CluStream-GT: Online Clustering for Personalization in the Health Domain

Eoin Martino Grua
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
e.m.grua@vu.nl

Mark Hoogendoorn
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
m.hoogendoorn@vu.nl

Ivano Malavolta
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
i.malavolta@vu.nl

Patricia Lago
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
p.lago@vu.nl

A.E. Eiben
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
a.e.eiben@vu.nl

ABSTRACT

Clustering of users underlies many of the personalisation algorithms that are in use nowadays. Such clustering is mostly performed in an offline fashion. For a health and wellbeing setting, offline clustering might however not be suitable, as limited data is often available and patient states can also quickly evolve over time. Existing online clustering algorithms are not suitable for the health domain due to the type of data that involves multiple time series evolving over time. In this paper we propose a new online clustering algorithm called CluStream-GT that is suitable for health applications. By using both artificial and real datasets, we show that the approach is far more efficient compared to regular clustering, with an average speedup of 93%, while only losing 12% in the accuracy of the clustering with artificial data and 3% with real data.

CCS CONCEPTS

• **Computing methodologies** → *Cluster analysis.*

KEYWORDS

online clustering, clustering, time series, e-Health

ACM Reference Format:
Eoin Martino Grua, Mark Hoogendoorn, Ivano Malavolta, Patricia Lago, and A.E. Eiben. 2019. CluStream-GT: Online Clustering for Personalization in the Health Domain. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI '19), October 14–17, 2019, Thessaloniki, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3350546.3352529

1 INTRODUCTION

Personalisation in the health domain can contribute greatly to an improved wellbeing among patients [7, 12, 16]. For example, personalisation entails selecting a dedicated intervention for a patient that is most likely to improve the patients health state. Applications vary from more medical cases in hospitals [2] to mobile health apps such as sport trackers [10] or apps to battle depression [11] . Performing personalisation in the health domain is challenging: bad suggestions are highly undesirable, data can be limited for specific types of health-related domains, and doing real life experiments to collect more data is not trivial at all.

Clustering is often considered a valuable step in providing personalised support or recommendations to users (see e.g. [6, 8]) in order to have enough data to base recommendations on. Using clustering, like-minded users are grouped, and recommendations (or the aforementioned interventions) are found that are relevant for the users in the group. Mostly, this clustering is done in an offline fashion, i.e., once the clusters are determined they do not change in real-time and are only updated in a batch mode over possibly long intervals. While this is fine for companies such as Netflix and Amazon with a user base of millions and without rapid changes of user preferences, for the health domain this might come with severe disadvantages: (1) the number of users could be very limited and one would want to exploit the most recent data of all patients, and (2) the health state of users can vary greatly and change rapidly over time. Hence, clustering in a real-time fashion is much more desirable.

In the literature, there are various algorithms that allow for the online updating of clusters. Well-known examples include CluStream [1], ODAC [14], and others [4, 5]. They do however not fit the health domain well.

In the health domain, we mostly consider measurements over time per patient, and intend to cluster on a patient level. This means that online clustering approaches need to cope with: (1) new patients arriving, and (2) new data of known patients coming in. Both situations potentially require an update of clusters, while in existing approaches only the second case is tackled, assuming the number of data points (in our case patients) do not change as time progresses. In this paper, we present an online clustering approach for the health care domain that is called CluStream-GT (standing for: CluStream for Growing Time-series). It is able to online cluster patients with evolving time series (i.e. an increasing amount of data per patient over time). This approach is an extension of the popular CluStream approach. CluStream-GT online clusters inputted time-series by first checking if the data is meant as an update of an already clustered patient or is categorising a new one. In the former case it then decides if the newly updated time-series has to be re-clustered or can be kept in the already assigned cluster. Whilst in the latter

WI '19, October 14–17, 2019, Thessaloniki, Greece
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6934-3/19/10…$15.00
https://doi.org/10.1145/3350546.3352529

case it will always decide which cluster is better suited to include the new patient data. In order to evaluate the CluStream-GT, we use both real medical EEG data and artificial data. We compare the quality of the clusters found by CluStream-GT to k-means (cf. [9]) and ODAC (as it is the closest existing method to CluStream-GT) by having both algorithms re-cluster at each timepoint and storing the average silhouette score [15]. We also compare the total execution time of the approaches. We therefore analysed the two metrics and found a beneficial trade-off in the use of CluStream-GT.

