Accelerating Cluster Assignment for SeqClu

Author: Rami Al-Obaidi
Supervisor: MSc. Azqa Nadeem
Responsible Professor: Dr. Ir. Sicco Verwer

Faculty of Electrical Engineering, Mathematics & Computer Science
Delft University of Technology

Abstract

Clustering is a group of (unsupervised) machine learning algorithms used to categorize data into clusters. The most popular clustering algorithm is k-means clustering. K-means clustering clusters the data into k clusters where a cluster is represented by the mean of the data points called a centroid. Instead of using the mean as a centroid, a data point (medoid) can be used instead. This algorithm is called k-medoids algorithm. Both the algorithms work in an offline setting where all the data is known beforehand and usually use Euclidean distance to calculate the distance between any two points. SeqClu is another clustering algorithm for sequential data that works in an online setting and uses Dynamic Time Warping as its distance measure. It is based on k-medoids where it uses p sequences called prototypes, to represent a cluster. It assigns an incoming sequence to the cluster that has the lowest average distance between its prototypes and the incoming sequence. The issue in this approach is that many Dynamic Time Warping distance calculations need to be made which affects the clustering speed and using the average distance affects the clustering accuracy due to outliers being assigned as prototypes. This paper proposes an alternative algorithm with three variants for the cluster assignment process. This algorithm iterates through the prototypes in search for the closest prototype while excluding clusters that are deemed too far. It assigns the incoming sequence to the cluster of the closest prototype that it has found. Experiments on the UJI Pen Characters and UCR Synthetic Control datasets show an improvement in clustering speed and accuracy.

1 Introduction

Clustering is an unsupervised learning algorithm used to classify data points whose labels are unknown. The aim of a clustering algorithm is to minimize intra-cluster distance and maximize inter-cluster distance [1]. A popular clustering algorithm is *k-means clustering. K-means clustering algorithm* works by partitioning the data into k clusters. The center of a cluster is calculated by taking the mean of the data points that belong to that particular cluster [2]. However, *k-means clustering* is known to be sensitive to outliers [3]. An alternative algorithm called *k-medoids clustering* can be used to avoid this issue [4]. Instead of assigning a centroid to represent a cluster, a data point (medoid) is used to represent a cluster instead [5].

Sequential data is a type of data where the ordering is important. Time-series data is a type of sequential data. Sequential data can be from many sources like the stock market or sensors and it is used in many contexts like financial analysis or weather forecasting, etc [6, Chapter 5]. In a clustering algorithm, the distance between two objects needs to be calculated during runtime. In k-medoids clustering, a distance measure is used to compute the distance between the points and the medoids in order to assign them to an appropriate cluster. One of the most popular distance

measures is Euclidean distance. However, Euclidean distance is unsuitable for sequential data, and therefore, Dynamic Time Warping distance is used instead of Euclidean distance [7] [8].

SeqClu is a real-time sequence clustering using an online k-medoids algorithm. SeqClu uses multiple sequences to represent a cluster. These are the most representative points in a cluster and they are called prototypes. The default implementation of SeqClu uses 5 prototypes to represent a cluster. SeqClu works in an online setting where not all the sequences are known initially. The sequences arrive one at a time with an unknown interval between them. SeqClu clusters sequential data therefore it uses Dynamic Time Warping as a distance measure.

Some of the related work that can be considered is from K. Nakagawa et al. [9] who proposed a k-medoids clustering algorithm for stock prices [9]. Their work use a variant of Dynamic Time Warping (DTW) called Indexed DTW since stock prices are time series data. However, their proposed algorithm works on offline data. L. O'Callaghan et al.[10] proposed STREAM algorithm [10]. Their algorithm deals with online/streaming data just like SeqClu. It buffers new data points before clustering them and it uses k-medians instead of k-medoids. Another algorithm that cluster data streams is CluStream [11] which makes use of micro-clusters. There have been also other research focused on improving the cluster assignment process for clustering algorithms such as the work by J. Ortega et al. [12] who worked on improving the cluster assignment process of k-means clustering for big data [12].

Currently, the cluster assignment process of SeqClu needs to calculate the Dynamic Time Warping distance between an incoming sequence and every prototype which is a computationally expensive process. This paper describes different ways to improve the cluster assignment process and compares them to the baseline implementation of SeqClu to assess their performance and accuracy. In section 2, the existing algorithm is described in detail and the problems with the current process are identified. The main contributions are explained in section 3. It contains the new cluster assignment algorithm and how it tries to fix the problems identified in the previous section. In section 4, the experimental setup is introduced and elaborated on. In section 5, the results of the different solutions are shown and compared to the original SeqClu implementation using different datasets; results reproducibility is also discussed in this section. The results are also compared and investigated in order to determine if there is an improvement in the cluster assignment process in section 6. The paper is concluded in section 7.

