Journal of
Sensor and Actuator Networks

Article

Introducing and Comparing Recent Clustering Methods for Massive Data Management in the Internet of Things

Christophe Guyeux [1,*,†], Stéphane Chrétien [2,3,†], Gaby Bou Tayeh [1,†], Jacques Demerjian [4,†] and Jacques Bahi [1,†]

1 Femto-ST Institute, UMR 6174 CNRS, University of Bourgogne-Franche-Comté, 90000 Besançon, France; gaby.bou_tayeh@univ-fcomte.fr (G.B.T.); jacques.bahi@univ-fcomte.fr (J.B.)
2 National Physical Laboratory, Teddington, Middlesex TW11 0LW, UK; stephane.chretien@npl.co.uk
3 Laboratoire ERIC, Université Lyon 2, 69500 Bron, France
4 LaRRIS, Faculty of Sciences, Lebanese University, 90656 Fanar, Lebanon; jacques.demerjian@ul.edu.lb
* Correspondence: christophe.guyeux@univ-fcomte.fr
† These authors contributed equally to this work.

Received: 28 October 2019; Accepted: 4 December 2019; Published: 9 December 2019

Abstract: The use of wireless sensor networks, which are the key ingredient in the growing Internet of Things (IoT), has surged over the past few years with a widening range of applications in the industry, healthcare, agriculture, with a special attention to monitoring and tracking, often tied with security issues. In some applications, sensors can be deployed in remote, large unpopulated areas, whereas in others, they serve to monitor confined busy spaces. In either case, clustering the sensor network's nodes into several clusters is of fundamental benefit for obvious scalability reasons, and also for helping to devise maintenance or usage schedules that might greatly improve the network's lifetime. In the present paper, we survey and compare popular and advanced clustering schemes and provide a detailed analysis of their performance as a function of scale, type of collected data or their heterogeneity, and noise level. The testing is performed on real sensor data provided by the UCI Machine Learning Repository, using various external validation metrics.

Keywords: clustering techniques; clustering evaluation; Internet of Things

1. Introduction

The deluge of data manipulated and transiting in complex systems such as wireless sensor networks (WSN) for the Internet of Things (IoT) recently transformed, in a timely fashion, the field of data management and the associated technologies. The need for new scalable data analytics techniques that can guarantee quality of service, reliability, robustness. and a large battery-operated components' lifespan has accordingly become paramount. It has been observed that for many different settings, clustering is a tool of choice for categorizing and interpreting the data in very large databases when no supervision is possible due to scale and time constraints issues. Moreover, and perhaps even more importantly, clustering is also relevant from a technological viewpoint: As sensors-to-sink information transmission is often very expensive, sensors can be grouped into clusters, and compressed groupwise information can be sent in the place of the whole data. Using clusters has led to substantial gains in a network's lifetime in, e.g., hop networks. An example of this is how efficient node scheduling techniques can benefit from clustering by switching off certain sensors whilst enforcing coverage constraint, thereby leading to systematic reduction of energy consumption [1]. Another example where clustering can be extremely useful is at the sink level, where massive and heterogeneous data arrive from myriads of various locations in the network. Clustering may then play a major role in real-time screening, as in the case of setting up alerts for the presence of dangerous pollutants, or more generally in various applications of changepoint detection. Finally, clustering also takes place when an extremely large number of sensors are deployed over large remote areas, suffering from poor signal quality. One specific aspect of clustering in WSN is that data quality is often poor due to signal transmission problems or even due to the use of certain energy saving aggregation and scheduling strategies [2]. Clustering should therefore be tailored with these peculiarities in mind. In summary, in the face of massive and heterogeneous data, clustering is often an essential tool at the various stages of sensor network data analytics.

A noticeable specificity of in the study of Sensor Networks is that the nodes' properties may depend on a certain number of constraints which may affect how the clustering step should be addressed. For instance, some agents such as aggregators may only be able to access much lower computational power as compared to the sinks. One other constraint to take into account is the respective load of the network's nodes. For instance, WSNs may consist of a certain number of subnetworks, such as body sensor networks (BSN), in which the load might be small, but when many individuals wear such BSNs, the resulting collaborative body sensor network [3,4] may have to deal with huge amounts of internal communications (such as patients and doctors connected within a large hospital). As a result, some clustering tasks must be achieved with methods of very low complexity, while others must be very scalable or overcome specific hurdles, such as decentralization, etc.

Motivation and Contribution

Literature on WSNs or the Internet of Things (IoT), where clustering is frequently mentioned as a subroutine in scalable routing algorithms, is filled with very interesting specialized contributions. Nevertheless, general overviews and comparisons appear to be relatively scarce. Worse, most papers present the well-known K-means approach as the state-of-the-art method without any more careful assessment of the pitfalls associated with this method, e.g., the nonconvexity of the cost function which makes it hard to ascertain practical optimality (theoretical optimality being precluded by NP-completeness) or the possible presence of nonspherical clusters, to name a few.

The objective of the present article is to review recent clustering techniques and provide a more rigorous account of when and for which task they might be useful in the context of WSNs and the IoT. Moreover, this paper aims to demonstrate that a few clustering techniques, such as K-means, are not necessarily the magical solution for every clustering problem. Instead, choosing the most appropriate clustering technique(s) according to the given scale (sensor, aggregator, sink), the type of data collected, their heterogeneity, and the possible presence of noise is a difficult but essential problem which deserves proper consideration.

The contributions provided in this paper are the following:

- All the main important techniques which have been devised in the literature, including the necessary details for their implementation in the network, are described.
- A thorough comparison between the main important techniques which have been devised in the literature, as well as their respective advantages and disadvantages, is provided.
- Numerical experiments illustrating the performance of the methods are presented.
- An example with real data for a gas leak detection sensor network is analyzed to show how these various clustering techniques can respond in different ways to data received at the sink level.
- We demonstrate that choosing the correct method makes a difference via performance comparisons on a massive dataset, where some methods show excellent accuracy and scalability, whereas others appear completely inappropriate for the task.

This article is divided into several sections, the first one giving a relatively complete overview of clustering methods and algorithms that are relevant for application to IoT. Section 3 presents the various established approaches to clustering assessment, divided into two subcategories: those that assume known ideal clustering and those that do not assume such a prior ideal model. These clustering and assessment techniques are then tested through simulation experiments, but also through a case study of a WSN consisting of gas sensors with massive datasets. This article ends with a conclusion presenting a list of future research directions for the analysis of massive datasets in the context of WSNs and IoT.

2. An Overview of Modern Clustering Methods Applicable to the IoT

2.1. Introducing Clustering for IoT

The number of devices and objects that are connected to the Internet is increasing rapidly. These connected objects generate a lot data, which can be analyzed to identify trends and information for various purposes. This is where clustering for IoT [5–18] becomes highly demanded. The advantages that clustering provides are numerous, such as the fact that it enables the scalability of the IoT network and reduces the routing overhead by managing the routing decisions on the elected cluster-heads (CHs) [19,20]. Moreover, it helps with saving communication bandwidth and drastically reduces the overhead for topology maintenance. In addition, only the CHs and the gateway will form the backbone of the network, resulting in a simplified topology, reduced overhead, flooding, and collision. The end devices' only task is to connect to the CHs and forward the data without being affected by changes at the inter CH tier. The aggregation of collected data on the CH reduces the number of exchanged packets. Finally, various management strategies such as scheduling that could be implemented on the CH level can help to preserve energy resources and extend the lifetime of the network [21–23].

As can be seen, clustering can be used in a variety of ways to improve the quality and operational safety of wireless sensor networks, while extending their lifespan. However, the situation changes from one network to another, depending on its scope, the number of sensors considered, their resources, the quality of the data produced, etc. A single way of clustering objects cannot be appropriate for all these situations, nor for the very diverse reasons (aggregation, hop-by-hop routing, data clustering at sink level) requiring their implementation. Therefore, in the following, various recent clustering techniques and their evaluations are recalled or introduced, all of which have a potential interest in the Internet of Things.

2.2. Ellipsoid-Shaped Clusters

2.2.1. K-Means

Generalities

The K-means algorithm clusters the data by trying to separate individuals into groups of equal variance, thus minimizing inertia, or the sum of intracluster squares. Its typical use case is presented in Table 1, while pros and cons are detailed in Table 2. Given k initial centers, we want to partition $\Omega = \{x_1, ..., x_m\}$ into k disjoint subsets, trying to minimize the Euclidean distance between each x_i and its assigned center. In other words, we want to minimize the criterion:

$$\min \left\{ \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - c_i||, (C_1, ..., C_k) \in P(\Omega) \right\}$$

where:

- $P(\Omega)$ is the set of possible subsets of Ω.
- c_i is the center of the C_i cluster.

The initial centers can be chosen in various ways: random, with K-means++ (see below), etc., and the algorithm iterates as follows:

- Assign each x_i to its nearest center c_j
- Recalculate each center as an average of the x_i closest to it.

