16th International Conference on Space Operations, Cape Town, South Africa – 3 - 5 May 2021.

SpaceOps-2021,5,x1248

To Catch Them All: A Generic Approach for Pattern Detection in Time Series Satellite Telemetry Data

Clemens Schefels[a]*, Leonard Schlag[b], Lukas Dreier[c], Lucas Lincoln[c], Markus Steinbach[c]

[a] *German Space Operations Center, Deutsches Zentrum für Luft- und Raumfahrt e.,V., German Aerospace Center Münchener Straße 20, 82234 Weßling, Germany*, clemens.schefels@dlr.de
[b] *German Space Operations Center, Deutsches Zentrum für Luft- und Raumfahrt e.,V., German Aerospace Center Münchener Straße 20, 82234 Weßling, Germany,* leonard.schlag@dlr.de
[c] *Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany*
* Corresponding Author

Abstract

This paper presents a tool that enables automatic recognition of patterns in historical satellite telemetry datasets. This tool uses pattern-matching techniques to identify substantially similar patterns to a user selected interval within a time series. In this sense, the software serves as a powerful data exploration tool for datasets which are too large for manual inspection. The system is designed to be robust to signal types, incomplete coverage, and inconsistent sampling, which are common issues with telemetry data. In detail, three distance-based algorithms, namely, Discrete Wavelet Transform, Dynamic Time Warping, and Adaptive Piecewise Constant Approximation, two probabilistic algorithms, namely, Gaussian Mixture Model, Hidden Markov Model, as well as an ensemble approach are implemented to cover and explore a wide variety of different pattern detection techniques. The system is evaluated on real satellite data and on the UCR (University of California, Riverside) Time Series Classification Archive.

Keywords: Telemetry, Time Series, Satellite, Data Analysis, Pattern Matching

Nomenclature

A = set of all possible compact subsets of a time series
A_l = subset of length l
A_p = set of all compact subsets in A of duration p
D = time series, tuple (x_i,t_i)
Δ = similarity measures
η_Δ = quality mapping
$\mathscr{L}_2$ = Euclidean distance
Q = query
S = sequence
t = time values
τ = quality threshold
x = value measurements

Acronyms/Abbreviations

Area Under ROC (AUROC),
Adaptive Piecewise Constant Approximation (APCA),
Deutsches Zentrum für Luft- und Raumfahrt / German Aerospace Center (DLR),
Discrete Fourier Transformation (DFT),
Discrete Wavelet Transform (DWT),
Dynamic Time Warping (DTW),
Gaussian Mixture Model (GMM),
German Space Operations Center (GSOC),
Hidden Markov Model (HMM),
Receiver Operating Characteristic (ROC),
University of California, Riverside (UCR),

1. Introduction

Nowadays, modern satellites are equipped with more and more sensors to keep track of satellites system status. For example, the two GRACE Follow-On satellites, operated by the German Space Operations Center (GSOC) at the German Aerospace Center (DLR), each collects about *80,000* housekeeping parameters many of which have to be inspected by the system engineers on a regular basis. Seeking for novel or specific behavioral patterns in these time series is a common task of system engineers. Based on these patterns, the engineers can evaluate the health status of the satellite, detect anomalies in its telemetry data, conduct predictive analyses, and investigate errors and system failures. Often, operational decisions are derived from these analyses and may result in changes in the scheduled routine of the satellite operations. The automatic identification of previous occurrences of a specific event or anomaly, through the recognition of the same pattern in the telemetry data, is essential to improve the efficiency of such analyses and speed up the workflow of satellite operations.

This paper presents a tool that uses pattern-matching techniques to identify substantially similar patterns to a user selected interval within a time series. In this sense, the software serves as a powerful data exploration tool for datasets which are too large for manual inspection. To present the system engineers with one generic tool that can be used on a wide variety of inhomogeneous telemetry data, five unique algorithms for pattern matching and one ensemble approach are included. In detail, three distance-based algorithms, namely, Discrete Wavelet Transform (DWT), Dynamic Time Warping (DTW), and Adaptive Piecewise Constant Approximation (APCA), as well as two probabilistic algorithms, namely, Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), are implemented to cover and explore a wide variety of different pattern detection techniques. Distance-based approaches include methods such as subsampling/filtering and comparing the Euclidean distance as well as more complex routines such as DTW or comparing distance in a Wavelet subspace. Probabilistic approaches represent the pattern as a model and measure the likelihood of a given test time-range coming from the same model. For the ensemble approach, the DTW, APCA, and GMM implementations, based on their promising and complementary performance on the test datasets, are combined.

The final evaluation of the system took place in two stages: first, a parametric study on the UCR (University of California, Riverside) Time Series Classification Archive [3] was conducted to test different settings for each algorithm. With this analysis, a set of optimal parameters was identified for each algorithm and insight into how to improve the algorithms and approaches was gained. Second, the algorithms were applied to satellite data to evaluate their performance and demonstrate the suitability of the tool for daily use.

In the future, this tool will be included into ViDA, the web-based platform for telemetry visualization and data analysis developed at GSOC.

This paper is structured as follows: Section 1 is this introduction. Section 2 is devoted as well to the analysis of the requirements for the software tool as to the formalization of the pattern recognition problem. Section 3 describes the initial statistical data exploration phase, followed by Section 4 with the literature review. Next section, Section 5, introduces the pattern matching workflow that is used by the software tool. Within Section 6, we evaluate our software tool and describe the results in Section 7. Section 8 is a conclusion and an outlook on future work.

2. Requirements Analysis and Problem Formalization

The goal of this project is a proof-of-concept prototype which enables the system engineers to more efficiently perform the aforementioned duties, specifically by providing a system which enables automatic recognition of events in the historical dataset which are substantially similar to a manually selected event.

2.1 Requirements Analysis

During the collection of the requirements for the pattern matching system, the following key challenges have been identified.

First, the large number of parameters: each satellite may provide tens of thousands of different time series data streams. Moreover, each parameter from a satellite may have different sampling rates, resolutions, types, or other characteristics. Additionally, different satellites may have a different set of parameters; and satellites reporting the same parameters may do so inconsistently (i.e., the resolution and sampling rate of parameter α from Satellite A is not, in general, the resolution and sampling rate of parameter α from Satellite B).

Second, the satellites have to deal with an inhomogeneous operating environment: the satellites experience a wide variety of conditions–there are regular periodic and diurnal variations as well as irregular variations due to solar activity, weather, sensor and power conditions, and other factors. There may also be updates to the system firmware, software, and operating modes over time. In practice this means that each period in the database matching a novel period λ may have different sampling rates, data representations, noise characteristics, bias, and coverage (data may be incomplete).