2 Investigating SeqClu

In order to investigate and improve SeqClu, the current algorithm of SeqClu needs to be fully outlined first. Afterward, potential flaws can be identified from the current algorithm.

2.1 Current algorithm

There are three important phases that take place when SeqClu is running, **cluster initialization**, **cluster assignment** and **cluster update**. Cluster initialization is done when SeqClu first starts up. Cluster assignment and cluster update phases are done for every incoming sequence that needs to be clustered. SeqClu clusters points in n clusters where each cluster is represented by p prototypes (default number of prototypes is 5).

Since SeqClu is still in an experimental stage, it uses an initialization strategy where every p sequences are assigned to each cluster.

When an incoming sequence needs to be assigned to a cluster, the Dynamic Time Warping distance is calculated between the incoming sequence and each prototype of a cluster. The incoming sequence is *assigned* to the cluster which has the lowest average distance between the incoming point and its prototypes. The pseudocode which can be found in algorithm 1, describes the cluster assignment process for an incoming sequence in detail.

Algorithm 1 Pseudocode for SeqClu cluster assignment

```
function ClusterAssignment(sequence, clusters)
    assignedCluster ← {}
    minimumDistance ← ∞
    for all C ∈ clusters do
        distance ← (∑_{p∈C} DTW(sequence, p))/|C|
        if distance < minimumDistance then
            assignedCluster ← C
            minimumDistance ← distance
        end if
    end for
    return assignedCluster
end function
```

Once a sequence has been assigned to a particular cluster, the cluster's prototypes will need to be *updated* to reflect that a new sequence now belongs to this cluster. SeqClu updates the cluster's prototypes by swapping the prototype with the farthest distance from the incoming sequence by the incoming sequence itself. During cluster update, p DTW distance calculations need to be made.

2.2 Problems with the current algorithm

The focus of this paper is to improve the cluster assignment process. As can be seen from the pseudocode of the cluster assignment phase, the Dynamic Time Warping distance needs to be calculated between the incoming sequence and every prototype of every cluster. That means for n clusters with p prototypes, $n * p$ Dynamic Time Warping distance calculations have to be done to assign a **single** incoming sequence to a cluster. The time complexity of calculating Dynamic Time Warping distance is $O(n * m)$, where n and m are the lengths of the two sequences [13] which means it is a computationally expensive operation [14]. Therefore, an improvement in the cluster assignment process that can be thought of is how to reduce the number of Dynamic Time Warping distance calculations while continuing to represent a cluster with p prototypes and ensuring that the accuracy is not severely affected.

Another cause of concern with the cluster assignment process of SeqClu is calculating the mean distance between an incoming sequence and the prototypes of a cluster. Mean is susceptible to outliers therefore it might cause sequences to be assigned to incorrect clusters. Since SeqClu replaces the farthest prototype with the incoming sequence in the assigned cluster, an outlier can become a prototype (see the example in figure 1). An important point to investigate here is whether there is a better alternative to calculating the mean and how it will affect the accuracy of SeqClu.

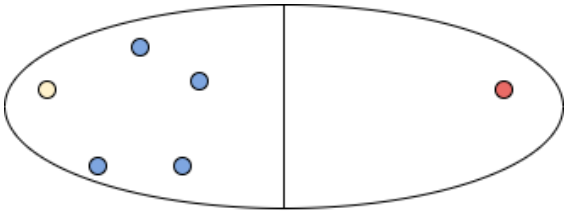

Figure 1: Even though all the sequences are on the left, the incoming sequence (red) replaces the yellow sequence as the new prototype due to the cluster update process of SeqClu

3 Improving SeqClu

At the end of section 2, two aspects were identified to improve the cluster assignment process. These two aspects are reducing the number of Dynamic Time Warping (DTW) distance calculations and avoid using the mean distance between a sequence and the prototypes of a cluster. This paper proposes a new algorithm and three variants that aim to reduce the number of DTW calculations and assign an incoming sequence to the cluster that has the closest prototype to the incoming sequence. The main idea behind the new algorithm is to exclude clusters that can be considered *very far* while searching for the closest prototype. The motivation behind this idea is that by excluding *sufficiently far* clusters during the iterative process, the total number of DTW distance calculations can be reduced. A simple visualization of the new algorithm can be seen in figure 2.

