SeqClu-PV: An extension of online K-medoids to efficiently cluster sequences real-time

Ruben E.C. te Wierik[1],
Supervisors: A. Nadeem[1], **S. Verwer**[1]

[1]Delft University of Technology

Abstract

Real-time sequence clustering is the problem of clustering an infinite stream of sequences in real-time with limited memory. A variant of the k-medoids algorithm called *SeqClu* is the suggested approach, representing a cluster with p most representative sequences of the cluster, called prototypes, to solve the problem of maintaining a high-quality representation of a cluster that requires little memory throughout time. However, the computational cost of this algorithm is considerable due to many distance computations that use Dynamic Time Warping (DTW), which is a computationally expensive distance measure that can be applied to sequences and is proven to be robust to noise and delays. Therefore, this paper proposes an extension of *SeqClu* called *SeqClu-PV*, characterised by a decision-making mechanism for updating prototypes that improves the balance between the number of distance computations and the cost incurred due to incorrect clustering and reviews its performance.

1 Introduction

Practical algorithms for clustering streams of sequences in an online setting would be helpful in a wide range of applications involving sequences of data generated in real-time, such as the analysis of internet traffic [20], Internet of Things (IoT) applications [15, 18, 3] and real-time systems involving multimedia [17]. Such sequences of data are referred to as time-series if an ordering can be inferred, as is the case in, for instance, handwriting data and any other sequences of data that are generated at a specific moment in time.

There is a vast literature on both online and offline clustering algorithms for various kinds of data, including time-series. The main conclusion drawn from the literature review is that none of the algorithms proposed in the found papers are suitable for the extension of the k-medoids algorithm involving multiple prototypes. A few other notions about each of the found papers are as follows.

Cardot et al. propose an online clustering algorithm based on the k-medoids algorithm [4]; however, this algorithm assumes high-dimensional data. Ailon et al., Liberty et al. and Cohen-Addad et al. propose promising online variants of the k-means algorithm [2, 11, 6]; however, these algorithms assume high-dimensional data. Grua et al. propose an exciting online clustering algorithm that assumes time-series data [8]. The algorithm introduces the concept of a microcluster that summarises the data within the cluster to obtain a more compact representation of the cluster. The idea of using more compact representations of clusters is also used in this paper. Choromanska and Monteleoni propose an algorithm that approximates the k-means objective by applying batch clustering algorithms to sliding windows of the data stream [5]; however, this algorithm assumes high-dimensional data and is more aimed at minimizing the objective rather than the computational complexity. Islam et al. propose a buffer-based online clustering algorithm for evolving data streams [9] that served as inspiration for the buffer proposed in this paper to improve the algorithm's robustness to the order in which data arrive. Like Grua et al., the authors also discuss the concept of microclusters but instead assumes high-dimensional data. Lastly, Kobren et al. propose an exciting online hierarchical clustering algorithm that works for large numbers of clusters and data points and assumes high-dimensional data [10].

As for relevant literature on offline clustering of time-series data, the reader is referred to various papers, most notably an extensive review of time-series clustering by Aghabozorgi et al. that provides a lot of insight on the challenges this problem poses [1]. Moreover, Yu et al. propose the idea of *candidate medoids* [21] that is used as inspiration for the buffer of candidate prototypes proposed in this paper.

This paper proposes a variant of *SeqClu* called *SeqClu with prototype voting* (SeqClu-PV). *SeqClu* is an online and sequence-based variant of the k-medoids algorithm that aims to tackle the problem of clustering a stream of data in an environment where data are clustered when they arrive and can only be processed once due to limited available memory. In the algorithm, k clusters are represented using p prototypes, which are the p most representative data points in a cluster and are meant to characterize the clusters. The research aims to find a combination of efficient decision-making mechanisms for updating prototypes that have the optimal balance between the number of distance computations and the cost incurred due to incorrect clustering. To this end, *SeqClu-PV* extends *SeqClu* with a feature called *prototype voting*, which improves the accuracy with which the centres of mass

of the clusters are modelled and therefore improves the overall accuracy of the algorithm. Furthermore, this feature can be paired with another feature that approximates the distance from an incoming sequence to a cluster to reduce distance computations. The final feature proposed in this paper involves delaying the updating of prototypes by introducing a buffer for candidate prototypes to improve further the algorithm's decision-making regarding the updating of prototypes.

The protocol that defines how prototypes are updated has a significant influence on the performance of the clustering algorithm. Suppose high-quality prototypes are discarded and replaced by prototypes of poor quality. In that case, the cluster that the prototypes characterize is no longer accurately represented, and incoming data might be assigned to the wrong cluster as a result. An example situation in which this is the case is shown in figure 6 in the appendix.

Considering the research from a more practical perspective: the most notable application of the proposed algorithm would be real-time analysis of internet traffic to detect malicious internet traffic. Using the algorithm for this purpose would improve the security of many web applications since these applications would have a better opportunity to defend themselves against malicious intent. Finally, the research is especially relevant considering the recent advances in technology: the algorithm in this paper could be a powerful addition to a world where systems operate more in a real-time setting where data is processed as soon as generated to assist humans more quickly and adequately.

This paper shows that a more efficient algorithm capable of clustering infinite streams of sequential data in an online setting is established and proves that the performance gain of the algorithm is substantial through a series of experiments that evaluate the performance using various evaluation metrics. The paper is organized as follows.

First, section 2 provides a formal description of the problem that will serve as the basis for the remaining content in the paper. Then, in section 3, section 4 and section 5, the modifications to the baseline implementation are presented and a mathematical foundation for these modifications as both mathematical and intuitive proofs is provided. Section 6 follows up with a description of the experiments that were carried out. The obtained results from the experiments and the conclusions drawn from these results are then discussed in section 7. Next, section 8 discusses the reproducibility of the research and provides a few ethical considerations related to the research. Finally, the paper concludes with section 9 and section 10 that summarize the conclusions drawn and suggests possible improvements and questions that arose during the research process but have not been answered, respectively.

2 The clustering problem

The problem of clustering an infinite stream of sequences in an online setting using *Seqclu* is as follows. Consider an infinite set, S, of sequences in $\mathbb{R}^d$, where $d \geq 1$ and new members of this set arrive over time. Important to note is that a sequence can be uni- or multivariate. $S_{[0,T]}$ is defined as a finite set of sequences that have been received from $t = 0$ up until and including $t = T$ and S_τ is defined as a finite set of sequences that have been received at $t = \tau$, where $t \in \mathbb{N}$ is a positive integer representing the moment in time, and $t = 0$ is the moment when the algorithm is started. Time is quantified using ticks, where an iteration of *SeqClu* is executed at every tick. A few definitions that are essential for understanding the problem are as follows.

Definition 1. The Dynamic Time Warping distance between two sequences, x and y, is denoted as $\mathrm{DTW}(x, y)$.

Definition 2. A cluster C is a set of $p \mid p \in \mathbb{Z},\ p \geq 1$ sequences in $\mathbb{R}^d$, where $d \geq 1$ and where the sequences represent the prototypes of the cluster.

Definition 3. The sum of distances between some sequence x in $\mathbb{R}^d$, where $d \geq 1$, and all prototypes in some cluster C is $\mathrm{SD}_{C,x} = \sum_{y \in C} \mathrm{DTW}(x, y)$. The average distance between some sequence x in $\mathbb{R}$ and all prototypes in some cluster C is $\mathrm{DTW}^{\mathrm{AVG}}_{C,x} = \frac{\mathrm{SD}_{C,x}}{|C|}$.

Definition 4. The set of sums of distances between any of the prototypes in a cluster C and all other prototypes in that cluster is $\mathrm{SD}_C = \{\mathrm{SD}_{C,x} \mid x \in C\}$. The average sum of distances between any of the prototypes in a cluster C and all other prototypes in that cluster is $\mathrm{SD}^{\mathrm{AVG}}_C = \frac{\sum_{x \in C} \mathrm{SD}_{C,x}}{|C|}$. The average distance between any of the prototypes in a cluster C is $\mathrm{DTW}^{\mathrm{AVG}}_C = \frac{\mathrm{SD}^{\mathrm{AVG}}_C}{|C|-1}$.

Definition 5. The set of $q \mid q \in \mathbb{Z},\ q \geq 1,\ q \leq p$ most representative prototypes of a cluster C is defined as $\mathbb{P}_{C,q}$. These most representative prototypes are defined as those prototypes in C that maximize a function that models the value of the prototype.

Definition 6. The objective of the *SeqClu* algorithm is to choose a set of k sets of p cluster prototypes at time t, $\{C_t \mid \forall (x, y) \in C_t \times C_t\ (x \neq y)\}$, which represents a set of clusters where no sequence can be a prototype for more than one cluster, that minimizes the cost function given in **Definition 7**.

Definition 7. The cost of the solution at time t, denoted as the set of clusters C_t, is defined as $\Phi(t) = \sum_{\tau \leq t} \sum_{x \in S_\tau} \mathrm{DTW}^{\mathrm{AVG}}_{A_x,x}$, where A_x is the set of p prototypes that represents the cluster that a sequence x is assigned to.

The goal is to design *approximation algorithms* due to the online nature of *SeqClu* that try to approximate the solution of the same algorithm in the offline version of *SeqClu*, where all data are known upfront.

3 Approximating the distance to a cluster

The strategy for updating prototypes characterises the online baseline algorithm. The procedure for processing an incoming sequence x, and updating the prototypes in the process, is as follows. The algorithm computes the distance $\mathrm{DTW}^{\mathrm{AVG}}_{C,x}$ to every cluster $C \in C_t$ and assigns the sequence x to the cluster $C_{\min}$ that minimizes this distance. The sequence also becomes a prototype for this cluster and replaces the prototype y that maximizes $\mathrm{DTW}^{\mathrm{AVG}}_{C_{\min},y}$, regardless of whether or not the incoming sequence is a better prototype than the one that is discarded for the incoming sequence.