With these challenges in mind, the requirements of the pattern matching system are as follows:

- Input: a period of data of a user selected duration from a single parameter (i.e., the origin time series), the start time and end time of the period determined by the user (i.e., the origin pattern).
- Output: based upon the input novel period, a list of periods which are similar to this novel period based on some metrics. The returned periods should be identifiably similar to the input period by human inspection.
- Users: technically proficient personnel, therefore, the system's methods should be understandable to such a userbase and not appear to said personnel as a black box.
- Resulting system: it is targeting future adoption into our DLR workflow which has an existing UI, plotting system, and database access systems.

2.2 Formal Statement of Problem

The problem to find (all) occurrences of a selected pattern in a time series, is formalized as follows: Given a sequence of N value measurements $\{x_1,x_2,...,x_n\} \in \mathbb{R}^N$ and an associated sequence of N strictly monotonically increasing time values $\{t_1,t_2,...,t_n\}$, the sequence of N tuples $D := \{(t_i, x_i)\}$ is called a *time series* and can be partitioned into (potentially overlapping) compact subsets arbitrarily. This set of all possible compact subsets is called A.

Furthermore, we define A_p as the set of all compact subsets in A of duration p. The most general problem is, given a user-selected query subset $Q \in A$, to find the most similar matching subsets from A, where similarity is best defined by human intuition considering the question "*are these two sequences similar?"*.

The time series is consistently sampled if $t_{k+1}-t_k$ is constant $\forall k \in [1, N]$, and inconsistently sampled otherwise. Moreover, the time series has incomplete coverage if it is otherwise consistently sampled but is missing periods of data. In general, data from satellites may be consistent- or inconsistently sampled; and often has incomplete coverage.

Though some intuition of a similar pattern would allow patterns to be stretched or squeezed to different durations, we will only investigate *strict-duration* queries, in which we attempt to match regions of the same duration as the query. That is to say: Given a subset $Q \in A$ of length l, we perform a strict-duration query if we seek matching subsets only in A_l, that is, in regions of equal length to the query Q.

2.2.1 Similarity

We wish to return the most-similar regions from our database to a query sequence Q. What makes a region similar? Is a region I similar to query Q? Similarity can mean different things to different people, or even to the same person at different times, and we must remain aware of these facts as we attempt to define some measures Δ such that a sequence S is similar to $Q \text{ if } \Delta(\vec{Q},\vec{S})<\varepsilon \text{ for some } \varepsilon$.

The challenge here is twofold: We must determine what an appropriate measure is Δ, and in addition we must have a match quality in order to rank the matches; allowing the return of only the most-similar matches. This match quality of (Q, S) is the result of a function $\eta_\Delta(Q, S) \in [0, 1]$ which should have the following properties:

- $\eta_\Delta(Q, Q) \approx 1$
- $\eta_\Delta(Q, S) > \eta_\Delta(Q, T)$ if Q is more similar to S than it is to T

We must determine the appropriate and quality mapping η_Δ for measure Δ such that only intuitively similar sequences achieve a high match quality. First, we will discuss similarity measures Δ.

2.2.2 Similarity Measures

In general, we have two types of similarity measures to consider: $\mathscr{L}_p$-type distance measures and probabilistic measures.

$\mathscr{L}_p$-type Distance Measures: A common measure of the distance (or inversely, similarity) between two signals is an $\mathscr{L}_p$ norm:

$$\Delta_{\mathscr{L}_p}(\vec{x},\vec{y})=\left(\sum_{i=1}^{N}|x_i-y_i|^p\right)^{\frac{1}{p}} \qquad (1)$$

Though there is potential value to using a $\mathscr{L}_1$ norm for robustness against impulsive noise, the $\mathscr{L}_2$ Euclidean distance is optimal for Gaussian, independent and identically distributed measurement error [1]. Given the popularity of the Euclidean distance in time series matching literature and provided its optimality, we selected to use it throughout our distance-based approaches. We consider in all cases a bias-compensated Euclidean distance, in which we compare the query *(Q)* and sequence in question *(S)* after compensating each by their mean values:

$$\widetilde{\Delta}(Q,S)=\Delta_x(Q-Q_{avg},S-S_{avg}) \qquad (2)$$

with

$$Q_{avg}=\frac{1}{n}\sum_{i=1}^{N}Q_i,\ S_{avg}=\frac{1}{n}\sum_{i=1}^{N}S_i \qquad (3)$$

and Q_i, S_i the ith component of Q and S, respectively.

Match Quality: The bias compensated $\mathscr{L}_2$ measure provides an unbounded measure of dissimilarity. Practically, we wish to work with a metric which has been normalized into a perfect score *(1.0)* for perfectly similar signals and a *0*-score for completely non-matching signals. We have designed a mapping from Euclidean Distance to Match Quality which captures the intent of our solution.

We wish that a perfect signal–one identical to the query–equals *1.0* $(\eta_{\mathscr{L}_2}(Q,Q)=1.0)$. The Euclidean distance between these signals is *0* $(\Delta_{\mathscr{L}_2}=0)$, therefore it is suggested to have a mapping of the form

$$\frac{f(Q)-\Delta_{\mathscr{L}_2}}{f(Q)} \qquad (4)$$

with *f(Q)* being some arbitrary function–examples may be the maximum or mean functions. We can select the most appropriate function using the knowledge of our end-use case.

There is in general no bound to the amount of distance a signal can be from the query signal, given that each signal has a different value range we cannot make assumptions about the maximum and minimum values of a signal. However, it is not required that a *0.0* be assigned only to the worst possible match; only that any match with a *0.0* would under no conditions be considered similar by a ground engineer. Phrasing that in a manner which we can translate to our mapping defines *f(Q)* to be the norm of *Q*, and the final quality metric is:

$$\frac{\left(\sum|Q_i|^2\right)^{\frac{1}{2}}-\left(\sum|Q_i-S_i|^2\right)^{\frac{1}{2}}}{\left(\sum|Q_i|^2\right)^{\frac{1}{2}}} \qquad (5)$$

This quality metric operates nicely in that it scales appropriately with signals of differing ranges. It is a measure of how much error between signals there is in relation to how large the signal itself is.

Probabilistic Distance: In general, we introduce how probabilistic models work in 5.2.2. Nevertheless, we will introduce the similarity context in probabilistic terms. Assume we have two realizations which are drawn from a probabilistic model. We fit a model for one of these realizations with the maximum likelihood method. From this we get an estimation of the hyperparameters for the assumed probabilistic model.

Next, we wish to check whether the underlying model is also appropriate for other realizations. Typically, this is done with a likelihood ratio test, but due to its complexity and computational cost we implemented a different but similar method.

In [2], the authors fit several models and calculate the respective likelihoods for queried time series. Afterwards, they decide based on the likelihood of the fit model whether it is appropriate or not. In our case, we calculate the likelihood for the second realization (region of interest) based on the query's underlying model. Now we compare the ratio of likelihoods determine if the assumption (that both realizations rely on the same underlying model) is true or not. This ratio of the likelihoods gives us a measure of similarity. If this ratio is large than the model represents the second realization very well. If it is small the model fitted for the first realization does not represent the one in question.