At the first iteration, one prototype is selected from each cluster, the DTW distances between the incoming sequence and the selected prototypes are calculated and stored. From figure 2, we can see this selection by the green circle on one prototype from each cluster. The shortest distance between the incoming sequence and the selected prototypes is retrieved and is denoted as d_{min}. The parameter ϕ controls how far a cluster should be from the incoming sequence (compared to the closest cluster to the incoming sequence) in order to be excluded for the next iteration. Clusters are included in the next iteration if they **satisfy** the following condition:

$$\frac{d_i - d_{min}}{d_{min}} \leq \phi$$

where d_i is the distance between the incoming sequence and the currently selected prototype for cluster i. If one cluster remains, then the sequence is assigned to the last remaining cluster. In subsequent iterations, a new prototype is selected for each cluster that passed the previous iteration. As shown in figure 2, cluster 3 didn't pass the first iteration so no new prototype is selected from it. The DTW distances between the incoming sequence and the newly selected prototypes are calculated. The newly selected prototype of cluster x replaces that of the previously selected prototype of the same cluster if it is closer to the incoming sequence than the old one. From figure 2, cluster 2 now has a closer prototype than the one selected from the previous iteration so it replaces it. Clusters are excluded just like the first iteration and if one cluster remains then the sequence is assigned to that cluster. If it is the last iteration, no filtering is done and the sequence is assigned to the cluster which has the closest prototype. In the worst case, $n * p$ DTW distance calculations will still need to be done (no cluster was excluded) but this should not happen very often. Algorithm 2 contains the full pseudocode of this new algorithm.

As mentioned earlier, the new algorithm has (three) different variations which differ in how a prototype is selected at each iteration. The three variants are described as follows:

1. Variant 1 starts with a random prototype (from each cluster) at the first iteration and selects another random prototype in subsequent iterations.

2. Variant 2 also starts with a random prototype but in subsequent iterations, it selects the closest prototype to the one from the previous iteration which has not been yet selected. Distances between prototypes of each cluster are calculated during cluster initialization and updated during the cluster update phase.

3. Variant 3 sorts the prototypes of each cluster by the average DTW distance between each prototype for every cluster. In another words, a prototype s ranks the highest in a cluster F (represented as the set of its prototypes) if it has the lowest $(\sum_{w \in D} DTW(w, s))/|F|$, where $D = F - \{s\}$. Variant 3 starts with the prototype that has the lowest average DTW distance to other prototypes and continues in ascending order. Distances between prototypes are calculated and updated just like variant 2.

Variants 2 and 3 incur **extra DTW** distance calculations during cluster initialization as they need to maintain the distance between the prototypes of each cluster. This extra costs is equivalent to $n * \binom{p}{2}$.

Algorithm 2 Pseudocode for improved cluster assignment

Require: *clusters* ▷ The prototypes of the clusters
Require: $f : A \to B$ ▷ Selects a prototype from a cluster

```
1:  function ClusterAssignment'(sequence, φ)
2:      assignedCluster ← {}
3:      D ← {(c, ∞) | c ∈ clusters}
4:      for i ← 1, p do
5:          D ← selectPrototypes(sequence, D)
6:          if i ≠ p then
7:              (k', v') ← min(D)                          ▷ Closest cluster
8:              for all (k, v) ∈ D do
9:                  if k ≠ k' ∧ (v − v')/v ≥ φ then
10:                     D ← D − {(k, v)}
11:                 end if
12:             end for
13:             if D = {(k, v)} then                       ▷ Only one cluster remains
14:                 return k
15:             end if
16:         else
17:             assignedCluster ← min(D)
18:         end if
19:     end for
20:     return assignedCluster
21: end function

22: function SelectPrototypes(sequence, D)
23:     D' ← ∅
24:     for all (k, v) ∈ D do
25:         v' ← DTW(sequence, f(k))        ▷ Get a new prototype and find the distance to it
26:         if v' < v then
27:             D' ← D' ∪ {(k, v')}
28:         else
29:             D' ← D' ∪ {(k, v)}
30:         end if
31:     end for
32:     return D'
33: end function
```

4 Experimental Setup

The experiment aims to compare how the original algorithm holds against the new algorithm with its three variants. Due to resource constraints, three values for ϕ will be used for the three variants which are 0.75, 1.0, and 1.25. The purpose of the experiment is not to find an optimal ϕ but rather to show that the new algorithm improves the clustering process. The experiment will