2 APPROACH

In this section, we formally define the problem we are addressing, followed by an explanation of the proposed algorithm.

2.1 Problem Description

We assume that we have a set of users U: $u_1, \dots, u_k$ (patients in our case) which can generate health related data. The health data contains a number of features that are measured around the users at each time point: $\{f_1, \dots, f_n\}$. The domain of each feature f_i is denoted by F_i. The values of these features are measured over time, we assume that at each time point when a measurement is performed, all feature values are measured. For a measurement at time point t for user u_i, the value of the feature f_j is noted as $v(u_i, f_j, t)$. We use $S_{t_0}^{t_1}(u_i)$ to denote the time series of a user u_i, containing vectors of values of all features over all time point between t_0 and t_1. Furthermore, $t_{start}(u_i)$ is used to specify the time when the time series started for user u_i and $t_{end}(u_i)$ when it ended.

Our task is to cluster the series of values over all features for the users. Here, the start, end, and length of the series of the users can vary freely across the users. In order to specify our algorithm, we assume that some aggregation function a is available that summarises the the entire time series into a single number that can be compared to other time series (i.e. this is our distance metric). We require the following property of the function (to simplify, we assume one feature f_j here):

$$a(S_{t_0}^{t_k}(u_i)) = a(a(S_{t_0}^{t_k-1}(u_i)), S_{t_k}^{t_k}(u_i))) \tag{1}$$

This property allows updating the aggregate of the time series without having to maintain a history of all values.

2.2 CluStream-GT algorithm

To solve the clustering problem, we deploy an approach similar to CluStream. CluStream works based on microclusters. These are used as intermediate step before the clustering of the entire dataset is used. They are initialised offline with a small sub-sample of the dataset. Microclusters are specified by means of five components that summarise the data within the microcluster:

- the sum of all values of the datapoints
- the sum of all squared values of the datapoints
- the sum of all time points associated with the datapoints
- the sum of the squared values of all time points associated with the datapoints
- the number of datapoints contained in the microcluster

Given our formal notation before, we slightly adjust the microcluster definition to make it suitable for our setting:

- the sum of all values of the aggregation function of the users in the microcluster: $\sum_{\forall u \in U_i} a(S_{t_{start}(u)}^{t_{end}(u)}(u))$
- the sum of all squared values of the aggregation function: $\sum_{\forall u \in U_i} \left(a(S_{t_{start}(u)}^{t_{end}(u)}(u))\right)^2$
- the sum of the last time points for all users: $\sum_{\forall u \in U_i} t_{end}(u)$
- the sum of the squared values of the last time points for all users: $\sum_{\forall u \in U_i} (t_{end}(u))^2$
- the set of users contained in the microcluster ($U_i \subset U$)

As a second step, these microclusters are used as datapoints in a standard clustering approach resulting in macroclusters. Having the microclusters as an intermediate step saves valuable storage space, but also computational effort in the clustering. Of course, it is essential to have appropriate microclusters that group users in a suitable way. CluStream therefore, when new data arrives, assigns the new data point to an existing microcluster if it is sufficiently alike, or its own microcluster in case it is too different. In the latter case, two existing microclusters are merged. Microclusters containing too many old datapoints can also be removed. In our setting, life becomes slightly more complicated as data points are now time series. Hence, we can have two cases: (1) a new datapoint arrives in a time series of an existing user/patient, or (2) a datapoint arrives of a new patient. We need to accommodate for both cases. Algorithm 1 shows our adjusted version of CluStream to accommodate for this setting.

