Marginal Distribution Clustering of Multi-variate Streaming IoT Data

Daniel Puschmann*, Payam Barnaghi* and Rahim Tafazolli*
*Institute for Communication Systems (ICS)
University of Surrey, Guildford, United Kingdom
Email: {d.puschmann, p.barnaghi, r.tafazolli}@surrey.ac.uk

***Abstract*—Analysis and processing of time-series data has been studied over the past decades and has resulted in several promising algorithms. Clustering continuous data is an interesting subset of unsupervised learning models to process and categorise the data. Over a long period and with a large number of samples most conventional data streams will converge to Gaussian distributions. The existing clustering methods for continuous data are usually suitable for this type of data. However, with growing volumes of real world observation and measurements, (i.e. Internet of Things data), the data sets become more volatile and have multi-variate distributions. In this paper we propose a dynamic and adaptable clustering algorithm for multi-variate time-series data clustering. We have evaluated our work against some well-known time-series clustering methods and have shown how the proposed method can reduce the complexity and perform efficient in multi-variate data streams.**

***Index Terms*—Internet of Things; Data Stream Clustering; Concept Drift;**

I. Introduction

With the emergence of the Internet of Things (IoT), dealing with multi-variate data streams has become a key issue as real-world observations and measurements have to be processed and analysed in real-time. Increasingly more physical devices are connected to the Internet, gathering and communicating continuous data, making unsupervised approaches that can deal with the huge amount, velocity and heterogeneity of the data more important than ever. Since the real-world data streams are usually not producing stationary data, the methods have to be able to adapt to changes in the data stream and process the data accordingly.

However, most of the existing approaches are either not adaptive to changes in data streams, require domain-knowledge about the data stream in advance or provide the results only when a clustering request is sent (and not in real-time) [1], [2], [3], [4], [5].

This paper proposes a dynamic approach to cluster evolving multi-variate data streams based on the data distributions. The proposed algorithm automatically determines the best number of clusters based on the current distribution. It remains adaptive to changes in the data stream and can identify the emergence of new clusters. We describe how the marginal distributions of multi-variate data streams are used to cluster the data in a dynamic and adaptive algorithm. Our proposed algorithm will require a memory buffer and will keep a set of samples in a first-in-first-out buffer that is used to obtain current statistics about the data stream such as the average value and the variance to detect the changes in the data distribution.

II. Problem Statement

Data stream processing requires real-time methods that are adaptive to changes in the data stream without human supervision. Data stream clustering provides an unsupervised and automated approach for grouping relevant data into clusters. With the rise of the IoT, there are an increasing amount of applications using real-time data streams from sensing devices. However, accessing the data is not enough, there is a need for approaches that can handle the speed, volume and heterogeneity of the IoT data streams and extract information abstractions from the data streams. Clustering methods allow to organise the data into groups in which similar data items are placed next to each other. These groups can then further be processed to label and analyse the data [6].

III. Related Work

Clustering of data streams is a well-explored research topic. Initially the algorithms were motivated by the fact that the data sets and data bases that had to be clustered were becoming too large to be processed in memory. Therefore one-pass approaches had to be developed such as STREAM [1] and BIRCH [2]. However, these methods work best for data sets with a stationary data distribution. Applying this method to an evolving data stream will lead to sub-optimal clustering results over time.

StreamKM++ [3] clusters the data stream by calculating a representative set called the core-set by constructing a tree that contains representative points. k-means++ is then applied on the core-set points multiple times and the best clustering result is chosen. This method needs to know the number of clusters in advance, which can not always be provided in practice.

Since then several stream clustering approaches have been introduced that can produce reliable and high-quality clusters even when the data stream is evolving over time such as CluStream [4] and DenStream [5]. Like StreamKM++, CluStream is not able to identify the right number of clusters from the data stream automatically and requires human supervision. Both CluStream and DenStream split the stream clustering task into two phases. An online phase, where micro-clusters are created and maintained, which give a memory

efficient representation of the data stream and an offline phase that is triggered whenever a clustering request is issued. The offline step takes the micro-cluster representations as input and applies commonly used clustering approaches such as DBSCAN [7] or k-means++ [8] on top of them. The outcome of this clustering phase are called macro-clusters.
For a more detailed and comprehensive discussion of stream clustering approaches we refer to the survey by Silva *et al.*.

