HUMBOLDT-UNIVERSITÄT ZU BERLIN

MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT

INSTITUT FÜR INFORMATIK

Evaluating Downsampling algorithms for visualization as a preprocessing step for Time Series Segmentation with ClaSP

Master Thesis Exposé

Melanie Wüstner - melanie.wuestner@student.hu-berlin.de

5. July 2024

First Supervisor:	Dr. rer. nat. Patrick Schäfer
Second Supervisor	Prof. Dr. Matthias Weidlich

Contents

1 Introduction

One real-world example of critical and highly interesting time series data are electrocardiograms. Considering the quite common scenario of a long-term or 24-hour cardiac monitoring, the diagnostic process of analyzing the monitored data can be very time consuming. Without any technical support the attending physician will have to review the whole series regarding the presence of any conspicuous event whereby missing one of those events may cause serious consequences for the patient. Especially cardiac events during sportive or high emotional activity can be hard to discover. The task of *Time Series Segmentation* can provide a reliable solution for this particular real-world example by preprocessing the monitored data such that physicians can be provided with already segmented series or a list of exact points where the monitored heartbeat pattern changes. By being provided these so-called Change Points the physicians tasks reduce from recognizing each possible event or change to analyzing only the beginning of each segment to evaluate if it represents a conspicuous event.

However, similar to many other machine learning tasks high accuracy time series segmentation algorithms degenerate in scalability regarding runtime and memory usage for large input time series. Especially for image data preprocessing steps, such as compression and downsampling, methods are used to reduce the memory consumption of data whilst keeping only important features to allow the respective machine learning task to still perform well regarding accuracy. In the field of time series downsampling is by now primarily used for visualization purposes. The time series segmentation tool ClaSP [1] works provenly well on its task in terms of accuracy but is in need of improvement in terms of runtime and memory usage. Exactly as for image data these two aspects can for example be improved by processing smaller data samples, which can be obtained using downsampling tools. Downsampling tools for visualization aim to provide a representation that visually still shows all important information of the original data like characteristic structure and significant peaks in the time series by selecting or aggregating significant data points. The motivation of this thesis is based on the assumption that visually characterizing information are similar to features needed for machine learning tasks. Practically spoken, if the reduced data visually still provides all characteristic information of the time series it also could be used for machine learning tasks like segmentation without trading too much performance regarding the accuracy of the model.

The main goals of this thesis are to first provide an overview of downsampling methods for time series visualization that potentially can be used as preprocessing step for the segmentation task. Using a selection of these methods it shall be analyzed and evaluated, if they are capable of solving the known issues of the segmentation tasks whilst keeping an accuracy not significantly lower, equal to or even higher than the accuracy performance achieved beforehand using the downsampling. For the evaluation purpose the time series segmentation tool ClaSP [1] will be used.

2 Background

2.1 Time Series Segmentation

To represent the general idea and process of segmentation, a time series can be defined as follows.

DEFINITION 2.1.1: A *Time Series* T is an ordered sequence of $n \in \mathbb{N}$ tuples consisting of an uni- or multivariate value v and a time component t, i.e., $T = \{(v_1, t_1), (v_2, t_2), \dots, (v_n, t_n)\}$ [7].

A value-time pair (v_i, t_i) of a time series can also be referred to as *datapoint* d. The above formal definition of a time series can be used to define a subsequence of a time series.

DEFINITION 2.1.2: A *Subsequence* $T_{i,j}$ is a part of a time series of length $j - i + 1$ consisting of one or multiple consecutive value-time-pairs, i.e., $T_{i,j} = \{(v_i, t_i), \dots, (v_j, t_j)\}$, where $1 \leq i \leq j \leq n$ [1], [7].

In the context of the time series segmentation tool ClaSP (see section 3.1) subsequences of a fixed length are also referred to as *windows*. The length of these windows as defined in *Definition 2.1.2* is called *window size*.

Assuming the time series was generated from a certain underlying process it maybe visually reflects state changes in this process. When a subsequence of a time series represents the exact period in between two state changes of the underlying process it is also called *Segment* in the context of time series segmentation.

DEFINITION 2.1.3: A *Segment* S is a disjunctive subsequence of a time series which includes all data points from one state change of the underlying process to the next one.

The offsets at which the state of the process changes in the time series and a segment begins or ends is also referred to as *Change Point*.

DEFINITION 2.1.4: A *Change Point* C is an annotation of the starting or ending point of a segment S and therefore represents a state change in the underlying process that generated the time series [8].