Algorithm 1 CluStream-GT pseudocode

```
1:  procedure CLUSTREAM-GT(new_data, id)
2:      $t \leftarrow current\_time$
        // We have seen the patient already
3:      if id is known then
4:          $m\_c \leftarrow get\_micro\_cluster(id)$
5:          $prev\_t \leftarrow get\_last\_t(id)$
6:          $prev\_aggr \leftarrow get\_prev\_aggregate\_value(id)$
7:          $new\_aggr \leftarrow calculate\_aggregate(get\_prev\_data(id),$
8:                          $new\_data)$
        // We see only a small change, just update the microcluster properties
9:          if $|new\_aggr - pre\_aggr| < \delta$ then
10:             $m\_c \leftarrow update\_microcluster(m\_c, id, t, prev\_t, new\_aggr,$
11:                         $prev\_aggr)$
        // Major change, remove from microcluster, find the best fitting cluster and update
12:         else
13:             $m\_c \leftarrow remove\_from\_microcluster(m\_c, id, prev\_t, prev\_aggr)$
14:             goto find
        // New patient
15:     else
16:         $new\_aggr \leftarrow calculate\_aggregate(null, new\_data)$
17: find:
18:         $m\_c \leftarrow find\_best\_microcluster(id, t, new\_aggr)$ // New patient fits in an
    existing microcluster
19:         if $distance(new\_aggr, t, m\_c) < max\_boundary$ then
20:             $m\_c \leftarrow update\_microcluster(m\_c, id, t, null, new\_aggr, null)$
        // Microclusters need to merge, patient in a new microcluster
21:         else
22:             $[m\_c\_1, m\_c\_2] \leftarrow min_{m\_c\_1, m\_c\_2 \in micro\_clusters} distance(m\_c\_1,$
23:                         $m\_c\_2)$
24:             $m\_c\_1 \leftarrow merge\_microclusters(m\_c\_1, m\_c\_2)$
25:             $m\_c\_2 \leftarrow null$
26:             $m\_c\_2 \leftarrow update\_microcluster(m\_c\_2, id, t, null, new\_aggr, null)$
```

In our extension, we consider the update when a new patient arrives the same as with CluStream (of course, using our aggregation function again). In case a new datapoint for an existing patient arrives we only update the properties of the microcluster that patient had already been assigned to in case it results in a minor adjustment of the aggregate value. Otherwise, we go to a full blown re-clustering. This process can be seen in line 8 of Algorithm 1 where the adjustment is judged against a threshold value δ.

In the pseudocode, the additions that were made to CluStream are seen from line 1 to 12 in Algorithm 1. We also assume that within the health domain we want to retain the information of all the clustered time-series, no matter how old they are. If they would ever have to be deleted, the user of the algorithm can do so manually, but we do not want to give permission to the algorithm to automatically remove information. Therefore we have removed the possibility of the algorithm deleting micro-clusters.

For the offline cluster creation (or macro-clustering process) CluStream-GT works similarly to CluStream. It uses k-means but uses the micro-clusters as the input data. Furthermore, the centroids are chosen as the k most populated micro-clusters. This can be easily achieved by analysing the last element of each micro-cluster tuple and choosing the k highest ones. This macro-clustering process has the clear advantage of not requiring storage of the whole time-series dataset as we use only use the micro-clusters, making our approach better suited for cases with limited resources.

3 EXPERIMENTAL SETUP

In this section we explain the experimental conditions and evaluations we have used to test CluStream-GT's performance. We compare CluStream-GT against two alternatives: (1) k-means clustering in each iteration, and (2) ODAC (cf. [14]).

Scenarios. To assess the difference in performance we performed tests for three scenarios: two were done with generated synthetic data and one with the use of a real world dataset. For the generated data conditions we utilised sine functions to generate our time-series. In the first test (henceforth referred to as: the base case) we had each time-series associated to a specific set of parameters inputted in the sine function. This includes the function itself and noise surrounding the curve. In the second test (henceforth referred to as: the advanced case) each time-series started with an associated set of parameters, analogous to the base case. However, at each timestep the considered time-series had a 10% chance of changing its parameters and therefore have its data generated by a different sine function. This extra factor was introduced as a method of representing the potential change in behaviour that can be observed in time series associated with human behaviour, especially within the health domain (e.g. vital signs getting better, mood improving, more frequent physical activity, etc.). To add a more practical test scenario we have also used of a real dataset. This dataset is a collection of EEG recordings that were published by Andrzejak et al. in 2001 [3].