IV. Methodology: Theory

In this Section we describe the ideas behind our approach based on exemplary data sets and provide further detail how the cluster centroids can be estimated from the marginal distributions. We call our approach MinDCluster, because we are using the MargINal Distributions to cluster the data streams. Section IV-C describes the mathematical models used to design the proposed algorithms.

A. The Clustering Approach

In multi-variate data, analysing the different distributions of the individual features can give good estimations to find areas with a high concentrations of points in the data stream. By looking at the Probability Density Function (PDF) of a feature, the data is flattened to this dimension. The local maxima in the PDF show accumulations of values within this feature. We analyse each feature individually and then combine the gained insights. We pick one local maxima from the PDF for each feature at random, which gives us a candidate for a cluster centre. All possible combinations of local maxima form the space of all candidates for cluster centroids. After the candidates have been determined, all the cluster centroid candidates that do not have a minimum amount of data samples in their vicinity are filtered and no longer considered. The selected points using maxima in the PDF provide approximations of the actual centres. Two additional steps have to be carried out until we find the optimal cluster centroids. The first step is to move each centroid candidate closer in the middle of all surrounding data points. After this adjustment step has been carried out, it can be the case that two candidates are now very close to each other and they might be part of the same cluster. We merge them by replacing them with a new cluster candidate in the middle of them. These steps are carried out until convergence. The adjustment and merge process is further described in Section IV-C.
For easier understanding we demonstrate this concept on a simple example with 2-dimensional data. It should be noted that while easier to show and describe in two dimensions, the concept and the resulting approach are still applicable with higher dimensional data. Figure 1 shows the 1000 data samples, the histograms of the data distributions of both features, the estimated PDFs and their respective local maxima. Furthermore, Figure 1 shows how the cross points of the local maxima can be used to get candidates for the cluster centroids. A more detailed description of the selection of the appropriate cross-points is given in Section IV-B.
The red circles show the possible cluster centroids. As a

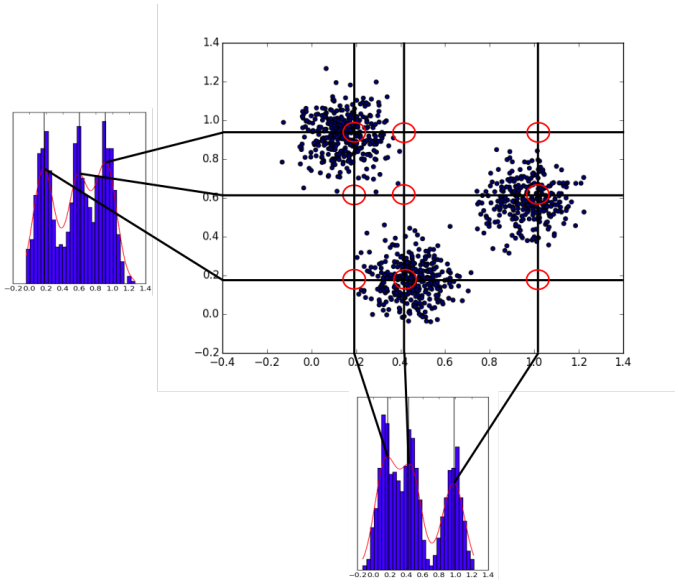

Fig. 1: Cluster centroid candidates are determined through local maxima cross-points

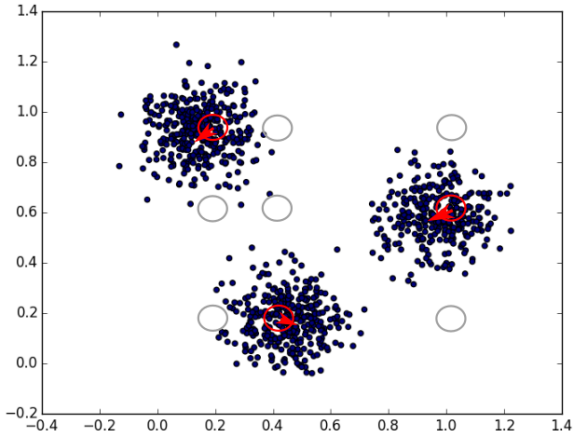

Fig. 2: Filtered and adjusted cluster centroids

first step, all points within a predefined or from the data stream calculated ε-radius are assigned to each candidate. Only centroid candidates that have been assigned more than a threshold of minimum points are considered as possible cluster centroids. All the remaining candidates are adjusted to the mean value point of their cluster. In the case that two cluster centroids are close to each other after this step; the cluster centroids are replaced by the mean value point between them and their corresponding clusters are merged. The filter, adjustment and merge step are repeated until convergence. In our experiments convergence never took more than 10 iterations. To avoid non-convergence a maximum iteration step of 20 is chosen. We discuss both the estimation/selection of ε and the threshold of minimum points in Section V.
All points that are not within an ε-radius of one of the cluster