DEFINITION 2.1.5: A *Segmentation* TS of a time series in the context of change points into $k \in \mathbb{N}$ segments is an ordered sequence of $k - 1$ change points [1] such that $TS = \{C_1, C_2, \dots, C_{k-1}\}$ where $1 \leq k < n$.

DEFINITION 2.1.6: *Time Series Segmentation* is the machine learning task of finding a meaningful segmentation TS of a time series T. A segmentation in the context of *Definition 2.2.5* is meaningful when each change point represents a state change in the underlying process that generated the time series [1].

2.2 Downsampling

Downsampling is a data processing method, that aims to reduce the data points of some given original data by selection or aggregation of data points [2].

DEFINITION 2.2.1: Downsampling algortihms will be categorized into two groups in context of this thesis. Algorithms that use a selection of data points to reduce the resolution of the original data are referred to as *Selective Downsampling*. Algorithms using any kind of aggregation of data will be referred to as *Aggregative Downsampling*.

Downsampling algortihms therefore return a subset of the original data points that still represent the data as good as possible. Measures for the latter differ in the different application areas of downsampling. When used for visualization purposes, as often applied to time series, one common measure is a human visual comparison of the downsampled and the original data [3], [4]. In contrast, a measure called *Visual Representativeness* can be assessed, to achieve a metric-based evaluation.

DEFINITION 2.2.2: The *Visual Representativeness* of a downsampled time series is measured by their visual approximation to the original data, which can be measured using image-based metrics [2].

The reduction of the original data to the output resolution of a downsampling algorithm can be described using a factor, which in this thesis we will call *Compression Ratio*.

DEFINITION 2.2.3: The ratio by which the given input data points are reduced is called *Compression Ratio*. The compression ratio CR is calculated using the original number of data points $\#OD$ and the target resolution $\#TD$ of the downsampling algorithm, i.e., $CR = \frac{\#TD}{\#OD}$.

Generally, four major algorithms exist that downsample time series for visualization, which are used as base for extensions, improvements or merges.

EveryNth / The intuitive algorithm: A rather intuitive algorithm for selective downsampling is the EveryNth algorithm. As the name indicates the EveryNth algorithm selects every n^{th} datapoint from the original data. For this approach the parameter n, more precisely $\frac{1}{n}$ equals the compression ratio as defined previously [2].

MinMax: The MinMax algorithm separates the time series into multiple buckets or sub-sequences. For each bucket only the datapoints with minimum value v_{min} and maximum value v_{max} are preserved. Considering the bucket size B a time series of length n would be separated into $\frac{n}{B}$ buckets. After downsampling each bucket consist of only two datapoints $d_{min} = (v_{min}, t_{vmin})$ and $d_{max} = (v_{max}, t_{vmax})$, one of them containing the minimum and the other the maximum value of the original bucket together with their corresponding time component. The order of these datapoints is defined by the order of them in the original time series, i.e.

d_{min} and d_{max} can also be defined as d_1 and d_2, where depending on their index in the original time series d_1 may include the minimum or the maximum and d_2 the corresponding opposite. The resulting downsampled time series T' then consists of the datapoints of all downsampled buckets such that $T' = \{d_{11}, d_{21}, \dots, d_{1b}, d_{2b}\}$ where $b \in \mathbb{N}$ equals the bucket index with $1 \leq b \leq \frac{n}{B}$. While having a good memory complexity this method may fail on alternating behavior in the time series [2]. As for each bucket two values are preserved the target resolution of MinMax equals $2\frac{n}{B}$. The compression ratio of this algorithm therefore can be calculated using the formula $CR = \frac{2}{B}$, where $1 < B \leq n$, depending only on the bucket size.

M4: One major issue of MinMax can be found in the visualization of the downsampled time series. As this approach only preserved the minimum and maximum value of each bucket connecting lines of each two consecutive buckets may get lost in the process and need to be approximated. This approximation may lead to errors in the final visualization of the downsampled data. In 2014 Jugel et al. proposed a solution approach for this issue with the M4 [5]. In addition to the datapoints with the minimum and maximum value M4 preserves the starting and ending point of each bucket $d_{start,b} = (v_{tmin,b}, t_{min,b})$ and $d_{end,b} = (v_{tmax,b}, t_{max,b})$, resulting in four aggregates per bucket such that $T' = \{d_{start,1}, d_{11}, d_{21}, d_{end,1}, \dots, d_{start,b}, d_{1b}, d_{2b}, d_{end,b}\}$ considering the same index and value definitions as previously defined for MinMax. This approach shall preserve connections between buckets and therefor provide an improved visualization of the downsampled time series. For a bucket size $B \geq 4$ the compression ratio for M4 equals $CR = \frac{4}{B}$.

