A framework for pattern mining and anomaly detection in multi-dimensional time series and event logs

Len Feremans[1], Vincent Vercruyssen[2], Wannes Meert[2], Boris Cule[1], and Bart Goethals[1]

[1] Department of Mathematics and Computer Science, University of Antwerp, Belgium
{firstname.lastname}@uantwerpen.be
[2] Department of Computer Science, KU Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract. In the present-day, sensor data and textual logs are generated by many devices. Analyzing these time series data leads to the discovery of interesting patterns and anomalies. In recent years, numerous algorithms have been developed to discover interesting patterns in time series data as well as detect periods of anomalous behaviour. However, these algorithms are challenging to apply in real-world settings. We propose a framework, consisting of generic transformations, that allows to combine state-of-the-art time series representation, pattern mining, and pattern-based anomaly detection algorithms. Using an early- or late integration, our framework handles a mix of multi-dimensional continuous series and event logs. Finally we present an open-source, lightweight, interactive tool that assists both pattern mining and domain experts to select algorithms, specify parameters, and visually inspect the results, while shielding them from the underlying technical complexity of implementing our framework.

1 Introduction

Discovering interesting patterns and anomalous periods in heterogeneous time series data is often the main interest of people generating and analyzing these data. In the past decades, the field of pattern mining has developed a large body of algorithms to automatically discover different types of interesting patterns, such as *frequent itemsets* and *sequential patterns* [15]. However, these algorithms are difficult to use for domain experts that are not familiar with their inner workings. Moreover, the algorithms require the data to be preprocessed to the proper format and the type and quality of the patterns being found is largely dependent on the choices made in the preprocessing steps. If a dataset consists of multiple time series or dimensions this becomes even more problematic. Recent *pattern-based algorithms for anomaly detection* in time series suffer from the same drawbacks [4,7] .

More importantly, when a new sequential pattern mining or a pattern-based anomaly algorithm is presented, important time series representation choices are often only discussed in the experimental design, and a review of alternative representations is often out of scope. Therefore, we propose several techniques for preprocessing and reducing continuous time series, and event logs. A wealth of itemset and sequential pattern mining algorithms has been developed in the past decades [5]. These pattern mining algorithms are optimized towards specific, *built-in* constraints, such as mining closed itemsets or mining sequential patterns satisfying temporal constraints efficiently [11]. We propose to add generic *external* constraints for reducing the set of discovered patterns independently from any specific algorithm, such as temporal constraints. Doing so allows for more flexibility. Finally, for anomaly detection we generalize two methods for computing an anomaly score based on patterns: frequent pattern based outlier factor (FPOF) [7] and pattern-based anomaly detection (PBAD) [4], that employs an isolation forest to predict anomaly scores based on distance-weighted pattern embedding. We extend both algorithms to support multiple time series and ensembles of pattern sets.

We propose a framework that allows for flexibility at the cost of being slightly less efficient by supporting several *generic transformations*, namely: (i) generic transformations and aggregations for continuous time series, (ii) a transformation that creates a transaction or sequence database for both single, *multi-dimensional*, and *mixed* continuous and discrete time series data, (iii) a transformation that re-computes support based on *temporal constraints* compatible with any pattern mining algorithm, and (iv) two *anomaly detection algorithms* that are compatible with any pattern mining algorithm and can handle pattern sets mined from multiple dimensions. By providing these generic transformations, end-users have the freedom to create new compositions not considered by the original authors. For example, instead of frequent sequential patterns, an end-user of our framework can mine a set of sequential patterns using an alternative interestingness measure [3,12], subsequently apply temporal constraints, and then use these patterns as input to an anomaly detection algorithm. We have implemented this framework and made it available as an open-source, interactive time series pattern mining tool (TIPM) that enables an iterative, exploratory workflow for preprocessing, finding patterns, discovering anomalies, and visualizing time series data and patterns.

2 Method

The general workflow of our framework is shown in Fig.1. In this section we discuss each step in our framework, also shown in the overview in Fig. 2.