A key observation regarding this baseline algorithm is that many distance computations must be done for every processed sequence. The algorithm first performs $k \times p$ distance computations to determine the cluster that the incoming sequence should be assigned to and then performs another $p \times p$ distance computations to determine the prototype that maximizes the average distance between that prototype and all other prototypes in that cluster, which is then replaced by the incoming sequence. Luckily, most of the distance computations in the second step can be memoized and reused in the next iteration of the algorithm since only one prototype changes at every iteration. However, the distance computations in the second step have to be executed at every algorithm iteration. Hence this step is the most sensible step for which to identify improvements involving approximation.

The first mechanism proposed in this paper is the idea of approximating the distance between an incoming sequence and a cluster. The algorithm can achieve this approximation by computing the distance between that sequence and a subset of the prototypes rather than the entire set. The mathematical foundation for this idea is established using the definition below.

Definition 8. An approximation of the average distance from an incoming sequence x to the cluster prototypes is the average distance from the incoming sequence to the $p' < p$ most representative prototypes of the cluster and is defined as $\mathrm{DTW}^{\mathrm{AVG}'}_{C,x,p'} = \mathrm{DTW}^{\mathrm{AVG}}_{\mathbb{P}_{C,p'},x}$.

To conclude, the idea of approximating the distance between an incoming sequence and a cluster involves computing the average distance between that incoming sequence and the p' most representative prototypes and assuming this distance is equal to the average distance between the incoming sequence and all prototypes. The assumption can be quite dangerous if the p' most representative prototypes do not represent all prototypes well since the accurate distance to some cluster could be significantly different from the approximated distance to the same cluster; an example of a situation in which this is the case is provided in figure 7 in the appendix.
Hence a definition for representativeness is provided below and used to define an upper bound for the error in the approximated distance from some incoming sequence to some cluster.

Definition 9. The representativeness of a prototype x that is a member of some cluster C is defined as $\mathrm{REP}_{C,x} = \frac{\mathrm{SD}^{\mathrm{AVG}}_{C}}{2 \times \mathrm{SD}_{C,x}}$.

This definition of representativeness can be viewed as the degree to which a cluster prototype represents all prototypes in that cluster. The higher the representativeness of the prototype, the lower the average distance between that prototype and all other prototypes in the cluster, the better the distance computation from some sequence x to that prototype approximates the average distance from x to all prototypes. A lemma stating that the representativeness has an upper bound of 1, and proof of this lemma is as follows.

Lemma 1. The maximum value of the representativeness of a cluster prototype is 1. In mathematical notation, this translates to $\max_{x \in C} \mathrm{REP}_{C,x} = 1$.

Proof. Consider a cluster C containing infinitely many prototypes in which all prototypes are the same, except for one of the prototypes. Consider the distance between any of the infinitely many identical prototypes and the one unique prototype to be d. In that case, the average sum of distances $\mathrm{SD}^{\mathrm{AVG}}_{C}$ is $\frac{(|C|-1)\times d + (|C|-1)\times d}{|C|}$. Intuitively, the infinitely many identical prototypes should be highly representative of the cluster considering all these prototypes have a distance of 0 to each other and a distance of d to the unique prototype and are therefore very close to the other prototypes overall. Evaluating $\mathrm{REP}_{C,x}$ where x is one of the infinitely many identical prototypes yields $\frac{(\frac{(|C|-1)\times d + (|C|-1)\times d}{|C|})}{2 \times d}$, which can be simplified to $\frac{2\times(|C|-1)\times d}{2\times|C|\times d}$, which can be further simplified to $\frac{|C|-1}{|C|}$. Considering the size of the set C is infinitely large, we can obtain the upper bound for the representativeness as $\lim_{|C|\to\infty} \frac{|C|-1}{|C|} = 1$. □

A definition that is required to define the error in the approximated distance from an incoming sequence to a cluster as well as a definition of the error itself can be found below.

Definition 10. The average representativeness of the p' representative prototypes of some cluster C is defined as $\mathrm{REP}^{\mathrm{AVG}}_{C,p'} = \frac{\sum_{x \in \mathbb{P}_{C,p'}} \mathrm{REP}_{C,x}}{|\mathbb{P}_{C,p'}|}$.

Theorem 1. The error in the approximated distance from any incoming sequence to a cluster C is defined as $E^{\mathrm{DTW}}_{C} = (1 - \mathrm{REP}^{\mathrm{AVG}}_{C,p'}) \times \frac{\sum_{x \in \mathbb{P}_{C,p'}} \mathrm{SD}_{C \setminus \mathbb{P}_{C,p'},x}}{p - p'}$.

To clarify **Theorem 1**, we can define an error factor that is the average degree of inaccuracy in the approximation of the average distance between x and all cluster prototypes in C. This error factor is scaled by the average distance from all representative prototypes, $\mathbb{P}_{C,p'}$, to all non-representative prototypes, $C \setminus \mathbb{P}_{C,p'}$.
The proof of **Theorem 1** is left as a future work due to the complexity of the mathematical proof. An interesting fact to note is that the triangle inequality does not hold for the *Dynamic Time Warping* distance measure, which might affect the ability to prove **Theorem 1**.

The error as defined above is used to determine whether or not assigning an incoming sequence to a cluster by approximation is justified. The definition for a function that decides whether or not a pair of distance computations is ambiguous is as follows.

Definition 11. The function $isAmbiguous\colon (x, C^1, C^2) \to boolean$ that takes an arbitrary sequence x and two clusters, C^1 and C^2, is defined as $|\mathrm{DTW}^{\mathrm{AVG}'}_{C^1,x,p'} - \mathrm{DTW}^{\mathrm{AVG}'}_{C^2,x,p'}| \leq \max(E^{\mathrm{DTW}}_{C^1}, E^{\mathrm{DTW}}_{C^2})$.

The above definition is used to determine whether or not the algorithm risks assigning a sequence to the wrong cluster if approximated distances are used to determine the cluster that

the sequence is closest to. If any ambiguity exists, the algorithm calculates the distance to all p prototypes instead to establish more accurate distances from the sequence to the clusters and then assigns the sequence to the closest cluster using these more accurate distances.

4 Buffering prototype candidates

A second idea to improve the algorithm's accuracy and therefore account for the expected loss of accuracy due to the approximation of the distance from incoming sequences to the clusters is to buffer incoming sequences that could represent a cluster and become a prototype. This mechanism is a trade-off where low memory usage is traded off for a slight improvement in accuracy due to delaying deciding whether or not to update the prototypes. A definition of all principles that are used to establish this mechanism can be found below.

Definition 12. The minimum average representativeness that the p' representative prototypes of some cluster should have is defined as $\mathrm{REP}^{\mathrm{MIN}}$.

Definition 13. The computation of the average distance between an incoming sequence x and some cluster C is accurate enough if the average representativeness of the p' representative prototypes of the cluster $\mathrm{REP}^{\mathrm{AVG}}_{C,p'}$ is equal to or greater than the minimum average representativeness $\mathrm{REP}^{\mathrm{MIN}}$.

The above two definitions can be used to decide whether or not the approximation of the distance of some sequence to some cluster is satisfactory enough. The higher the average representativeness, the higher the degree of approximation of the average distance between some sequence x and all prototypes of the cluster C. Intuitively, this translates to: the higher the average representativeness, the higher the likelihood that a candidate prototype is a better prototype than one of the current prototypes and, therefore, the higher the quality of candidate prototypes. If the prototypes are not representative enough, the accurate average distance to the cluster prototypes is computed instead and used to determine whether or not x is a candidate prototype.

With the above two definitions to decide whether or not to rely on an approximation of the distance to a cluster to decide whether or not an incoming sequence is a candidate prototype, the function that decides whether or not an incoming sequence is a candidate prototype for some cluster is defined as follows.

Definition 14. The function $candidate\colon (C, \mathbb{P}_{C,p'}, x) \rightarrow boolean$ determines if an incoming sequence x is a candidate to become a prototype of the cluster C. In the case where the prototypes of cluster C meet the condition specified in **Definition 13**, the function returns $true$ if $\mathrm{DTW}^{\mathrm{AVG}'}_{C,x,p'} < \mathrm{DTW}^{\mathrm{AVG}}_{C} + E^{\mathrm{DTW}}_{C}$ and $false$ otherwise. In the case where the prototypes of cluster C do not meet the condition specified in **Definition 13**, the function returns $true$ if $\mathrm{DTW}^{\mathrm{AVG}}_{C,x} < \mathrm{DTW}^{\mathrm{AVG}}_{C}$ and $false$ otherwise.

The above definition can be used to determine whether or not incoming sequences are candidate prototypes for any of the clusters and are then stored in a buffer up to the point where the maximum capacity of the buffer is reached. When this happens, all the items in the buffer need to be processed and added to the set of prototypes for some cluster if they turn out to be better prototypes than one of the current prototypes in that cluster. The next problem is how to decide which prototype to update, which is discussed in the following section.

5 Voting for prototypes

As mentioned in section 3, the online baseline algorithm promotes an incoming sequence x to a prototype for the cluster $C_{\min}$ that minimizes $\mathrm{DTW}^{\mathrm{AVG}}_{C_{\min},x}$ and replaces the prototype y that maximizes $\mathrm{DTW}^{\mathrm{AVG}}_{C_{\min},y}$, regardless of whether or not the incoming sequence is a better prototype than the one that is discarded for the incoming sequence.

A problem with this approach is that the cluster's centre of mass is not modelled correctly. The online baseline algorithm merely attempts to minimize the average distance between all prototypes of some cluster and does not consider the similarity of prototypes to the earlier assigned sequences. The online baseline algorithm processes new sequences after initializing the prototypes for all clusters, part of which are promoted to prototypes since these sequences decrease the average distance between prototypes. Suppose most of the later processed sequences assigned to that cluster are very similar to each other. In that case, the closest prototype should become a prototype of higher value than the other prototypes since it is most similar to the sequences assigned to the cluster overall. A hypothetical situation in which one of the prototypes is of higher value than the other prototypes is depicted in figure 8 in the appendix.