Match Quality: This ratio is used directly as the match quality. The quality could exceed *1* since it is possible that the second realization is represented better by the model than the first one which only indicates that the realization is represented very well by this underlying probabilistic model. We ultimately use as a match quality the maximum of the described ratio or *1.0*.

3. Statistical Data Exploration

Before researching applicable algorithms and evaluating benchmarks, we selected representative parameters highlighting the broad range of behaviors satellite telemetry can show. In an initial statistical data exploration phase, we familiarized ourselves with the data and distilled the key challenges it poses w.r.t. pattern matching approaches. This phase concluded with the description of various test cases used to evaluation.

In addition to these parameters, a publicly available dataset [3] of labeled time series data was utilized for parametric optimization of the algorithms. This dataset will not be described in this section, see [3] for details regarding this dataset. The subset of classified data used from this dataset were EGC200, ECGFiveDays, FordA, OliveOil, PowerConds, and Computers.

Table 1. Statistics over analyzed parameters.

Name	Duration	Unique Values	Max	Min	Std	Mean	Mean Frequency
Param. 1	250 days	333	2457.00	0.00	139.29	277.47	1.01Hz
Param. 2	250 days	116	2457.00	0.00	10.28	324.25	1.01Hz
Param. 3	346 days	1296	1715.00	259.00	97.92	838.60	0.02Hz
Param. 4	393 days	2836122	3.26	-3.14	1.81	-0.00	0.10Hz
Param. 5	348 days	29946733	2.14e+09	-2.14e+09	1.07e+09	3.41e+04	1.00Hz
Param. 6	806 days	692	-5.00	-2806.00	37.09	-929.82	1.00Hz
Param. 7	366 days	150	346.00	31.00	118.74	152.38	0.65Hz
Param. 8	826 days	151	2865.00	2715.00	24.99	2788.57	0.14Hz
Param. 9	826 days	2	120.00	30.00	18.40	116.07	0.33Hz
Param. 10	365 days	1489	2584.00	1092.00	165.10	1799.98	1.00Hz
Param. 11	365 days	79	1.41e+09	5.78e+08	2.68e+07	1.19e+09	1.00Hz
Param. 12	365 days	11	20.00	0.00	0.39	17.00	1.19Hz

3.1 Data Acquisition

In total, twelve telemetry parameters from one of GSOCs active missions were chosen for further analysis and evaluation of the pattern detection methods. The time series for each parameter include both high frequent telemetry acquired during ground station contacts as well as lower frequent data from processed dumps. The parameters include, e.g. pressure and reaction wheel friction sensors, voltages, temperatures as well as mode and status flags. In total, the time series combined duration is over *5,500* days containing more than *300,000,000* data points.

3.2 Data Preprocessing

In the scope of this project, the conscious decision to not apply any preprocessing in the form of cleaning or filtering the data was taken. The unmodified data best fits the end use case as the researched methods should be useable by ground engineers. Not only do engineers also work on unfiltered raw data most of the time, but problems in the data might very well describe the pattern which they are looking for.

3.3 Statistical Data Exploration

Before looking for suitable algorithms, the key features and difficulties stemming from the heterogenous data typical in our field were analyzed. An overview is given in Table 1, showing the wide range of used parameters. Both high and low precision float parameters, e.g. *Param. 4* and *Param. 7* as well as parameters representing discrete entities such as status flags or booleans, e.g. *Param. 9* and *Param. 12*, are present.

In addition, another important aspect of telemetry is the irregular sampling rate. The decimal values of the mean frequency in Table 1 hints at this feature. For low earth orbit satellites, the sampling rate can change over time depending on e.g. the mission phase, planned tasks and whether there is a contact or not. For some algorithms, an irregular sampling rate can pose tremendous challenges as they are dependent on equidistant data points as their input.

3.4 Identified Challenges

Overall, the analysis of the time series led to the identification of six key challenges algorithms will face when recognizing patterns in satellite telemetry described in Table 2. In addition to the statistical analysis, the identified challenges also stem from the envisioned use case. Optimally, engineers should be able to find any kind of pattern in past telemetry, no matter the shape, duration or type of telemetry parameter.

Table 2. Identified challenges.

Inconsistent sampling	Variable pattern duration
Instantaneous events	Low Pattern-to-noise-ratio
Discrete or continuous signals	Patterns characterized by frequency variation

3.5 Testcase Definition

After a qualitative exploration of the data, a labelled dataset is required in order to evaluate the performance and quality of each algorithm and compare various implementations. The labelled dataset was manually crafted using a custom tool specifically developed for this purpose. It allows us to label various patterns in our sample data and at the same time evaluate the performance of the implemented algorithms w.r.t. the labelled patterns. Based on the aforementioned challenges, and using the developed framework, we labeled *776* regions over *26* testcase patterns from the *12* telemetry parameters.

4. Literature Review

From literature review, there are three different pattern matching approaches that have to fulfill our requirements, discussed in Section 2:

- Distance-based approaches, and would include basic methods as subsampling/filtering and comparing the Euclidean distance as well as more complex routines such as Dynamic Time Warping [4] or comparing distance in a Wavelet subspace [5].
- Probabilistic approaches, which represent the pattern as a model and measure the likelihood a given test range comes from the same model (such as Hidden Markov Models [6] or Gaussian Mixture Models [7]
- Symbolic approaches, which transform the time series to a string of symbols and compare string similarity to measure distance. These are frequently used in bioinformatics, language processing, and streaming and can be very fast given some prior knowledge of the dataset at hand [8], [9], [10]. Because we desire a robust, general matching system without prior training, we did not pursue symbolic approaches in this work.

We will discuss in further detail Dynamic Time Warping, Probabilistic Methods, and the APCA algorithm (see Section 5.2). First, we may further decompose a time series matching workflow (including any of the aforementioned algorithms) into a set of tasks. Though not a perfect taxonomy, it serves as a useful framework for both discussing the following work and for structuring the software module.

5. Pattern Matching Workflow

An overview of the time series pattern matching workflow is shown in Fig. 1; with a few examples for each task of the process. The Acquisition task is outside the scope of this work and may differ from project to project. For our purposes, data is contained in simple text-files, each containing a duration of data on the order of 1-2 years. The preprocessing, scanning, measuring, and postprocessing are now discussed.

The characterization of the different types of time series matching algorithms is mainly based on the different measures of similarity. The main idea is to take a suitable measure for the representation of the time series. The overview of time series matching workflow shows clearly that different tasks can be combined in several ways, i.e. use a wavelet decomposition before apply a probabilistic algorithm. This gives the user several possibilities to customize the algorithms.

One may notice that we neglected advanced machine learning methods like deep learning. We want to have a high explainability of the model which is not guaranteed in advanced methods. Approaches such as deep learning would have offered a bunch of new possibilities and can be studied in a separate project.