2.1 Time series representation for pattern mining

Non-stationary time series. First, we must consider that time series naturally exhibit a large amount of variation. Typically, event logs are sparse and contain small periods where there is a burst of events. Continuous sensor values are often assumed to be autocorrelated, meaning that the next value is closely related to

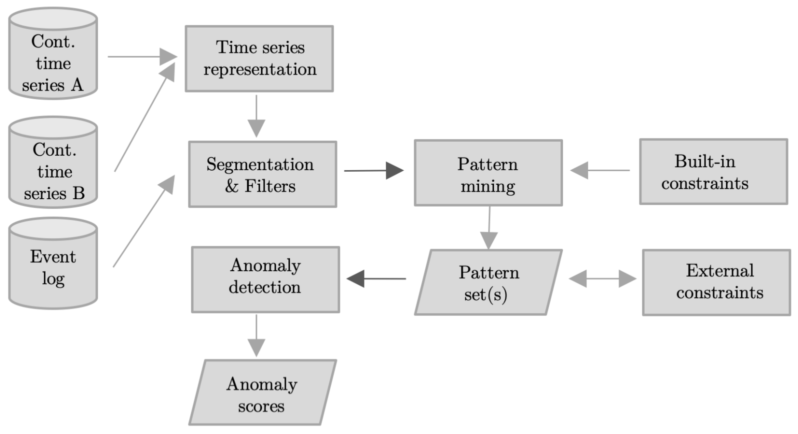


Fig. 1: Workflow of our framework.

the previous value in the raw signal. A stationary time series has the property that the mean, variance, and autocorrelation structure do not change over time. Some authors assume that there is a global trend and that after correcting for this trend, the time series is stationary. We will not make such assumptions and consider that different periods have different means, variance, and autocorrelation structures.

Dealing with outliers. Two strategies are available in the framework. First, removing the outliers by capping values that deviate a user-specified number of standard deviations from the mean. Second, keeping the outliers and discretising them along with the rest of the data. Which strategy to use depends on the use case and can be freely chosen by the user. When applying a pattern-based anomaly detection technique, we are mainly interested in patterns that occur frequent in normal regions, and prefer the first strategy.

Time series dimensionality reduction. A straightforward transformation to reduce time series is piecewise aggregate approximation (Paa) [8]. We set a window duration w and replace all continuous values in each window with the window mean, effectively downsampling a time series S by a factor $|S|/w$. In practice, we often want to downsample each time series as we are more interested in patterns that span a larger period. Note that Paa allows more flexibility than symbolic aggregate approximation [10]. The latter assumes that the time series values are normally distributed, which is rarely the case in a non-stationary time series.

Discretisation. After reducing dimensionality, we discretise continuous time series using *equal-width* or *equal-length* bins. As a rule-of-thumb, equal-width discretisation is applied if the observations are normally or uniformly distributed over the bins. If this is not the case, equal-length binning with a slightly larger number of bins can be selected by the end-user. The goal of discretisation is to have good coverage of items that occur in at least 5% of segments.

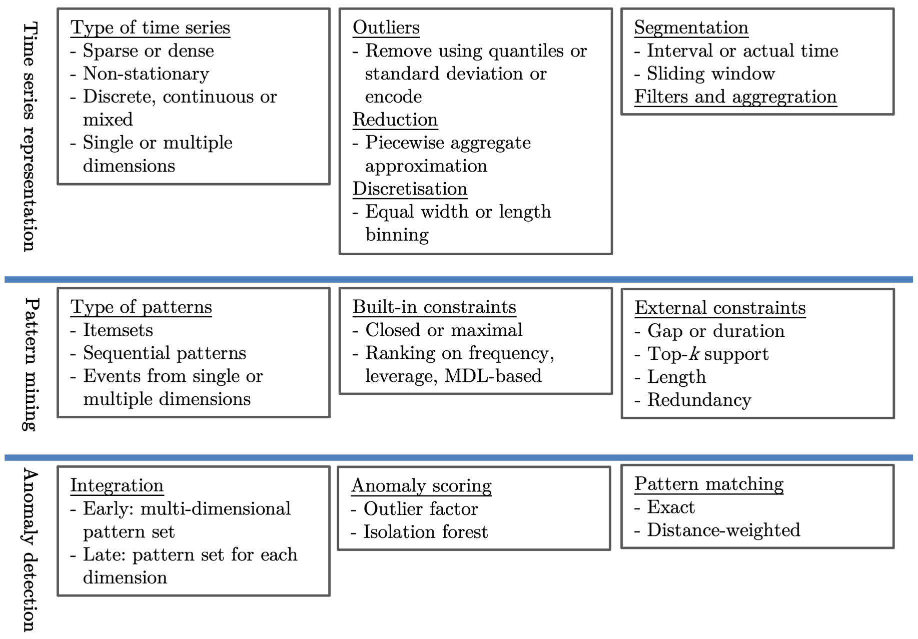


Fig. 2: Detailed overview of our framework.