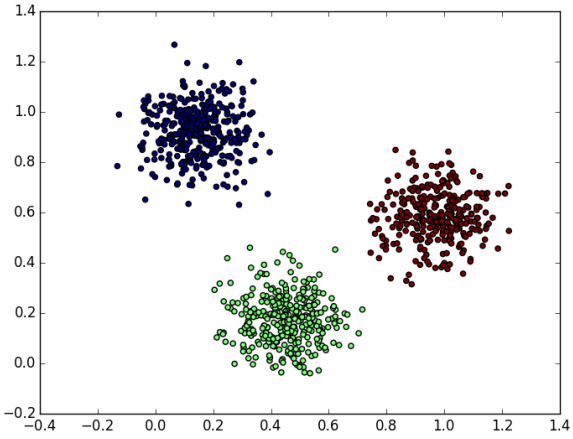

Fig. 3: Result of the clustering approach

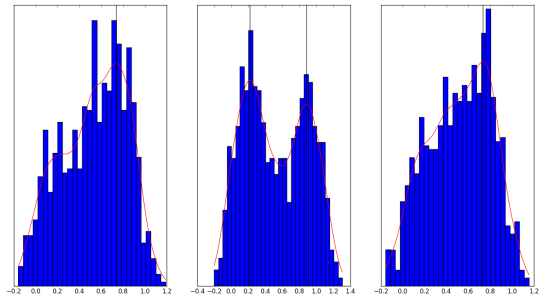

Fig. 4: Problematic distributions: Feature 1 and Feature 3 have only one local maximum.

centroids are assigned to the nearest cluster by minimising the intra-cluster distances. This is the most computationally complex step in the method. The performance of the method can therefore be increased by minimising the points that are left unassigned after the first step in the approach. In Section V we discuss how we can achieve this improved performance. The final result of running the clustering approach is shown in Figure 3.

B. Local Maxima Cross-points

We estimate the PDF for each feature and calculate the local maxima. The maxima give the information about the values that have agglomerations of data in their respective dimension. If we look at the cross-points of the local maxima (see Section IV-A and Figure 1) we get strong candidates for cluster centroids in the multi-dimensional data. In the case that we have data distributions with multiple local maxima as shown in the example given in Section IV-A, we have enough candidates to find all cluster centroids.

However, there is a problematic case, which we encountered during the evaluation of our method, if we have data distributions as shown in Figure 4. Here feature one and three have just one local maximum at 0.78 and 0.77 respectively and feature two has two maxima at 0.22 and 0.89. If we take the cross-points of the distributions, we then end up with just two candidates for the cluster centroids $(0.78, 0.22, 0.77)$ and $(0.78, 0.89, 0.77)$. However, if we look closely at the distributions, we see that even though there is only one maximum, we can take all the points away around this maximum and then there are still areas with a high density of points.

We solved this issue by first taking the crosspoint with the most data samples around itself. We assigned these points to the cluster and then take away all points that were assigned this way. By recomputing the PDFs of the remaining data, we get new local maxima and therefore new local cross-points. By taking the cross-point with the most data samples around it, we can determine the next dense cluster of the given points. This process should be repeated until there are no dense agglomerations of points left.

C. Mathematical Background

In this Section we give a more detailed mathematical description of the models and ideas used in the presented approach.

1) Probability Density Function: In order to analyse the data distributions of the features of the data stream, we need an efficient method to estimate a function that can represent the data distribution. Kernel Density Estimation (KDE) was independently developed by Rosenblatt in 1956 [9] and Parzen [10] in 1962. The kernel used to estimate the function has very little impact on the resulting function compared to the bandwidth parameter. For practical purpose the use of a Gaussian kernel is preferable. There are effective and efficient rules that can be used for automated bandwidth selection (such as Scotts rule [11] and Silvermans rule [12]). Using one of these solutions can effectively turn the approach into a non-parametric method.