Note that there are various tricks, based on triangular inequality, for example, to speed up the process. On the other hand, if K-means generally converges quite well with Euclidean distance, this convergence is less assured with other distances. Finally, the K-means has a K-Median variant, based, as its name suggests, on the median. As a result, it is less sensitive to outliers, but slower (including sorting to obtain the median).

Examples of the application of K-means in the context of IoT can be found in refs. [21,24,25]. For instance, k-means clustering is used for data irregularities detection and pattern matching in ref. [24], to extend the lifetime of the network in ref. [21], and in ref. [25] for anomaly detection.

Table 1. Use case of clustering techniques.

Method	Parameters	Cluster Size	Nb of Clusters	Geometry
Affinity propagation	damping sample preference	distinct	large nb of clusters; does not scale up with the nb of individuals	non flat
Birch	numerous	large data sets	large nb of clusters	non flat
DBSCAN	neighborhood size	distinct	Very large set of individuals average number of clusters	non flat
GMM	nb of clusters	not scalable	not scalable	flat
Hierarchical clustering	nb of clusters link type, distance	distinct	many clusters many samples	hierarchical
K-means	nb of clusters	regular	not too many	flat
K-Medoids	nb of clusters	regular	not too many	flat
Mean-shift	bandwidth	distinct	many	non flat
Spectral	nb of clusters	distinct	small	non flat

2.2.2. K-Means++

The choice of initial centers is crucial for K-means. It has a significant impact on the results. K-means++ is an initialization algorithm. It avoids choosing two initial centers too closely. Its principle is as follows:

- The initial centers are chosen randomly, but with a non-uniform probability law.
- The probability of choosing a point as the initial center is proportional to the square of the distance from that point to the already selected centers.

This can be written in algorithmic form as listed in Algorithm 1.

Algorithm 1 K-means++ initialization

```
Require: $K$: Number of clusters
Require: $m$ elements $x_i$ to be clustered
  $c_1 \leftarrow x_1$
  for $k = 1, ..., K - 1$ do
    $s_0 \leftarrow x_0$
    for $i = 1, ..., m$ do
      $s_i = s_{i-1} + min(||x_j - c_1||^2, ..., ||x_j - c_i||^2)$
    end for
    pick $r$ randomly in $[s_1, s_m]$
    find $l$ such that $r$ belongs in $[s_l, s_{l+1}]$
    $c_j \leftarrow x_l$
  end for
```

Table 2. Pros and cons of clustering techniques.

Methods	**Advantages**	**Disadvantages**
Affinity propagation	- Nb of clusters not required - One main parameter: the damping factor [1]	- Quadratic complexity in the nb of points - There may not be convergence
Birch	- Outlier deletion, data reduction - If large number of subclusters is desired	- Does not adapt very well to large data [2]
DBSCAN	- No need to provide a priori the nb of clusters - Useful for identifying outliers, noise - Can find clusters of arbitrary size and shape - Works well for clusters of homogeneous density	- Works less well than others when the clusters to be found have different densities [3] - Does not work well in very high dimension
GMM	- Intuitive algorithm associated with maximum likelihood maximisation - Soft clustering: each point belongs to all clusters, but with different probabilities - Allows an estimation of density - model based approach that comes with easy to implement information theoretic penalties for model selection (number of clusters, etc.)	- Not scalable
Hierarchical clustering	- Nb of clusters not required - The most appropriate cluster nb can be chosen afterwards - Choice of distance not critical - Works well when data is naturally hierarchical allows you to recover this hierarchy	- Its specificity for hierarchical data - High impact of outliers - Slow: $O(n^3)$
K-means	- Fast (linear complexity) - Easy to implement - Proven in many applications - Scalable: for a large number of individuals and a reasonable number of clusters [4]	- We have to choose the nb of clusters - Not very good, outside of spherical clusters - Consistency problem: different runs produce different clustering, due to its random initialization
K-Medoids	- Robust against noise and presence of outliers	- We have to choose the nb of clusters
Mean-shift	- No need to provide a priori the nb of clusters - The centers of the balls converge towards the places of highest density	- Window size can be difficult to fix
Spectral	- Useful when cluster structure is highly non-convex - Very effective with sparse affinity matrix	- Nb of clusters must be provided - Not recommended for large nb of clusters

Notes: [1] increasing it would tend to reduce the number of clusters; [2] it is generally preferable to use MiniBatchKMeans; [3] HDBSCAN* solves this issue; [4] via Mini Batch K-means.

As we can see, as an input, K-means++ receives the m elements x_i to cluster (without the centers). At the output, it produces the initial K centers for K-means. Among the advantages of the method, K-means++ offers convergence guarantees and has a parallelized version. However, this technique requires finding the nearest center to each element for each use, and this explodes with the number of centers.

Examples of the application of K-means++ in the context of IoT can be found in refs. [19,20]. For instance, K-means++ clustering is used for finding the optimal path for hop-by-hop routing in ref. [19], and for an energy efficient routing algorithm in ref. [20].

2.2.3. K-Medoids

The K-Medoids [26] or PAM (partitioning around medoids) is an adaptation of the K-means, seeking to minimize the distance between the points of clusters and their center, the difference being that the medoids are necessarily points of the set to be clustered (which is not necessarily the case of the K-means centers). Thus, a medoid can be seen as the individual of a cluster whose average dissimilarity to the individuals of said cluster is minimal: It is to some extent the most central individual of the cluster.

The PAM algorithm is the most common realization of the K-Medoids, but other approaches exist in the literature (e.g., a method based on Voronoi iterations). PAM follows a gluttonous approach, which does not guarantee to find the optimum but is much faster than an exhaustive search. Its operation is as follows:

- Initialization: choose k from n points for medoids.
- Associate each point with its closest medoid.
- As long as the cost of configuration decreases:
  - For each medoid m and each non-medoid o:
    - Exchange m and o, associate each point with its closest medoid, and recalculate the cost (sum of the distances of the points to their medoids).
    - If the total cost of the configuration has increased in the previous step, cancel the exchange.

The complexity of this algorithm is in $O(k(n-k)^2)$ and can be improved to $O(n^2)$.

Examples of the apllication of K-Medoids in the context of IoT can be found in refs. [22,23]. For example, in ref. [22], K-Medoids clustering is used to improve the low energy adaptive clustering hierarchy (LEACH) and extend the lifetime of the network. Similarly, In ref. [23], the K-Medoids is used to improve the data routing and reduce energy consumption.

2.2.4. Gaussian Mixture Model

One of the main concerns about *K-means* is its naive use of the mean value for the cluster center.

- We can, for example, have two circular clusters of the same center and different radii, which K-means will not be able to capture.
- Similarly, if the clusters are better modelled using an ellipsoid-shaped subset, K-means might not converge to the correct solution.

The Gaussian mixture model (GMM) [27] gives more flexibility to K-means by assuming that each point in the observed sample has been drawn from a Gaussian distribution, with parameters depending on the cluster it belongs to. As a result, we no longer assume circular clusters but allow for elliptical shapes in 2D, 3D, etc. Each cluster will then be defined by the associated probability of belonging to it, its mean vector, and Covariance matrix.

More precisely, GMM assumes that the points follow a distribution:

$$\sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k)$$

with $\mu_1, ..., \mu_K$ the cluster centers, $\pi_1, ..., \pi_K$ the probability weights, and $\Sigma_1, ..., \Sigma_K$ the variance–covariance matrices of each cluster. It is an unsupervised and parametric model, as we are looking for a distribution in the form of a Gaussian distribution mean. The parameters are optimized according to a maximum likelihood criterion to get as close as possible to the desired distribution. This procedure is most often done iteratively via the expectation-maximization (EM) algorithm [28], and its Kullback proximal generalizations [29,30], Gauss–Seidel version [31].

One of the main advantages of the Gaussian mixture model is that it has statistical underpinnings that allow for various penalizations allowing a principled model order selection based on, e.g., AIC or BIC [27] of ICL [32], or even sparsity, inducing a penalized version of the maximum likelihood approach [33].

The EM algorithm works as follows:

1. We select the number of clusters and randomly initialize the parameters of the Gaussian distributions of each cluster;
2. Given these Gaussian distributions for each cluster, we calculate the probability that each point belongs to each cluster: The closer a point is to a Gaussian center, the more likely it is to belong to the associated cluster (Expectation part);
3. Based on these probabilities, we re-estimate the Gaussians: We calculate a new set of parameters of Gaussian distributions, in order to maximize the probability of the points to be in the clusters (Maximization part). These new parameters are calculated using a weighted sum of the point positions, the weights being the probabilities that the points belong to this cluster;
4. We repeat this until the distributions no longer change, or almost: We evaluate the log-likelihood of the data to test convergence (this is the objective function to be increased).