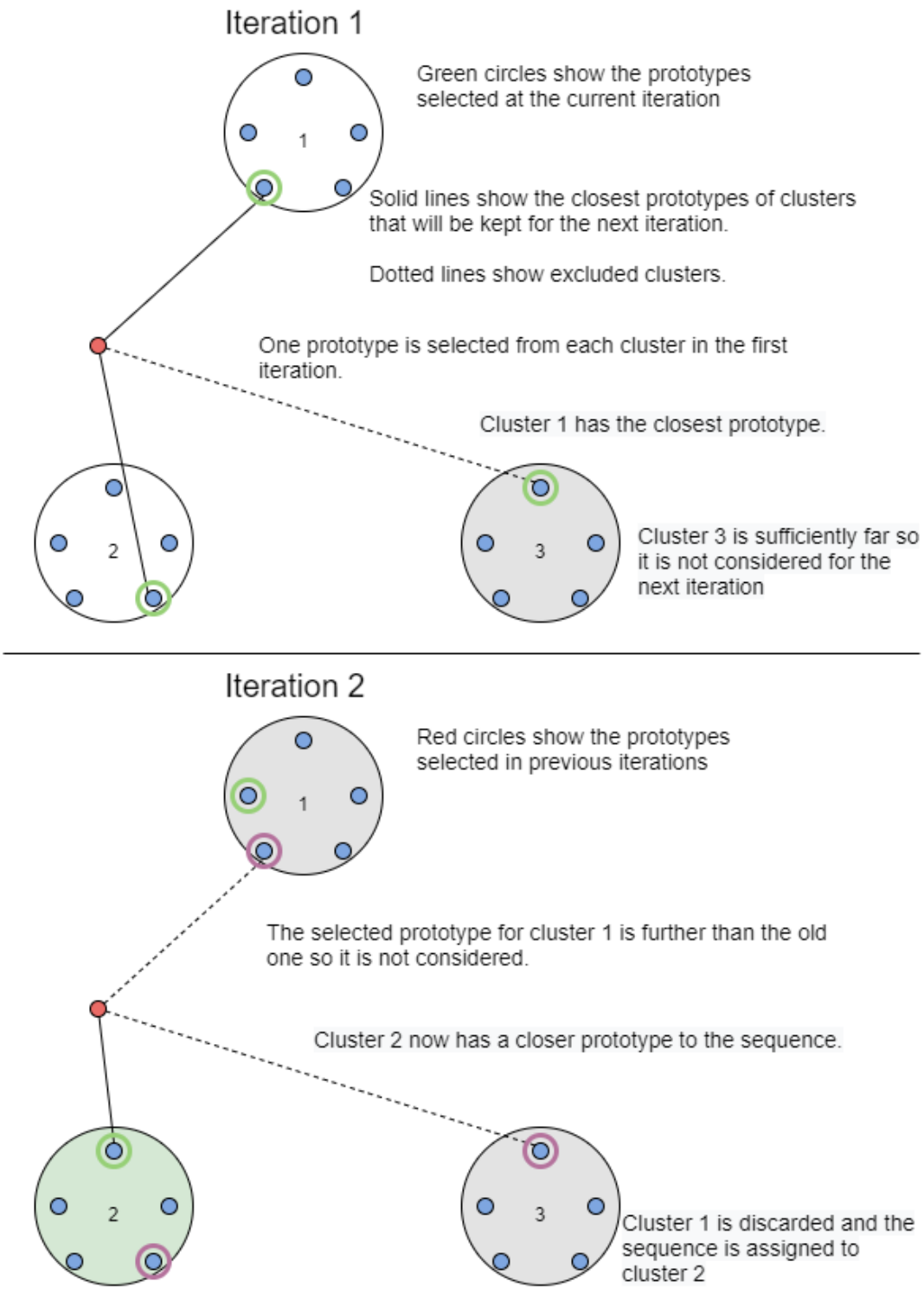

Figure 2: Visualization of the new algorithm

randomize the data stream (except for the first $n*p$ sequences as they are needed for initialization) and the same stream is used on all the implementations. The stream is randomized again and this is repeated x times in order to get reliable results and mitigate the effect of randomness. The goal of the experiment is to check if the new algorithm (with its three variants) improves the clustering speed while aiming for an improvement in the clustering accuracy.

At the end of each trial, performance metrics are calculated for each implementation and they are stored. At the end of the experiment, the mean and the standard deviation of each metric and implementation are calculated. The performance metrics are the following:

1. Total time to assign every incoming sequence to a cluster.
2. Total number of Dynamic Time Warping distance computations during the cluster assignment process (this also includes the extra Dynamic Time Warping distance computation incurred during initialization for variants 2 and 3).
3. Average Dynamic Time Warping distance computations for assign an incoming sequence to a cluster.
4. The (micro) F_1-score which is calculated as follows:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Once these statistics are calculated, a bar chart is created for each metric to compare the obtained results between the original SeqClu implementation and the three variants (with different ϕ values). Statistical hypothesis testing is used to check if there is a significant difference between the original SeqClu implementation and the three variants. The Wilcoxon signed-rank test is used as a statistical hypothesis test in this experiment setup as the samples are paired and it can't be guaranteed that the data is normally distributed for all the metrics (e.g., the average DTW distance calculations during cluster assignment for the original SeqClu algorithm is always $n * p$) [15] [16]. Two datasets will be used for the experiment which will be elaborated more on in section 5.

The source code for the experimental setup is developed in Python programming language [1]. The source code contains the original SeqClu implementation and the implementation of the new improved cluster assignment algorithm with its three variants. It also contains the statistics calculation used to make conclusions about the performance of each implementation. Multiple external libraries are used in the source code to set up the entire experiment. FastDTW [17] is used to calculate the Dynamic Time Warping distance between two sequences. Numpy [18], Scipy [19], Matplotlib [20], Scikit-learn [21], Tslearn [22], and Pandas [23] are used for different functionalities like calculating metrics and statistics.

