Min_Points** — The amount of points needed within the range of epsilon to be considered a cluster

Terms:

1. **Noise** — This is a point that does not have enough neighbors within epsilon to be part of a cluster (including itself)
2. **Border Points** — This is a point that has neighbors within epsilon but not enough neighbors to be a core point. These points make up the edge of the cluster
3. **Core Points** — Points that have the Min Points required within epsilon (including itself). These points along with border points will form a cluster

Reference: DBSCAN with Python

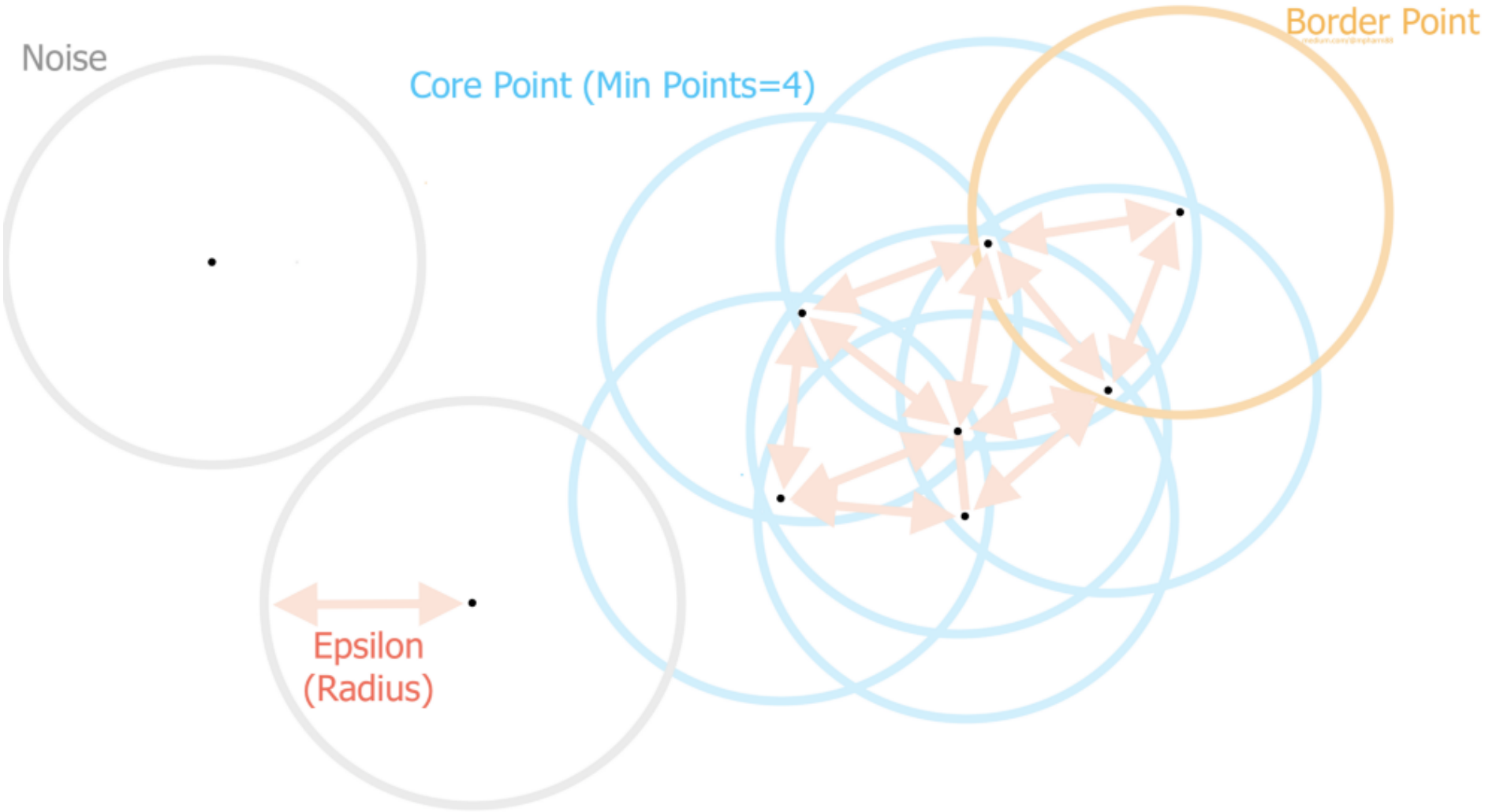


Figure: DBSCAN, by Mikio Harman

Directly Density Reachable

Directly Density Reachable: Data-point a is directly density reachable from a point b if -

$|N(b)| \geq MinPts$ i.e. b is a core point.

$a \in N(b)$ i.e. a is in the epsilon neighborhood of b.

Considering a border point and a core point, we can understand that notion of directly density reachable is not symmetric, because even though the core point falls in the epsilon neighborhood of border point, the border point doesn't have enough $MinPts$, and thus fail to satisfy both conditions.

Reference: DBSCAN Algorithm: Complete Guide and Application with Python Scikit-Learn

Density Reachable

Density Reachable: Point a is density reachable from a point b with respect to $\in$ and $MinPts$, if -

For a chain of points $b_1, b_2, ...b_n$, where $b_1 = b$ and $b_n = a$, such that $b_i + 1$ is directly density reachable from b_i.