Finally, let us note the following remarks: (1) K-means is a special case of GMM, i.e., with a constant covariance per component (GMM is much more flexible in terms of covariance than K-means); (2) rhe EM depends in particular on the number of times it is launched (100 is a good choice), and strongly from the initialization: We can randomly extract k observations of X as means of the initial components (k: number of Gaussians) or choose them by K-means++, which is preferable.

As far as we know, the GMM clustering technique has not yet been implemented in any IoT related application

2.3. Density-Based Clustering

2.3.1. Mean-Shift

Mean-shift [34] is a clustering algorithm based on a sliding window (ball), which searches for dense areas of points by moving the centroids of the balls. We try to locate the center of each class, updating the candidates as an average of the points in the window. The windows are finally postprocessed to eliminate overlaps.

This method works as follows:

1. For nucleus, we consider a sliding ball centered at a random point C and of radius r. Through a hill climbing approach, this nucleus is iteratively moved to an area of higher density, until convergence;
2. This is done by moving the center of the ball towards the center of gravity of the ball's points, which causes the ball to move to denser areas;
   - As the ball has been moved, there are potentially new points, and, therefore, a new center of gravity;
   - Otherwise, we end up converging.
3. Points 1 and 2 are operated with various randomly placed balls, or according to a grid adapted to the data, and when several balls overlap, the less dense one is removed.

Once convergence is established, the remaining balls are the clusters. Refer to Tables 1 and 2 for the context of use, as well as the advantages and disadvantages of this method.

Examples of the application of mean-shift in the context of IoT can be found in refs. [5,6]: in ref. [6], for instance, mean-shift clustering is used for localization and tracking in IoT applications.

2.3.2. DBSCAN

The density-based spatial clustering of applications with noise (DBSCAN [35]) is another density-based clustering: It looks for high-density areas and then extends clusters from them. Its general operating principle is as follows.

1. We start from a point P of the data, not yet visited;
    - If there are enough neighbors (min_samples-1) at ε from this point, clustering starts with P as the first (core) point of the cluster;
    - Otherwise, the point is labeled as noise (which can later integrate a cluster) and visited;
2. The points at distance ε of P integrate the cluster of P, then all the points at ε of these points, etc., until it is no longer possible to expand the cluster;
3. We start again with a point not visited, and this until the points are exhausted. At the end, any point will either be in a cluster or labeled as noise.

DBSCAN is therefore similar in its operation to the mean-shift, although it has some advantages, see Table 2. It has an optimized version called HDBSCAN*, which extends DBSCAN "by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters" [36].

Examples of the application of DBSCAN in the context of IoT can be found in refs. [7–9]. In ref. [7], the DBSCAN clustering algorithm is used for anomaly detection, while in ref. [8], an improved version of DBSCAN is used for uncertain data clustering problems. Finally, DBSCAN clustering is used to detect faulty sensor data in ref. [9].

2.4. Tree-Based Clustering

2.4.1. Hierarchical Clustering

Hierarchical clustering algorithms are either top–down or bottom–up: We start from points seen as simple clusters, which we iteratively aggregate in pairs, until we reach a single final cluster. This bottom–up approach is precisely what is called agglomerative hierarchical clustering, and this hierarchy of clusters can be represented as a dendrogram: The root is the final single cluster, the leafs being the data points. Unlike the agglomerative version of the hierarchical clustering, the divisive starts with a single cluster encompassing all points then iterates divisions until only clusters at one point are obtained.

The agglomerative algorithm can be summarized as follows.

1. Each data point is treated as a single cluster, and a distance between clusters is fixed—for example, the average linkage: the average distance between the points of two clusters;
2. At each iteration, the two clusters with the shortest distance are merged, and the height of the new node in the dendrogram is the similarity between said clusters;
3. Repeat 2 until you only get one cluster left;
4. If a predefined number of clusters is wanted, step number 2 is stopped when the number is reached.

Its divisive version can be easily deduced from this. Various distances can be used:

single linkage: The distance between two clusters is the distance corresponding to the two most similar points.

complete linkage: As above, but with the least similar points.
average linkage: The average distance.
Ward: Minimization of the sum of the squares of distances within each cluster. It is an approach of the variance minimization type, and therefore a kind of K-means coupled with a hierarchical agglomerative approach.

2.4.2. Birch

Birch [37] is an online learning algorithm, memory-efficient, which can be seen as an alternative to Mini Batch K-means. It builds a tree, in which the centroids of the clusters are at the level of the leaves. These can either be the centroids of the final clustering or they can be passed as an input to another clustering algorithm, such as agglomerating clustering.

The data tree consists of nodes, each node consisting of a number of subclusters. The maximum number of subclusters in a node is determined by the connection factor. Each subcluster stores a variety of information necessary for the inline process, such as the number of samples in that subcluster or the average of its points providing the centroid. In addition, each subcluster can also have a node as a child if the subcluster is not in a leaf.

Every new individual is introduced to the root. It is merged with the nearest subcluster and the information of this subcluster is updated, which is recursively made until it reaches a leaf.

The parameters of this method are as follows:

- The threshold: The radius of the subcluster obtained by merging a new individual and the nearest subcluster must be less than this threshold. If this is not the case, a new subcluster is initialized. A low value of this threshold therefore multiplies the number of node breakdowns and, thus, the clusters;
- The branching factor: The maximum number of subclusters in each node. If a new individual entering a node causes the number of subclusters to exceed this branching factor, then the node is split into two, which distribute the subclusters. The parent subgroup of this node is deleted, and two new subclusters are added as parents of these two cut nodes;
- Number of clusters after the last clustering step.

Note that, to the best of our knowledge, the Birch clustering technique has not yet been considered in any IoT-related application.

2.5. Other Methods

2.5.1. Spectral Clustering

Spectral clustering [38] applies K-means to a low dimensional immersion of the affinity matrix between samples, i.e., to the normalized Laplacian matrix:

$$D^{-1/2}(D-A)D^{-1/2},$$

where D is the degree matrix of the graph whose adjacency matrix is the affinity A. This affinity matrix is constructed using:

- Either a kernel function, for example, the RBF $e^{-\gamma d(X,X)^2}$ or the heat kernel $e^{-\beta \frac{d(X,X)}{std(d)}}$, where d is the distance between individuals, while β and γ are hyperparameters to set up;
- Or a connectivity matrix in the k-nearest neighbors;
- Or, via the precomputed parameter, an affinity matrix provided by the user.

As with other clusterings detailed here, the pros and cons of this technique are provided in Tables 1 and 2.

A study of Laplacian-eigenvector-based clustering in the context of overlapping clusters is available in ref. [39].

Examples of the application of spectral clustering in the context of IoT can be found in refs. [10–12]. For instance, in ref. [10], spectral clustering is used to detect disconnected segments of the sensor network caused by depletion of battery or physical tempering of nodes. Conversely, in ref. [11], spectral clustering is used to detect and eliminate sensor nodes that are not working properly.

2.5.2. Affinity Propagation

Affinity propagation (AP [40]) is a clustering algorithm based on the notion of passing messages between data points. As with the K-Medoids, the AP is looking for models ("exemplars"), i.e., points in the dataset that could be good cluster representatives. The AP starts from a similarity matrix and exchanges messages (real numbers) between the points to be clustered until quality clusters emerge naturally. During message exchange, the algorithm identifies exemplary points, or models, that are able to properly describe a cluster.

Let s be a similarity matrix for the points to be clustered, in which $s(i,i)$ designates the "preference" of the i entry, which means how likely the i element is to be found as a model. These $s(i,i)$ are typically initialized at the median of the similarities, knowing that initializing them at a value close to the smallest of the similarities will lead to fewer clusters (and vice versa for a value close to the maximum).

The algorithm alternates two message passing steps, updating the following two matrices:

- The R matrix of competence (responsibility), in which the element (i,k) quantifies how much x_k is justified as a model for x_i, compared to the other candidates.
- The availability matrix A, in which the element (i,k) represents how appropriate it would be for x_i to take x_k as a model, once all the other points have been considered as a model.

These two matrices, which can be seen as log-probability tables, are initialized with 0's. The following steps are then iterated:

- First, we circulate the skill updates: $r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\}$;
- Then the availability is updated by:
  - $a(i,k) \leftarrow \min \min \left(0, r(k,k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k))\right)$ for $i \neq k$; and
  - $a(k,k) \leftarrow \sum_{i' \neq k} \max(0, r(i',k))$.

Iterations are performed either until the cluster boundaries no longer move or after a predefined number of iterations. The models are extracted from the final matrices as being self-competent and self-available elements (i.e., $r(i,i) + a(i,i)) > 0$).

Examples of the application of Affinity propagation clustering in the context of IoT can be found in refs. [13–16]. For instance, in ref. [13], AP is used to improve the LEACH protocol, while in refs. [14,15], improved versions of the AP for sensor data clustering are proposed. Finally, in ref. [16], AP is used for optimal routing path selection.