5 Results

5.1 UJI Pen Characters dataset

The first dataset that will be used is the UJI Pen Characters dataset [24]. It consists of 3-dimensional data. Due to resource constraints, the experiment will only use seven characters from the dataset which are C, W, S, O, 1, 2, and 6. There are 281 sequences in the dataset that corresponds to these classes, where each sequence has a length of 35. The performance of the algorithms on this dataset can be found in figures 3, 4, 5, and 6.

[1] python.org

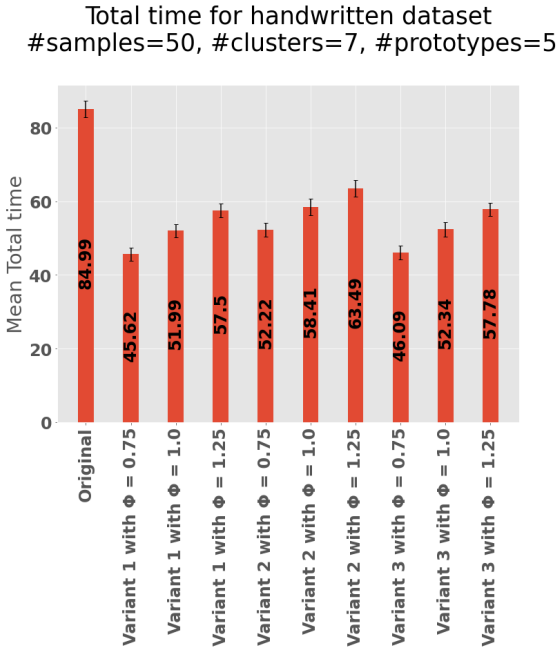

Figure 3: Total time to cluster the UJI Pen Characters dataset with different algorithms

5.2 UCR Synthetic Control dataset

The second dataset that will be used is the synthetic control dataset from the UCR time-series datasets archive [25]. It consists of univariate data. It contains 300 sequences (for the train set) which are divided into 6 classes (50 sequences per class). Each sequence has a length of 60. The performance of the algorithms on this dataset can be found in figures 7, 8, 9, and 10.

5.3 Result Reproducibility

The source code mentioned in section 4 can be found from the project's GitHub repository[2]. The project repository contains the source code and all the documentation needed to understand the source code. The datasets used in this paper are also included in the project's repository. The source code is included as a Jupyter Notebook[3]. The version of Python used in the source code is Python 3.8. The repository includes a requirements file that indicates the dependencies that need to be installed.

The results shown in subsections 5.1 and 5.2 were obtained using the same source code. The relevant hardware specifications of the device which was used to obtain the results are Intel Core i7-8750H @ 2.20GHz and 16 GB RAM. The device was running the latest version (21H1) of the Windows 10 operating system at the time of the experiment.

6 Discussion

The obtained results are similar between the two datasets. The new algorithm shows a big improvement in clustering speed. The new algorithm shows a substantial decrease in total time

[2]github.com/rami-ra/SeqClu-ClusterAssignment
[3]jupyter.org

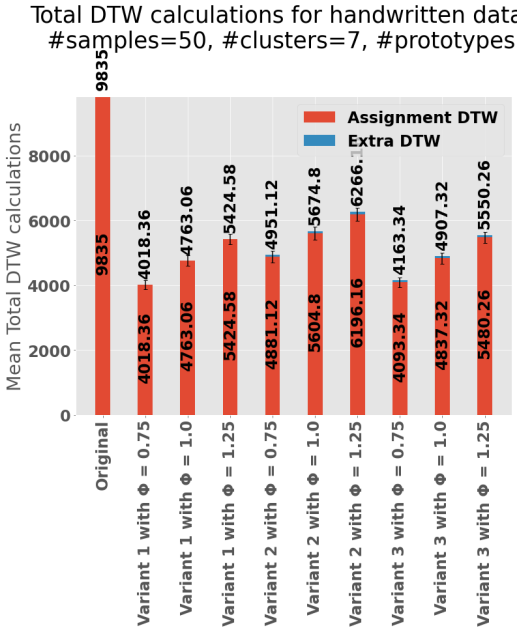

Figure 4: Total number of DTW calculations to cluster the UJI Pen Characters dataset with different algorithms including extra DTW calculations incurred during initialization (if any)

for clustering, total DTW distance calculations for clustering (including the extra DTW distance calculations incurred during initialization) for both the datasets. From the figures, we can see there is an improvement in the F-score for both the datasets for variants 1 & 3 (for all ϕ values) while there is a (slight) drop in the F-score for variant 2 across all ϕ values when compared to the original implementation of SeqClu. A substantial improvement can also be seen when considering the number of DTW distance calculations to cluster a single sequence. The original algorithm will always use $n * p$ DTW distance calculations to cluster a sequence but in the best performing case (variant 1 with $\phi = 0.75$), there is a decrease of $\approx$50% and a decrease of $\approx$30% in the worst performing variant (variant 2 with $\phi = 1.25$). As expected, excluding *sufficiently far* clusters improves the clustering speed greatly, and assigning the incoming sequence to the cluster which has the closest prototype improves clustering accuracy.