Performance metrics. The two aspects investigated across all of our tests were accuracy of the clustering and speed of execution. To asses clustering accuracy, we decided to utilise the silhouette score [15]. For the execution time, we kept track of the total length of the execution of the algorithms thereby performing the experiments on the same machine with no other processes open in order to minimise potential variance.

Algorithm setup. All of the tests are set up to represent a realistic scenario for all techniques (CluStream-GT, k-means and ODAC). We therefore update at every single timestep as in the health care domain data can be scarce and therefore all available data should be exploited as much as possible to create the most up-to-date clusters. This means that we re-cluster every time any form of new data is given to the algorithm. That includes both a new time-series and any amount of new data related to an already clustered one.

The k-means used as benchmark clustered using the means of each time-series present in the dataset. The mean was used since it was also utilised as our selected aggregate function (described in Section 2) for distance computation during the online phase of CluStream-GT. Both ClusStream-GT and k-means clustering require the number of clusters to be set. To make the results comparable we fixed this value to $k = 3$ for both cases as initial experiments have shown this to be the best value for all scenarios. Our replication package contains the full experimental setup implemented in Python as used to perform these experiments[1].

Experimental Conditions. For both generated cases, we had a starting population of 120 time-series uniformly distributed over three clusters. The experiments were performed over the course of 30, 60 and 90 simulated days, with each case repeated 30 times. At each day there was a 50% chance of adding new time-series to the dataset (simulating the addition of a new patient). The amount added ranged from one to five chosen with the use of a uniform distribution. It is important to note that this feature of our experiment was not used whilst testing ODAC. This is because ODAC cannot work on datasets with changing numbers of timeseries. Per day, each generated time-series consisted of 24 points. This was selected to simulate pooling done once at each hour of the day.

For the real dataset, we ran tests on three cases: 100, 200 and 300 total patients. Each patient had a time-series containing 4097 individual datapoints. Each one of these scenarios was repeated five times. ODAC was run only a single time as it lacks stochasticity and therefore would always return the same results.

4 RESULTS

Section 4.1 illustrates the results we gathered from the two generated data test cases, whilst Section 4.2 does so with the EEG dataset results. As we could never compute the silhouette score for ODAC we illustrate those results separately (see Section 4.3).

4.1 Results from the generated data

We will first discuss the results for the base case. Figure 1 illustrates the distribution of the average silhouette scores obtained for each of the test scenarios. The CluStream-GT mean averages at 0.86 and the median to 0.9, whilst the k-means mean and median average at 0.93. Secondly, it is interesting to note the skewed distribution occurring for all cases of CluStream-GT. Furthermore, we can observe some runs that result in differences deviant enough from the mean to be classified as outliers. The potential cause of this behaviour could be attributed to poor initialisation of the micro-clusters within those runs, which then followed with worse overall clustering and therefore a lower silhouette score. Nevertheless, the distributions

[1]https://github.com/EMGrua/CluStream-GT

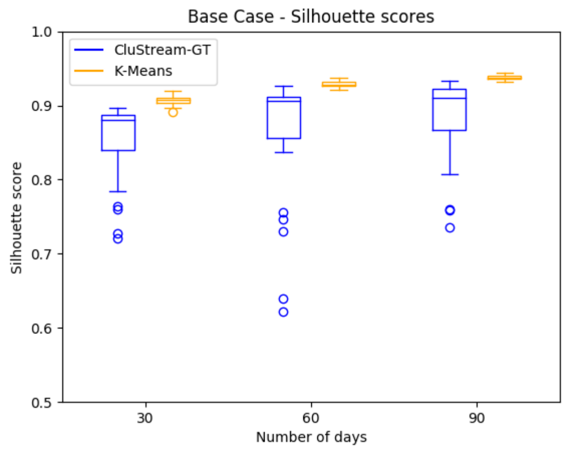

Figure 1: Decrease of the the average silhouette score using CluStream-GT compared to k-means (Base Case)