Largest Triangle: The general idea of the largest triangle algorithms is to assess the effective area of each data point in a time series and use it as the aggregation instead of minimum and maximum. The calculation of this effective area depends on the variation of the algorithm.

For the Largest Triangle One Bucket (LTOB) variant the effective area of each point equals the area size of the triangle that is formed with the two neighboring points. Afterwards as in MinMax and M4 the time series is split into a certain number of equally sized buckets [6].

Another variant is the Largest Triangle Three Buckets (LTTB) approach. LTTB addresses the question of LTOB, if the effective area of a data point can be restricted to its neighboring points. Instead of using only single neighboring points LTTB calculates the effective area using the previous and following bucket. The very first and last point are associated to the first and last bucket. For each bucket the point resulting in the highest effective area considering the selected point in the previous and next bucket shall be selected. The first point of the triangle, e.g. the point selected in the first bucket is always fixed, as it was calculated before. The second point is the one to be selected, the question arises how to determine the point to use of the third bucket. The intuitive approach is to brute force calculate the effective area for all possible combinations, which would be inefficient. A better approach is to fix a temporary point in the third bucket and use it for the calculation [6].

Finally, in both variants only one point per bucket having the highest effective area is selected. As for LTOB one point per bucket is selected the compression ratio equals $CR = \frac{1}{B}$. For LTTB also one point per bucket is selected, but additionally the first and last point of the time series are preserved. The splitting into buckets is done for the datapoints excluding the first and last one, resulting in a target resolution of $\#TD = \frac{n-2}{B} + 2$. The compression rate for LTTB equals accordingly $CR = \frac{n-2}{Bn} + \frac{2}{n}$ and therefore is, in contrast to the compression rate of the algorithms presented before, depending on the length n of the original time series.

3 Related Work

Although not much research has been done on applying downsampling algorithms as a preprocessing step on time series, there is related research on subtopics available. The subsections in this chapter include a time series segmentation tool called “ClaSP”, some existing downsampling algorithms for time series visualization as well as already existing benchmarks for time series segmentation.

3.1 ClaSP

In 2021 Schäfer et al. introduced a self-supervised method for time series segmentation called Classification Score Profile (ClaSP) [1]. The classification score profile is a function, where each local maximum represents an optimal position for a change point. To obtain this profile first the time series is split into virtual change points. Each of these splits is transformed into a binary classification problem by labelling all windows (as defined in section 2.1) extracted on the left side from the split with 0 and right from the split with 1. On certain extracted features of the windows the unsupervised k-Nearest-Neighbor classifier is trained to output a cross validation score. A high score is interpreted as high dissimilarity between the windows on the left and right of each split. All scores obtained for each split result in the final classification score profile, where each highest score, e.g. each maximum, indicates a change point. From the obtained change points by the classification score profile a segmentation of the time series can be provided as per *Definition 2.1.5*.

3.2 Downsampling algorithms

In section 2.2 a general categorization of downsampling algorithms for visualization of time series has been provided. Steinarsson [6] provided an overview of current and state of the art downsampling algorithms for visualization purposes in 2013 which is used as a starting point for the focus on downsampling algorithms. Several implementations of these algorithms already have been published, which will be used for the master thesis main objective. Van der Donckt et al. [9] published the Python library *tsdownsample* in 2023 which includes several optimized algorithms. A TypeScript library called *downsample* [10] including some state of the art downsampling algorithms for visualization can be found on Git.

Additionally, some extensions of the basic algorithms introduced in section 2.2 and novel approaches have been proposed. In 2017 Xu et al. [4] introduced a large-scale downsampling algorithm based on map-reduce programming. The idea of this implementation was to improve runtime issues of downsampling methods for large-scale time series. Franco et al. [3] addressed the issue of M4 to be only applicable to univariate time series. In 2022 they proposed an extension that made M4 available for multivariate data. Lastly, Donckt et al. [11] proposed a two-step hybrid implementation of the LTTB and MinMax approach (see section 2.2) in 2023 that extends the classic LTTB algorithm with improved scalability.