In order to confirm these results, the Wilcoxon signed-rank test will be used to compare the original algorithm and three variants using $\phi = 1.0$ against each other. The statistical test shows that there is a significant difference when comparing the original algorithm with the three variants across all the metrics for both the datasets except in F-score between the original algorithm and variant 2 for the UCR Synthetic Control dataset. Variant 2 shows a significant difference when compared to variants 1 & 3 across all metrics for both datasets. Variants 1 & 3 don't show a significant difference in F-score for UCR Synthetic Control and no significant difference between them with regards to the total time for both datasets.

As can be seen from the statistical test results, the new algorithm is a significant improvement over the original algorithm. It has a significant improvement in the clustering speed across all the variants. Variants 1 & 3 also improve the clustering accuracy while variant 2 doesn't do well in terms of improving the clustering accuracy. The hypothetical reason that variant 2 fails in this regard is that it selects new prototypes that are the closest to the current ones. However, this doesn't mean that they are nearer to the incoming sequence so it can cause some clusters to be discarded even if they have not yet selected prototypes that are nearer to the incoming sequence.

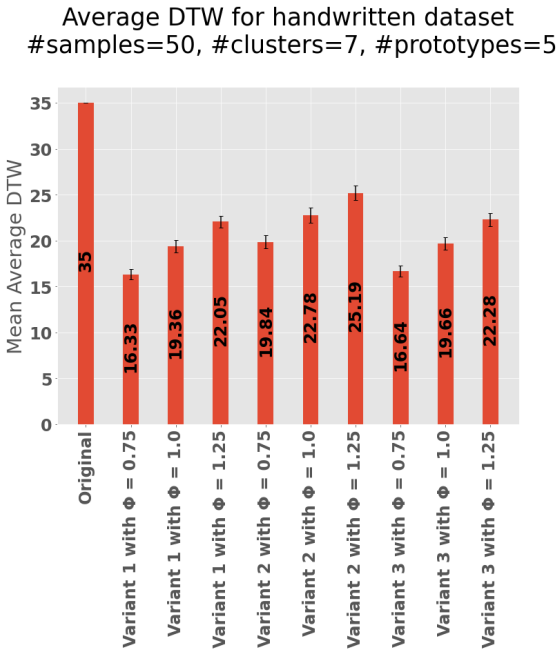

Figure 5: Average number of DTW calculations to cluster one point in the UJI Pen Characters dataset with different algorithms

It can also lead to fewer clusters being discarded so more DTW distance calculations are needed. The same issue is present in variants 1 & 3 but it occurs less due to the way these two variants select the prototypes.

7 Conclusions & Future Work

In this paper, a new (iterative) algorithm has been proposed to accelerate the cluster assignment process for SeqClu. SeqClu is a real-time sequence clustering using an online k-medoids algorithm. This iterative algorithm iterates over the prototypes of the cluster in order to find the closest prototype while excluding clusters that are deemed *not sufficiently close* (controlled by the ϕ value). It assigns a sequence to the cluster that has the closest prototype while the original algorithm assigned it to the cluster that has the lowest average distance between its prototypes and the sequence. This algorithm has three variants that differ in how they select a prototype at each iteration. Variant 1 selects a (new) prototype from a cluster randomly at each iteration. Variant 2 starts with a random prototype from a cluster in the first iteration, and in subsequent iterations, it selects the closest (not previously selected) prototype to the previously selected prototype. Variant 3 ranks the prototypes of a cluster by the average Dynamic Time Warping distance between them in ascending order; it uses this order to selects prototypes at each iteration. This algorithm aims to decrease the number of Dynamic Time Warping distance calculations by excluding far clusters and attempts to improve the accuracy by assigning sequences to the cluster with the closest prototype.