5.1 Workflow Tasks

In this section, we will introduce the workflow of the time-series pattern matching that is followed by our software tool. The single tasks will be discussed in detail.

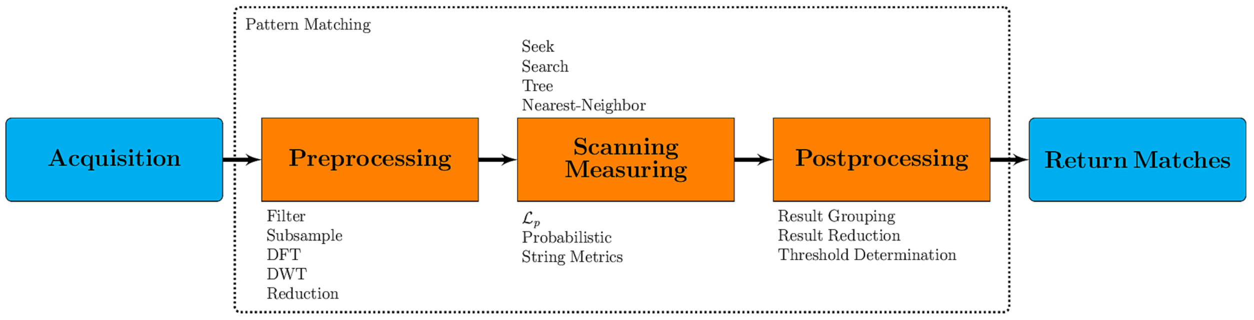

Fig. 1. A breakdown of the tasks (in orange and inside of the dotted box) of the time-series pattern matching workflow, with examples of potential methods for each task.

5.1.1 Preprocessing

We consider preprocessing to include not only traditional data cleaning operations but also fundamental transformation to the data, and therefore part of the matching algorithm as a whole. Besides well-known preprocessing methods such as normalizing and standardizing, the transformation to a new representation such as the Discrete Wavelet Transformation (DWT) or Discrete Fourier Transformation (DFT) are included in the preprocessing task. The mapping to a representation as a probabilistic model is also contained here and it demonstrates that different approaches (like DTW and a probabilistic method) can be combined using our multi-task approach.

5.1.2 Scanning and Measuring

There are two fundamental ways in which one can search through a time series data set to evaluate quality of each potential matching subset. First, let us define our terminology for this section.

Given the set of A_p compact subsets of length A from time series $D := (t_i, x_i)$ (see Section 2.2), it is obvious to assert that each subset of A_p may be uniquely identified (indexed) by its lowest contained timestamp t_i and that we may sort them to be in monotonically increasing order in time:

$$\forall a_j, a_k \in A_p \quad j,k \in \mathbb{N}^+ \quad j<k : min_{t_i} a_j < min_{t_i} a_k$$

Note that this is not true for A_p in general, wherein elements must be identified by at least two components, for example starting time and length. This is a simplification made possible by restricting to a strict-duration query. Given strict-duration query we say that we search for matches to query Q in A_l if we evaluate the match quality of at least a substantial subset of A_l against Q. We seek for matches in A_l if we systematically evaluate $\eta_\Delta(Q, a_j)$ in monotonically increasing order – $\eta_\Delta(Q, a_j+1)$ is evaluated after $\eta_\Delta(Q, a_j)$.

A seek can be performed in reverse order (newest timestamps to oldest) or trivially parallelised by partitioning the series into non-overlapping sections.

To note a few important distinctions: Seeking is clearly a type of search, and is the type utilized throughout this project. Promising methods of search that are not seeking take advantage of efficient datastructures (often trees or tree-like structures) and to allow faster retrieval of a match than a seek. A notable example are F-Trees, a type of R*-tree populated wherein levels are built based on the proximity of the subsets Fourier coefficients [11]. These are applicable in cases where we search many same-length patterns on one dataset; this is not our use case.

In general, we don't seek every possible region of length l in the time series (see Section 6.1). Instead, we seek through the time series at a stride that is some fraction of the initial query length. This stride length is a parameter that can be adjusted. When this stride is fixed, we refer to it as a fixed-walk.

A smart-walk (see Algorithm 1) proceeds as a fixed-walk except after evaluations where $\eta_\Delta > thresh$, wherein the stride length for the next evaluation increases to some value nx.

After seeking through the time series, the similarity of the potential matches has to be measured. The different types of time series matching algorithms with their different measures of similarity, will be discussed in Section 5.2.

Algorithm 1. Smart-walk

```
Input: $Q$ query, $D := d_i$ data in closed subranges of length $len(Q)$
Parameter: $X$ fixed stride length, $Y$ stepover length, $\tau$ Quality Threshold
Output: Vector of match qualities $\vec{\eta}$

 1: $\vec{\eta} \leftarrow list$
 2:  do
 3:      $i \leftarrow 1$
 4:      if $\eta_\Delta(Q,d_i) > \tau$ then
 5:          $\vec{\eta}.append(\eta_\Delta(Q, d_i))$
 6:          $i \leftarrow 1 + Y$
 7:      else
 8:          $i \leftarrow 1 + X$
 9:  while Not at the end of time series
10: return $\vec{\eta}$
```

5.1.3 Postprocessing

After a query has been performed, the result is a list of tuples, each tuple containing the range of a match exceeding the threshold and the quality of that match, η_{Δ}. The amount of these matches and the distribution can be a challenge for end users – it may not be useful to return *10* matches all overlapping the same pattern; the goal of the system is to use a single region tuple to alert the user to a single matching pattern.

This goal is complicated by the interplay in a variety of parameters: Lowering the match quality threshold will increase the number of regions returned; raising it may cause false negatives. Decreasing the stride length will increase the number of matches returned and result in many overlapping matches on a single pattern (especially for a time-scaling measure such as DTW), increasing it introduces the risk that a pattern is missed because it doesn't align well in time with the stride period.

In an attempt to improve the user experience, we have designed two postprocessing routines and included them in our parametric studies of algorithm performance. In the first, best-in-group postprocessing, every contiguous set of matches (those whose endpoint overlaps the following startpoint) is reduced by selecting only the best match in that set. In the case that a single pattern of interest truly exists in that group, the best match should be the most accurate pattern overlap.

The second method is Combine-group postprocessing, in which each contiguous group is reduced to a single match with a range starting at the minimum startpoint of the group and ending at the maximum endpoint of the group. The quality in this case is inherited from the best match within the group.

Finally, postprocessing of the type none is self-explanatory – the raw list of returned regions from the algorithms is maintained.

5.2 Implementation

Five algorithms were selected to be implemented and evaluated, with three as the focus. The algorithms were selected based on a desire for both variety (in strengths and weaknesses, as well as in fundamental method of use) and a high likelihood of success on pattern matching the supplied datasets. These algorithms include three from a distance category, and two from the probabilistic category.