3.3 Benchmarks

The time series segmentation benchmark (TSSB) [12] has been constructed using datasets from the UEA & UCR time series classification repository and is provided via a Python library on GitHub. TSSB has already successfully been used with the time series segmentation tool ClaSP (see section 3.1). The benchmark currently contains 75 time series, each created by concatenating grouped time series by label. This benchmark therefore can be considered semi-synthetic [1]. Each time series in the concatenated group represents a segment of the complete time series. The offsets at which the complete time series has been concatenated are annotated as change points (see section 2.1).

Another more natural benchmark resulted out of the human activity segmentation challenge [13]. In a collaboration with students this benchmark was created by collecting and annotating human motion sensor data from the real world. The benchmark consists of 250 multivariate natural time series with activity segments and annotated change points.

Both benchmarks have annotated change points. They have already been successfully used for the experimental evaluation of the time series segmentation tool ClaSP [1].

4 Objectives

The overall main objective of this thesis is to evaluate the general possibility of using downsampling methods for time series visualization as a preprocessing step for the time series segmentation tool ClaSP and to evaluate the sense of purpose for the very same regarding runtime and memory usage whilst keeping improved or not significantly lower accuracy. From this main objective multiple categories of sub-objectives derive.

Overview of downsampling methods for time series: As a first step to evaluating downsampling methods as a preprocessing step on time series an overview shall be provided on the above mentioned existing downsampling methods for time series visualization purposes. The purpose of this overview is to give a general idea of already existing approaches, their performance and application areas, as well as it shall serve as a starting point to find methods, that can be evaluated regarding their applicability as a preprocessing step.

Parameter adjustments: With a selection of available methods comes a list of parameters that may have to be adjusted for the new purpose of the method. The first part of this objective is to have a closer look at the compression ratio (see section 2.2). Some time series of one dataset may not be as easy to compress as others, which derives the question if and down to which level the available methods allow the adjustment of the compression ratio. Another adjustable parameter is the window size in the created segments of a time series. For ClaSP an optimal window size has been figured out for the raw time series. The question arises if this optimal window size would still apply for downsampled data. For this matter it shall be evaluated, if and how the window size in the segments should be adjusted to suit the downsampled data.

Relation analysis: For this objective it shall be observed if the performance of the methods for visualization correlates with their performance as a preprocessing step. Shortly described it shall be figured out, if better performing methods in visualization are also performing better for preprocessing. In addition, if not, it may be analyzed if another correlation can be figured out. As one possible correlation, it is to be observed, if specific downsampling methods or categories perform better on certain kinds of time series and, if so, if these dependencies can be predicted based on characteristics of the time series and method categories.

Performance and evaluation measures: The last objective derives from the above relation analysis. In order to evaluate relations between or based on performances it first has to be analyzed if the measures used for visualization methods are also applicable to the new purpose as preprocessing. If the performance and evaluation measures of the visualization task of the methods does not apply to the new context of preprocessing, it is to be figured out, what measures can be used to achieve meaningful evidence of the performance of the methods in the new context. Apart from a general performance measure an additional measure must be found to weigh up possible runtime and memory usage improvements against accuracy loss.

5 Approach

To achieve the above objectives first research is to be done on current downsampling methods. This includes algorithms for time series visualization as well as recent advances and state of the art downsampling approaches as preprocessing for image data. The latter can provide a general insight into the possibilities and performance of downsampling as preprocessing. For the parameter adjustments different approaches for each parameter such as proportional adjustment will be tested and evaluated regarding their impact on the performance. Relation analysis will be done by categorizing downsampling methods and datasets into groups. For each pair of these groups correlations can be observed regarding the performance of ClaSP. Performance measures are assessed by figuring out the exact statement of each measure and evaluating them regarding their applicability on preprocessing purposes. The overall main goal of this thesis can be achieved in a two-step evaluation.

Step 1: First it needs to be proven that downsampled time series still contain enough significant information for the segmentation tool ClaSP to work keeping sufficient accuracy. This can be done by first splitting the original data at their actual labelled change points. Afterwards each split can be downsampled individually. Concatenating the downsampled splits and saving the offsets between each two splits as new true change points provides a labelled downsampled sample. ClaSP can now be executed on samples generated with this process, the results can be evaluated as usual. With the evaluation results of the first step it can be assessed, if ClaSP still performs generally well on selected features of the original data.