We apply the KDE on a window of time series data. Let $d(w) = [X_1, X_2, \ldots, X_n]$ be the time series data of a given window size w. As shown by Parzen [10] we can take the sample distribution given in Equation 1. This allows us to estimate the probability density function $f_n(x)$ based on the data samples of size n with the formula given in Equation 2, where $K(\cdot)$ is called the weighting function satisfying the kernel requirements given in Equation 3 and Equation 4, therefore referred to as the kernel. The kernel is dependent on the chosen bandwidth h. The bandwidth is chosen as $h = h(n)$ such that $\lim_{n\to\infty} h(n) = 0$. This leads to the expected value of $f_n(x)$ approaching $f(x)$ and $f_n(x)$ is a good estimation of the PDF.

$$F_n(x) = (1/n)\{observations \leq x \; among X_1, X_2, \ldots, X_n\} \tag{1}$$

$$f_n(x) = \int_{-\infty}^{\infty} 1/h K(\frac{x-y}{h}) \, \mathrm{d}F_n(y) = \frac{1}{nh} \sum_{j=1}^{n} K(\frac{x - X_i}{h}) \tag{2}$$

$$\int_{-\infty}^{\infty} K(y) \, \mathrm{d}y = 1 \tag{3}$$

$$K(-y) = K(y) \tag{4}$$

2) Local Maxima: In order to identify the possible candidates for cluster centroids we have to find the local maxima. After we have computed the PDF, we can determine the extrema. Let $f_i(x)$ be the PDF of feature i and $f_i'(x)$ be the derivative of this function. The necessary condition for a local extrema is that $f_i'(x) = 0$. If this condition is fulfilled in a point x then we have to examine $f_i'(x \pm \varepsilon)$ for $\varepsilon << 1$. Only if $f_i'(x - \varepsilon) > 0$ and $f_i'(x \pm \varepsilon) < 0$ we have a sufficient condition for a local maxima where $x \pm \varepsilon$ defines the infinitely small area around point x.

3) Adjusting Centroids: Although the cross-point candidates give a strong estimation of where the actual cluster centroids are, they are not necessarily ideally located in their cluster. After we have assigned all points within the data window to one of the centroid candidates, we adjust the centroid candidate in the middle of their respective cluster. Let $c_i = [x_i1, x_i2, \ldots, x_im]$ be the i-th cluster. We then set the centroid candidate C_i to the point that minimises the sum squared Euclidean distances for all cluster points as shown in Equation 5

$$C_i = \frac{x_i1 + x_i2 + \ldots + x_im}{m} \tag{5}$$

4) Merging of Cluster Centroids: After adjusting the cluster centroids, we need to identify if there are separate clusters that should be considered as one cluster because of the vicinity of their centroids. In this case the clusters must be merged. For each cluster centroid, we store the minimum distance of all distances to the other cluster centroids. Let $d_{mins} = [d_1, d_2, \ldots, d_n]$ be the list of minimum distances between the cluster centroids. We want to find small outlier distances of d_{mins}.

We use the definition of the outlier region introduced by Davies and Gather in 1993 [13] to find outlier distances. The outlier region measure is given by Equation 6, where σ is the standard variance, μ the expected value, z_q the q quintile of the Normal distribution with standard deviation one and mean zero.

$$out(\alpha, \mu, \sigma^2) = \{d_i : |d_i - \mu| > z_{1-\alpha/2} * \sigma\} \tag{6}$$

We are only interested in outliers on the low end of the distances, as large distances signify that the cluster centroids are far apart. We use the criterion given in Equation 7 to determine if a distance d_i is a small outlier distance.

$$d_i < -(z_{1-\alpha/2} * \sigma + \mu) \tag{7}$$

V. Description of Algorithms

The estimation of the centroid candidates is performed by algorithm 1. It follows the concepts presented in Section IV-B. The PDFs of each feature of the data sequence d_w are estimated using KDE (see Section IV-C1). Then the maxima are calculated and fed into the function which determines the cross-points. As the size of the data window is d_w is w, the running time of the PDF estimation lies in $\mathcal{O}(w)$; the same

Algorithm 1 DETERMINECENTROIDCANDIDATES(d_w, ε, α)