Segmentation. Before pattern mining we create fixed-size *sliding windows*, or *segments*. We specify the window duration and increment in time units or steps. Setting segmentation parameters is largely domain-specific. For instance, if the length of the datasets is two hours, but measurements (or events) are sampled every second, then finding patterns within 1 minute makes sense. The window duration and increment are important parameters towards pattern mining since they directly determine the length of the average window (or transaction), and the total number of windows. In practice, patterns are limited in length so we must ensure that windows are of moderate size. We ensure this by either setting a relatively small value for the duration or by reducing time series dimensionality.

Filters and aggregation. Finally, our framework supports basic filtering and aggregation on the time series, as well as generic SQL queries. Filtering is useful if we want to model only a part of the dataset. For instance, an end-user can filter the time series on time, on periods where certain event codes (e.g., warning/error codes) occur, or periods where some continuous variable exceeds a certain threshold. This has the advantage that end-users can mine and discover interesting patterns local to certain event or condition. Finally, we provide options to aggregate values within each window and compute summary statistics such as min, mean, max, count and unique.

Automatically selecting parameters. Using our framework we can manually change parameters for representing time series. Good parameters can be selected using domain knowledge or set interactively in a trial-and-error way. However, it is possible to select parameters using a wrapped approach for anomaly detection. For instance, let l be the duration, i the increment, w the window for Paa, and b the number of bins. We can select a parameter setting from the parameters space $\Omega = \{l, i, n, b\}$ using random search and select those settings that optimise an evaluation metric on the anomaly scores.

2.2 Pattern mining

Frequent pattern mining. After preprocessing, an end-user can discover patterns for each dimension that is either discrete or has been transformed to a discrete representation. Our current framework integrates with the Spmf library [5] containing more than 40 algorithms for itemset and sequential pattern mining, covering efficient algorithms for mining frequent, closed, and maximal itemsets and sequential patterns, top-k sequential patterns ranked on leverage [12], and a set of sequential patterns compressed using minimal description length [9]. For the brevity of this paper, we will not discuss details of these algorithms and refer to existing work [5,15]. Itemset and sequential pattern mining algorithms require a suitable representation for enumerating patterns and computing support. Itemset mining algorithms require a *transaction database*. This databases is created by generating a transaction, or unordered set of *items*, for each window. Likewise, sequential pattern mining algorithms require a *sequence database* where for each window, we create a chronologically ordered list of items (if two events happen at the same time, this is also encoded). Each item is encoded using an integer identifier and either represents an event code or discretised continuous value. When applying a pattern mining algorithm, our framework creates a transaction (or sequence) databases in the background, transparent to the end-user. We encode events using items, consisting of integer identifiers, and decode item identifiers to report human-readable patterns.

External constraints. A recent benchmark study found that temporal constraints for pattern mining in time series are of high importance [16]. Our framework computes occurrences of itemsets and sequential patterns, reported by any algorithm, and computes the occurrences that have a minimum duration in each window, by looking at the raw dataset. If the minimal occurrence does not satisfy *temporal constraints* on maximal duration and maximal gap (time between two pattern items in one occurrence), we remove the occurrence and re-compute the support for each pattern. In addition, we provide basic external constraints for filtering patterns on the minimum and maximal *length*, filtering the top-k patterns on *support*, and removing *redundant* patterns using a threshold on Jaccard similarity, i.e., if two patterns cover mostly the same transactions, we filter the pattern with the lowest support.