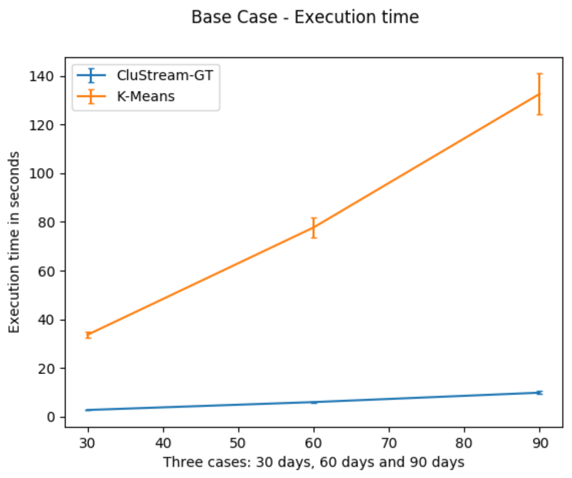

Figure 2: Decrease of the the average execution time using CluStream-GT compared to k-means (Base Case)

favour a smaller difference with the IQR ranging from the smallest Q1 equal to 0.84 to the highest Q3 equal to 0.92. Differently is the execution time trends of the two techniques (shown in Figure 2). We clearly observe that with the growing amount of data the saved execution time also grows. This can be certainly attributed to the use of the micro-cluster tuples for the generation of the macro-clusters, which caps the amount of data used to cluster. Therefore, the time increase is only due to the higher number of runs of the online component needed by CluStream-GT to update the micro-clusters. No matter the amount of data, CluStream-GT provides at least a 90% speedup.

Examining now the silhouette score for the advanced case, we observe a bigger difference between CluStream-GT and k-means, and a less skewed distribution for all the scenarios of CluStream-GT (as shown in Figure 3) as compared to the base case. In this case we observe a number of outliers, although smaller than with the base case. The overall higher difference between the two approaches is to be expected as CluStream-GT is trying to cluster a now far more complex time-series with only the use of the meta-data contained in the micro-cluster tuples. This provides somewhat of an advantage to our benchmark k-means which has access to the mean of each time-series present in the generated dataset.

Finally, we examine the execution times recorded for each scenario of the advanced case (shown in Figure 4). Similar to the execution times of the base case, CluStream-GT minimally grows as the data does, whilst the execution time of k-means continues to grow. This indicates that the more data is clustered and the higher is the speed gain achieved by using CluStream-GT. In fact in the 90 days test case we achieve an average 94.7% speed-up.

4.2 Results from the real dataset

We start by analysing the results collected from the silhouette scores (shown in Figure 5). The mean values recorded from both algorithms are extremely similar, with only a small loss in the silhouette score

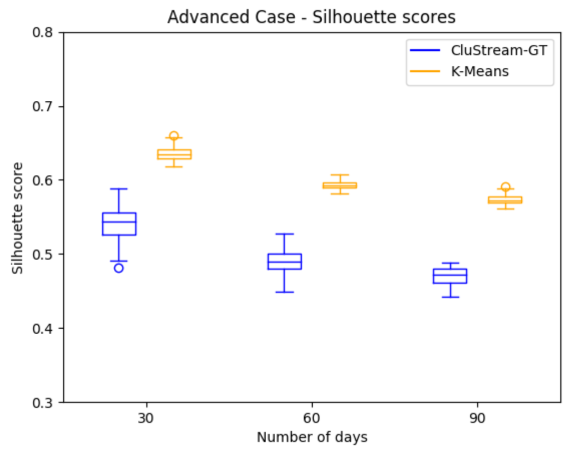

Figure 3: Decrease of the the average silhouette score using CluStream-GT compared to k-means (Advanced Case)