This prototype of higher value should not be discarded while it is considered of high value. However, in the online baseline algorithm, this prototype is discarded, and the representation of the cluster diverges from the centre of mass. After the purple sequence arrives in figure 8, the prototypes are the green, purple and right-most dark red sequences. Then, the two blue sequences arrive and replace the purple and green prototypes in that order. To tackle this issue, this paper proposes a third mechanism called *prototype voting* to address modelling the centre of mass of a cluster.

The idea behind *prototype voting* is to keep track of how often an incoming sequence that is not a candidate prototype is closest to any of the prototypes.

Definition 15. The amount of times a sequence has been observed as being closest to a prototype $P \in C$ compared to all other prototypes in C at time t is defined as $O_{P,t}$. The total amount of times a sequence has been observed as being closest to any of the prototypes of some cluster C at time t is defined as $O_{C,t} = \sum_{P \in C} O_{P,t}$. The *weight* of a prototype $P \in C$ at time t is defined as $w_{C,P,t} = \frac{O_{P,t}}{O_{C,t}}$.

The above definition of the *weight* of a prototype can be used in combination with the representativeness of a prototype discussed in section 3 to define the *value* of a prototype as a linear combination of the two.

Definition 16. The function $prototypeValue\colon (C, P, t) \rightarrow [0, 1]$ computes the *value* of a prototype as a linear combination of the *weight* and *representativeness* of the prototype P and is defined as $\alpha \cdot w_{C,P,t} + \beta \cdot \mathrm{REP}_{C,P}$ where $\frac{\beta}{\alpha} = r$ and r is the ratio that defines the importance of the representativeness relative to the importance of the weight.

An issue that remains with the above definition is what happens to the weight when a new prototype replaces an existing prototype. If the new prototype is added to the set of prototypes, the prototype will not be valuable since it does not have any votes. Moreover, the votes of the prototype that was replaced by the new prototype are discarded; therefore, information on the cluster's centre of mass is lost. A way to tackle this problem is to distribute the votes of the discarded prototype to the updated prototypes depending on the relative distance from a prototype to the discarded prototype. A definition of the algorithm for distributing the votes of a discarded prototype as pseudo-code is as follows.

Algorithm 1: Procedure for updating prototypes

Result: The votes of some discarded prototype P_{old} have been distributed to the updated prototypes in C.

fractions $\leftarrow$ empty(p); // creates empty array of size p
sumOfDistances $\leftarrow$ 0;
foreach prototype $P \in C$ **do**
  sumOfDistances += $\mathrm{DTW}(P, P_{old})$;
end
$i \leftarrow 0$;
foreach prototype $P \in C$ **do**
  fractions[i] = $1 - \frac{\mathrm{DTW}(P, P_{old})}{\mathrm{sumOfDistances}}$;
  i++;
end
sumOfFracs $\leftarrow$ sum(fractions); // sums values in array
fractions /= sumOfFracs; // divide every value by sum
$i \leftarrow 0$;
foreach prototype $P \in C$ **do**
  $O_{P,t}$ += int(fractions[i] $\cdot\, O_{P_{old},t}$);
  i++;
end

6 Experimental setup

The experiment compares four variants of the *SeqClu-PV* algorithm to the two baseline variants of the *SeqClu* algorithm, namely the offline variant based on Partitioning Around Medoids (PAM) and the online variant provided at the start of the research project and explores the strengths and weaknesses of each variant.

The four variants differ in whether or not the two features discussed in section 3 and section 4 respectively are enabled. Moreover, the voting for cluster prototypes proposed in section 5 is enabled for all four variants.

All algorithm variants are single-threaded and implemented in Python using the package *FastDTW* [16] for the distance computations. The variants of the *SeqClu-PV* algorithm will be evaluated on two open-source data sets: a data set of handwritten characters [7, 12] and a data set of gestures performed with the Pebble smartwatch [14].

The experiment is divided into three stages. First, both the experimental- and baseline variants of the algorithm are run on all data sets with a base set of parameters. Since the offline baseline variant is a deterministic algorithm, it only needs to be run once on each data set. All other variants are non-deterministic algorithms and are hence run 30 times on each data set, where the order of incoming sequences is randomized, to observe the average behaviour of each variant. This part of the experiment aims to observe each experimental variant's performance on a sensible set of parameters compared to the two baseline variants mentioned earlier.

Second, the experimental variant *SeqClu-PV-A*, which stands for *SeqClu-PV* with the approximation of the distance to a cluster and without buffering, is run 30 times on one of the data sets for each parameter set described as follows. The *SeqClu-PV* algorithm takes three additional parameters as input, which are the size of the buffer, the minimum average representativeness of a cluster that is required to reliably approximate the distance between an incoming sequence and that cluster and the factor that determines how many times the representativeness of a prototype is more important than the weight of a prototype. The latter parameter is used to compute the value of a prototype as a linear combination of its representativeness and weight, as discussed in section 5. This part of the experiment aims to study the influence of each parameter to gain insight into the volatility of the parameters and suitable values for specific parameters. Since one of the parameters, namely the buffer size, involves the buffering feature discussed in section 4, the experiment where the value of the buffer size is varied uses the experimental variant *SeqClu-PV-B* (without approximation of the distance to a cluster and with buffering) instead.

Last, the experimental variant *SeqClu-PV-A* is run on all data sets with the base set of parameters and increasing numbers of prototypes and representative prototypes to study the effect of the number of prototypes on the algorithm's performance.

The experimental variant *SeqClu-PV-A* is used in the second and third stage of the experiment because this variant theoretically requires fewer distance computations to compute the result and likely performs worst out of all variants due to the approximation of the distance to a cluster discussed in section 3 and the fact that this variant has the buffering feature disabled. Therefore, this variant is the most practical in terms of computational cost and memory requirements and likely benefits most from analysing the effects of the parameters.

The performance of the algorithm is measured in various ways, starting with the number of distance computations, the macro F1-score and the accuracy of the algorithm to provide a high-level overview of the overall performance of the algorithm in terms of quality of the obtained set of clusters. The definition of accuracy is as follows.

Definition 17. The following definitions take as input two lists of predicted labels $\hat{y}$ and actual labels y. The accuracy of some list of predicted labels $\hat{y}$ given the actual labels y is defined as $accuracy(y, \hat{y}) = \frac{\mathrm{TP}(y,\hat{y})+\mathrm{TN}(y,\hat{y})}{\mathrm{TP}(y,\hat{y})+\mathrm{TN}(y,\hat{y})+\mathrm{FP}(y,\hat{y})+\mathrm{FN}(y,\hat{y})}$, where *TP*, *TN*, *FP* and *FN* are defined as the number of true positives, false positives, true negatives and false negatives respectively.

Second, for the experimental variants of the *SeqClu-PV* algorithm that approximate the distance to a cluster, the sequences that are clustered by approximation are recorded and used to compute the following three measurements:

- the number of sequences that were correctly clustered;
- the number of sequences that were correctly clustered and incorrectly clustered by the online baseline variant;
- the number of sequences that were incorrectly clustered and correctly clustered by the online baseline variant.

These data are transformed into fractions, where each number is divided by the total amount of sequences clustered by approximation. Third, for the experimental variants of the *SeqClu-PV* algorithm that have buffering of prototype candidates enabled, the sequences that are buffered are recorded and used to compute the same three data as mentioned above. These data are again transformed into fractions, where each number is divided by the total amount of buffered sequences. Last, for the experimental- and online baseline variants of the algorithm, the prototypes of all clusters after the algorithm finished are recorded and compared to the prototypes of all clusters after the offline baseline algorithm finished, which yields a percentage of correct prototypes. This metric is used to study the effectiveness of the *prototype voting* discussed in section 5: the eventual representation of clusters obtained by the experimental variants should match the representation obtained by the offline variant since this representation is the most optimal one from a perspective where all data can be considered any number of times.

To verify that the experimental variants of the *SeqClu-PV* algorithm outperform the online baseline algorithm, a statistical test called the *Wilcoxon signed-rank test* is used to determine whether or not there is a significant difference between two paired samples of evaluation metrics generated from the first stage of the experiment [13]. These samples are paired since a pair of evaluation metrics was generated for every independent run of the first stage of the experiment by using some randomized stream of sequences as input for both the experimental variant of the *SeqClu-PV* algorithm and the online baseline algorithm. For the second and third stage of the experiment, the *Mann-Whitney U test* is used instead since the samples generated from these experimental stages are independent due to the experimental setup when conducting these stages of the experiment [13]. The reason why non-parametric statistical tests are used is because the data are not normally distributed, as is shown in appendix C [13]. The *Shapiro-Wilk test* is used to test if the data are normally distributed; the resulting p-values can be found in the caption of the charts [13].

7 Experimental results and discussion

This section presents the experimental results described in section 6 and discusses the conclusions drawn from these results. The entire set of experimental results can be found as both tables and charts in appendix D and appendix E, respectively. Some conclusions regarding the used data sets and a few abbreviations used in the charts are provided below, after which the results themselves are presented and discussed.

Conclusions regarding the used data sets

A few conclusions drawn from the information about the data sets provided in appendix A, among which visualizations of the data sets as t-SNE [19] charts, that are used in the remainder of the discussion of the results are as follows. From figure 4, it is concluded that the *UJI-PenCharacters* [7, 12] data set is quite noisy and that the clusters of classes *2*, *5* and *9* are sparse and very close to each other. The sequences in these clusters are scattered across half of the y-axis, which indicates that some sequences belonging to the same cluster are relatively far away from each other. This fact makes finding a good representation of the cluster that yields a relatively low distance to the cluster for all sequences belonging to that cluster quite a challenge. In conclusion, the *UJI-PenCharacters* [7, 12] data set is presumably rather difficult to cluster accurately due to the clusters of classes *2*, *5* and *9*.