5.2.1 Distance

Wavelet matching is implemented in a straightforward manner. The preprocessing task consists of a Discrete Wavelet Transform using Haar wavelets at a depth of *4* levels. This results in a set of coefficients in a generated wavelet basis. The measure is directly the Euclidean distance between these coefficients – that is, we measure the proximity in the decomposed wavelet basis, rather than reconstructing the time series from the truncated wavelet representation. This results in an algorithm with fast performance as it has vastly decreased length in search space. The scan method is selectable between a fixed-stride walk and a smart walk through the time series. Postprocessing may be selected from any of the previously described postprocessing routines.

Dynamic Time Warping (DTW) is a popular time series matching procedure in which the query may be mapped surjectively to the search subset, effectively pairing each query sample to one or more search samples. Taking the optimal mapping (the mapping which results in the lowest Euclidean distance) under some restrictions (endpoint continuity, maximum time scaling limits) performs a nonlinear scaling of time within the search sequence. This agrees well with intuitive ideas of similarity, in which a peak of the same size and shape, but a few samples further in time than expected, is considered by a human observer a very similar sequence. In comparison, a direct Euclidean distance sample-by-sample produces a significant penalty on small temporal deviations. The dynamic time warping routine used is a mature implementation by the authors of [12], and described therein.

Adaptive Piecewise Constant Approximation (APCA) [13] is an approximation of a time series that partitions it into *n* (a hyperparameter) segments of variable length. Each of these segments is represented by the mean of the data points it contains. The lengths of the individual segments are chosen in such a way as to achieve the most accurate approximation of the time series under the limitation of n segments. This is achieved by a discrete Haar wavelet transformation in which the largest coefficients are retained. This results in an approximation that preserves the essential structural properties, but on the other hand abandons non-essential structures such as noise. During a seek operation a region is transformed into an APCA representation by partitioning it into n segments, same as the query, with segment starting points in the same relative locations. Euclidean distance is measured between these two low-dimensional representations. Since these are relatively inexpensive operations, an APCA matching algorithm is performant with respect to runtime. Furthermore, APCA is particularly suitable for the approximation of time series with sharp edges. But less for natural or sinusoidal signals.

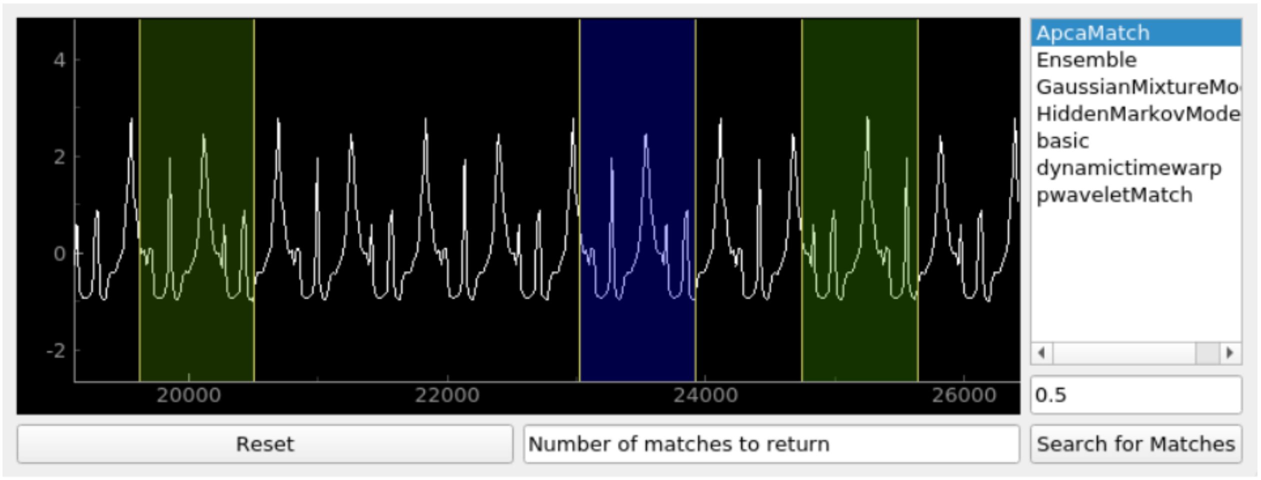

Fig. 2.Screenshot of the prototype GUI, developed for interactively finding matched patterns of interest.

5.2.2 Probabilistic

At a first glance, it may seem unintuitive that we use probabilistic models to compare time series. In this approach, we consider the time series as a realization of the probabilistic model by representing the observed time series by an underlying model. The main structure of the time series is reflected in this model, where the next point in the time series is expected probabilistically.

As preprocessing, we must transform the time series into a probabilistic model. To get such a model, the observed time series is fit with an Expectation-Maximization-Algorithm generating hyperparameters of our underlying model. Knowing these parameters, one can calculate the likelihood prior, if the model is correct, for the realized time series. We evaluate this likelihood on the query range, and then by any chosen scan method compare the likelihood of each subset of interest. The measure is a ratio of the query likelihood and the subset of interest likelihood. One can compare it to the likelihood ratio which is used in statistical tests to accept or reject a hypothesis. This procedure stays the same for all probabilistic models. In the following, we will consider two of them. It has to be stressed that it is not the same as the likelihood ratio test and does not have the statistical properties of a likelihood ratio test. Nevertheless, it fits our needs in terms of comparing the similarity.

Gaussian Mixture Model (GMM) is essentially the mixture of several Gaussian distributions. One hyperparameter is the number of Gaussian models one wants to include in the GMM. The different unique Gaussian distributions will be weighted. In particular, the number of Gaussian models represents the different states of our observed time series. Each different state of the time series can be characterized by a different Gaussian distribution, i.e. the plateaus which can be represented by different means of the Gaussian distribution or the increase of the variation which can be represented through a higher variance.

Hidden Markov Model (HMM) is a more sophisticated probabilistic model. HMMs have two components. On the one hand are the observed emissions and on the other hand the hidden states. The connection between both is the emission probability. Every hidden state has a distribution for the emission probability and they form a Markov chain of different transition probabilities. This is applied to a time series in a manner analogous to the GMM. The characteristics of the observed values are modeled through the probabilistic model (in this case, a HMM). The HMM, in comparison to GMM, additionally takes the transition between states into account capturing these transitions into the model. Through the additional information the complexity rises and also the computational cost for this model.

5.2.3 Ensemble

We found each algorithm to have strengths and weaknesses in regards to patterns that are reliably matched. A natural extension was to combine the algorithms in an ensemble approach. This is common in time series forecasting work as it decreases the variance and often leads to better results than the singular algorithms.

For our ensemble approach, we utilized the DTW, APCA, and GMM implementation, based on their promising and complementary performance on the test datasets. Details of this study, showing the relative performance of each algorithm across datasets, are found in Section 6.2. In the ensemble approach, each algorithm is independently executed. If any algorithm considers the region as a match it is considered as an ensemble match (an OR ensembleing process). More sophisticated ensemble procedures are conceivable. Since we want to find a robust algorithm against several challenges we relied on the simplest ensembling rule to avoid overfitting for certain test cases.