Density reachable is transitive in nature but, just like direct density reachable, it is not symmetric.

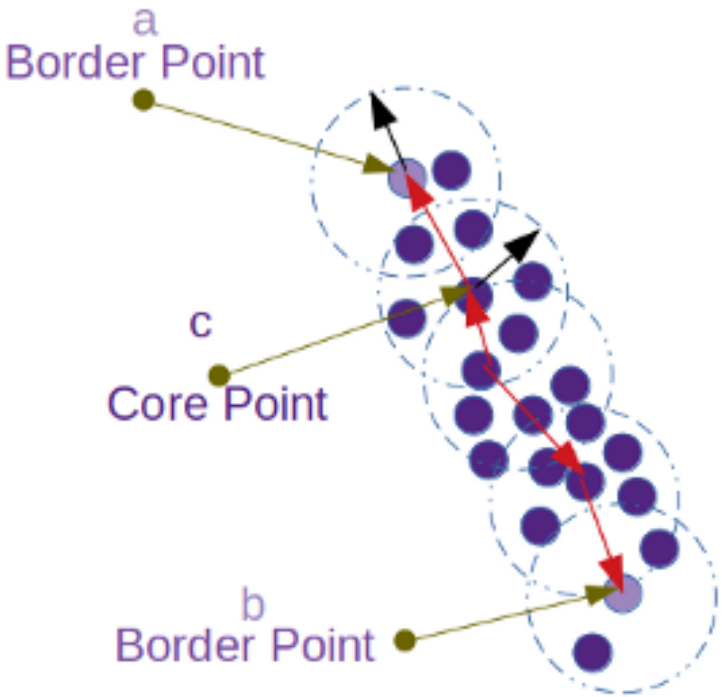
Reference: DBSCAN Algorithm: Complete Guide and Application with Python Scikit-Learn

Density Connected

Density Connected: There can be cases when 2 border points will belong to the same cluster but they don't share a specific core point, then we say that they are density connected if, there exists a common core point, from which these borders points are density reachable. As you can understand that density connectivity is symmetric. Definition from the Ester et.al. paper is given below —

"A point a is density connected to a point b with respect to $\in$ and $MinPts$, if there is a point c such that, both a and b are density reachable from c w.r.t. to $\in$ and $MinPts$."

Reference: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise



a, b are Density Reachable from a core point c.

a, b are called Density Connected points.

Figure: DBSCAN points explanation

Reference: DBSCAN Algorithm: Complete Guide and Application with Python Scikit-Learn

DBSCAN Pros

1. It requires minimum domain knowledge.
2. It can discover clusters of arbitrary shape.
3. Efficient for large database, i.e. sample size more than few thousands.
4. The user does not need to know how many clusters exist in the data.
5. DBSCAN works best when the clusters are of the same density (distance between points).
6. DBSCAN identifies outliers as noise, instead of classifying them into a cluster.

DBSCAN Cons

1. DBSCAN algorithm fails in case of varying density clusters.
2. Fails in case of neck type of dataset.
3. Does not work well in case of high dimensional data.

Mean Shift

Mean shift is a non-parametric density based clustering algorithm. Like DBSCAN, the user does not need to tell the algorithm the number of clusters. It is sometimes referred to as the Mode-seeking algorithm.

"Given a set of data points, the algorithm iteratively assigns each data point towards the closest cluster centroid and direction to the closest cluster centroid is determined by where most of the points nearby are at. So each iteration each data point will move closer to where the most points are at, which is or will lead to the cluster center. When the algorithm stops, each point is assigned to a cluster."

Essentially, the algorithm shifts each point toward the mode of a points within a certain radius.

Bandwidth is the primary parameter for mean shift clustering. It is essentially the window size for the kernels'

Reference: ML, Mean-Shift Clustering

Mean Shift Algorithm

1. Initialize the data points as cluster centroids.

2. Repeat the following steps until convergence or a maximum number of iterations is reached:

   1. For each data point, calculate the mean of all points within a certain radius (i.e., the "kernel") centered at the data point.
   2. Shift the data point to the mean.
   3. Identify the cluster centroids as the points that have not moved after convergence.
   4. Return the final cluster centroids and the assignments of data points to clusters.

Reference: ML, Mean-Shift Clustering

Mean shift used kernel density estimation (KDE) to estimate the underlying distribution (or probability density function) for the data set. A kernel is placed on each data point, which is essentially a weighting function. Adding these up creates the probability surface.