From figure 5, it is concluded that the *GesturePebbleZ1* [14] data set contains clusters that are a lot denser and does not exhibit any noise. The clusters of classes *2*, *4* and *6* are likely easy to cluster since they do not overlap with any other clusters. The clusters of classes *1* and *5* are intertwined; hence most mistakes are probably made there. Moreover, part of the cluster of class *3* is closer to the cluster of class *4* than it is to the rest of the cluster of class *3*, which might be an additional cause of mistakes. In conclusion, the *GesturePebbleZ1* [14] data set is presumably easier to cluster accurately than the *UJI-PenCharacters* [7, 12] data set due to clusters being denser and the data set being less noisy.

Legend for the charts presented in this section

Dist.	Number of distance computations that the algorithm did during execution..
Acc.	The accuracy of the solution provided by the algorithm.
F_1	The macro F_1-score.
Prot.	The percentage of correct prototypes at the end of the algorithm.
App.↑	The percentage of sequences that were correctly clustered by approximation and incorrectly clustered by the online baseline algorithm.
App.↓	The percentage of sequences that were incorrectly clustered by approximation and correctly clustered by the online baseline algorithm.
Buff.↑	The percentage of buffered sequences that were correctly clustered and incorrectly clustered by the online baseline algorithm.
Buff.↓	The percentage of buffered sequences that were incorrectly clustered and correctly clustered by the online baseline algorithm.
PV-AB	*SeqClu-PV* with both approximating the distance from some sequence to a cluster and buffering of candidate prototypes enabled.
PV-A	*SeqClu-PV* with approximating the distance from some sequence to a cluster enabled and buffering of candidate prototypes disabled.
PV	*SeqClu-PV* with both approximating the distance from some sequence to a cluster and buffering of candidate prototypes disabled.
PV-B	*SeqClu-PV* with approximating the distance from some sequence to a cluster disabled and buffering of candidate prototypes enabled.
BL	The online baseline variant of the *SeqClu* algorithm.

Table 1: Explanation of a few abbreviations used in the tables.

Performance of experimental variants of *SeqClu-PV*

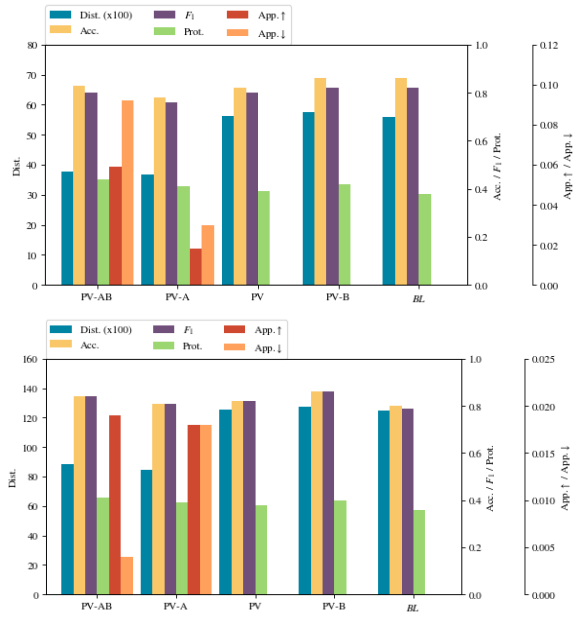

Figure 1: The upper chart contains the results for the *UJI-PenCharacters* [7, 12] data set, the lower chart contains the results for the *GesturePebbleZ1* [14] data set. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic variant is the variant to which all other variants present in the figure are compared with the *Wilcoxon signed-rank* test described in section 6, the results of which can be found in appendix D.

Performance of *SeqClu-PV-A* on first data set

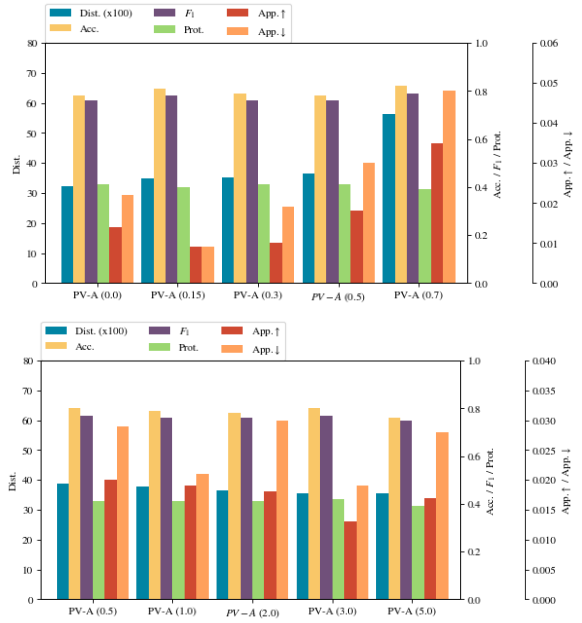

Figure 2: Both charts contain results for the *UJI-PenCharacters* [7, 12] data set. The upper chart varies in the parameter that defines the minimum representativeness. The lower chart varies in the parameter that defines the importance of the representativeness respective to the weight of prototypes. The values for these parameters are contained in the variant name. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic variant is the variant to which all other variants present in the figure are compared with the *Mann Whitney U* test described in section 6, the results of which can be found in appendix D.

Performance of *SeqClu-PV-A* on second data set

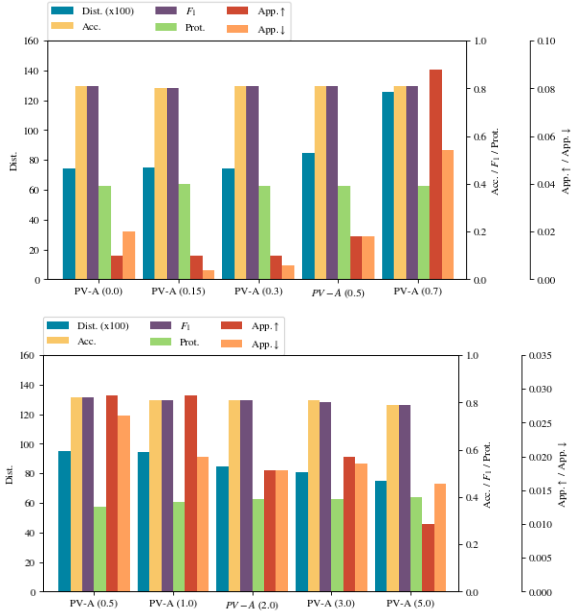

Figure 3: All charts contain results for the *GesturePebbleZ1* [14] data set. The upper chart varies in the parameter that defines the minimum representativeness. The lower chart varies in the parameter that defines the importance of the representativeness respective to the weight of prototypes. The values for these parameters are contained in the variant name. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic variant is the variant to which all other variants present in the figure are compared with the *Mann Whitney U* test described in section 6, the results of which can be found in appendix D.

Discussion

From the provided charts in figure 1, figure 2, and figure 3, the following interesting observations can be made.

First, for the noisy *UJI-PenCharacters* [7, 12] data set, the *PV-B* variant achieves the same accuracy as the online baseline variant and a higher percentage of correct prototypes. The *PV-A* variant shows a considerable loss of accuracy. However, the loss in accuracy is half as significant, and the percentage of correct prototypes is the highest of all variants for the *PV-AB* variant. The variants that approximate the distance to a cluster make more errors than the online baseline variant for the sequences clustered by approximation. The F_1 score of all variants is lower than the accuracy, which is sensible considering the class distribution of the *UJI-PenCharacters* [7, 12] data set is imbalanced and suggests that all variants suffer from either low precision or low recall. Moreover, for the well-separated *GesturePebbleZ1* [14] data set, all experimental variants outperform the online baseline variant. The *PV-B* variant again achieves the highest accuracy, and the *PV-A* variant the lowest accuracy. The *PV-AB* variant again achieves the highest percentage of correct prototypes. All experimental variants cluster sequences clustered by approximation or buffered with higher accuracy than the online baseline variant.

Second, for the noisy *UJI-PenCharacters* [7, 12] data set and the *PV-A* variant, the optimal value for the parameter that defines the minimum representativeness is *0.15*, resulting in an increase in the accuracy and F_1 score and in a decrease in the sequences that were wrongly clustered by approximation.

The optimal value for the parameter that defines the importance of the weight respective to the value of a prototype is *3.0*, which means the weight determines 25% of the value of a prototype.

Last, for the well-separated *GesturePebbleZ1* [14] data set, the optimal value for the parameter that defines the minimum representativeness is *0.15*, since the balance between the accuracy gain for sequences clustered by approximation and the number of distance computations is optimal. The optimal value for the parameter that defines the importance of the weight respective to the value of a prototype is *3.0* since the balance between the performance gain and the number of distance computations is optimal.

8 Responsible research

This section contains information on the code and environment used to obtain the results presented in this paper to improve the reproducibility of this research and discusses ethical implications of the research.

The code used to obtain the results presented in this paper is available on GitHub[1], this repository contains both the online- and offline baseline implementation as well as an implementation containing the mechanisms proposed in this paper. The algorithm can be launched using a command-line interface using either of the two data sets discussed in section 6 and any combination of parameters. The documentation for how to use this command-line interface can be found in the repository. The repository also contains the obtained research results, consisting of both the logged output and Excel sheets that record several evaluation metrics for all experiments carried out. Moreover, the code is made available as a Python package via PyPI[2].