2.5.3. Constrained Clustering

There is a separate branch of clustering that integrates the notion of constraints. Thus, in COP (constrained pairwise) K-means [41], it is specified that this and that individual must be linked, while that and that other must not, whereas MinSizeKmeans forces a minimum size for clusters. Other types of semisupervised clustering can be found in the literature, such as seeded-KMeans, constrainted-KMeans, pairwise constrained K-means (PCK-means), metric K-means (MK-means), or metric pairwise constrained K-means (MPCK-means), to name a few. The implementation of such active semi-supervised clustering algorithms can be found, e.g., in ref. [42], while ref. [43] contains the Python implementation of most usual clustering methods.

3. Clustering Evaluation: State-Of-The-Art

There are two different approaches to validation of clustering results. In external validation, we have access to the real clusters, and we want to measure to what extent the considered algorithm is able to recover the ground truth. In internal validation, we do not have access to the ground truth, and an alternative hint about the relevance of the number and shape of the clusters is needed. External validation provides more relevant and meaningful measures for testing on simulated data, but the hypothesis of having access to the desired solution is obviously not realistic in practical implementations. Many libraries already implement most of these metrics, such as ref. [43].

3.1. External Validation

3.1.1. Homogeneity, Completeness, and V-Measure

If we know the expected labels, we can define some intuitive measures based on an analysis of conditional entropy. Thus, Rosenberg and Hirschberg [44] defined the following two objectives, which are obviously desirable for any clustering:

Homogeneity: Each cluster contains only members of a single class;
Completeness: All members of a given class are in the same cluster.

These two scores are between 0 and 1 (the higher the value, the better the clustering), and the *V-measure* is the harmonic mean of the latter two.

In more detail, homogeneity and completeness are defined mathematically as follows:

$$h = 1 - \frac{H(C \mid K)}{H(C)}, \quad c = 1 - \frac{H(K \mid C)}{H(K)}$$

where:

- $H(C \mid K)$ is the conditional entropy of classes knowing clusters, i. e. $-\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n} log \frac{n_{c,k}}{n_k}$,
- $H(K)$ is the entropy of the class, equal to $-\sum_{c=1}^{|C|}\frac{n_c}{n} log \frac{n_c}{n}$,

with n the number of individuals, n_c and n_k that of the c class and the k cluster, and $n_{c,k}$ the number of individuals in the c class assigned to the k cluster, and, for the completeness, the adaptation of the latter. Finally, the V-measurement is the harmonic mean of h and c, i.e.,

$$v = 2\frac{hc}{h+c}.$$

The following points can be noted. First of all, a permutation of the class or cluster labels will not change the value of these scores. Further, homogeneity and completeness are not symmetrical: Changing true labels with predicted labels will move them from one score to another (in other words, homogeneity_score(a, b) = completeness_score(b, a)). V-measure, for its part, is symmetrical; it is equivalent to mutual information, with the arithmetic mean as an aggregation function. This V-measure can be used to evaluate the agreement of two independent clustering on the same data set.

Among the advantages of these measures, we can mention the fact that a clustering with a bad V-measure (i.e., close to 0) can be analyzed qualitatively, in terms of homogeneity and completeness, to better interpret the type of error made. On the other hand, there is no hypothesis on the structure of clusters, so we can compare various clustering approaches (K-means vs. spectral, etc.)

Conversely, these methods have some disadvantages. First of all, the actual labeling must be known to calculate these scores. Moreover, these metrics are not adjusted to account for chance: Depending on the number of individuals, clusters and real classes, completely random labeling will not necessarily produce the same values of homogeneity, completeness, and therefore V-measurement. In particular, random labeling is not necessarily zero, especially for a large number of clusters. However, this problem can be ignored if the number of individuals exceeds 1000 for a number of clusters of less than 10. If this is not the case, prefer the adjusted Rand index (ARI) recalled below.

3.1.2. Adjusted Rand Index

The Rand index (RI) calculates a similarity between two clusterings, by looking at each peer of individuals and counting those that are or are not in the same cluster, depending on whether you are in actual or predicted clustering:

$$RI = \frac{a+b}{\binom{n}{2}}$$

where a is the number of points that are in the same cluster for both clusterings, b is for those that are in a different cluster for both clusterings, and n is the total number of samples. As such, the RI does not guarantee that random assignment will produce a value close to 0. This is why this raw index is "adjusted to account for chance", which gives the ARI (adjusted Rand index) score:

$$ARI = \frac{RI - Expected\_RI}{max(RI) - Expected\_RI}.$$

The ARI, which is symmetrical, measures the similarity and the consensus of two assignments, ignoring permutations and normalizing against what would have happened by chance. The ARI score has various advantages:

- Random labeling leads to an ARI close to 0, regardless of the number of points and clusters, which is not the case for RI or V-measure;
- The ARI varies between -1 and 1 included. Negative values are for independent labeling, when similar clusterings have a positive ARI (1 for an exact match);
- No hypothesis on the structure of the clusters: we can therefore compare K-means to spectral clustering, leading a priori to very different structures.

Its main disadvantage is that it assumes that the actual expected labeling is known. To have a full understanding of the values returned by the ARI, it can be noted that a label placing all individuals in the same clusters is complete, but not always pure, and is therefore penalized. In addition, the ARI is symmetrical, so labeling leading to pure clustering with members from the same classes, but with unnecessary class splits, is also penalized. Finally, if the class members are completely separated in different clusters, then the assignment is totally incomplete, and therefore, the ARI is very low.

3.1.3. Fowlkes–Mallows Score

When we know the classification of individuals whose clustering is done, we can calculate the Fowlkes–Mallows FMI score [45] as the geometric mean of the precision and recall per pair:

$$FMI = \frac{TP}{\sqrt{(TP+FP)(TP+FP)(TP+FN)}}$$

where:

- TP is the number of true positives: The number of pairs of points in the same cluster in the real and predicted labeling;
- FP is the number of false positives: Pairs with the same real labeling, but in different predicted clusters;
- FN is the number of false negatives: Pairs in the same predicted clusters, but with different actual labels.

Clearly, labels can be swapped or renamed, leading to the same score. further, a perfectly predicted clustering will have an FMI of 1, while a clustering independent of the real classes of 0. The FMI has various advantages:

- Unlike mutual information or V-measure, a random clustering will have an FMI score of 0, regardless of the number of clusters or individuals;

- A value close to 0 indicates largely independent labeling, while a value close to 1 indicates clustering agreement;
- Two equal labels (with permutation) have an FMI of 1;
- Finally, there is no hypothesis on the structure of clusters.

Its main disadvantage is that the actual labeling must be known in order to use this measure.

3.1.4. Based on Mutual Information

Mutual Information

Entropy is the amount of uncertainty for a partitioning $U = (U_i)$, which can be defined by:

$$H(U) = -\sum_{i=1}^{|U|} P(i)\log(P(i))$$

where $P(i) = |U_i|/N$ is the probability that a randomly drawn object will end up in the U_i cluster. The mutual information between two partitionings U and V is defined by [46]:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

where $P(i,j) = |U_i \cap V_j|/N$ and $P'(j) = |V_j|/N$ are probabilities in the obvious sense. This mutual information can also be written as follows:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N}\log\left(\frac{N|U_i \cap V_j|}{|U_i|.|V_j|}\right)$$

This is a symmetrical measurement less than or equal to 1. Values close to 0 indicate that the labels are largely independent, while values close to 1 indicate a significant agreement between the two clustering. However, perfect labeling does not necessarily have an MI of 1; on the other hand, random labeling has a negative MI. This score is independent of permutation or renaming, but it requires knowledge of true labeling, and unlike AMI defined below, MI is not adjusted to account for chance.

Normalized Mutual Information

It is defined on the basis of mutual information as follows [47]:

$$NMI(U,V) = \frac{MI(U,V)}{mean(H(U),H(V))}$$

This is again a symmetrical measurement and increased by 1. Similarly, values close to 0 indicate that the labels are largely independent, when values close to 1 indicate a significant agreement between the two clustering. In addition, perfect labeling has an NMI of 1, and random labeling has a negative NMI. Finally, in the absence of true labeling, the NMI can be used as a model selection criterion: Clusterings on various hyperparameters producing a score of 1 are agreed.

It should also be noted that this score is independent of permutation or renaming, but like all measurements in this subsection, it requires knowledge of true labeling. Finally, unlike the AMI defined below, the NMI is not adjusted to account for chance.

Adjusted Mutual Information

The adjusted mutual information, which is a more recent technique than normalized mutual information, is for its part adjusted to account for chance [47]:

$$AMI(U,V) = \frac{MI(U,V) - E\{MI(U,V)\}}{\max\{H(U),H(V)\} - E\{MI(U,V)\}},$$

where $E\{MI(U,V)\}$ is the expected mutual information between two random clusterings.