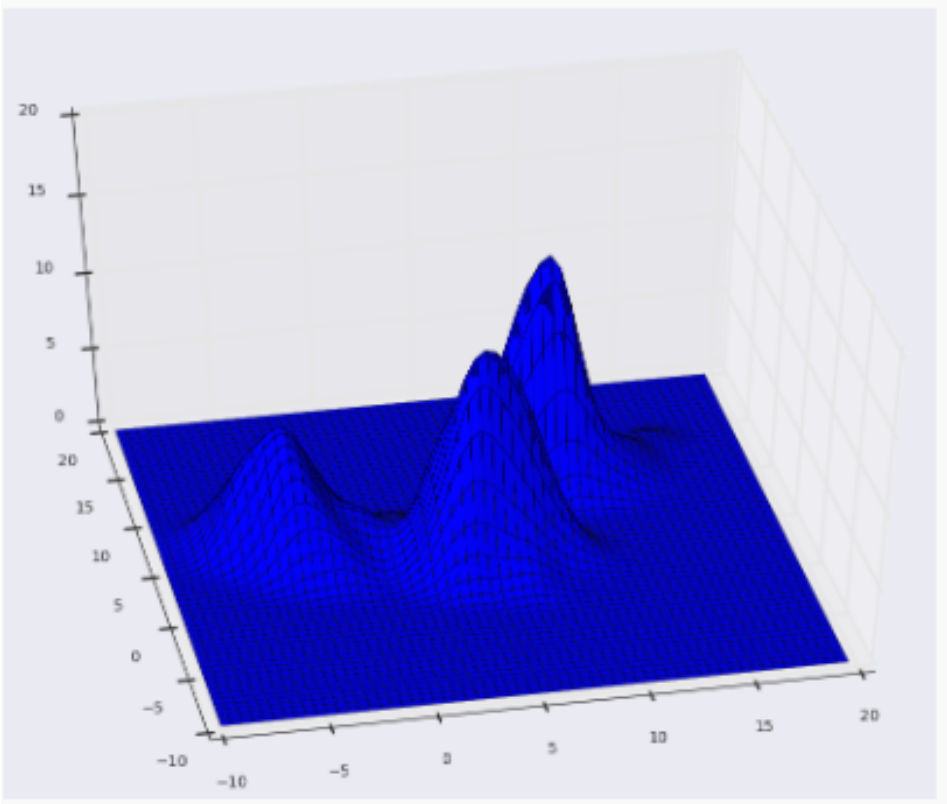


Figure: Mean Shift Surface Plot

Reference: Mean Shift

Mean Shift Pros

1. Finds variable number of modes
2. Robust to outliers
3. General, application-independent tool
4. Model-free, doesn't assume any prior shape like spherical, elliptical, etc. on data clusters. Thus it works very well for spherical-shaped data
5. Just a single parameter (window size h) where h has a physical meaning (unlike k-means)

Mean Shift Cons

1. Output depends on window size
2. Window size (bandwidth) selection is not trivial
3. Computationally (relatively) expensive
4. Doesn't scale well with dimension of feature space

OPTICS

Ordering Points To Identify the Clustering Structure (OPTICS) is a density based algorithm that is similar to DBSCAN, it can extract clusters of varying shapes and densities and is particularly useful in large data sets with high-dimensionality.

OPTICS has the following concepts in addition to those of DBSCAN.

1. "**Core Distance:** it is the minimum value of radius required to classify a given point as a core point. if the given point is not a Core point, then its Core Distance is undefined."

2. "**Reachability Distance:** it is defined with respect to another data point 'q'. The Reachability distance between a point p and q. Note that The Reachability Distance is not defined if 'q' is not a Core point."

Additionally, OPTICS uses K Nearest Neighbors approach instead of a radius measurement when determining core points

Reference: OPTICS CLUSTERING (Intro)

Core Distance

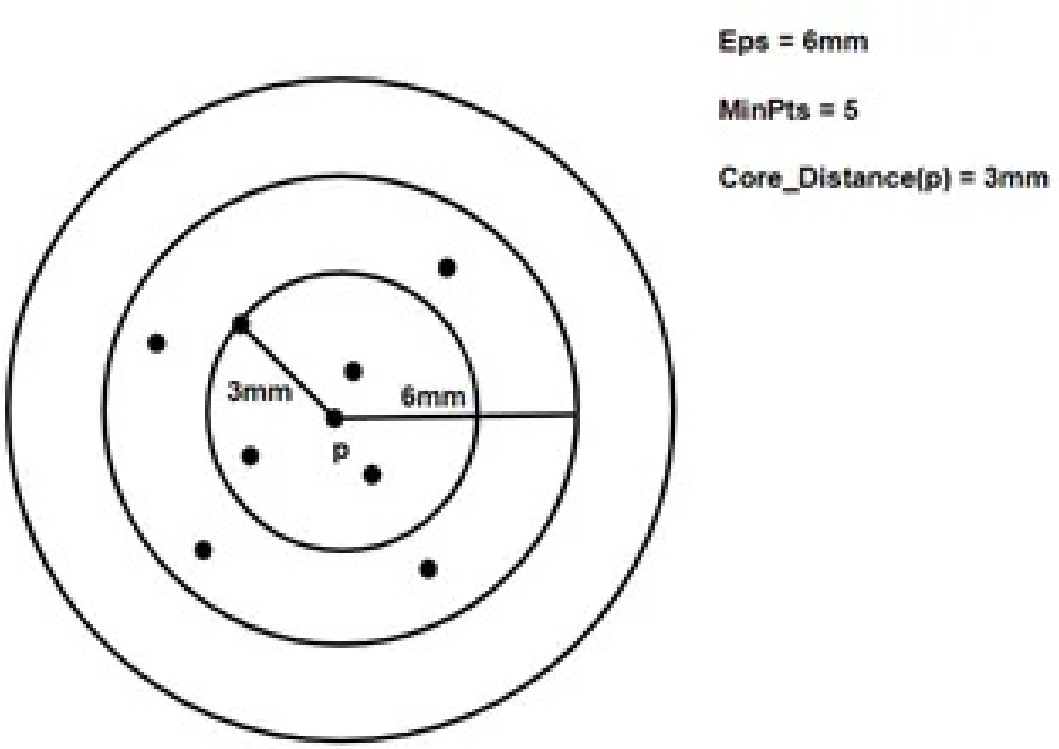


Figure: Core Distance

Reference: OPTICS CLUSTERING (Intro)