To conclude with a discussion about the ethical implications of this research, one should consider the consequences of false positives to the people involved with the software that employs the algorithm proposed in this paper. An excellent example of this would be the internet traffic of some user who does not have malicious intent unjustly being marked as malicious internet traffic. This user suffers the consequences of the defensive mechanism implemented in the software employing the algorithm proposed in this paper.

One of the conclusions that can be drawn from the results presented and discussed in section 7 is that the experimental variants of *SeqClu-PV* seem to perform equally well or better than the online baseline algorithm in terms of accuracy, thanks to the powerful *prototype voting* mechanic discussed in section 5 that constitutes the main contribution of this paper. As a result, the probability that false positives occur is lower; however, sadly, not (close to) zero due to the online nature of the algorithm. For people or companies who are considering employing the algorithm proposed in this paper in their software to improve the security of the software, it is essential to note that false positives still occur and that they carry the responsibility to deal with these false positives properly as part of the guarantees of their software.

Modern computer systems are great additions to the world we live in since they can assist humans in carrying out tasks. However, it is essential to realize that these systems are not perfect. Human supervision and sometimes even intervention are necessary to ensure a good experience with these systems and fairness for all humans involved.

[1] https://github.com/rtewierik/SeqClu-PV

[2] https://pypi.org/project/SeqClu-PV/

9 Conclusion

This paper presents an extension to the *SeqClu* algorithm, called *SeqClu-PV*, for clustering a stream of sequences in an online setting. The algorithm employs a novel mechanism called *prototype voting* to model the centre of mass of a cluster more accurately. Moreover, the algorithm can approximate the distance between a sequence and a cluster, thereby reducing the required distance computations. Lastly, the algorithm can employ a buffer to delay updating the representation of a cluster, to reduce the risk of incorrectly updating said representation due to not knowing data that will arrive in the future. Through a series of experiments, it is proven that the *prototype voting* mechanism is a powerful addition to the *SeqClu* algorithm that results in a more optimal balance between the number of distance computations and the cost incurred due to incorrect clustering. The experiments demonstrate that the *SeqClu-PV-A* variant minimizes the number of distance computations at the expense of accuracy for noisy and imbalanced data sets and yielding an increase in performance for well-separated and dense data sets. The *SeqClu-PV-AB* variant achieves an even more optimal balance between the number of distance computations and the cost incurred due to incorrect clustering at the expense of extra memory.

10 Future work

A few future research recommendations to answer the remaining questions that the research project that resulted in this paper could not cover are as follows.

As mentioned in section 3, it could be interesting to prove **Theorem 1** provided in the same section to strengthen the theoretical foundation of the research discussed in this paper.

In section 5, the value of a prototype is defined as a linear combination of its *representativeness* and *weight*. It could be interesting to explore other functions that model the value of a prototype differently and test if these functions yield better results in terms of the quality of prototypes throughout the algorithm's execution.

Another idea that was not explored is modelling the risk of not updating prototypes to help decide when to update prototypes if the buffer is not yet full. This risk could be modelled by using data such as the tick at which any of the prototypes of some cluster were last updated, the tick at which a specific cluster prototype was last observed as being closest to an incoming sequence compared to other prototypes and the number of sequences added to a cluster since the prototypes were last updated. Logically, the algorithm's time complexity would increase due to having to keep track of additional information. However, the algorithm would be more capable of using all available information to make the right decisions at critical moments that can be identified using the risk metric, thereby improving the algorithm's accuracy.

References

[1] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering - A decade review. *Information Systems*, 53:16–38, 2015. ISSN 03064379. doi: 10.1016/j.is.2015.04.007. URL http://dx.doi.org/10.1016/j.is.2015.04.007.

[2] Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. In Y Bengio, D Schuurmans, J Lafferty, C Williams, and A Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2009/file/4f16c818875d9fcb6867c7bdc89be7eb-Paper.pdf.

[3] Sivadi Balakrishna, M. Thirumaran, R. Padmanaban, and Vijender Kumar Solanki. An efficient incremental clustering based improved K-Medoids for IoT multivariate data cluster analysis. *Peer-to-Peer Networking and Applications*, 13(4):1152–1175, 2020. ISSN 19366450. doi: 10.1007/s12083-019-00852-x.

[4] Hervé Cardot, Peggy Cénac, and Jean Marie Monnez. A fast and recursive algorithm for clustering large datasets with k-medians. *Computational Statistics and Data Analysis*, 56(6):1434–1449, 2012. ISSN 01679473. doi: 10.1016/j.csda.2011.11.019. URL http://dx.doi.org/10.1016/j.csda.2011.11.019.

[5] Anna Choromanska and Claire Monteleoni. Online clustering with experts. *Journal of Machine Learning Research*, 22:227–235, 2012. ISSN 15337928.

[6] Vincent Cohen-Addad, Benjamin Guedj, Varun Kanade, and Guy Rom. Online k-means Clustering. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1126–1134. PMLR, 2021. URL http://proceedings.mlr.press/v130/cohen-addad21a.html.

[7] Dheeru Dua and Casey Graff. {UCI} Machine Learning Repository, 2017. URL http://archive.ics.uci.edu/ml.

[8] Eoin Martino Grua, Mark Hoogendoorn, Ivano Malavolta, Patricia Lago, and A. E. Eiben. ClustreaM-GT: Online clustering for personalization in the health domain. *Proceedings - 2019 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019*, pages 270–275, 2019. doi: 10.1145/3350546.3352529.

[9] Md Kamrul Islam, Md Manjur Ahmed, and Kamal Z. Zamli. A buffer-based online clustering for evolving data stream. *Information Sciences*, 489:113–135, 2019. ISSN 00200255. doi: 10.1016/j.ins.2019.03.022. URL https://doi.org/10.1016/j.ins.2019.03.022.

[10] Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. A hierarchical algorithm for extreme clustering. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Part F1296:255–264, 2017. doi: 10.1145/3097983.3098079.

[11] Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. *Proceedings of the Workshop on Algorithm Engineering and Experiments*, 2016-Janua:81–89, 2016. ISSN 21640300. doi: 10.1137/1.9781611974317.7.

[12] D. Llorens, F. Prat, A. Marzal, J. M. Vilar, M. J. Castro, J. C. Amengual, S. Barrachina, A. Castellanos, S. España, J. A. Gómez, J. Gorbe, A. Gordo, V. Palazón, G. Peris, R. Ramos-Garijo, and F. Zamora. The UJIpenchars database: A pen-based database of isolated handwritten characters. *Proceedings of the 6th International Conference on Language Resources and Evaluation, LREC 2008*, (January):2647–2651, 2008.

[13] Evie McCrum-Gardner. Which is the correct statistical test to use? *British Journal of Oral and Maxillofacial Surgery*, 46(1):38–41, 2008. ISSN 02664356. doi: 10.1016/j.bjoms.2007.09.002.

[14] Antigoni Mezari Mezari and Ilias Maglogiannnis. Gesture recognition using Symbolic Aggregate approximation and Dynamic Time Warping on Motion Data. ACM, 2018. doi: 10.1145/3154862.3154927.

[15] Daniel Puschmann, Payam Barnaghi, and Rahim Tafazolli. Adaptive Clustering for Dynamic IoT Data Streams. *IEEE Internet of Things Journal*, 4(1):64–74, 2017. ISSN 23274662. doi: 10.1109/JIOT.2016.2618909.

[16] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007. ISSN 15714128. doi: 10.3233/ida-2007-11508.

[17] Shingo Tamura, Keiichi Tamura, Hajime Kitakami, and Kaishi Hirahara. Clustering-based burst-detection algorithm for web-image document stream on social media. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pages 703–708, 2012. ISSN 1062922X. doi: 10.1109/ICSMC.2012.6377809.

[18] Xin Tao and Chunlei Ji. Clustering massive small data for IOT. *2014 2nd International Conference on Systems and Informatics, ICSAI 2014*, (Icsai):974–978, 2015. doi: 10.1109/ICSAI.2014.7009427.

[19] L J P van der Maaten and G E Hinton. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*, 9(nov):2579–2605, 2008. ISSN 1532-4435.

[20] Yu Wang, Yang Xiang, Jun Zhang, Wanlei Zhou, and Bailin Xie. Internet traffic clustering with side information. In *Journal of Computer and System Sciences*, volume 80, pages 1021–1036. Academic Press Inc., 8 2014. doi: 10.1016/j.jcss.2014.02.008.

[21] Donghua Yu, Guojun Liu, Maozu Guo, and Xiaoyan Liu. An improved K-medoids algorithm based on step increasing and optimizing medoids. *Expert Systems with Applications*, 92:464–473, 2018. ISSN 09574174. doi: 10.1016/j.eswa.2017.09.052.

A More information about the data sets

The data sets that are used in the experiments are described in section 6. This appendix provides more information about these data sets to the reader. The distribution of samples for both data sets described in section 6 is provided in subsection A.1. The data in both data sets were also used to compute a matrix of pair-wise distances of all pairs of sequences in the data sets and generate t-SNE [19] charts with this matrix to visualize the data in two dimensions. The t-SNE [19] charts for both data sets can be found in subsection A.2.

A.1 Distribution of samples for the data sets

The distribution of samples for the *UJI-PenCharacters* [7, 12] data set is as follows.

Class	W	O	2	5	9
Number of instances	44	66	22	22	22

Table 2: The distribution of samples for the *UJI-PenCharacters* [7, 12] data set.

The distribution of samples for the *GesturePebbleZ1* [14] data set is as follows.

Class	1	2	3	4	5	6
Number of instances	49	45	54	53	52	51

Table 3: The distribution of samples for the *GesturePebbleZ1* [14] data set.