This is again a symmetrical measurement and bounded by 1. Perfect labeling has an AMI of 1, and random labeling has an AMI close to 0: values close to 0 indicate that the labels are largely independent, when values close to 1 indicate a significant agreement between the two clusterings. As before, the AMI can be used as a model selection criterion (clusterings on various hyperparameters producing a score of 1 are the consensus). Finally, this score is independent of permutation or renaming.

3.2. Internal Validation

3.2.1. Elbow Method

The "Elbow method" is a technique that considers the elbow that appears when the number of clusters is plotted on the x-axis and the percentage of variance explained on the y-axis [48]. This method is old and rather rudimentary, although popular.

In practice, it can be calculated in two ways, for each potential number of clusters k:

- We sum the intracluster variances, which we divide by the overall variance;
- The sum of squared errors (SSE) is calculated: the sum, on all clusters, of the square distances between the points and their centroids.

For instance, in Figure 1, the elbow in the SSE appears for a number of clusters equal to 3.

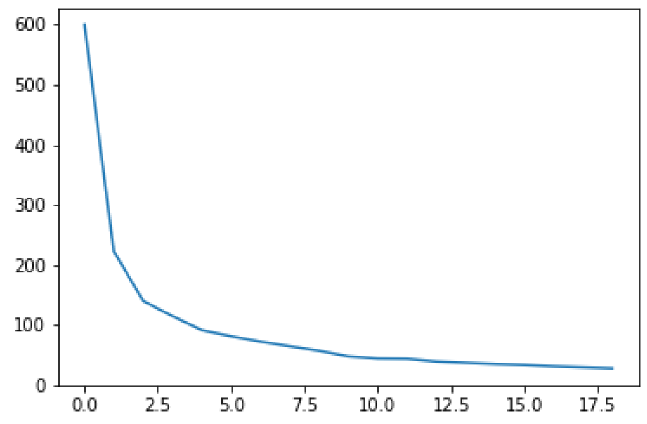

Figure 1. Illustration of the Elbow method: sum of squared errors (SSE) versus nb of clusters.

The following points can then be noted. For the K-means method, the average of each cluster is its centroid. For normalized data, the mean involved in the total variance is the origin. Finally, it is ultimately an F-test in the variance approach.

3.2.2. Indices

Calinski-Harabaz

In the absence of knowledge of actual labels, the *Calinski–Harabaz* index [49], also known as the variance ratio criterion, measures how well clusters are defined. This score is defined as the ratio between the average of the dispersions between clusters and the intracluster dispersion; the larger the score, the better.

More precisely, for k clusters, this score is equal to:

$$s(k) = \frac{tr(B_k)}{tr(W_k)} \frac{N-k}{k-1},$$

where:

- $W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$ is the intracluster dispersion;
- $B_k = \sum_{q=1}^{k} n_q (c_q - c)(c_q - c)^T$ is the dispersion matrix between groups;

with N the total number of points and c its center, when C_q is the set of points in the cluster q, n_q its number of points, and c_q its center.

The advantages of this metric are that the score is calculated quickly, and it is higher when the clusters are dense and well separated, which is usually expected in good clustering. On the other hand, the Calinski–Harabaz index is generally broader for convex clusters, which does not fit well, for example, with clusters generally obtained by density-based techniques such as DBSCAN.

Davies–Bouldin

This index is defined by the average similarity between each cluster $(C_i)_{i=1...k}$ and its most similar C_j. For the purposes of this index, this similarity is defined as the $R_{i,j}$ compromise between:

- The diameter s_i of the cluster C_i: average distance between each point of this cluster and its centroid;
- $d_{i,j}$: the distance between the centroids of the i and j clusters.

A simple way to construct such a measure, so that it is positive and symmetrical, is:

$$R_{i,j} = \frac{s_i + s_j}{d_{i,j}}.$$

The Davies–Bouldin index is then defined by [50]:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{i,j}.$$

Clusters are better separated when the Davies–Bouldin index is low. The highest score is 0, and partitioning is better when the index is low. The main advantages of this index is that:

- It can be calculated without the knowledge of true clustering;
- The calculation of the DB is simpler than the Silhouette scores, described hereafter;
- The index is calculated only from distances.

Conversely, its disadvantages are:

- It does not evaluate very well in a situation of nonconvexity;
- A good value produced by this index does not mean that the information collected is maximum.

Dunn

The Dunn index [51] is an internal clustering validation measure, which is calculated as follows:

- For each cluster, the distance between each point of the cluster and the points of the other clusters is calculated. The minimum of these distances corresponds to the intercluster separation (min.separation);

- The distance between the points of each cluster is calculated, its maximum diameter being the intracluster distance reflecting the compact nature of the clustering.

  Dunn's index is then equal to:
$$D = \frac{min.separation}{max.diameter}$$

If clusters are compact and well separated, the diameter of the clusters should be small when the distance between clusters should be large. Thus, a good clustering is associated with high values of this index.

3.2.3. Silhouette

The silhouette method [52] allows estimating the consistency of the points belonging to clusters. The silhouette value measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). For each point, this value is between -1 and 1:

- Close to 1, the object fits very well with its own cluster and very badly with other clusters;
- Near 0, the data are at the border of two clusters;
- Close to -1, the object would fit better in the neighboring cluster: probably a bad assignment.

If most objects have a high silhouette value, then clustering is appropriate. Otherwise, it probably indicates too few or too many clusters. Let:

- $a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j)$ be the average distance between i point and the other points of its cluster, measuring how much i fits well with its own cluster (value is the smaller the better the assignment is); and
- $b(i) = \min_{i \neq j} \frac{1}{|C_j|} \sum_{j \in C_j} d(i,j)$ be the smallest average distance between i and the points of all other clusters, the average dissimilarity of a point i to a cluster C being seen as the average of the distances between i and the points of C; the cluster with the smallest average dissimilarity is therefore the cluster close to i.

The silhouette value of object i is then $s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$ if the i cluster has more than one point, and 0 otherwise.

The average silhouette value on all points of a cluster assesses the extent to which the points of the cluster are closely grouped. The average value over the entire dataset indicates whether it has been appropriately clustered. If there are too many or too few clusters, some of them will typically have much weaker silhouettes than other clusters. Thus, the display and averages of the silhouettes can be used to determine the natural number of clusters in a dataset.

4. Comparison of Modern Clustering Techniques for the IoT

4.1. Experimental Protocol: A Network of Gas Sensors

The objective of this section is to show that the various clustering methods mentioned above may behave differently depending on the IoT application in question, and that it is therefore important to consider the framework in which the study is conducted. Typical scenarios for the use of each method have been described in a previous section, and we have indicated the particularities and application contexts associated with each method, the situations in which it can be expected to work well, as well as those in which these methods behave poorly. The metrics used to assess the quality of the results produced have been introduced to show that various measures provide different types of information on the performance of the various clustering techniques presented in this article. All this is illustrated in this section, in a context of the Internet of Things already mentioned in this article: the difficult case of a network of sensors deployed for gas detection.

The data we consider are based on a concrete experiment produced by Alexander Vergara of the University of California San Diego. It is entitled "Gas Sensor Array Drift Dataset at Different Concentrations Data Set" and can be found on the UCI Machine Learning Repository [53]. In detail, this archive contains 13,910 measurements from 16 chemical sensors exposed to 6 different gases (ethanol, ethylene, ammonia, acetaldehyde, acetone, and toluene) at various concentration levels. The scenario here is that the sink receives the values of these 13,910 measurements, and it must cluster them in order to find 6 clusters corresponding to each of the 6 gases. In this clustering application at IoT, the operator knows which sensors they have deployed and the types of gases they want to detect. In other words, they know the number of clusters to recover, and since they themselves have exposed their network to these 6 gases at various concentrations, they know the true cluster. Its goal is to find the method to best separate all its measurements in order to make its sensor network operational once this cluster-based postprocessing has been implemented. We would like to note that the gas sensors are not a generalized version of any IoT system, and different data sets collected by different types of sensor networks can yield different results than the one obtained in this experiment.

In the following, we therefore evaluate the clustering techniques for which it is possible to determine the expected number of clusters beforehand. In detail, these are K-means with random initialization or following K-means++, agglomerative clustering (we consider here, for comparison purposes, the linkages of type ward, complete, and average), the Gaussian mixture model, and Birch. First of all, it should be noted that spectral clustering was excluded from the study: The size of the dataset was too large and led to a memory error, showing that this technique should be excluded in the case of IoT networks producing a large amount of data (as expected following our previous presentation).