Multi-dimensional pattern mining. Thus far, pattern mining algorithms only work on a *single*-dimensional event log or continuous time series, after preprocessing. Our framework makes it possible to uncover patterns with events from multiple dimensions. If more that one input dimension is selected for pattern mining in our framework, we create a *multi-dimensional* transaction (or sequence) database in the background, transparent to the end-user. For itemset mining, we create a transaction for each window by adding events from multiple time series to a single set. For sequential pattern mining, we create a sequence transaction for each window by adding events from multiple dimensions sorted chronologically. We differentiate between events from different dimensions by encoding the item identifier to reflect the source dimension.

Pattern explosion in time series. While the pattern mining community has gone through great lengths in creating efficient algorithms for different tasks, time series remain a difficult data source for efficient pattern mining. For example, imagine a time series that contains a sequence of 20 values, that occurs frequently. Because this series is frequent, any subsequence will also be frequent, thereby generating an exponential number of patterns. In general, time series generate a lot of patterns due to naturally occurring autocorrelation. The problem becomes even worse when two or more dimensions are added, especially if different time series dimensions are highly correlated. In practice, mining maximal patterns with relatively high support in each dimension separately seems to work well. Alternatively, we can change the representation of the time series. In our experience, we find that using pattern sets with more than a few thousand of patterns rarely results in higher accuracy.

2.3 Pattern-based anomaly detection

We provide two algorithms for anomaly detection: a generalised version of FPOF and a generalised version of PBAD. Both methods take a set (or sets) of patterns as input and compute an anomaly score between 0.0 (normal) and 1.0 (abnormal) for each segment. If we select an increment of a single timestamp we can compute an anomaly score at each timestamp.

Generic outlier factor. FPOF [7] computes a score for each segment S_i in time series S, given a pattern set $\mathbf{P}$, based on the total number of patterns matching each segment, denoted by $P_k \prec S_i$:

$$p(S_i, \mathbf{P}) = 1.0 - \frac{|\{P_k | P_k \in \mathbf{P} \text{ and } P_k \prec S_i\}|}{|\mathbf{P}|}.$$

The authors only consider closed itemsets over a single dimension, but we can extend FPOF to compute this score for any pattern set, such as sequential patterns, and for multiple pattern sets mined over multiple dimensions. To compute the score over two patterns sets, $\mathbf{P}_1$ and $\mathbf{P}_2$, we can compute

$$p(S_i, \mathbf{P}_1 \cup \mathbf{P}_2) = 1.0 - \frac{|\{P_k | P_k \in \mathbf{P}_1 \cup \mathbf{P}_2 \text{ and } P_k \prec S_i\}|}{|\mathbf{P}_1 \cup \mathbf{P}_2|}.$$

It is trivial to extend this formula to d dimensions. The only requirement is that for computing a match from dimension d, we check if the pattern mined from dimension d matches the segment of the corresponding dimension. Note that multiple pattern sets can also be mined over the same dimension, but using different algorithms and settings. For example, we can combine both itemsets and sequential patterns in a single dimension.

Generic isolation forest of distance-weighted occurrences. PBAD employs an isolation forest, based on an embedding of both maximal itemsets and sequential patterns for each continuous and discrete dimension [4]. For continuous variables, the authors use a distance-weighted match to match both itemsets and sequential patterns with each original, non-discretised, segment. For example, itemset $P_k = \{\text{“0.5”}, \text{“0.6”}\}$ will be close to segment $S_1 = (\mathit{0.50}, \mathit{0.61}, 0.11, 0.10)$ and further from segment $S_2 = (\mathit{0.31}, \mathit{0.42}, 0.12, 0.04)$. We generalise PBAD by decoupling the pattern mining from the anomaly detection phase. Concretely, we can use the distance-weighted embedding and isolation forest on any ensemble of pattern sets. This allows combining different preprocessing, pattern mining, and external constraints available in our framework. We extend PBAD to two dimensions as follows. Assume we have two pattern sets $\mathbf{P}_1$ and $\mathbf{P}_2$. First, we compute the distance-weighted match between each pattern and each window for continuous time series, and the exact match for discrete (or multi-dimensional) time series. We now have two matrices of dimensions $|S| \times |\mathbf{P}_1|$ and $|S| \times |\mathbf{P}_2|$, and can represent each segment S_i using a feature vector (or embedding) of length $|\mathbf{P}_1| + |\mathbf{P}_2|$. Finally, we feed this embedding to an isolation forest to compute anomaly scores.