A.2 t-SNE charts for the data sets

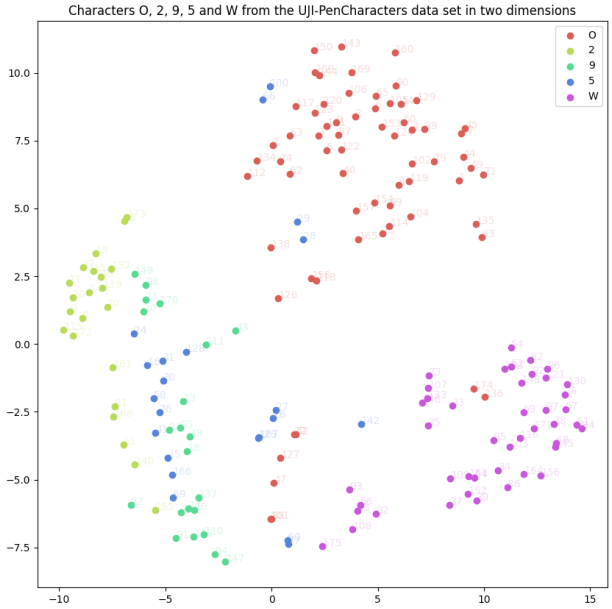

Figure 4: From the chart is concluded that the *UJI-PenCharacters* [7, 12] data set is quite noisy and that the clusters of classes *2*, *5* and *9* are sparse and very close to each other. The sequences for these clusters are scattered across half of the y-axis, which indicates that some sequences belonging to the same cluster are relatively far away from each other. This fact makes finding a good representation of the cluster that yields a relatively low distance to the cluster for all sequences belonging to that cluster quite a challenge. In conclusion, the *UJI-PenCharacters* [7, 12] data set is presumably rather difficult to cluster accurately due to the clusters of the classes *2*, *5* and *9*.

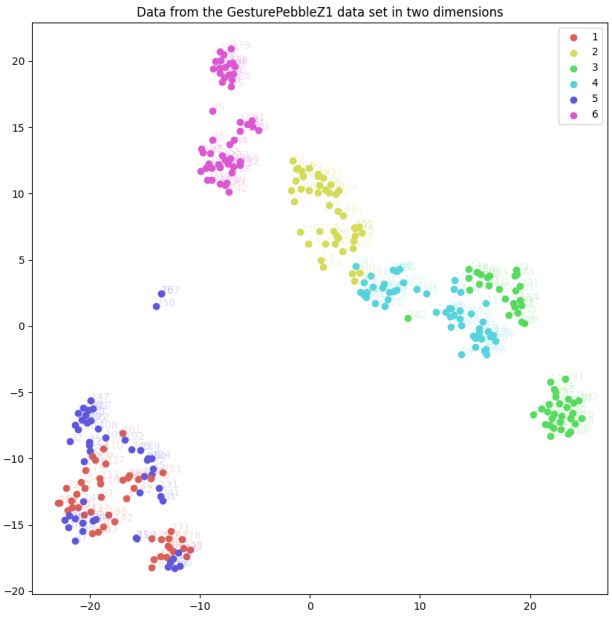

Figure 5: From the chart is concluded that the *GesturePebbleZ1* [14] data set contains clusters that are a lot denser and does not exhibit any noise. The clusters of classes *2*, *4* and *6* are likely easy to cluster since they do not overlap with any other clusters. The clusters of classes *1* and *5* are intertwined; hence most mistakes are probably made there. Moreover, part of class *3* is closer to the cluster of class *4* than it is to the rest of the cluster of class *3*, which might be an additional cause of mistakes. In conclusion, the *GesturePebbleZ1* [14] data set is presumably easier to cluster accurately than the *UJI-PenCharacters* [7, 12] data set due to clusters being denser and the data set is less noisy.

B Figures for clarification of some principles

This appendix presents some visuals that can clarify potential confusion after reading the contents provided in section 1, section 3 and section 4. The visuals can be found in the subsections below.

B.1 Problem with the online baseline algorithm

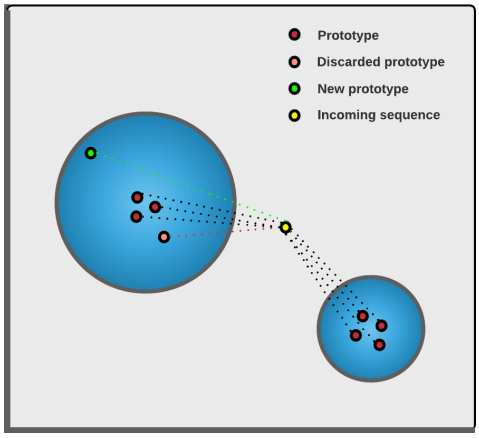

Figure 6: The figure shows the consequence of replacing a high-quality prototype with a low-quality prototype in two dimensions. The incoming sequence would be assigned to the left cluster if the distance computation included the discarded prototype instead of the new prototype, but is assigned to the right cluster instead, since the average distance from the incoming sequence to the right cluster is lower than the average distance from the incoming sequence to the left cluster with the new prototype that is a lot further away from the incoming sequence than the discarded prototype.

B.2 Problem with approximating the distance to a cluster

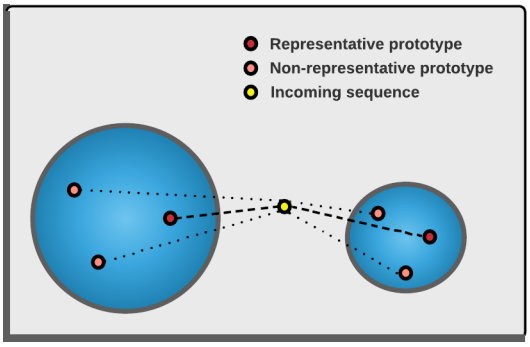

Figure 7: The distance between the incoming sequence and representative prototype of the left cluster is lower than the average distance between the sequence and all prototypes; this causes the sequence to be assigned to the left cluster, which is the wrong cluster considering the average distance to all prototypes of the right cluster is lower than the same distance for the left cluster.

B.3 Example of inability to model the centre of mass of a cluster

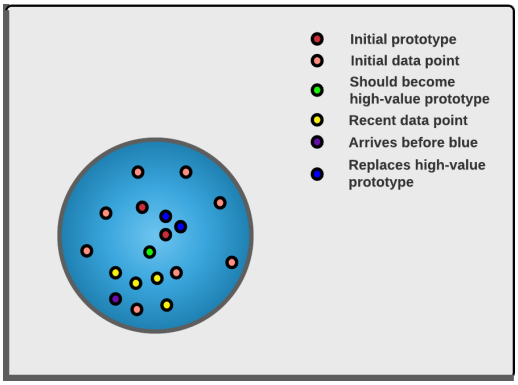

Figure 8: A hypothetical situation in which one of the prototypes is of higher value compared to the other prototypes due to very similar sequences being assigned to the cluster after initialization and improvement of the prototypes. The green sequence should become a high-value prototype, but is instead replaced by the blue sequences in the online baseline algorithm. The algorithm starts with the green and dark red sequences as initial prototypes, the order of arrival is yellow, purple and blue.

C Distribution of the research results as histograms

This appendix presents the distribution of some of the research results as histograms to show that the research results are not normally distributed and parametric assumptions are therefore not satisfied. The *Wilcoxon signed-rank test* and the *Mann-Whitney U test* are used instead of the *paired samples t-test* and the *independent samples t-test* for that reason [13]. The charts are generated from the samples of evaluation metrics that were generated from running the experimental variant *SeqClu-PV-A* on a baseline set of parameters. These samples are used in statistical tests for all three stages of the experiment described in section 6 and therefore provide solid proof that the parametric assumptions are not satisfied in the research results.

SeqClu-PV-A with baseline set of parameters

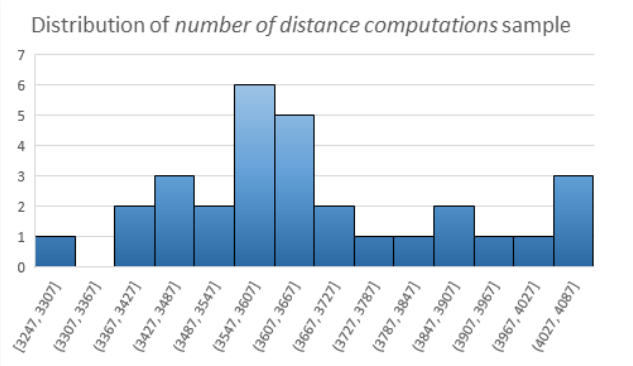

Figure 9: The distribution of the *number of distance computations* sample generated from 30 independent runs of the *SeqClu-PV-A* variant with a baseline set of parameters. The p-value resulting from the *Shapiro-Wilk test* [13] used to test if the data is normally distributed is 0.11.

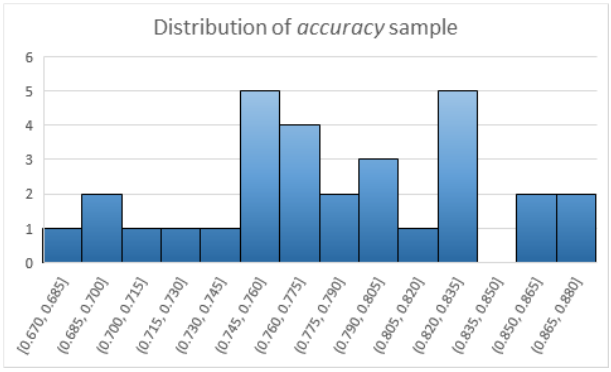

Figure 10: The distribution of the *accuracy* sample generated from 30 independent runs of the *SeqClu-PV-A* variant with a baseline set of parameters. The p-value resulting from the *Shapiro-Wilk test* [13] used to test if the data is normally distributed is 0.79.