Reachability Distance

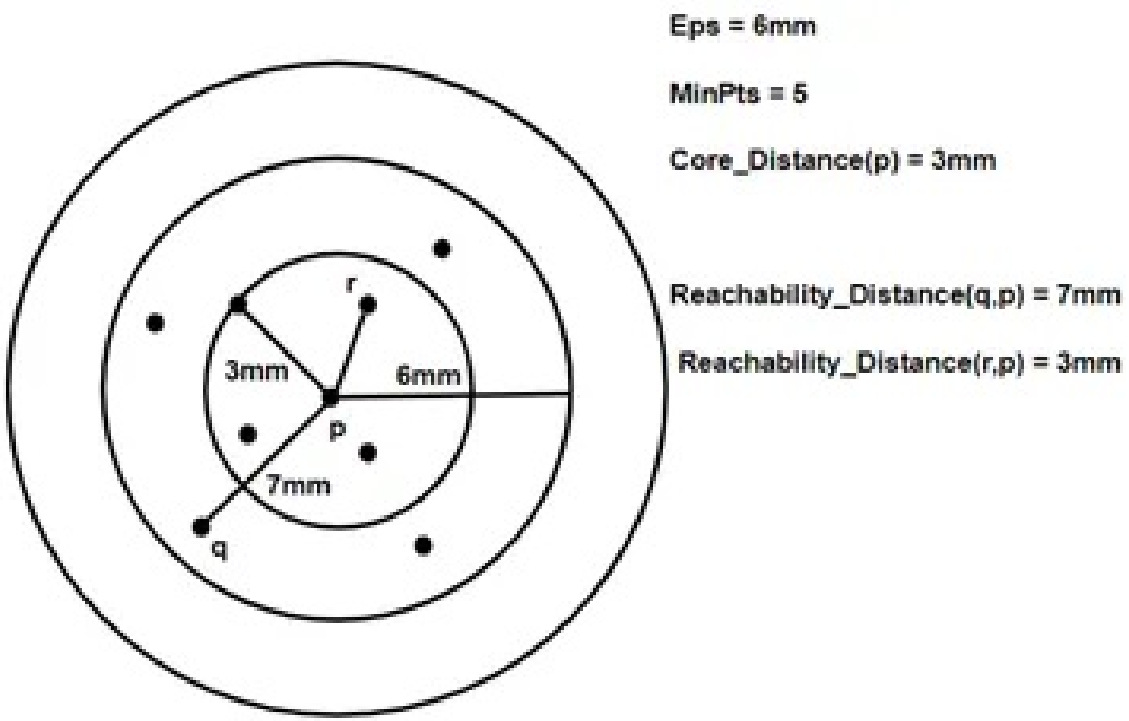


Figure: Reachability Distance

Reference: OPTICS CLUSTERING (Intro)

OPTICS Pros

1. "OPTICS clustering doesn't require a predefined number of clusters in advance."
2. "Clusters can be of any shape, including non-spherical ones."
3. "Able to identify outliers(noise data)."

Reference: OPTICS CLUSTERING (Intro)

OPTICS Cons

1. "It fails if there are no density drops between clusters."
2. "It is also sensitive to parameters that define density(radius and the minimum number of points) and proper parameter settings require domain knowledge."
3. OPTICS requires more memory than DBSCAN and has a much higher runtime complexity (longer execution time).
4. Requires fewer parameters than DBSCAN.
5. Does not explicitly identify noise points, but points with high reachability distances can be considered noise.

Reference: OPTICS CLUSTERING (Intro)

Table of Contents

Affinity Propagation

"Affinity Propagation was first published in 2007 by Brendan Frey and Delbert Dueck in Science. In contrast to other traditional clustering methods, Affinity Propagation does not require you to specify the number of clusters. In layman's terms, in Affinity Propagation, each data point sends messages to all other points informing its targets of each target's relative attractiveness to the sender. Each target then responds to all senders with a reply informing each sender of its availability to associate with the sender, given the attractiveness of the messages that it has received from all other senders. Senders reply to the targets with messages informing each target of the target's revised relative attractiveness to the sender, given the availability messages it has received from all targets. The message-passing procedure proceeds until a consensus is reached. Once the sender is associated with one of its targets, that target becomes the point's exemplar. All points with the same exemplar are placed in the same cluster."

Reference: Affinity Propagation Algorithm Explained)

Exemplars

Exemplars a data point that best represents its respective cluster.