Require: time series data sequence d_w, radius used to find core points of cluster ε, minimum percentage of the data a cluster has to have core points α
Ensure: List of centroid candidates for further adjusting/merging
1: $min\_num\_points = len(inp) * \alpha$
2: $finished =$ **False**
3: $data = d_w$
4: **while not** $finished$ **do**
5: $PDFs = pdfEstimation(data)$
6: $maxima = []$
7: **for** pdf **in** $PDFS$ **do**
8: $maximum = getmaxima(pdf)$
9: $maxima.append(maxima)$
10: **end for**
11: $C_{candidates} = getCrosspoints(maxima\_values)$
12: $l_{max}, count = mostDenseCluster(C_{candidates}, data, \varepsilon)$
13: {Assign label l_{max} to the points}
14: {in the most dense cluster}
15: **if** $amount < min\_num\_points$ **then**
16: {No more dense clusters left}
17: $finished =$ **True**
18: **else**
19: {Remove already assigned points}
20: $data = removeAssigned(data)$
21: **end if**
22: **end while**

holds true for the calculation of the maxima. The centroid candidates are set to the cross-points of the maxima and are examined. The candidate that has the most points around itself is set as the cluster centroid and all surrounding points are not considered. The process is repeated until no more dense cluster can be found in the data. This results in k iterations, where k is the number of centroid candidates that can be found. Therefore the running time of Algorithm 1 will be $\mathcal{O}(kw)$ in total.

Algorithm 2 adjusts the centroids according to the formula

Algorithm 2 ADJUSTCENTROIDS(C, ld)

Require: List of centroid candidates C, Labelled time series data sequence ld
Ensure: List of adjusted centroid candidates for further merging
1: **for** $label, centroid$ **in** $centroids$ **do**
2: $C_i = \frac{x_i1 + x_i2 + \ldots + x_im}{m}$
3: **end for**

given in Section IV-C3 andruns in $\mathcal{O}(m)$. Because the dimensionality of the multi-variate features that are considered in clustering is usually low in practice, this equates to $\mathcal{O}(1)$.

The identification and merging of near cluster centroids is shown in Algorithm 3. First the distance matrix between the cluster centroids is computed. Because the distance matrix is a square symmetric matrix, we can compute the lower triangular and use this part of the matrix for the further computations. That way we keep all the necessary information we need, while improving the performance. The nested for-loop runs in $\mathcal{O}(k^2)$ where $k = length(C)$ is the number of centroid candidates.

Algorithm 4 combines the algorithms 1, 2 and 3 to determine the cluster centroids. Iteratively, the candidates are calculated and adjusted and near cluster centroids are merged until con-

Algorithm 3 MERGECENTROIDS(C, z_q)

Require: List of centroid candidates C, q quintile z_q
Ensure: List of centroid candidates, where near centroids are merged

D = {Distance matrix between centroids}
d_{mins} = {Minimum distances between centroids}
$\sigma = std(d_{mins})$
$\mu = mean(d_{mins}$
for $i, distances$ **in** D **do**
  for $j, distance$ **in** $distances$ **do**
    if $j > i$ **and** $distance < -z_q * \sigma + \mu$ **then**
      $merge(C_i, C_j)$
    end if
  end for
end for

Algorithm 4 DETERMINECENTROIDS(d_w, ε, α)

Require: time series data sequence d_w, radius used to find core points of cluster ε, minimum percentage of the data a cluster has to have core points α
Ensure: List of centroids C

$C_{candidates} = determineCentroidCandidates(d, \varepsilon, \alpha)$
$ld = mdCluster(C_{candidates}, d_w)$ {Labelled data}
$centroids = []$
while True do
  $C = adjustCentroids(C_{candidates}, ld, epsilon)$
  $C = mergeCentroids(C, ld, epsilon)$
  if $C_{candidates} == centroids$ **then**
    break {We converged}
  else
    $C_{candidates} = C$
    $ld = mdCluster(C_{candidates}, d_w)$
  end if
end while

vergence.We suggest to limit the iteration steps to a maximum in order to avoid non-convergence in corner cases where the centroids might be adjusted back and forth. In our practical experiments, convergence happened after less than 10 iteration steps and we limited the maximum iterations to 20.