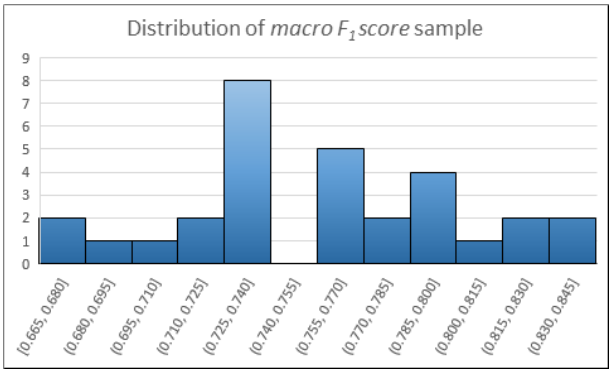

Figure 11: The distribution of the *macro* F_1 *score* sample generated from 30 independent runs of the *SeqClu-PV-A* variant with a baseline set of parameters. The p-value resulting from the *Shapiro-Wilk test* [13] used to test if the data is normally distributed is 0.75.

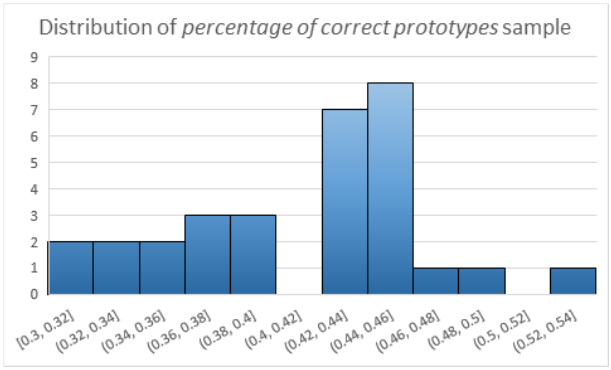

Figure 12: The distribution of the *percentage of correct prototypes* sample generated from 30 independent runs of the *SeqClu-PV-A* variant with a baseline set of parameters. The p-value resulting from the *Shapiro-Wilk test* [13] used to test if the data is normally distributed is 0.11.

D Experimental results as tables

This appendix presents the results of the experiments described in section 6. The section is divided up into three according to the three experiments described in section 6 and shows the results of every experiment in readable tables. A few abbreviations used in the tables are explained in table 1.

D.1 Performance of experimental variants of *SeqClu-PV*

Variant	Dist.	Acc.	F1	Prot.	App.↑	App.↓	Buff.↑	Buff.↓
PV-AB	**3768**	**0.83**	**0.80**	**0.44**	0.059	0.092	0.019	0.004
PV-A	**3662**	**0.78**	**0.76**	0.41	0.018	0.030	-	-
PV	**5630**	**0.82**	**0.80**	0.39	-	-	-	-
PV-B	**5760**	0.86	0.82	**0.42**	-	-	0.078	0.120
BL	*5580*	*0.86*	*0.82*	*0.38*	-	-	-	-

Variant	Dist.	Acc.	F1	Prot.	App.↑	App.↓	Buff.↑	Buff.↓
PV-AB	**8842**	**0.84**	**0.84**	**0.41**	0.019	0.004	0.086	0.033
PV-A	**8449**	0.81	**0.81**	0.39	0.018	0.018	-	-
PV	**12523**	**0.82**	**0.82**	0.38	-	-	-	-
PV-B	**12714**	**0.86**	**0.86**	**0.40**	-	-	0.084	0.047
BL	*12456*	*0.80*	*0.79*	*0.36*	-	-	-	-

Table 4: The upper part contains the results for the *UJI-PenCharacters* [7, 12] data set, the lower part contains the results for the *GesturePebbleZ1* [14] data set. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic row is the variant to which all other variants present in the table are compared with the *Wilcoxon signed-rank test* described in section 6. Bold values indicate that the underlying distributions for the corresponding metric of the two variants are significantly different. Dashes represent values that could not be determined for the given variant.

D.2 Effect of parameters on the performance of *SeqClu-PV*

Variant	Dist.	Acc.	F1	Prot.	Buff.↑	Buff.↓
PV-B (5)	**5662**	**0.84**	0.81	0.41	**0.113**	**0.205**
PV-B (10)	**5710**	0.84	0.81	0.41	0.076	**0.163**
PV-B (15)	*5760*	*0.85*	*0.82*	*0.42*	*0.078*	*0.12*
PV-B (30)	**5782**	0.85	0.81	**0.44**	0.064	**0.085**
PV-B (40)	**5949**	**0.84**	0.81	0.42	0.080	**0.092**

Variant	Dist.	Acc.	F1	Prot.	App.↑	App.↓
PV-A (0.0)	**3225**	0.78	0.76	0.41	0.014	0.022
PV-A (0.15)	**3483**	**0.81**	**0.78**	0.40	0.009	**0.009**
PV-A (0.3)	**3506**	0.79	0.76	0.41	0.010	0.019
PV-A (0.5)	*3662*	*0.78*	*0.76*	*0.41*	*0.018*	*0.030*
PV-A (0.7)	**5615**	**0.82**	**0.79**	**0.39**	**0.035**	**0.048**

Variant	Dist.	Acc.	F1	Prot.	App.↑	App.↓
PV-A (0.5)	**3883**	**0.80**	0.77	0.41	0.02	0.029
PV-A (1.0)	**3791**	0.79	0.76	0.41	0.019	0.021
PV-A (2.0)	*3662*	*0.78*	*0.76*	*0.41*	*0.018*	*0.030*
PV-A (3.0)	**3537**	**0.80**	0.77	0.42	0.013	0.019
PV-A (5.0)	**3563**	0.76	0.75	**0.39**	0.017	0.028

Table 5: All table parts contain results for the *UJI-PenCharacters* [7, 12] data set. The upper part varies in the parameter that defines the buffer size. The middle part varies in the parameter that determines the minimum representativeness. The lower part varies in the parameter that determines the importance of the representativeness respective to the weight of prototypes. The values for these parameters are contained in the variant name. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic row is the variant to which all other variants present in the table are compared with the *Mann-Whitney U test* described in section 6. Bold values indicate that the underlying distributions for the corresponding metric of the two variants are significantly different.

Variant	Dist.	Acc.	F1	Prot.	Buff.↑	Buff.↓
PV-B (5)	**12573**	**0.84**	**0.84**	0.40	0.096	**0.069**
PV-B (10)	**12641**	0.85	0.85	0.41	0.082	**0.065**
PV-B (15)	*12714*	*0.86*	*0.86*	*0.40*	*0.084*	*0.048*
PV-B (30)	**12944**	0.86	0.86	0.41	0.075	**0.032**
PV-B (40)	**13123**	**0.87**	**0.87**	0.40	0.08	**0.025**

Variant	Dist.	Acc.	F1	Prot.	App.↑	App.↓
PV-A (0.0)	**7408**	0.81	0.81	0.39	0.010	**0.02**
PV-A (0.15)	**7408**	0.80	0.80	0.40	0.010	**0.004**
PV-A (0.3)	**7439**	0.81	0.81	0.39	**0.010**	**0.006**
PV-A (0.5)	*8450*	*0.81*	*0.81*	*0.39*	*0.018*	*0.018*
PV-A (0.7)	**12527**	0.81	0.81	0.39	**0.088**	**0.054**

Variant	Dist.	Acc.	F1	Prot.	App.↑	App.↓
PV-A (0.5)	**9496**	0.82	0.82	**0.36**	0.029	0.026
PV-A (1.0)	**9427**	0.81	0.81	0.38	0.029	0.020
PV-A (2.0)	*8450*	*0.81*	*0.81*	*0.39*	*0.018*	*0.018*
PV-A (3.0)	**8062**	0.81	0.80	0.39	0.020	0.019
PV-A (5.0)	**7501**	**0.79**	**0.79**	0.4	0.010	0.016

Table 6: All table parts contain results for the *GesturePebbleZ1* [14] data set. The upper part varies in the parameter that defines the buffer size. The middle part varies in the parameter that defines the minimum representativeness. The lower part varies in the parameter that determines the importance of the representativeness respective to the weight of prototypes. The values for these parameters are contained in the variant name. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic row is the variant to which all other variants present in the table are compared with the *Mann-Whitney U test* described in section 6. Bold values indicate that the underlying distributions for the corresponding metric of the two variants are significantly different.

D.3 Effect of number of (representative) prototypes on the performance of *SeqClu-PV*

Variant	Dist.	Acc.	F1	App.↑	App.↓
PV-A (16/6)	**5600**	0.76	0.77	**0.001**	0.052
PV-A (12/6)	**5304**	0.77	0.76	0.009	0.016
PV-A (12/3)	**4176**	0.78	0.77	0.012	0.023
PV-A (8/6)	**4840**	0.80	0.77	0.012	**0.012**
PV-A (8/3)	*3662*	*0.78*	*0.76*	*0.018*	*0.030*

Variant	Dist.	Acc.	F1	App.↑	App.↓
PV-A (16/6)	**12967**	**0.90**	**0.90**	0.013	**0.001**
PV-A (12/6)	**12019**	**0.88**	**0.88**	**0.014**	**0.000**
PV-A (12/3)	**10276**	**0.87**	**0.86**	**0.043**	0.005
PV-A (8/6)	**10699**	0.81	0.81	**0.014**	**0.004**
PV-A (8/3)	*8450*	*0.81*	*0.81*	*0.018*	*0.018*

Table 7: The upper part contains the results for the *UJI-PenCharacters* [7, 12] data set, the lower part contains the results for the *GesturePebbleZ1* [14] data set. The variants differ in the number of (representative) prototypes. The values for these parameters are included in the variant name. The first value is the total number of prototypes, and the second value is the number of representative prototypes. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic row is the variant to which all other variants present in the table are compared with the *Mann Whitney U* described in section 6. Bold values indicate that the underlying distributions for the corresponding metric of the two variants are significantly different.

E Experimental results as charts

This appendix presents the results of the experiments described in section 6. The section is divided up into three according to the three experiments described in section 6 and shows the results of every experiment in readable charts. A few abbreviations used in the charts are explained in table 1.