The "responsibility" matrix R has values $r(i, k)$ that quantify how well-suited xk is to serve as the exemplar for x_i, relative to other candidate exemplars for x_i.

The "availability" matrix A contains values $a(i, k)$ that represent how "appropriate" it would be for x_i to pick x_k as its exemplar, taking into account other points' preference for x_k as an exemplar.

Reference: Affinity Propagation in ML

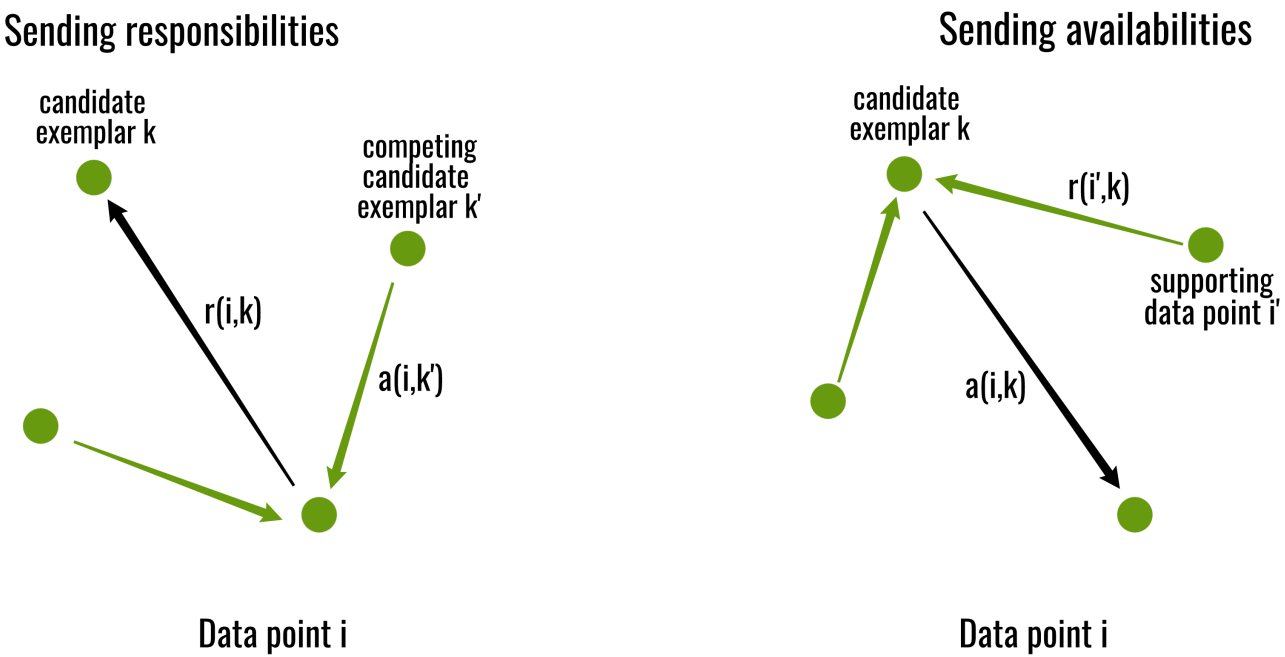


Figure: Affinity Propagation

Reference: Affinity Propagation in ML

Affinity Propagation Pros & Cons

Pros:

1. User does not need to specify the number of clusters
2. Identifies exemplars, which can provide insight into the structure of data

Cons:

1. Slow and memory heavy, thus not good for large datasets
2. User must specify sample preference and damping hyperparamaters
3. Assumes underlying clusters are globular

Reference: Advantages and disadvantages of each algorithm use in Machine Learning

Spectral Clustering

Spectral clustering is a way to cluster data that has a number of benefits and applications. It relies on the eigenvalue decomposition of a matrix, which is a useful factorization theorem in matrix theory. It is a technique with roots in graph theory, where the approach is used to identify communities of nodes in a graph based on the edges connecting them. The method is flexible and allows us to cluster non graph data as well.

Similarity Graphs

The $\in$-neighborhood graph: Here we connect all points whose pairwise distances are smaller than $\in$.

As the distances between all connected points are roughly of the same scale (at most $\in$), weighting the edges would not incorporate more information about the data to the graph.

Hence, the $\in$-neighborhood graph is usually considered as an unweighted graph.

Reference: A Tutorial on Spectral Clustering

Similarity Graphs

K-nearest neighbor graphs: Here the goal is to connect vertex vi with vertex v_j if v_j is among the k-nearest neighbors of v_i. However, this definition leads to a directed graph, as the neighborhood relationship is not symmetric. There are two ways of making this graph undirected.

The first way is to simply ignore the directions of the edges, that is we connect v_i and v_j with an undirected edge if v_i is among the k-nearest neighbors of v_j or if v_j is among the k-nearest neighbors of v_i. The resulting graph is what is usually called the k-nearest neighbor graph.

The second choice is to connect vertices v_i and v_j if both v_i is among the k-nearest neighbors of v_j and v_j is among the k-nearest neighbors of v_i. The resulting graph is called the mutual k-nearest neighbor graph. In both cases, after connecting the appropriate vertices we weight the edges by the similarity of their endpoints.