The three variants were tested against the original implementation of SeqClu across two datasets. They all showed a significant improvement in clustering speed. Variants 1 & 3 also showed a significant improvement in the clustering accuracy when compared to the original algo-

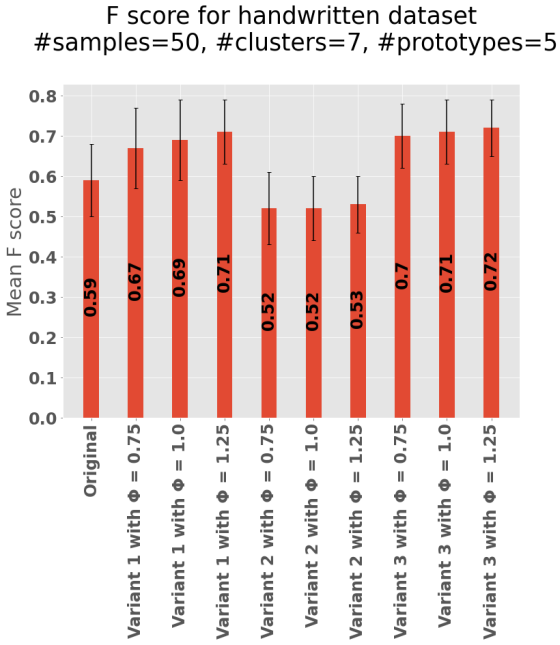

Figure 6: F score when clustering the UJI Pen Characters dataset with different algorithms

rithm. Variant 2 generally performed worse than variants 1 & 3 but performed better in terms of clustering speed when compared to the original algorithm. Variant 1 and Variant 3 perform similarly when compared to each other in terms of the DTW distance calculations but variant 3 needs $n * \binom{p}{2}$ extra DTW distance calculations during cluster initialization which means it scales worse when compared to variant 1.

Future improvements to the proposed algorithm can revolve around optimizing the value of ϕ, have a changing ϕ value during the iterative process, or explore different heuristics to select prototypes. The original SeqClu implementation was intended to cluster network traffic to detect malware. The new algorithm will also need to be tested on network traffic to check its performance in its intended context. Ethical considerations need to be taken into account when testing SeqClu and its new proposed improvement in their intended setting (clustering network traffic) to ensure that normal network traffic doesn't get classified as malicious (false positives) which could have wide implications on the network users.

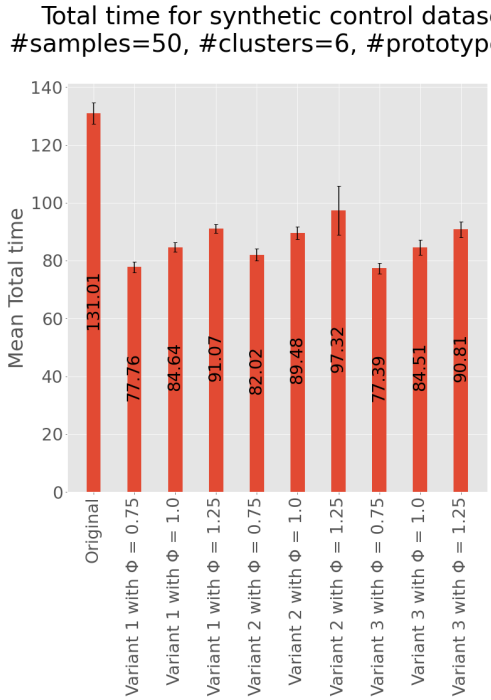

Figure 7: Total time to cluster the UCR synthetic control dataset with different algorithms

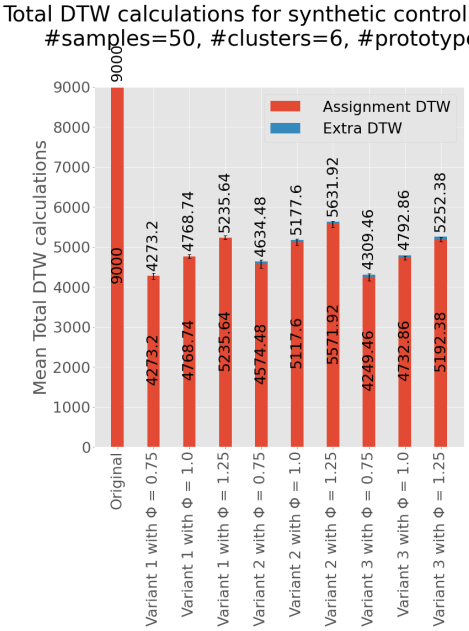

Figure 8: Total number of DTW calculations to cluster the UCR synthetic control dataset with different algorithms including extra DTW calculations incurred during initialization (if any)

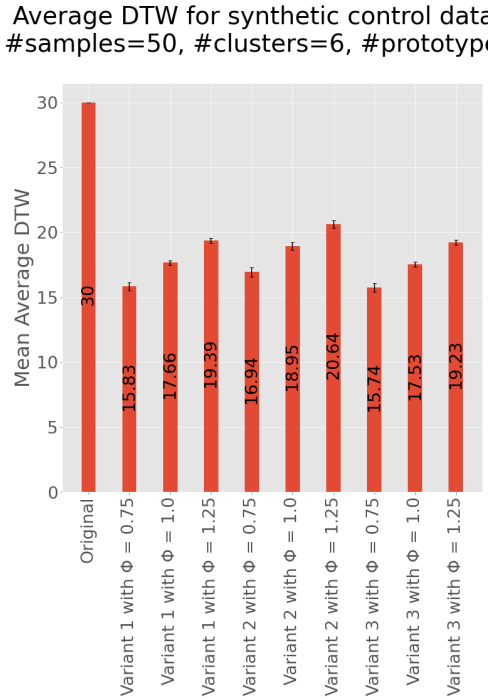

Figure 9: Average number of DTW calculations to cluster one point in the UCR synthetic control dataset with different algorithms