We also wish to show, in this study, that clustering work for the IoT often benefits greatly from being preprocessed, such as with an outlier removal step followed by a phase of dimensional reduction step. These two steps are generally neglected in the literature on clustering in an IoT context. However, we saw through the various clustering evaluations obtained that adding such steps greatly improve the results. Concerning the detection and exclusion of outliers, we considered the cases where such a step has not been carried out, then the implementation of a basic technique (i.e., isolation forest [54]), followed by an optimized version (extended isolation forest [55]). In terms of dimension reduction, we considered four scenarios: (1) the absence of such a step; (2) its realization by the most traditional method, namely, the PCA; (3) a modern version of the latter optimized to the problem we are interested in, namely, the sparse PCA [56] (sensors for 6 different gases, exposed to a single gas, should ideally lead to a sparse matrix); (4) and finally the famous t-SNE [57] technique. Regarding clustering quality measures, the following list was considered: homogeneity, completeness, adjusted Rand index, V-measure, Fowlkes–Mallows, and AMI.

The data were first standardized. Then, we performed (or not, as the case may be) the outlier removal operations followed by dimension reduction, before performing the clustering and evaluating its quality. The calculations were carried out in Python. For extended forest isolation, we used the library available in ref. [55]. The rest of the clustering methods, forest isolation, and various dimension reduction and evaluation techniques were performed with the scikit-learn [43] library. At the parameter level, reasonable default values were considered: For instance, the branching factor and the threshold were set at 50 and 0.5, respectively, for Birch, while the damping was set at 0.5 in the AP.

4.2. Results of the Survey

First of all, it should be recalled that we have deliberately chosen to consider, in this experiment, the problem of detecting various gases at different concentrations, by means of a sensor network. A great diversity in the variability in the results after clustering has been observed, ranging from random-like results (frequently obtained) to results that carry richer structural information, given the complexity of the task considered in this experiment. In order to visualize the results better and assess the results in all the tables in this section, values are highlighted using a three color code:

- Blue: The values highlighted in blue are the highest recorded quality metrics with no outlier removal;
- Green: The values highlighted in green are the ones showing improvement as compared to the values obtained with no outlier removal;
- Red: The values highlighted in red are the ones showing degraded performance as compared to the values obtained with no outlier removal.

The clustering technique is ranked according to who has the highest values for the majority of the quality metrics (the closer to 1.0, the better). In addition, as explained earlier in Section 3 all the metric values range from either 0 to 1 or from −1 to 1. However, using the python libraries mentioned in Section 4.1, we sometimes obtained values that exceeded 1 or were below −1. Therefore, these values are ignored and removed from the analysis.

Table 3 shows the obtained results when only the outlier removal operations (or not) have been performed without any reduction in dimensionality afterward. It is clear that, on one hand, the K-means, AC Ward, and the Birch clustering techniques have been improved in term of quality when the Isolation Forest is used to remove outliers. On the other hand, only AC Ward and Birch have been improved when the extended IF is used. However, looking at GMM, which had the best results (excluding Fowlkes–Mallows) when no outlier removal has been performed, a reduction in quality has been recorded when Isolation Forest and extended IF are used. Moreover, despite the improvements in other clustering techniques, none of the values in green matched or exceeded the quality metric values highlighted in blue. Therefore, we can conclude that none of the outlier removal techniques improved the clustering results. GMM without any outlier removal is by far the best.

Table 4 shows the measured quality metrics of the different clustering techniques when the PCA dimensionality reduction method is used after the removal (or not) of outliers. Indeed, a small improvement can be noticed for K-means, K-means++, AC complete, and Birch, regardless of the outlier removal technique that has been used. However, GMM shows an improvement only with extended IF. Similarly to the previous results, the best-performing clustering technique, AC Ward, showed a reduction in quality instead of improvement, with both Isolation Forest and extended IF being used. Moreover, both AC complete and AC complete had a sufficient improvement in their "Fowlkes–Mallows" and "Completeness" values, respectively, to surpass the matching blue metric values of AC Ward. However, still having the best homogeneity, V-measure, ARI, AMI, and AC Ward, without any outlier removal, could be considered the best choices.

In Table 5, where sPCA is used to reduce the dimension of the dataset, a minor improvement can be found for K-means and K-means++, with both outlier removal techniques. However, AC complete and GMM only showed partial improvement with extended IF and Isolation Forest, respectively. In this table, we notice that no clustering technique dominated over the others in terms of quality metrics; thus, we have no clear "winner" yet. However, AC Ward has been improved enough with extended IF in order to surpass the blue values in homogeneity, ARI, V-measure, and AMI. Therefore, the combination of extended IF, sPCA, and AC Ward could be considered better than the others.

The previous experiments clearly show that the performance can have large fluctuations, depending on how the methods are combined in the pipeline. This is corroborated by the reported experiments in Table 6. In this table, we discover that when t-SNE is considered as the dimensionality reduction method, AC average, GMM, and Birch have an improved performance with both Isolation Forest and extended IF. However, K-means and K-means++ only showed improvement with the former, and AC Ward only showed improvement with the latter. This time, Birch is the one that stands out as the clustering technique with the majority of values colored in blue. By looking at the results obtained using Isolation Forest and extended IF, we also observe that AC average outperforms all other approaches in terms of clustering quality. Therefore, we can safely conclude that the combination consisting of Isolation Forest for outlier removal, t-SNE for dimensionality reduction, and AC average for clustering is by far the best combination for the given data set.

Table 3. Clustering scores without dimensionality reduction.

Outliers	Clustering	Homogeneity	Completeness	ARI	V-Measure	Fowlkes–Mallows	AMI
None	kmeans	0.1983	0.2471	0.0797	0.2201	0.2936	0.2196
	kmeans++	0.1739	0.2409	0.0772	0.2020	0.3021	0.2015
	AC ward	0.1757	0.2337	0.0869	0.2006	0.3006	0.2001
	AC complete	0.0003	0.1554	−3.324	0.0006	0.4189	−5.143
	AC average	0.0003	0.1554	−3.324	0.0006	0.4189	−5.143
	GMM	0.3196	0.3613	0.2449	0.3392	0.4012	0.3388
	Birch	0.1811	0.3273	0.0916	0.2332	0.3564	0.2327
Isolation Forest	kmeans	0.2059	0.2619	0.0850	0.2306	0.3020	0.2301
	kmeans++	0.1608	0.2233	0.0772	0.1869	0.3031	0.1865
	AC ward	0.1795	0.2341	0.0854	0.2032	0.2981	0.2028
	AC complete	0.0003	0.1550	−3.581	0.0006	0.4184	−5.419
	AC average	0.0003	0.1550	−3.581	0.0006	0.4184	−5.419
	GMM	0.2783	0.3034	0.1669	0.2903	0.3291	0.2899
	Birch	0.1856	0.3274	0.0922	0.2369	0.3557	0.2365
Extended IF	kmeans	0.1962	0.2447	0.0794	0.2177	0.2934	0.2173
	kmeans++	0.1731	0.2401	0.0768	0.2012	0.3018	0.2007
	AC ward	0.1792	0.2363	0.0889	0.2038	0.3017	0.2034
	AC complete	0.0003	0.1551	−3.503	0.0006	0.4185	−5.334
	AC average	0.0003	0.1551	−3.503	0.0006	0.4185	−5.334
	GMM	0.2728	0.3460	0.1826	0.3051	0.3661	0.3047
	Birch	0.1806	0.3293	0.0939	0.2333	0.3581	0.2328

Table 4. Clustering scores on data reduced by PCA.

Outliers	Clustering	Homogeneity	Completeness	ARI	V-Measure	Fowlkes–Mallows	AMI
None	kmeans	0.1676	0.1893	0.0822	0.1778	0.2682	0.1773
	kmeans++	0.1674	0.1891	0.0820	0.1776	0.2681	0.1771
	AC ward	0.2302	0.2716	0.1131	0.2492	0.3072	0.2488
	AC complete	0.0848	0.1794	0.0690	0.1152	0.3474	0.1146
	AC average	0.0108	0.2496	−0.002	0.0208	0.4115	0.0199
	GMM	0.2071	0.2580	0.0950	0.2298	0.3043	0.2294
	Birch	0.1006	0.2026	0.0115	0.1345	0.3206	0.1339
Isolation forest	kmeans	0.1752	0.1920	0.0863	0.1833	0.2665	0.1828
	kmeans++	0.1754	0.1923	0.0862	0.1835	0.2667	0.1830
	AC ward	0.1889	0.2335	0.0763	0.2088	0.2915	0.2084
	AC complete	0.0745	0.2994	−0.001	0.1194	0.3746	0.1186
	AC average	0.0410	0.1675	−0.006	0.0659	0.3601	0.0651
	GMM	0.2006	0.2346	0.0954	0.2162	0.2920	0.2158
	Birch	0.1282	0.2751	0.0178	0.1749	0.3351	0.1743
Extended IF	kmeans	0.1746	0.1916	0.0855	0.1827	0.2664	0.1822
	kmeans++	0.1750	0.1924	0.0857	0.1833	0.2668	0.1829
	AC ward	0.1479	0.1761	0.0485	0.1608	0.2558	0.1603
	AC complete	0.1315	0.3068	0.0219	0.1841	0.3423	0.1835
	AC average	0.0080	0.2333	-0.001	0.0155	0.4131	0.0146
	GMM	0.2181	0.2553	0.1022	0.2352	0.3011	0.2348
	Birch	0.1522	0.2705	0.0696	0.1948	0.3361	0.1943

Table 5. Clustering scores on data reduced by sPCA.