Reference: A Tutorial on Spectral Clustering

Similarity Graphs

The fully connected graph: Here we simply connect all points with positive similarity with each other, and we weight all edges by s_{ij} . As the graph should represent the local neighborhood relationships, this construction is only useful if the similarity function itself models local neighborhoods.

An example for such a similarity function is the Gaussian similarity function $s(xi, xj) = exp(-(xi - xj(2/(2\sigma^2))$, where the parameter σ controls the width of the neighborhoods. This parameter plays a similar role as the parameter $\in$ in case of the $\in$-neighborhood graph.

Reference: A Tutorial on Spectral Clustering

Graph Laplacians

The main tools for spectral clustering are graph Laplacian matrices.

The unnormalized graph Laplacian $L = D - W$

The normalized graph Laplacians - There are two matrices which are called normalized graph Laplacians in the literature.

Both matrices are closely related to each other and are defined as $L_{sym} := D-1/2LD-1/2 = I-D-1/2WD-1/2Lrw := D-1L = I-D-1W.$

We denote the first matrix by L_{sym} as it is a symmetric matrix, and the second one by L_{rw} as it is closely related to a random walk Eigenvectors and Eigenvalues.

Reference: Introduction to Spectral Clustering

Spectral Clustering Pros & Cons

Pros:

1. Clusters are not assumed to be any certain shape/distribution, in contrast to say k-means. This means the spectral clustering algorithm can perform well with a wide variety of shapes of data.
2. Works quite well when relations are approximately transitive (like similarity)

Cons:

1. Need to choose the number of clusters k, although there is a heuristic to help choose
2. Can be costly to compute, although there are algorithms and frameworks to help
3. For very large datasets computing eigenvectors is computationally expensive and becomes the bottleneck

Table of Contents

Anomaly Detection

"Anomaly Detection is the technique of identifying rare events or observations which can raise suspicions by being statistically different from the rest of the observations." [1]

Anomaly detection can be supervised, unsupervised, or semi-supervised.

There are two broad categories of anomaly detection, outlier detection and novelty detection.

Note: Any clustering algorithm that we haves discussed can potentially be used as unsupervised anomaly detection: A "normal" cluster and an "abnormal" cluster

Any classification algorithm can be used as supervised anomaly detection: A "normal" class and an "abnormal" class.

Outlier vs. Novelty Detection

In **outlier detection**, you are looking for anomalies that may be in your dataset. Outlier detection is an unsupervised approach. It is used when you suspect there is anomalous data points that you wish to remove prior to using the data.

In **novelty detection**, you are planning on identifying anomalies that may exist in new data. Novelty detection is either supervised or semi-supervised.

Local Outlier Factor

Local Outlier Factor (LOF) is an unsupervised method of outlier detection. An anomaly score, called the local outlier factor, is generated for each input sample. The LOF is a measure of a samples deviation for the density with respect to its neighbors. The lower the density score, the more likely that the sample is an outlier. Neighbors are loosely defined as the previous samples that are closest to the current one. A "nearest neighbors" algorithm is used to generate neighborhoods.

LOF Continued

LOF can be used for both Outlier Detection and Novelty Detection.

Parameters of Note:

1. **n_neighbors** - this is a variable for the number of neighbors to use for a comparison.
2. **algorithm** - this is a variable for choosing the algorithm to be used to generate neighborhoods. The options are **'auto'**, **'ball_tree'**,**'kd_tree'**,**'brute'**. Note that **'auto'** selects the best method based on the parameters passed by the user.
3. **metric** - this variable to for the distance metric that will be used. 'euclidean' is one of the most common, but Sklearn offers a large number of options for distance metrics.

They are all listed here: SK Learn LOF

LOF Outlier Code

lof = pipeBuild_LocalOutlierFactor(algorithm=['ball_tree','kd_tree'],**novelty=[False]**)

names=['LOF']

pipes=[lof]

skn.gridsearch_outlier(names=names,pipes=pipes,X=x,y=y,plot_number=3)

LOF Novelty Code

lofn =
pipeBuild_LocalOutlierFactor(algorithm=['ball_tree','kd_tree'],**novelty=[True]**)

names=['LOF Novelty']

pipes=[lofn]

gridsearch_outlier(names=names,pipes=pipes,X=x,y=y,plot_number=3)