by CluStream-GT compared to k-means. Furthermore, the standard deviation for each case was minimal. This suggests reliable clustering over repetitions and therefore reliable clustering overall. This is somewhat in contrast with the generated data, where both CluStream-GT and k-means showed wider standard deviations, reinforcing our assumption that the deviation in the generated data is due to the noisier nature of said data. Moving to execution times we observe the huge advantage that using CluStream-GT gives over k-means. In Figure 6 we see that whilst k-means drastically increases its execution time with the increase of data, CluStream-GT barely increases. This leads to a difference in execution time that becomes more substantial the bigger the dataset is. Taking the case of 300 patients the average execution time for k-means is of 20000 seconds (5 hours and 33 minutes) whilst CluStream-GT's average execution time is only 1036 seconds (17 minutes and 20 seconds). This effectively is a 95% improvement.

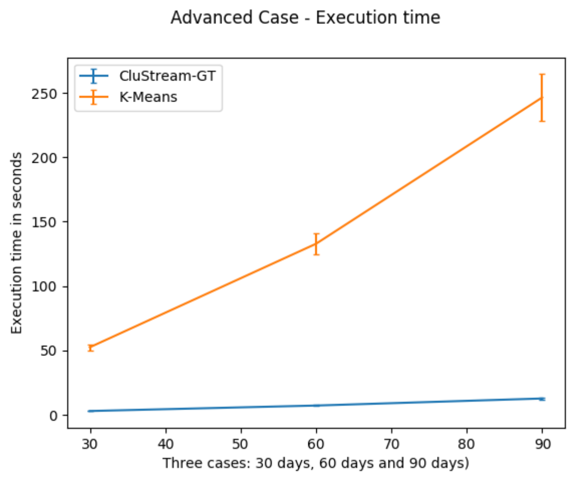

Figure 4: Decrease of the average execution time using CluStream-GT compared to k-means (Advanced Case)

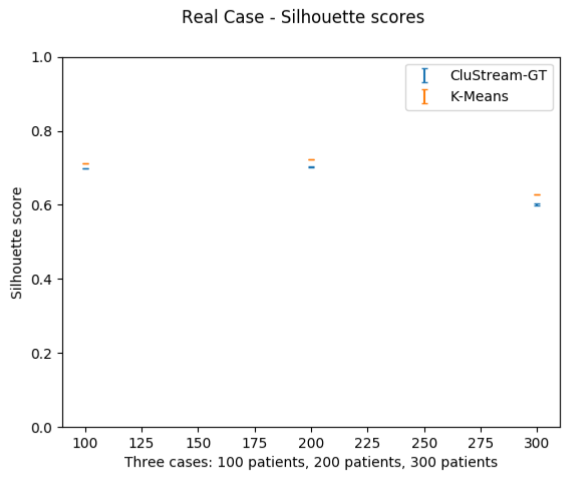

Figure 5: Decrease of the average silhouette score using CluStream-GT compared to k-means (Real Case)

4.3 Results obtained by the use of ODAC

Over all tests, ODAC consistently maintained only one node of its tree structure, hence clustering all data under one cluster. As a result, it was impossible for us to measure the silhouette score. In order to make such measure, ODAC would have had to result in at least two separate clusters. The reason for this behaviour seems to stem from the algorithm stalling on the first node, splitting and aggregating consecutively. This type of behaviour has also been reported by the authors of ODAC as well [13]. A cause could be that updates are performed at each time step, while in experiments using ODAC often batches of time points are used. ODAC was also consistent in the registered execution times. For all cases tested, CluStream-GT was, on average, 98% faster than ODAC (as reported in Table 1). This is an expected consequence, given that ODAC increases in speed with an increasing number of leaves, otherwise needing to