A challenging detail in the assembling is determining a match quality *[0, 1]* based upon the independent match qualities returned from each algorithm. Ultimately, the qualities are scaled uniformly and summed to generate an overall ensemble quality measure from each independent algorithm.

5.2.4 Prototype GUI

In addition to the label tooling described in Section 3.5, we developed a pattern-matching GUI which may be used immediately by the system engineers to select regions of interest from arbitrary time series, and explore the patterns matching this selected region as returned by various algorithms. Within the GUI (see Fig. 2), the prototype allows selection of a region of interest (marked in blue) and returns color-coded matches (red-green mapping low-quality to high-quality matches). The algorithm to use is selected in the right-hand-side dialog box.

6. Evaluation

6.1 Performance Evaluation

Investigations into the appropriate performance metrics to evaluate our system yielded a single most-popular measure: The receiver operating characteristic (ROC), and to summarize into a single value from *[0, 1]*, the area-under this curve (AUROC) [14]. This measure indicates how successful a binary classifier is by plotting the true-positive rate against the false-positive rate. A perfect classifier in this space appears as a step function to *1*, a random classifier is represented by the diagonal.

We have implemented our own version of this routine. We consider a return a true positive if it overlaps a labeled range at least *40%*, and a false positive if it does not. A false negative is represented by any labeled pattern that is not found by the algorithm. For the number of true negatives, we must determine the total population of possible regions in our time series – that is: how large is the strict-duration sets A_l?

A lower bound would be every subset we explicitly evaluated by the scan method, however this is inappropriate because our measure would then be very sensitive to changes in step size and stepover size. More intuitively, taking large steps does not remove regions from A_l, it only pre-emptively assigns a negative response by the classifier. Therefore, these should be included in the total population.

An upper bound for the total population is infinite – we can check a region infinitesimally close to the last region; but this is also inappropriate. Assuming for simplicity of discussion a fixed sampling rate, it is tempting to use as a total population the set of every range of length l starting from a unique sample. This is not fundamentally better than the continuous case; and would clearly result in near-zero false positive rates, as the number of true negatives dwarfs the possible number of matching regions.

To resolve this and allow ourselves to use the ROC tool, we again must turn to the user experience (and to the ultimate goal of the project): At what distance would regions be spaced such that a user would identify any closed region as belonging to the same pattern region, and any further spacing as belonging to a different possible pattern region? Human pattern perception is relative; as a result we suggest using a fraction of the query pattern length as the effective spacing to determine a total population. By intuition we suggest this value is somewhere in the range of *1/8*

Algorithm 2. ROC performance measure

```
     Data: ResultMatches, LabeledMatches
     Result: Percent of matches for x% of matches
1:   Sort ResultMatches descending by quality and add midpoint
2:   Add midpoint to ManualLabels
3:   usedMatches = 0
4:   for Each ResultMatch ∈ ResultMatches do
5:       usedMatches + = 1
6:       Find closest LabeledMatch ∈ LabeledMatches
7:       Get overlap of ResultMatch and LabeledMatches
8:       if Overlap is at least 40% then
9:           if overlappedLabels >= x ∗ Length(LabeledMatches) then
10:              return int(x ∗ Length(LabeledMatches)/usedMatches
```

to *1/4* of the query length, and conservatively choose *1/8*. Therefore, we assume the total population (for the purposes of ROC plotting) as:

$$8 * \frac{TotalTimeseriesDuration}{QueryDuration} \qquad (6)$$

6.2 Computational Experiments

The final evaluation of the system took place in two stages: first, a parametric study on the UCR data afforded a view into the pareto-front of settings for each algorithm. This allows us to make suggestions on optimal parameters for each algorithm; affords insight into how to improve the algorithms and approaches; and indicates the relative value of each component of our pipeline.

We also apply the two best performing algorithms to the satellite test data to evaluate performance.

6.2.1 Parameter Study

There are a number of parameters for each algorithm to vary – examples are the bounds for time-scaling limits in DTW; the number of segments in APCA; an appropriate depth for Wavelet decomposition; and the number of hidden states in a HMM. Conservative selections for these parameters were used, based on experimentation and existing values used in literature.

In addition to these parameters, there are a number of parameters which affect all algorithms; these parameters were used for an optimization study on the UCR dataset utilizing a compute cluster. These parameters are:

Analysis of the performance at each of these iterations is performed using the AUC metric. For brevity and effective visualization, this document will present the results in aggregate: at each combination of parameters, the sum of all AUC scores across test cases is used as the final score. For *14* test cases, this would result in a perfect classifier having a score of *14*. A classifier with random (coin-flip) classification on every test case would score a *7*.

Table 3. This parameter study results in a total of over *6000* pattern matching searches.

Symbol	Description	range
τ	Quality measure threshold, below which a potential match is discarded	*0.1 − 0.75*
α	Step size fraction. The fraction of the query length (in time) which is used as the stride for a seek	*1*, *1/2*, *1/4*, *1/8*
β	Stepover fraction. The fraction of the query length to step over in the case of a smart walk. $\beta = 0$ is defined to be a fixed-walk	off, *1*, *1/2*
P	Postprocessing method	none, best, combine

6.2.2 Performance Study

For the performance study, we conducted a simplified representative study with the most promising algorithms from the UCR study (GMM and APCA) at a low match threshold (*0.01*) with a fixed-walk at *1/8* query length. This provides the purest result for a AUROC curve and indicates the general performance of these two algorithms on the our dataset (see Section 3.1).

7. Results and Discussion

A sampling of the results from the parameter study are presented in Fig. 3, Fig. 4, and Fig. 5. Representative results from the performance study are found in Fig. 6.

Fig. 3 shows the aggregate sum on AUC scores on each of the 14 UCR testcases. We note that the baseline Wavelet approach is a poor classifier. In general for all algorithms a finer stepsize ($\alpha = 8$ results in a step *1/8* the query length) provides better results; likewise, a lower threshold results in more effective classification. These points must be balanced with runtime since the runtime scales linearly with the stepsize. Increasing the threshold improves runtime as well, but to a lesser extent than the stepsize (any item below the threshold is discarded from memory).

Better performance is observed with no postprocessing (see Fig. 4) than with the post-processing routines. Here we observe limitations with AUROC as the metric for our performance. A greater number of false positives has a less pronounced effect on the AUROC measure than it does on user experience: a user may tolerate only a certain number of false positives before discarding the tool. As the postprocessing routines were added to improve this user-experience, it is unsurprising to see them have a negative effect on the AUROC score.