Outliers	Clustering	Homogeneity	Completeness	ARI	V-Measure	Fowlkes–Mallows	AMI
None	kmeans	0.1975	0.2441	0.0796	0.2184	0.2937	0.2179
	kmeans++	0.2039	0.2515	0.0885	0.2252	0.2989	0.2248
	AC ward	0.2198	0.2851	0.0852	0.2482	0.3102	0.2478
	AC complete	0.0706	0.3567	0.0062	0.1179	0.3887	0.1171
	AC average	0.0129	0.2305	−0.002	0.0244	0.4098	0.0235
	GMM	0.2249	0.2754	0.1017	0.2476	0.3095	0.2472
	Birch	−6.291	1.0	0.0	−1.258	0.4190	−2.293
Isolation forest	kmeans	0.2297	0.2666	0.0979	0.2468	0.2981	0.2463
	kmeans++	0.2297	0.2666	0.0979	0.2468	0.2981	0.2464
	AC ward	0.2247	0.2793	0.0764	0.2490	0.3023	0.2486
	AC complete	0.0732	0.2810	0.0168	0.1162	0.3781	0.1155
	AC average	0.0074	0.2255	-0.001	0.0143	0.4133	0.0134
	GMM	0.2388	0.2839	0.0990	0.2594	0.3074	0.2590
	Birch	-3.143	1.0	0.0	−6.287	0.4186	−6.287
Extended IF	kmeans	0.2313	0.2685	0.0981	0.2485	0.2985	0.2481
	kmeans++	0.2083	0.2522	0.0893	0.2281	0.2981	0.2277
	AC ward	0.2484	0.3249	0.1045	0.2816	0.3294	0.2812
	AC complete	0.1168	0.3113	0.0084	0.1699	0.3485	0.1693
	AC average	0.0079	0.2364	-0.001	0.0154	0.4133	0.0145
	GMM	0.2082	0.2560	0.0943	0.2296	0.3030	0.2292
	Birch	3.1444	1.0	0.0	6.2889	0.4187	1.1054

Table 6. Clustering scores on data reduced by t-SNE.

Outliers	Clustering	Homogeneity	Completeness	ARI	V-Measure	Fowlkes–Mallows	AMI
None	kmeans	0.3519	0.3476	0.2418	0.3497	0.3723	0.3494
	kmeans++	0.3537	0.3495	0.2427	0.3516	0.3730	0.3513
	AC ward	0.3552	0.3593	0.2223	0.3572	0.3618	0.3569
	AC complete	0.3123	0.3571	0.2209	0.3332	0.3793	0.3328
	AC average	0.2922	0.3467	0.1994	0.3171	0.3684	0.3168
	GMM	0.3443	0.3556	0.2138	0.3498	0.3587	0.3495
	Birch	0.3547	0.3608	0.2370	0.3578	0.3749	0.3574
Isolation forest	kmeans	0.3883	0.3857	0.2904	0.3870	0.4134	0.3867
	kmeans++	0.3888	0.3862	0.2908	0.3875	0.4137	0.3872
	AC ward	0.3405	0.3527	0.2436	0.3465	0.3840	0.3462
	AC complete	0.2992	0.3311	0.2338	0.3143	0.3801	0.3140
	AC average	0.4221	0.5147	0.3666	0.4638	0.5066	0.4635
	GMM	0.3608	0.3617	0.2666	0.3613	0.3957	0.3610
	Birch	0.3648	0.3696	0.2820	0.3672	0.4104	0.3669
Extended IF	kmeans	0.3272	0.3233	0.2238	0.3253	0.3572	0.3249
	kmeans++	0.3272	0.3233	0.2238	0.3253	0.3572	0.3249
	AC ward	0.4051	0.4041	0.2972	0.4046	0.4201	0.4043
	AC complete	0.2394	0.2432	0.1434	0.2413	0.2974	0.2409
	AC average	0.3611	0.3723	0.2281	0.3666	0.3705	0.3663
	GMM	0.3723	0.3802	0.2343	0.3762	0.3732	0.3759
	Birch	0.4109	0.4201	0.2926	0.4154	0.4208	0.4151

5. Conclusions

In this paper, we surveyed and classified modern clustering schemes applicable to the Internet of Things (IoT). We categorized the different approaches, highlighted their specificities and their domain of applicability, and we presented and explained the state-of-the-art clustering evaluation methods. Our findings were illustrated on the "Gas Sensor Array Drift at Different Concentration" data set, provided by the UCI Machine Learning Repository. For this dataset, we compared the most relevant clustering schemes and tested various combinations of outlier removal and dimensionality reduction approaches. Great diversity in the quality of clustering was observed, leading to the conclusion that there is no such thing as a simple rule for deciding the best clusters. In particular, the assumption that K-means should be used regardless of the application context did not appear to be true, and we were able to conclude that, instead, removing outliers using the Isolation Forest approach, reducing the dimensionality using t-SNE, and clustering using AC average yielded the best results for these data.

Author Contributions: Conceptualization, C.G., G.B.T. and S.C.; methodology, C.G. and S.C.; software, C.G., G.B.T. and J.D.; validation, S.C. and J.B.; formal analysis, C.G. and G.B.T.; investigation, C.G. and G.B.T.; resources, J.D.; data curation, C.G. and G.B.T.; writing—original draft preparation, C.G.; writing—review and editing, C.G., G.B.T. and S.C.; visualization, G.B.T.; supervision, C.G.; project administration, C.G. and G.B.T.; funding acquisition, J.B.

Funding: This research was funded by the Interreg RESponSE project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Farhat, A.; Guyeux, C.; Makhoul, A.; Jaber, A.; Tawil, R. On the coverage effects in wireless sensor networks based prognostic and health management. *Int. J. Sens. Netw. IJSNET* **2018**, *28*, 125–138. [CrossRef]
2. Farhat, A.; Guyeux, C.; Makhoul, A.; Jaber, A.; Tawil, R.; Hijazi, A. Impacts of wireless sensor networks strategies and topologies on prognostics and health management. *J. Intell. Manuf.* **2017**, *30*, 2129–2155. [CrossRef]
3. Boudargham, N.; Makhoul, A.; Bou Abdo, J.; Demerjian, J.; Guyeux, C. Efficient Hybrid Emergency Aware MAC Protocol for Wireless Body Sensor. *Sensors* **2018**, *18*, 3572. [CrossRef] [PubMed]
4. Boudargham, N.; Bou Abdo, J.; Demerjian, J.; Makhoul, A.; Guyeux, C. Efficient Cluster Based Routing Protocol for Collaborative Body Sensor Networks. In Proceedings of the Sensornets 2019, 8th International Conference on Sensor Networks, Prague, Czech Republic, 26–27 February 2019.
5. Xie, Q.Y.; Cheng, Y. K-Centers Mean-shift Reverse Mean-shift clustering algorithm over heterogeneous wireless sensor networks. In Proceedings of the 2014 Wireless Telecommunications Symposium, Washington, DC, USA, 9–11 April 2014; pp. 1–6. [CrossRef]
6. Zhou, Q.; Li, X.; Xu, Y. Mean Shift Based Collaborative Localization with Dynamically Clustering for Wireless Sensor Networks. In Proceedings of the 2009 WRI International Conference on Communications and Mobile Computing, Yunnan, China, 6–8 January 2009; Volume 2, pp. 66–70. [CrossRef]
7. Saeedi Emadi, H.; Mazinani, S.M. A Novel Anomaly Detection Algorithm Using DBSCAN and SVM in Wireless Sensor Networks. *Wirel. Pers. Commun.* **2018**, *98*, 2025–2035. [CrossRef]
8. Pan, D.; Zhao, L. Uncertain data cluster based on DBSCAN. In Proceedings of the 2011 International Conference on Multimedia Technology, Hangzhou, China, 26–28 July 2011; pp. 3781–3784. [CrossRef]
9. Narasegouda, S. Fault Detection in Sensor Network Using DBSCAN and Statistical Models. In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*; Satapathy, S.C., Udgata, S.K., Biswal, B.N., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 443–450.
10. Hu, H.; Wang, X.; Yang, Z.; Zheng, B. A Spectral Clustering Approach to Identifying Cuts in Wireless Sensor Networks. *IEEE Sens. J.* **2015**, *15*, 1838–1848. [CrossRef]
11. Kung, H.T.; Vlah, D. A Spectral Clustering Approach to Validating Sensors via Their Peers in Distributed Sensor Networks. In Proceedings of the 18th International Conference on Computer Communications and Networks, San Francisco, CA, USA, 3–6 August 2009; pp. 1–7. [CrossRef]