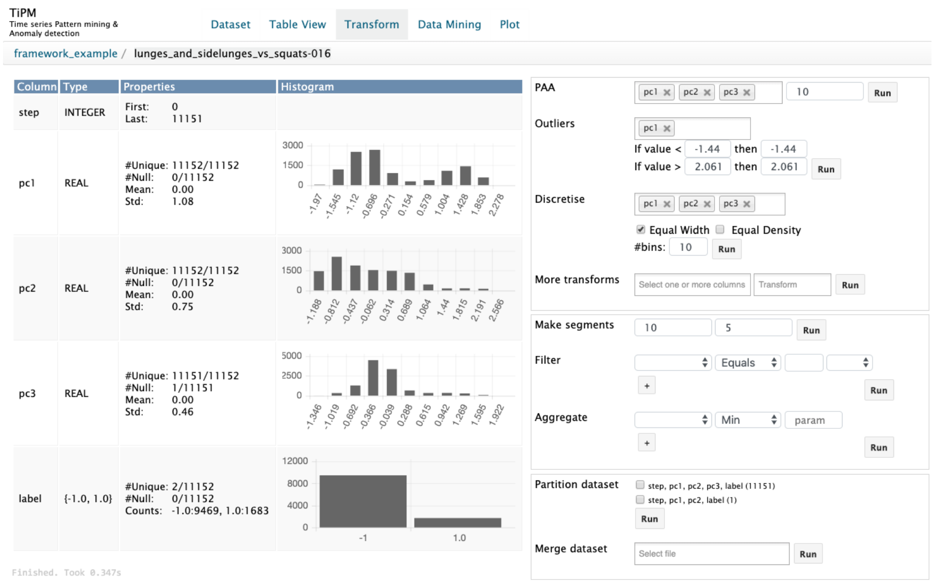
Multi-dimensional anomaly detection using early- and late integration. For anomaly detection in time series there exist two main strategies for dealing with multiple dimensions. In *early integration*, we mine patterns using a single multi-dimensional transaction (or sequence) database and then predict anomalies on a single pattern set. We can compute an anomaly score using the generic outlier factor or isolation forest. In *late integration*, we mine pattern sets in each dimension separately, and compute the anomaly score based on the union of these pattern sets using either anomaly detection method. As remarked in PBAD [4], it would be tempting to prefer early integration, because patterns directly represent multiple dimensions. However, this leads to more items and a larger search space, making pattern explosion more likely. We recommend late integration for larger time series, as the anomaly detection algorithms combine patterns from different dimensions.

2.4 Implementation of framework

We implemented the framework in Java as an open-source web-based application called TIPM[3].

Interactive workflow. For preprocessing, we can upload any dataset that contains at least a timestamp and one or more value columns. TIPM visualizes the histogram and summary statistics for each column and allows to transform continuous time series using our framework, as shown in Fig. 3. For subsequent pattern mining we apply algorithms implemented in SPMF [5]. Multi-dimensional

[3] Source and datasets available at `https://bitbucket.org/len_feremans/tipm_pub`



Column	Type	Properties	Histogram
step	INTEGER	First: 0 Last: 11151	
pc1	REAL	#Unique: 11152/11152 #Null: 0/11152 Mean: 0.00 Std: 1.08	
pc2	REAL	#Unique: 11152/11152 #Null: 0/11152 Mean: 0.00 Std: 0.75	
pc3	REAL	#Unique: 11151/11152 #Null: 1/11151 Mean: 0.00 Std: 0.46	
label	{-1.0, 1.0}	#Unique: 2/11152 #Null: 0/11152 Counts: -1.0:9469, 1.0:1683	

```
Finished. Took 0.347s
```

PAA: pc1 pc2 pc3, 10, Run
Outliers: pc1
If value < -1.44 then -1.44
If value > 2.061 then 2.061 Run
Discretise: pc1 pc2 pc3
[✓ Equal Width] Equal Density
#bins: 10 Run
More transforms: Select one or more columns, Transform, Run
Make segments: 10, 5, Run
Filter: Equals, +, Run
Aggregate: Min, param, +, Run
Partition dataset:
step, pc1, pc2, pc3, label (11151)
step, pc1, pc2, label (1)
Run
Merge dataset: Select file, Run

Fig. 3: Time series representation, pattern mining, anomaly detection, and visualisation are implemented interactively in TIPM.

mining transformations, external constraints and anomaly detection algorithms are implemented in our framework. For visualisation, TIPM can plot continuous time series values, transformed values, discrete event logs, labels, and segmentation, on different levels of granularity in time (raw, hourly, daily, yearly, etc.). For validation of data mining, we can render pattern occurrences and anomaly scores. TIPM saves intermediate files after each operation allowing end-users to undo any operation.