Step 2: As a second step it shall be evaluated if the change points from the original data can be derived from the downsampled representation. This can be achieved by adjusting the evaluation method of ClaSP slightly. First, the original data is downsampled and processed through ClaSP. The resulting predicted change points are now to be upscaled to the original data. One trivial approach to assess the change points of the original data is to upscale the change points of the downsampled data using the compression ratio as a factor. Further upscaling methodologies for the change points are to be discovered. The evaluation of the ClaSP model is now to be done on the upscaled change points from the downsampled data and the true change points from the original.

Each step is performed using a selection of different downsampling approaches and data sets. The final comparison is done based on memory usage, runtime and accuracy of ClaSP for each set of downsampling method and dataset. A critical distance diagram can be used to evaluate the significance of the obtained improvements or possible degradations.

6 References

[1] P. Schäfer, A. Ermshaus, und U. Leser, „ClaSP - Time Series Segmentation“, in *CIKM*, 2021, S. 10. doi: 10.1145/3459637.3482240.

[2] J. Van Der Donckt, J. Van Der Donckt, M. Rademaker, und S. Van Hoecke, „Data Point Selection for Line Chart Visualization: Methodological Assessment and Evidence-Based Guidelines“. arXiv, 3. April 2023. doi: 10.48550/arXiv.2304.00900.

[3] J. Franco, A. Garcia, und A. Gil, „Multivariate Adaptive Downsampling Algorithm for Industry 4.0 Data Visualization“, in *16th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2021)*, H. Sanjurjo González, I. Pastor López, P. García Bringas, H. Quintián, und E. Corchado, Hrsg., Cham: Springer International Publishing, 2022, S. 588–597. doi: 10.1007/978-3-030-87869-6_56.

[4] J. Xu, Y. Qiu, H. Zhang, M. Li, und M. Li, „Large-scale time series data down-sampling based on Map-Reduce programming mode“, in *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, März 2017, S. 409–413. doi: 10.1109/IAEAC.2017.8054047.

[5] U. Jugel, Z. Jerzak, G. Hackenbroich, und V. Markl, „M4: a visualization-oriented time series data aggregation“, *Proc. VLDB Endow.*, Bd. 7, Nr. 10, S. 797–808, Juni 2014, doi: 10.14778/2732951.2732953.

[6] S. Steinarsson, „Downsampling Time Series for Visual Representation“. Háskólaprent, Fálkagata 2, 107 Reykjavík, 1. Juni 2013. Zugegriffen: 17. Mai 2024. [Online]. Verfügbar unter: https://www.semanticscholar.org/paper/Downsampling-Time-Series-for-Visual-Representation-Steinarsson/37fd11c7c644a6b398cd388727ee2e58d71bc6d4

[7] M. Lovrić, M. Milanović, und M. Stamenković, „Algoritmic methods for segmentation of time series: An overview“, *J. Contemp. Econ. Bus. Issues*, Bd. 1, Nr. 1, S. 31–53, 2014.

[8] S. Aminikhanghahi und D. J. Cook, „A survey of methods for time series change point detection“, *Knowl. Inf. Syst.*, Bd. 51, Nr. 2, S. 339–367, Mai 2017, doi: 10.1007/s10115-016-0987-z.

[9] J. Van Der Donckt, J. Van Der Donckt, und S. Van Hoecke, „tsdownsample: high-performance time series downsampling for scalable visualization“. arXiv, 5. Juli 2023. doi: 10.48550/arXiv.2307.05389.

[10] J. J. Naništa, „janjakubnanista/downsample“. 23. Mai 2024. Zugegriffen: 31. Mai 2024. [Online]. Verfügbar unter: https://github.com/janjakubnanista/downsample

[11] J. Van Der Donckt, J. Van Der Donckt, M. Rademaker, und S. Van Hoecke, „MinMaxLTTB: Leveraging MinMax-Preselection to Scale LTTB“. arXiv, 29. April 2023. doi: 10.48550/arXiv.2305.00332.

[12] A. Ermshaus, „ermshaua/time-series-segmentation-benchmark“. 21. Mai 2024. Zugegriffen: 22. Mai 2024. [Online]. Verfügbar unter: https://github.com/ermshaua/time-series-segmentation-benchmark

[13] A. Ermshaus *u. a.*, „Human Activity Segmentation Challenge @ ECML/PKDD’23“, in *Advanced Analytics and Learning on Temporal Data*, G. Ifrim, R. Tavenard, A. Bagnall, P. Schaefer, S. Malinowski, T. Guyet, und V. Lemaire, Hrsg., Cham: Springer Nature Switzerland, 2023, S. 3–13. doi: 10.1007/978-3-031-49896-1_1.