The final clustering process, shown in Algorithm 5, is started after the cluster centroids have been determined using Algorithm 4. The first step of the algorithm assigns all the points that are within the ε-radius of a centroid to the respective cluster. After this step, there are still data samples that have not yet been assigned to a cluster label. We calculate the best fit for these points by calculating the average distance to the points of each cluster and assign them the cluster label with the smallest average distance.

Because the second step is computationally complex, we have to estimate the parameter ε in such a way that only the minority of the points are left unassigned after step one. To do this we calculate ε based on the minimum and maximum values in the data stream and the density of the data samples. To further improve the running time of this algorithm without losing cluster quality, in step two we can use a representative sample of the already clustered points to calculate the average distances. The run time can be further optimised by assigning the unlabelled points to the cluster centroid with the lowest Euclidean distance, however, doing so will decrease the cluster quality.

Algorithm 5 MINDCLUSTER(d_w, C, ε)

Require: time series data sequence d, List of centroids C, radius used to find core points of cluster ε
Ensure: Labelled data ld

{Initially assign all points inside the ε-radius to the centroids}
for p **in** d **do**
  for l, c **in** C **do**
    if $distance(p, c) < \varepsilon$ **then**
      {Assign label l to point p }
      break
    end if
  end for
end for
ud= {Data that has yet to be assigned to a cluster}
ld = {Data that has been assigned to a cluster, ld_i is the data assigned to cluster i}
for p **in** ud **do**
  for ld_i **in** ld **do**
    $avg\_dist_i = averageDistance(p, ld_i)$
  end for
  {Assign label i to p with minimum avg_dist_i}
end for

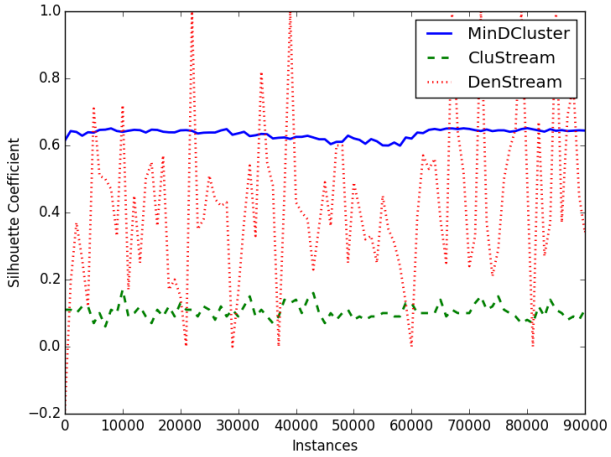

Fig. 5: Comparison to CluStream and DenStream in 3 Dimensions

VI. EVALUATION

The synthesised data sets used and the codes used to produce the evolving data streams are available online on our website[1]. The evaluation metric used is the silhouette Coefficient introduced by Pravilovic *et al.* [14]. The silhouette score is a value in the range $[-1, 1]$, where values close to -1 signify misclustering and values close to 1 signify a perfect clustering result. In the MOA framework [15], the value is normalised to be in the range $[0, 1]$, which is why we use the normalised silhouette score in the evaluations shown in Figures 5 and 6.

[1]http://iot.ee.surrey.ac.uk/#software

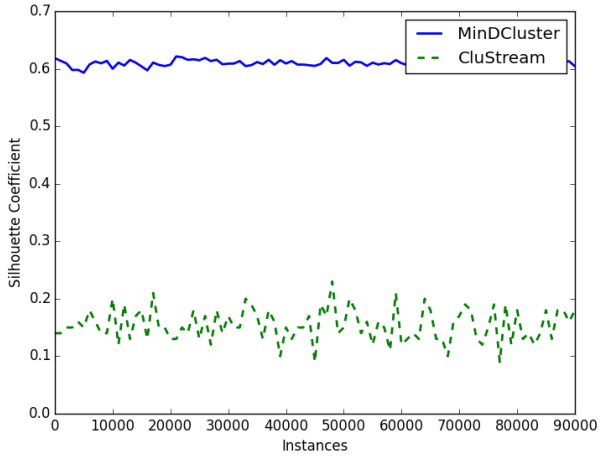

Fig. 6: Comparison to CluStream in 4 Dimensions

Figure 5 compares the cluster quality of this approach to DenStream and CluStream on a 3-dimensional data set. It can be seen that our approach consistently outperforms CluStream. Even though DenStream shows a higher cluster quality at a few points in time while processing the data stream, the overall cluster quality of DenStream is unstable and produces very low quality cluster at times. Our approach performs consistently well. Compared to the average cluster quality found in the CluStream approach, our approach has a 511.66% increase in cluster quality. Compared to DenStreams average cluster quality, our approach shows an 49.33% increase. Figure 6 shows the evaluation results on a 4-dimensional data sets. The implementation of DenStream in MOA was not able to produce a clustering result on this data set. Our approach consistently outperforms CluStream, with an average increase in cluster quality of 298.97%