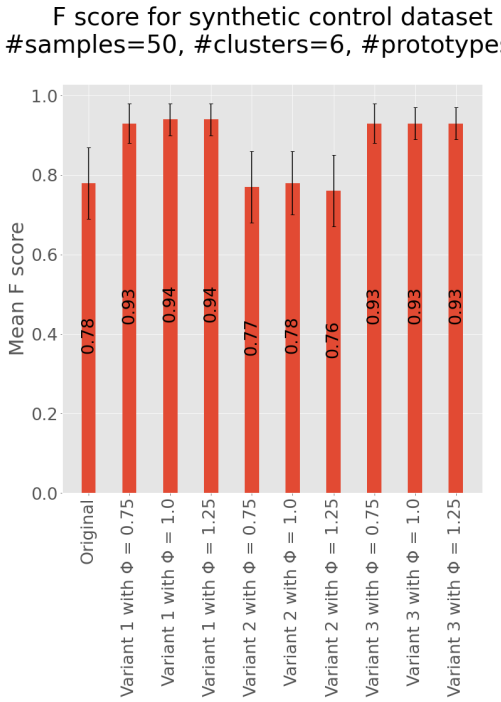

Figure 10: F score when clustering the UCR synthetic control dataset with different algorithms

References

[1] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985, ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(85)90224-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304397585902245.

[2] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[3] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3336–3341, 2009, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2008.01.039. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741740800081X.

[4] X. Jin and J. Han, "K-medoids clustering," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 564–565, ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_426. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_426.

[5] L. Kaufmann and P. Rousseeuw, "Clustering by means of medoids," *Data Analysis based on the L1-Norm and Related Methods*, pp. 405–416, Jan. 1987.

[6] P. Joshi, *Artificial intelligence with Python : build real-world artificial intelligence applications with Python to intelligently interact with the world around you*, English. Birmingham, UK: Packt Publishing, 2017.

[7] R. Ma and R. Angryk, "Distance and density clustering for time series data," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 25–32. DOI: 10.1109/ICDMW.2017.11.

[8] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," in *KDD workshop*, Seattle, WA, USA: vol. 10, 1994, pp. 359–370.

[9] K. Nakagawa, M. Imamura, and K. Yoshida, "Stock price prediction using k-medoids clustering with indexing dynamic time warping," *Electronics and Communications in Japan*, vol. 102, no. 2, pp. 3–8, 2019. DOI: https://doi.org/10.1002/ecj.12140. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ecj.12140. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ecj.12140.

[10] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," Feb. 2002, pp. 685–694, ISBN: 0-7695-1531-2. DOI: 10.1109/ICDE.2002.994785.

[11] C. Aggarwal, J. Han, J. Wang, P. Yu, T. Watson, and R. Ctr, "A framework for clustering evolving data streams," Jun. 2003.

[12] J. Ortega, N. N. A. Ortega, J. A. Ruiz-Vanoye, R. A. P. Rangel, S. Sáenz-Sánchez, J. M. Rodríguez-Lelis, and A. Rebollar, "A-means: Improving the cluster assignment phase of k-means for big data," *Int. J. Comb. Optim. Probl. Informatics*, vol. 9, pp. 3–10, 2018.

[13] P. Senin, "Dynamic time warping algorithm review," Jan. 2009.

[14] E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping to massive datasets," in *Principles of Data Mining and Knowledge Discovery*, J. M. Żytkow and J. Rauch, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 1–11, ISBN: 978-3-540-48247-5.

[15] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945, ISSN: 00994987. [Online]. Available: http://www.jstor.org/stable/3001968.

[16] A. Imam, M. Usman, and M. Chiawa, "On consistency and limitation of paired t-test, sign and wilcoxon sign rank test," *IOSR Journal of Mathematics*, vol. 10, pp. 01–06, Jan. 2014. DOI: 10.9790/5728-10140106.

[17] S. Salvador and P. Chan, "Fastdtw: Toward accurate dynamic time warping in linear time and space," 2004.

[18] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020. DOI: 10.1038/s41586-020-2649-2.

[19] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[20] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, "Tslearn, a machine learning toolkit for time series data," *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-091.html.

[23] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[24] D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[25] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, *The ucr time series classification archive*, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/, Oct. 2018.