E.1 Performance of experimental variants of *SeqClu-PV*

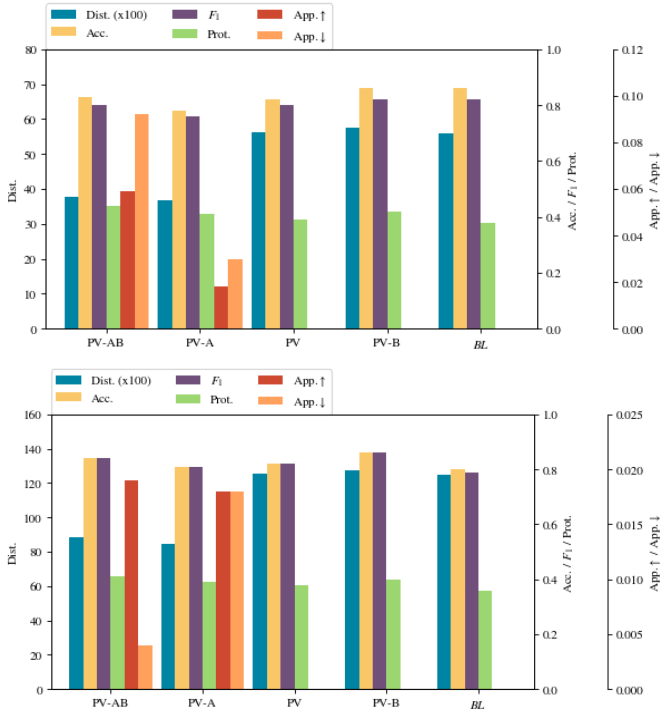

Figure 13: The upper part contains the results for the *UJI-PenCharacters* [7, 12] data set, the lower part contains the results for the *GesturePebbleZ1* [14] data set. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic variant is the variant to which all other variants present in the figure are compared with the *Wilcoxon signed-rank* test described in section 6, the results of which can be found in appendix D.

E.2 Effect of parameters on performance of *SeqClu-PV*

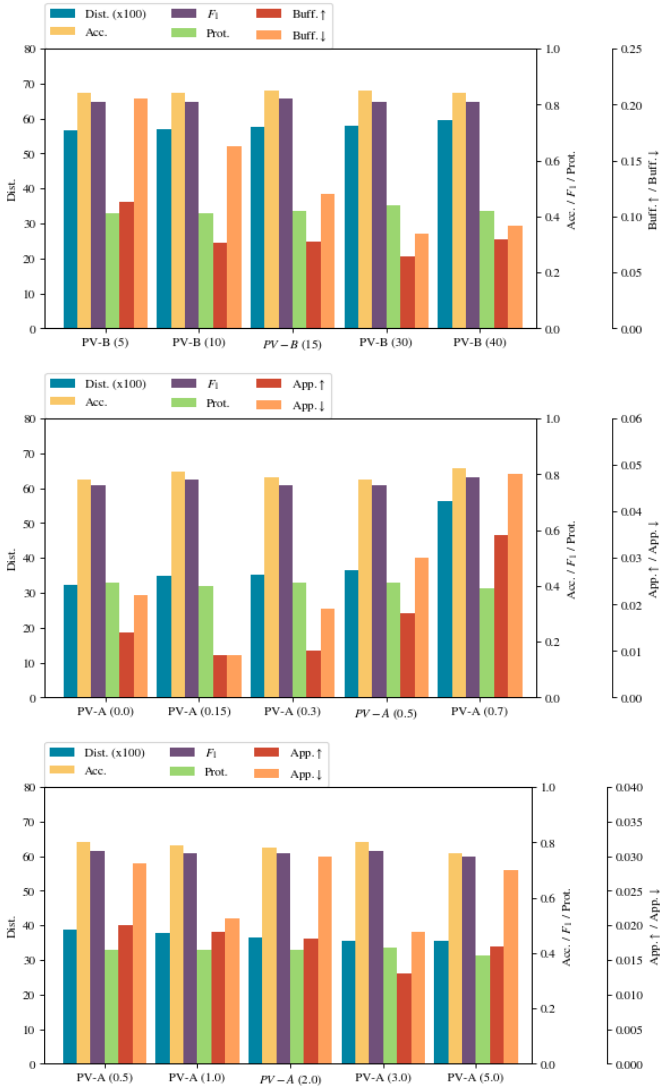

Figure 14: All charts contain results for the *UJI-PenCharacters* [7, 12] data set. The upper chart varies in the parameter that defines the buffer size. The middle chart varies in the parameter that defines the minimum representativeness. The lower chart varies in the parameter that defines the importance of the representativeness respective to the weight of prototypes. The values for these parameters are contained in the variant name. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic variant is the variant to which all other variants present in the figure are compared with the *Mann Whitney U* test described in section 6, the results of which can be found in appendix D.

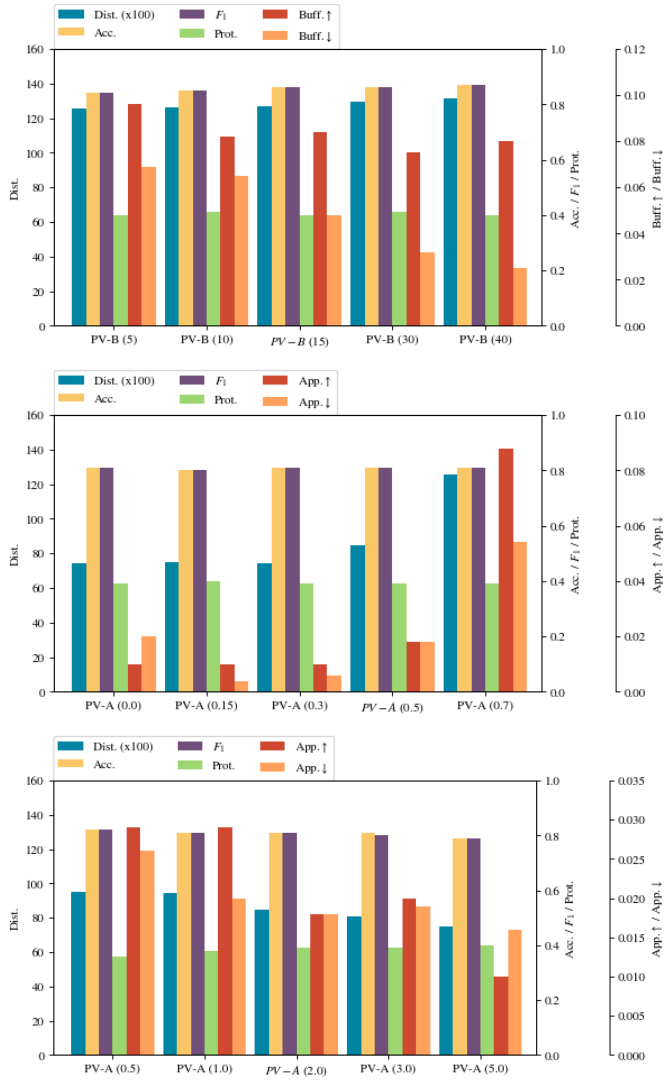

Figure 15: All charts contain results for the *GesturePebbleZ1* [14] data set. The upper chart varies in the parameter that defines the buffer size. The middle chart varies in the parameter that defines the minimum representativeness. The lower chart varies in the parameter that defines the importance of the representativeness respective to the weight of prototypes. The values for these parameters are contained in the variant name. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic variant is the variant to which all other variants present in the figure are compared with the *Mann Whitney U* test described in section 6, the results of which can be found in appendix D.

E.3 Effect of number of (representative) prototypes on the performance of *SeqClu-PV*

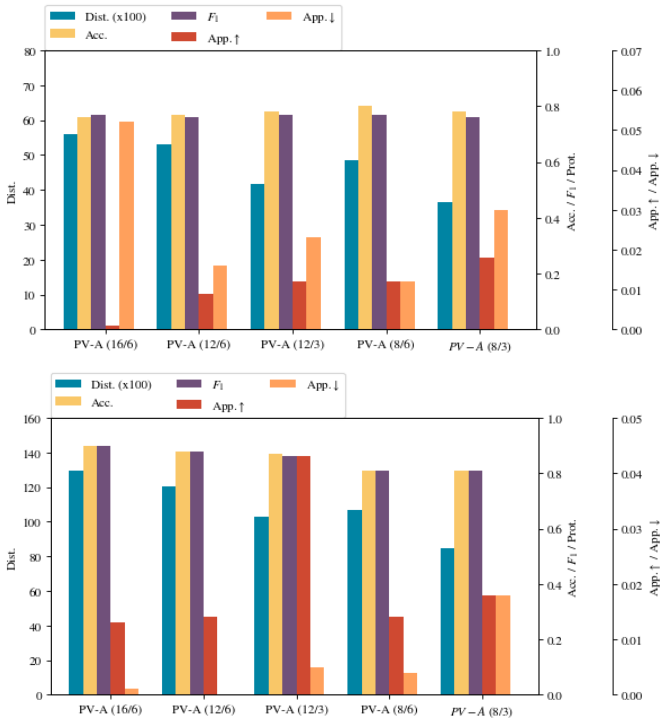

Figure 16: The upper chart contains the results for the *UJI-PenCharacters* [7, 12] data set, the lower chart contains the results for the *GesturePebbleZ1* [14] data set. The variants differ in the number of (representative) prototypes. The values for these parameters are contained in the variant name. The first value is the total number of prototypes, and the second value is the number of representative prototypes. The meaning of the abbreviated observed metrics can be found in table 1. The values are averages of the corresponding metric observed in 30 independent runs of some experiment. The italic variant is the variant to which all other variants present in the figure are compared with the *Mann Whitney U* test described in section 6, the results of which can be found in appendix D.