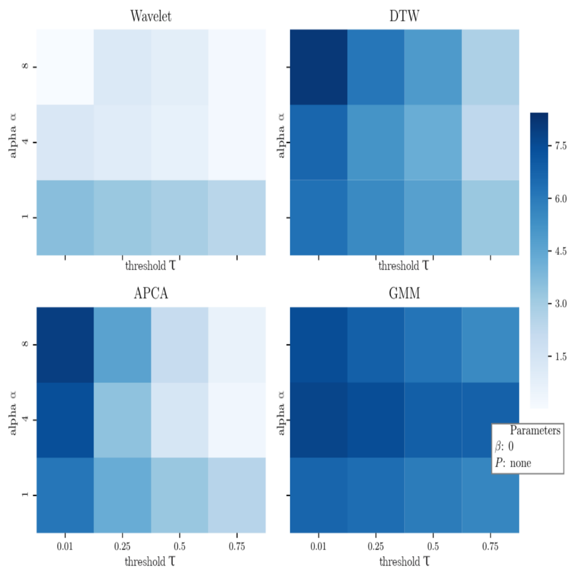

Fig. 3. Parameter study results on UCR dataset. Heatmap shows aggregate AUROC score over *14* test cases. Higher scores represent a better classifier, a *14.0* is a perfect classifier on all testcases. In this plot, smart walk is disabled and there is no postprocessing.

In general, the algorithms perform satisfactory on the classification problems. In most cases the primary problem resulting in a low AUC score is that not all labeled patterns are identified by the algorithms, no matter how low the threshold is set. Two reasons appear for this. The first is that we do not perform a complete scan through the time series in infinitesimally small strides – some pattern regions may not align with the scan stride, resulting in a failure to classify this region. Increasing to a finer and finer scan increases computation time and introduces the additional issues with user experience regarding overlapping regions.

The second problem reflects the nature of the work and was indicated in our introduction: what a similar pattern is determined by human intuition. As a result, any metric may fail to correlate with human judgement; and furthermore labeled datasets may not be precise and free of errors.

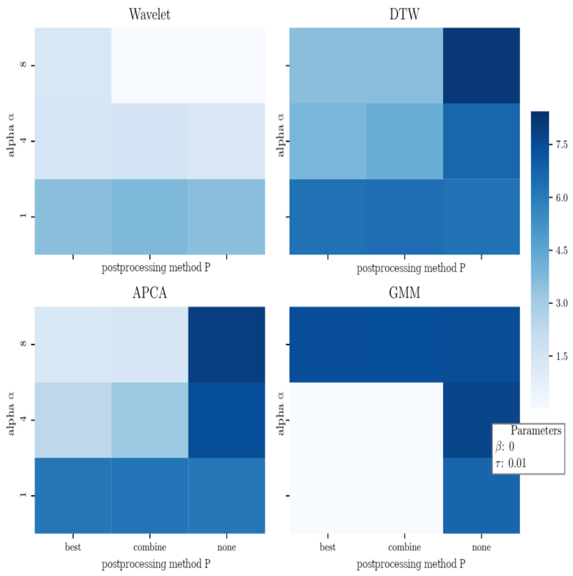

Fig. 4. Parameter study results on UCR dataset. Heatmap shows aggregate AUROC score over *14* test cases. Higher scores represent a better classifier. *14.0* is a perfect classifier on all testcases. In this plot, a fixed walk is performed at different step sizes α and *different* postprocessing methods. The threshold is fixed and smart walk is disabled. The ROC curves from the top-right squares (excepting Wavelet) are shown in Fig. 5.

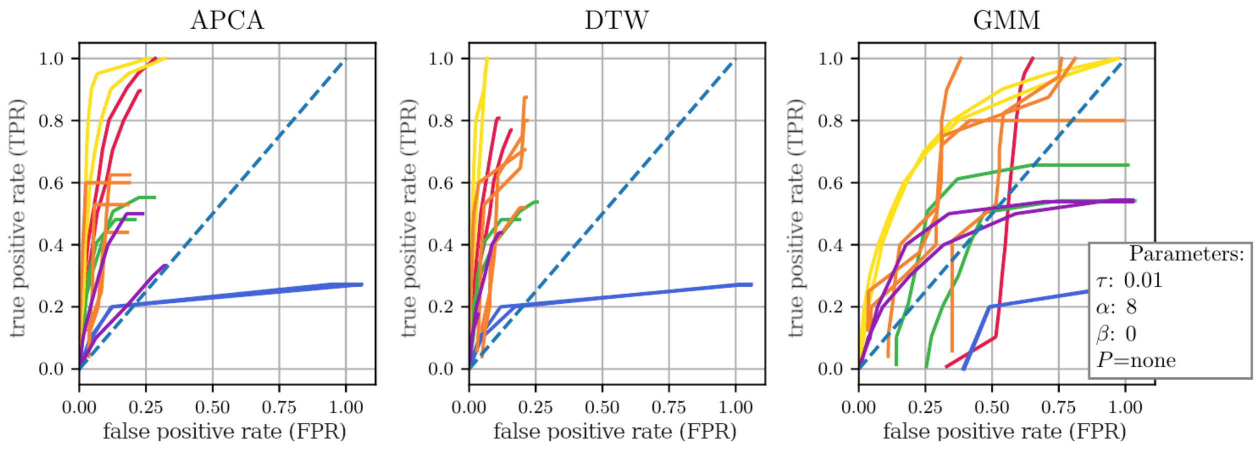

Fig. 5. Results across UCR data test cases of APCA, DTW, and GMM methods using a fixed walk with no postprocessing. Graphs correspond to data in Fig. 4. Colors represent one signal, lines sharing a color are different patterns to match within that signal. The dashed line represents a binary random classifier.

We may gain additional insight by reviewing two reasonable, but not best-case, parameterizations in the original ROC space. The first, Fig. 5 shows the performance of each algorithm at matching each of the *14* test cases in our UCR dataset. Here we see again the singular AUROC score may not reflect the user experience – the GMM method often scores well on AUROC, but its precision over the first *10%* of matches has many more false positives than the equivalent APCA, which has excellent performance with its first matches. In all three plots most test cases terminate around the same TPR – note the green test case. Perhaps in this case the human who labeled the testcase did so with a different eye for similarity than the Euclidean norm, DTW distance, or likelihood ratio do.

The same comparison between APCA and GMM translates to the satellite telemetry dataset (Fig. 6). APCA, when effective, generally returns reliable matches as the highest scoring regions. GMM, while often reaching *100%* of matches ultimately, return many false positives to the user as the highest quality regions.

8. Conclusions and Future Work

In summary, we have successfully developed a software suite which gives a technical user the ability to easily label new signal datasets for testing; incorporate new matching algorithms; evaluate the performance of algorithms; implement new performance measures; and most importantly gives a nontechnical user the ability to, through a GUI, select a range of time containing a pattern of interest be presented a set of reasonable-quality matching regions in the signal's history. The goals of the project as it was presented (and summarized in the Introduction chapter) have been achieved.