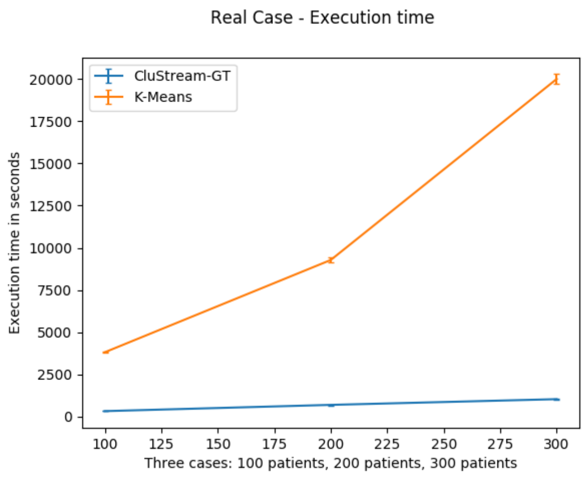

Figure 6: Decrease of the average execution time using CluStream-GT compared to k-means (Real Case)

	Clustream-GT	ODAC
Base Case 30 Days	2.4	197.8
Base Case 60 Days	4.6	398.2
Base Case 90 Days	6.9	605.8
Advanced Case 30 Days	2.7	197.8
Advanced Case 60 Days	5.7	402.7
Advanced Case 90 Days	8.8	610.1
Real Dataset 100 patients	327.4	19368
Real Dataset 200 patients	697.9	79965.7

Table 1: Execution times (in seconds) for Clustream-GT and ODAC on all executed tests

recompute all dissimilarities each time new data is clustered (a calculation that has a quadratic complexity on the number of data streams) [14].

5 DISCUSSION AND FUTURE WORK

The main contribution of our study is the development of an online clustering algorithm that can cluster growing timeseries.

We have developed this algorithm by modifying the already existing data stream clustering algorithm CluStream and so named ours CluStream-GT. We formalised CluStream-GT's function in Section 2 where we present pseudocode and explain the input and global variables used by the algorithm to perform the micro-cluster updates. We then evaluated our approach by the use of three test scenarios: two of them were executed using generated data, whilst the third one was performed using a real EEG dataset [3].

As described in Section 3, for all test cases we recorded the total execution time and the average silhouette score obtained by re-clustering at each timestep. We compared Clustream-GT against k-means and ODAC for three scenarios. ODAC clustered all data under one cluster for all experimental conditions, it was impossible to compute, and therefore compare, the silhouette score with that of CluStream-GT. CluStream-GT was 98% faster than ODAC on all cases. We explain this as ODAC, remaining on a structure of one node, had to execute under its worst case scenario. Therefore, having to recompute all dissimilarities at each new time step (an operation with quadratic complexity).

When comparing k-means with CluSteam-GT for the base case, CluStream-GT provides a good trade-off between accuracy and execution time by speeding up the performance by 92% whilst only loosing an average of 0.06 on the silhouette score. For the advanced case, the trade-off is similar as we lose an average of 0.1 on the silhouette score but still achieve significant speedup with CluStream-GT performing 94.5% faster. The bigger divide in silhouette score, as compared to the base case, can be explained by the increase in noise that the advanced case brings to the data due to the chance of timeseries suddenly switching behaviour and therefore making the clustering a more challenging task. This is especially apparent for CluStream-GT since it only uses the descriptive data contained in the micro-clusters for the formation of the final macro-clusters.

Lastly, in the test case performed with the EEG dataset CluStream-GT performed excellently. The speed-up was of at least 91% with it improving to 95% with the 300 patients run. This meant that on the machine used for testing k-means it took a total of 5 and a half hours whilst CluStream-GT only took a little more than 17 minutes. This was achieved with an extremely small trade-off on the silhouette score, with the worst case being the 300 patients run in which CluStream-GT had on average 0.028 less on the silhouette score.

For future work we would like to augment CluStream-GT with a mechanism to detect poor micro-cluster initialisation at an early stage. Whilst it was not a problem for the less noisy EEG data, we did record a few outlier cases in some of the runs in the generated data. We therefore aim to create such a mechanism in order to reduce or remove the possibility of such outliers appearing and therefore increasing the average silhouette score obtained.