Generate Sine & Square Waves

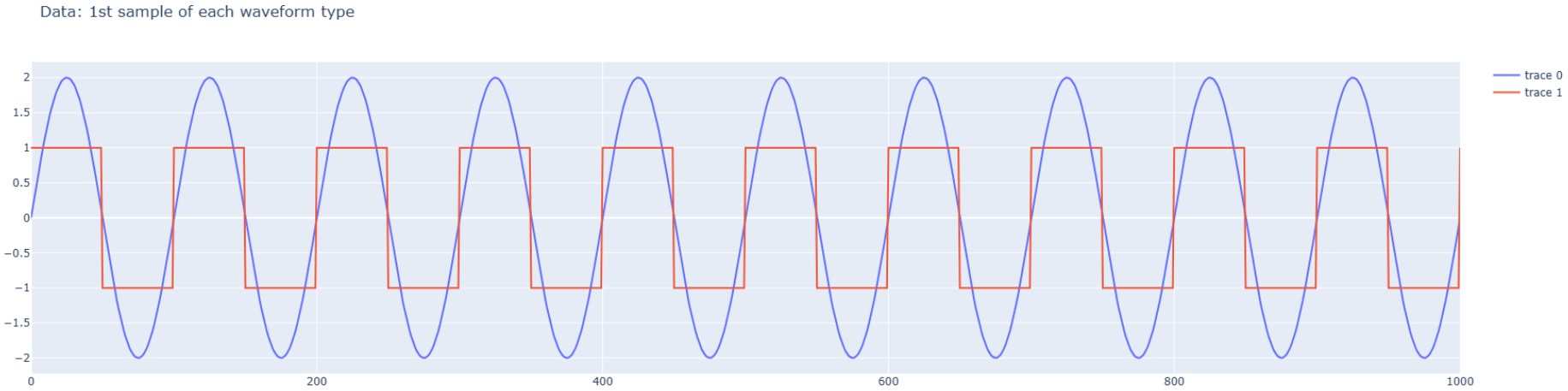


Figure: Normal Case: Sine Waves, all other Abnormal

LOF Outlier Output

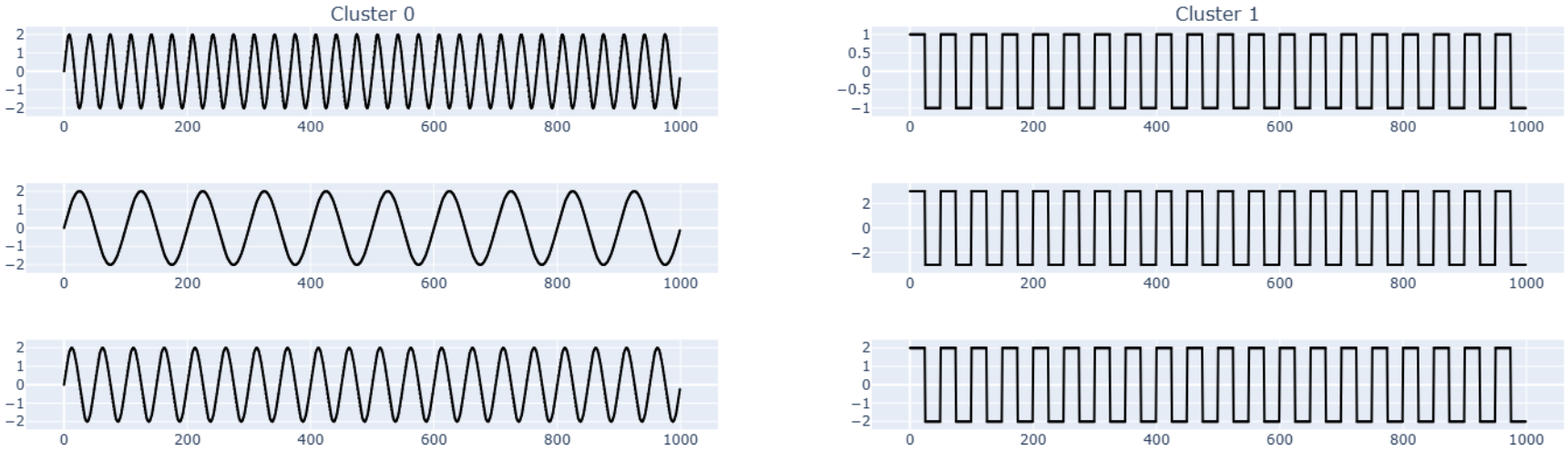


Figure: LOF Outlier Output

LOF Novelty Output

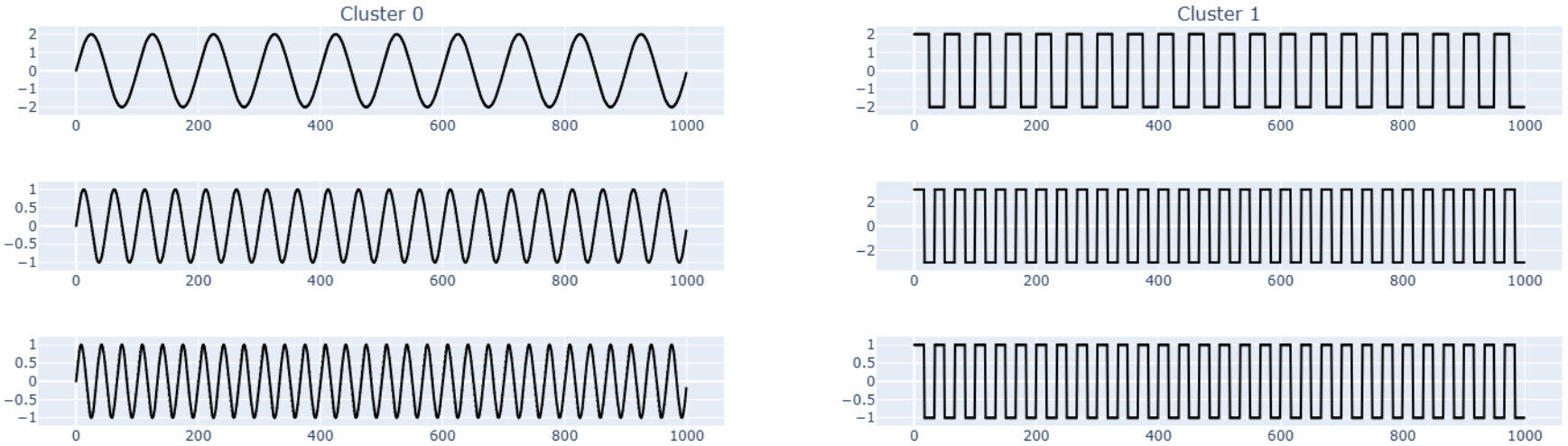


Figure: LOF Novelty Output

Isolation Forests

An **isolation forest** is an unsupervised machine learning algorithm for anomaly detection based on the Decision Tree algorithm. It uses an ensemble approach.

It randomly selects features from the data set and the randomly selects a split point between the minimum and maximum values of the selected feature. This approach will create short paths to anomalous data points. [ref]

Thus Isolation Forests begin by identifying outliers without attempting to first establish what is normal. Additionally, it does not use a distance measure that is typical of other algorithms. [ref]

Isolation Forests continued

It isolates the outliers by randomly selecting a feature from the given set of features and then randomly selecting a split value between the max and min values of that feature. This random partitioning of features will produce shorter paths in trees for the anomalous data points, thus distinguishing them from the rest of the data. [ref]