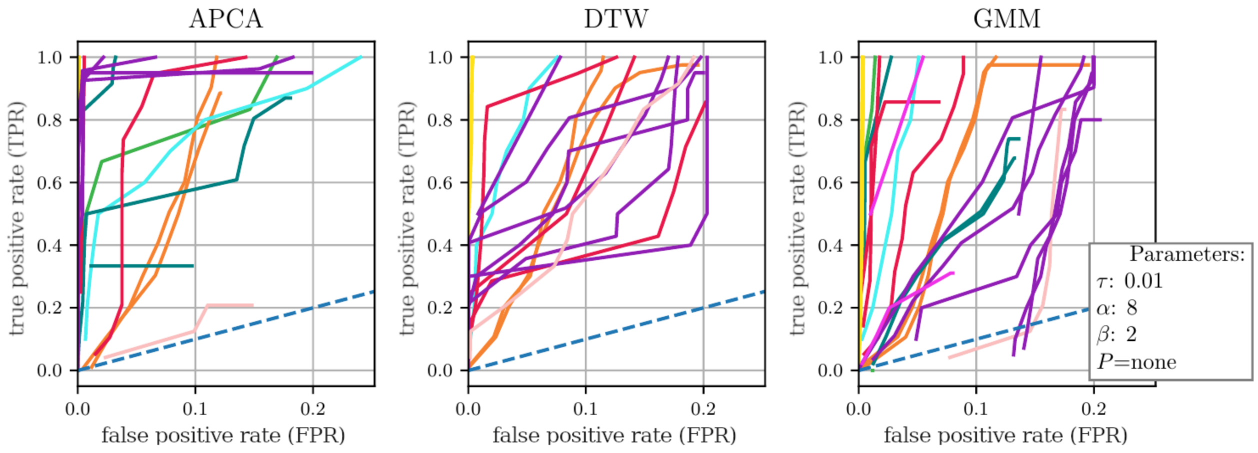

Fig. 6. Results across all satellite telemetry data test cases of APCA, DTW, and GMM methods using a smart walk with stepover length equal to one half the total query length. The threshold is *0.01* and the stepsize is *1/8* query length. Each line color represents one signal, lines that share colors are different test case patterns from the same signal. The dashed line represents the expected performance of a random binary classifier.

Furthermore, the software has a practical, professional design and will serve as a strong foundation for future work in this area at the at mission operations at GSOC. Six functional pattern-matching algorithms are included in the software – one of which (DTW) is the direct application of an existing library for this purpose [4]. Algorithm performance is challenging to measure; however the initial results are promising. Already the tool provides a useful set of results – further optimization of algorithm parameters should be undertaken.

A valuable exploration in the immediate future would be into evaluation of the software from a user perspective. Though AUROC results are higher at low thresholds and step sizes; human preferences for performant software and a clean set of results may encourage other parameterization realizations.

Acknowledgements

The authors are thankful to Dr. Martin Wickler, Dr. Edith Maurer, Dr. Tobias Göttfert, and all members of the MBT-Team from the DLR for their valuable support and fruitful discussions. Furthermore, our appreciation goes to Prof. Dr. Massimo Fornasier, Dr. Ricardo Acevedo Cabra, and Laure Vuaille from the Technical University of Munich for providing the creative set-up in course of the Data Innovation Lab.

References

[1] Yi, B.-K., and Faloutsos, C., Fast Time Sequence Indexing for Arbitrary Lp Norms, Proceedings of the 26th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, p. 385–394, https://dl.acm.org/doi/10.5555/645926.671689.

[2] Rebagliati, S., and Sasso, E., Pattern Recognition Using Hidden Markov Models in Financial Time Series, Acta et Commentationes Universitatis Tartuensis de Mathematica, Vol. 21, 2017, p. 25, https://doi.org/10.12697/ACUTM.2017.21.02.

[3] Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G., The UCR Time Series Classification Archive, July 2015, https://www.cs.ucr.edu/%7Eeamonn/time_series_data_2018/.

[4] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., and Keogh, E., Addressing Big Data Time Series: Mining Trillions of Time Series Subsequences Under Dynamic Time Warping, ACM Trans. Knowl. Discov. Data, Vol. 7, No. 3, 2013, https://doi.org/10.1145/2500489.

[5] Kin-Pong Chan, and Ada Wai-Chee Fu, Efficient Time Series Matching by Wavelets, Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337), 1999, pp. 126–133, https://doi.org/10.1109/ICDE.1999.754915.

[6] Esmael, B., Arnaout, A., Fruhwirth, R., and Thonhauser, G., Improving Time Series Classification Using Hidden Markov.Models, 12th International Conference on Hybrid Intelligent Systems (HIS), 2012, pp. 502–507, https://doi.org/10.1109/HIS.2012.6421385.

[7] Povinelli, R. J., Johnson, M. T., Lindgren, A. C., and Jinjin Ye, Time Series Classification Using Gaussian Mixture Models of Reconstructed Phase Spaces, IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 6, 2004, pp. 779–783, https://doi.org/10.1109/TKDE.2004.17.

[8] Lin, J., Keogh, E., Lonardi, S., and Chiu, B., A Symbolic Representation of Time Series, with Implications for Streaming Algorithms, Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03, 2003, pp. 2–11, https://doi.org/10.1145/882082.882086.

[9] Bagnall, A. J., Bostrom, A., Large, J., and Lines, J., The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version, CoRR, Vol. Abs/1602.01711, 2016, http://arxiv.org/abs/1602.01711.

[10] Schäfer, P., Scalable Time Series Similarity Search for Data Analytics, Ph.D. thesis, Humboldt-Universität zu Berlin, Germany, 2015, http://edoc.hu-berlin.de/docviews/abstract.php?id=42117.

[11] Agrawal, R., Faloutsos, C., and Swami, A., Efficient Similarity Search in Sequence Databases, Foundations of Data Organization and Algorithms, edited by D. B. Lomet, Springer Berlin Heidelberg, Berlin, Heidelberg, 1993, pp. 69–84, https://doi.org/10.1007/3-540-57301-1_5.

[12] Rakthanmanon, T., Keogh, E., Lonardi, S., and Evans, S., Time Series Epenthesis: Clustering Time Series Streams Requires Ignoring Some Data, Proceedings - IEEE International Conference on Data Mining, ICDM, 2011, pp. 547–556, https://doi.org/10.1109/ICDM.2011.146.

[13] Chakrabarti, K., Keogh, E., Mehrotra, S., and Pazzani, M., Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases, ACM Trans. Database Syst., Vol. 27, No. 2, 2002, p. 188–228., https://doi.org/10.1145/568518.568520.

[14] Pepe, M. S., Longton, G., and Janes, H., Estimation and Comparison of Receiver Operating Characteristic Curves, The Stata Journal, Vol. 9, No. 1, 2009, pp. 1–16, https://doi.org/10.1177/1536867X0900900101.