Furthermore, as mentioned in Section 3 we measure execution time by measuring the time difference with the python module Time. Whilst we minimised the risk of variance with repeated runs and assuring that the machine had no other processes open apart from our experiment, it would be desirable to repeat the experiments on other machines in order to further validate our findings.

Lastly, we have mentioned throughout our work that the execution time gap increases with the size of the data and have explained this phenomenon by CluStream-GT's use of the micro-clusters and lack of needing to store the entire dataset. However, we have not investigated how much more efficient CluStream-GT can be on storage space. This would be an interesting fact to investigate especially for CluStream-GT's therefore potential use on lower spec hardware, such as mobile devices.

REFERENCES

[1] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. 2003. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 81–92.

[2] Mamoun Almardini, Ayman Hajja, Zbigniew W Raś, Lina Clover, David Olaleye, Youngjin Park, Jay Paulson, and Yang Xiao. 2015. Reduction of readmissions to hospitals based on actionable knowledge discovery and personalization. In *Beyond Databases, Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery*. Springer, 39–55.

[3] Ralph G Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E Elger. 2001. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E* 64, 6 (2001), 061907.

[4] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. 2006. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*. SIAM, 328–339.

[5] Yixin Chen and Li Tu. 2007. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 133–142.

[6] Fabio Cortellese, Marco Nalin, Angelica Morandi, Alberto Sanna, and Floriana Grasso. 2009. Personality diagnosis for personalized ehealth services. In *International Conference on Electronic Healthcare*. Springer, 157–164.

[7] Irit Hochberg, Guy Feraru, Mark Kozdoba, Shie Mannor, Moshe Tennenholtz, and Elad Yom-Tov. 2016. Encouraging physical activity in patients with diabetes through automatic personalized feedback via reinforcement learning improves glycemic control. *Diabetes Care* 39, 4 (2016), e59–e60.

[8] Kyoung-jae Kim and Hyunchul Ahn. 2004. Using a clustering genetic algorithm to support customer segmentation for personalized recommender systems. In *International Conference on AI, Simulation, and Planning in High Autonomy Systems*. Springer, 409–415.

[9] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297.

[10] Anouk Middelweerd, Julia S Mollee, C Natalie van der Wal, Johannes Brug, and Saskia J Te Velde. 2014. Apps to promote physical activity among adults: a review and content analysis. *International journal of behavioral nutrition and physical activity* 11, 1 (2014), 97.

[11] Adam C Powell, John Torous, Steven Chan, Geoffrey Stephen Raynor, Erik Shwarts, Meghan Shanahan, and Adam B Landman. 2016. Interrater reliability of mHealth app rating measures: analysis of top depression and smoking cessation apps. *JMIR mHealth and uHealth* 4, 1 (2016).

[12] David Riaño, Francis Real, Joan Albert López-Vallverdú, Fabio Campana, Sara Ercolani, Patrizia Mecocci, Roberta Annicchiarico, and Carlo Caltagirone. 2012. An ontology-based personalization of health-care knowledge to support clinical decisions for chronically ill patients. *Journal of biomedical informatics* 45, 3 (2012), 429–446.

[13] Pedro Pereira Rodrigues, Joao Gama, and Joao Pedroso. [n. d.]. Hierarchical TimeÂŋSeries Clustering for Data Streams. ([n. d.]). http://alumni.cs.ucr.edu/~piyush/cs235_ppt.pdf [Online;].

[14] Pedro Pereira Rodrigues, Joao Gama, and Joao Pedroso. 2008. Hierarchical clustering of time-series data streams. *IEEE transactions on knowledge and data engineering* 20, 5 (2008), 615–627.

[15] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[16] Monica S Webb, Vani Nath Simmons, and Thomas H Brandon. 2005. Tailored interventions for motivating smoking cessation: using placebo tailoring to examine the influence of expectancies and personalization. *Health Psychology* 24, 2 (2005), 179.