It assumes "that anomalies are observations that are few and different, which should make them easier to identify. Isolation Forest uses an ensemble of Isolation Trees for the given data points to isolate anomalies." [ref]

An ensemble of decision trees is created to classify data points. Each tree is built using different subsets of the input features, similar to the random forest classifier. It divides the data space into "n buckets" to build each tree. Higher anomaly scores are given to data points that require fewer splits to become isoslated. [ref]

IF Pros & Cons

Pros:

1. Faster than other algorithms due to lack of distance or density measures [1] [2]
2. Works well on small datasets [1]
3. Popular algorithm [2]
4. Linear time complexity mean low computational resources [2]
5. Work well with high dimensional problems with irrelevant attributes [2]

Cons:

1. Algorithm cannot identify local anomaly points which can lower accuracy [2]
2. Since it is unsupervised, it can be difficult to determine if the algoritm is working correctly [2]

IF Code

```
iso = skn.pipeBuild_IsolationForest(n_estimators=[50,100])

names=['Isolation Forest']

pipes=[iso]

skn.gridsearch_outlier(names=names, pipes=pipes, X=x, y=y, plot_number=3)
```

IF Results

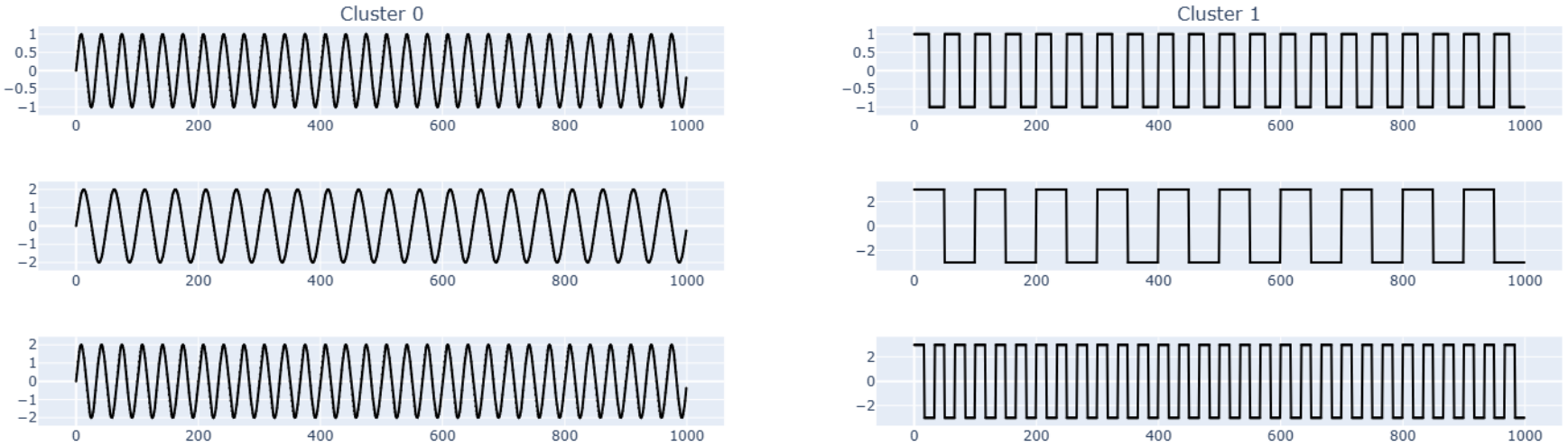


Figure: Isolation Forest Results

Table of Contents

Thank you!

E-mail: stephen.coshatt@uga.edu