12. Muniraju, G.; Zhang, S.; Tepedelenlioglu, C.; Banavar, M.K.; Spanias, A.; Vargas-Rosales, C.; Villalpando-Hernandez, R. Location Based Distributed Spectral Clustering for Wireless Sensor Networks. In Proceedings of the 2017 Sensor Signal Processing for Defence Conference (SSPD), London, UK, 6–7 December 2017; pp. 1–5. [CrossRef]
13. Sohn, I.; Lee, J.; Lee, S.H. Low-Energy Adaptive Clustering Hierarchy Using Affinity Propagation for Wireless Sensor Networks. *IEEE Commun. Lett.* **2016**, *20*, 558–561. [CrossRef]
14. ElGammal, M.; Eltoweissy, M. Location-Aware Affinity Propagation Clustering in Wireless Sensor Networks. In Proceedings of the 2009 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, Marrakech, Morocco, 12–14 October 2009; pp. 471–475. [CrossRef]
15. Wang, J.; Gao, Y.; Wang, K.; Sangaiah, A.K.; Lim, S.J. An Affinity Propagation-Based Self-Adaptive Clustering Method for Wireless Sensor Networks. *Sensors* **2019**, *19*, 2579. [CrossRef]
16. Sakthidasan, K.; Vasudevan, N.; Diderot, P.K.G.; Kadhiravan, C. WOAPR: An affinity propagation based clustering and optimal path selection for time-critical wireless sensor networks. *IET Netw.* **2018**, *8*, 100–106. [CrossRef]
17. Zhang, T.; Zhao, Q.; Shin, K.; Nakamoto, Y. Bayesian-Optimization-Based Peak Searching Algorithm for Clustering in Wireless Sensor Networks. *J. Sens. Actuator Netw.* **2018**, *7*, 2. [CrossRef]
18. Amaxilatis, D.; Chatzigiannakis, I. Design and analysis of adaptive hierarchical low-power long-range networks. *J. Sens. Actuator Netw.* **2018**, *7*, 51. [CrossRef]
19. Yang, X.; Yan, Y.; Deng, D. Research on clustering routing algorithm based on K-means++ for WSN. In Proceedings of the 2017 6th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 21–22 October 2017; pp. 330–333. [CrossRef]
20. Li, L.; Li, D.; Li, D. An Efficient Routing Algorithm based on K-means++ Clustering and Fuzzy for Wireless Sensor Network*. In Proceedings of the 2018 13th World Congress on Intelligent Control and Automation (WCICA), Changsha, China, 4–8 July 2018; pp. 716–720. [CrossRef]
21. Rajasegarar, S.; Leckie, C.; Palaniswami, M.; Bezdek, J.C. Distributed Anomaly Detection in Wireless Sensor Networks. In Proceedings of the 2006 10th IEEE Singapore International Conference on Communication Systems, Singapore, 30 October–1 November 2006; pp. 1–5. [CrossRef]
22. Bakaraniya, P.; Mehta, S. K-LEACH: An improved LEACH protocol for lifetime improvement in WSN. *Int. J. Eng. Trends Technol.* **2013**, *4*, 1521–1526.
23. Pavithra, M.; Ghuli, P. A novel approach for reducing energy consumption using k-medoids in clustering based WSN. *Int. J. Sci. Res. IJSR* **2015**, *4*, 2279–2282.
24. Mittal, R.; Bhatia, M.P.S. Wireless sensor networks for monitoring the environmental activities. In Proceedings of the 2010 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 28–29 December 2010; pp. 1–5. [CrossRef]
25. Tan, L.; Gong, Y.; Chen, G. A Balanced Parallel Clustering Protocol for Wireless Sensor Networks Using K-Means Techniques. In Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications (Sensorcomm 2008), Cap Esterel, France, 25–31 August 2008; pp. 300–305. [CrossRef]
26. Park, H.S.; Jun, C.H. A simple and fast algorithm for K-medoids clustering. *Expert Syst. Appl.* **2009**, *36*, 3336–3341. [CrossRef]
27. McLachlan, G.J.; Lee, S.X.; Rathnayake, S.I. Finite mixture models. *Annu. Rev. Stat. Its Appl.* **2019**, *6*, 355–378. [CrossRef]
28. McLachlan, G.; Krishnan, T. *The EM Algorithm and Extensions*; John Wiley & Sons: Hoboken, NJ, USA, 2007; Volume 382.
29. Chrétien, S.; Hero, A.O. Kullback proximal algorithms for maximum-likelihood estimation. *IEEE Trans. Inf. Theory* **2000**, *46*, 1800–1810. [CrossRef]
30. Chrétien, S.; Hero, A.O. On EM algorithms and their proximal generalizations. *ESAIM Probab. Stat.* **2008**, *12*, 308–326. [CrossRef]
31. Celeux, G.; Chrétien, S.; Forbes, F.; Mkhadri, A. A component-wise EM algorithm for mixtures. *J. Comput. Graph. Stat.* **2001**, *10*, 697–712. [CrossRef]
32. Biernacki, C.; Chrétien, S. Degeneracy in the maximum likelihood estimation of univariate Gaussian mixtures with EM. *Stat. Probab. Lett.* **2003**, *61*, 373–382. [CrossRef]
33. Chrétien, S.; Hero, A.; Perdry, H. Space alternating penalized Kullback proximal point algorithms for maximizing likelihood with nondifferentiable penalty. *Ann. Inst. Stat. Math.* **2012**, *64*, 791–809. [CrossRef]

34. Comaniciu, D.; Meer, P. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *5*, 603–619. [CrossRef]
35. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd* **1996**, *96*, 226–231.
36. McInnes, L.; Healy, J.; Astels, S. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.* **2017**, *2*. [CrossRef]
37. Zhang, T.; Ramakrishnan, R.; Livny, M. BIRCH: An efficient data clustering method for very large databases. In Proceedings of the ACM Sigmod Record, Montreal, QC, Canada, 4–6 June 1996; Volume 25, pp. 103–114.
38. Von Luxburg, U. A tutorial on spectral clustering. *Stat. Comput.* **2007**, *17*, 395–416. [CrossRef]
39. Chretien, S.; Jagan, K.; Barton, E. Clustering on low-dimensional latent manifolds via Laplacianeigenmaps when clusters overlap. *Meas. Sci. Technol.* **2019**, submitted.
40. Frey, B.J.; Dueck, D. Clustering by passing messages between data points. *Science* **2007**, *315*, 972–976. [CrossRef] [PubMed]
41. Wagstaff, K.; Cardie, C.; Rogers, S.; Schrödl, S. Constrained k-means clustering with background knowledge. *Icml* **2001**, *1*, 577–584.
42. Active-Semi-Supervised-Clustering Repository. Available online: https://github.com/datamole-ai/active-semi-supervised-clustering (accessed on 29 May 2019).
43. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
44. Rosenberg, A.; Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Prague, Czech Republic, 28–30 June 2007.
45. Fowlkes, E.B.; Mallows, C.L. A method for comparing two hierarchical clusterings. *J. Am. Stat. Assoc.* **1983**, *78*, 553–569. [CrossRef]
46. Meilă, M. Comparing clusterings—An information based distance. *J. Multivar. Anal.* **2007**, *98*, 873–895. [CrossRef]
47. Vinh, N.X.; Epps, J.; Bailey, J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.* **2010**, *11*, 2837–2854.
48. Thorndike, R.L. Who belongs in the family? *Psychometrika* **1953**, *18*, 267–276. [CrossRef]
49. Caliński, T.; Harabasz, J. A dendrite method for cluster analysis. *Commun. Stat. Theory Methods* **1974**, *3*, 1–27. [CrossRef]
50. Davies, D.L.; Bouldin, D.W. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **1979**, *2*, 224–227. [CrossRef]
51. Dunn, J.C. Well-separated clusters and optimal fuzzy partitions. *J. Cybern.* **1974**, *4*, 95–104. [CrossRef]
52. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]
53. UC Irvine Machine Learning Repository. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 17 July 2019).
54. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, ICDM '08, Pisa, Italy, 15–19 December 2008; IEEE Computer Society: Washington, DC, USA, 2008; pp. 413–422. [CrossRef]
55. Hariri, S.; Carrasco Kind, M.; Brunner, R.J. Extended Isolation Forest. *arXiv* **2018**, arXiv:1811.02141.
56. Zou, H.; Hastie, T.; Tibshirani, R. Sparse principal component analysis. *J. Comput. Graph. Stat.* **2006**, *15*, 265–286. [CrossRef]
57. Maaten, L.V.D.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.