VII. Conclusion

Clustering time-series data requires specific attention to the characteristics of the data and the aim of the process and its applications. Conventional data sets tend to lean toward Gaussian distributions over long-term [16]. The distribution of data over various features are also usually fixed or have limited variations. However, time-series data from IoT streams usually create dynamic and multi-variate data that require special adaptive clustering mechanisms. In this paper, we have presented a clustering algorithm designed for multi-variate data. We have evaluated our solution using experimental data sets and compared our results with some of the existing stream clustering methods. The evaluation results shows that our solution is capable of detecting the number of clusters automatically by analysing the feature distributions of the data stream. The clustering quality of the resulting clusters of our approach shows improvements from 49% up to 500% compared to competitor methods on the experimental data sets. The algorithms designed in this work will have a significant impact in dynamic and automated processing and clustering real-world IoT data sets.

Acknowledgment

The research leading to these results has received funding from the European Commissions Seventh Framework Programme for the CityPulse project under grant agreement no. 609035.

References

[1] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 359–366, 2000. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=892124

[2] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Databases Method for Very Large," *ACM SIGMOD Record*, vol. 25, no. 2, pp. 103–114, 1996.

[3] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "StreamKM++: A Clustering Algorithm for Data Streams," *Journal of Experimental Algorithmics*, vol. 17, no. 1, pp. 173–187, jul 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2133803.2184450

[4] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A Framework for Clustering Evolving Data Streams," in *Proceedings of the 29th international conference on Very large data bases*, 2003, pp. 81–92. [Online]. Available: http://dl.acm.org/citation.cfm?id=1315460http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8650

[5] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Conference on Data Mining*, no. 2, 2006, pp. 328–339. [Online]. Available: http://books.google.com/books?hl=en{&}lr={&}id=2sIT1rTMDFgC{&}oi=fnd{&}pg=PA328{&}dq=Density-Based+Clustering+over+an+Evolving+Data+Stream+with+Noise{&}ots=X8DxLBObX1{&}sig=JcBfOFMdXr9ffiaWuMdmYxdN9uw

[6] F. Ganz, P. Barnaghi, and F. Carrez, "Real World Internet Data," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3793–3805, 2013.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Conference on Knowledge Discovery & Data Mining*, vol. 2, 1996, pp. 226–231.

[8] D. Arthur, "k-means ++ : The Advantages of Careful Seeding," in *Proceedings of the 18th annual ACM-SIAM symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics*, 2007, pp. 1027–1035.

[9] M. Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 1956.

[10] E. Parzen, "On Estimation of a Probability Density Function and Mode Author (s): Emanuel Parzen Source : The Annals of Mathematical Statistics , Vol . 33 , No . 3 (Sep ., 1962), pp . 1065-1076 Published by : Institute of Mathematical Statistics Stable URL : http:," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[11] D. W. Scott, *Mutli-variate Density Estimation: Theory, Practice and Visualization*. New York: John Wiley & Sons, 1992.

[12] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall, 1986.

[13] L. Davies and U. Gather, "The Identification of Multiple Outliers," *Journal of the American Statistical Association*, vol. 88, no. 423, pp. 782–792, 1993. [Online]. Available: http://www.jstor.org/stable/2290763$\delimiter"026E30F$nhttp://www.jstor.org/stable/2290763?origin=crossref

[14] S. Pravilovic, A. Appice, and D. Malerba, "Integrating Cluster Analysis to the ARIMA Model for Forecasting Geosensor Data," *Foundations of Intelligent Systems*, pp. 234–243, 2014.

[15] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010. [Online]. Available: http://eprints.pascal-network.org/archive/00007194/

[16] E. Keogh and S. Kasetty, "On the Need for Time Series Data Mining Benchmarks : A Survey and Empirical," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, 2003.