Representing mixed-type real-world datasets. Many real-world datasets, such as SCADA data for wind turbines, contain missing values, non-continuous periods, and timestamped values stored together with event log data in a single file. In our framework, we stay close to this *tabular format* as this is most convenient for collaborating with domain experts who prefer to look at the raw data for validation. In addition, we provide two explicit temporal join operations: *partition* and *merge*. Partition takes a subgroup of columns having non-zero values and saves them in a separate table. This is useful for extracting event log data from continuous time series data. Merge is the opposite operation and takes the union of two tables and sorts them on time. If two column names match in both tables, merge takes the column value of the first table. For example, merge can be used to join time series datasets from multiple devices.

Scale to a large dataset. Most transformations in our framework are implemented using *streaming* techniques, thereby loading only a small set of rows at a time, instead of loading all data into main memory. By only loading and processing data in a streaming, or *paginated*, way, the interface and most preprocessing and postprocessing transformations can handle large time series with millions of samples in an instant of time. For pattern mining, we can manage resources by setting support to a relatively high value, and reducing time series as discussed before. A possible extension would be to support streaming pattern mining algorithms.

3 Experiments

In this section, we will illustrate our framework, implemented in TIPM, to mine a multivariate dataset. The time series dataset that was obtained by using a Kinect sensor to track the body movements during indoor physical exercises [1]. The goal is to assist people in performing exercises correctly. We focus on detecting incorrectly executed exercises during a continuous workout session consisting of 60 lunges and 10 squats. The ground-truth values are known. We remark that the original dataset consists of 75 time series, and we reduced this to 3 time series using principal component analysis [4].

First, we upload the time series in tabular file format. TIPM shows statistics and histograms for each time series (*pc1*, *pc2* and *pc3*) as well as the label as shown in Fig. 3. We can now select options to preprocess each time series. First we cap outlier values based on the 1% and 99% quantiles. Next, we compute and store the average value every 10 time steps to reduce the 3 continuous dimensions using PAA. We then apply equal-width discretisation with 16 bins. For multi-dimensional mining, our input dataset thus consists of 16×3 discrete items. We create sliding windows with a duration of 10 steps (or 3 seconds in absolute time) and an increment of 5, resulting in 223 windows of length 10 that overlap for 50%. With all continuous data time series represented as discretised segments, we start mining patterns. We opt for early integration, and select all three dimensions as input. We select an algorithm for mining maximal itemsets with a support of 20%. For reducing patterns, we remove itemsets that co-occur in at least 90% of windows, resulting in 999 itemsets. We compare this set of patterns by mining maximal sequential patterns with the same settings, resulting in 360 patterns. Finally, we run the generic outlier factor and compute an anomaly score for both types of pattern sets individually.

Fig. 4 shows the first minute of the Kinect dataset. TIPM shows the transformed time series and overlapping segments. We selected the top-5 most frequent maximal itemsets for visualisation. The first itemset is $\{pc1 = 1, 2, 4 \wedge pc2 = 1 \wedge pc3 = 4, 5\}$ which has a support of 56 (or relative support of 0.25). This means that 25% of segments contain both (discretised) values of 1, 2 and 4 in time series *pc1*, 1 in *pc2*, and 4 and 5 in *pc3*. Notice that the first frequent pattern, as well as the 2^{nd}, 3^{th} and 5^{th}, but not 4^{th}, almost never occur in any anomalous segment. Consequently, the patterns are examples of frequent interpretable patterns discriminative towards anomalies. TIPM allows to sort pattern on confidence towards normal (or abnormal) segments. We find that sequential patterns containing high values of *pc1* are the most predictive towards abnormal behaviour. Fig. 5 shows the anomaly scores over the entire 6 minute

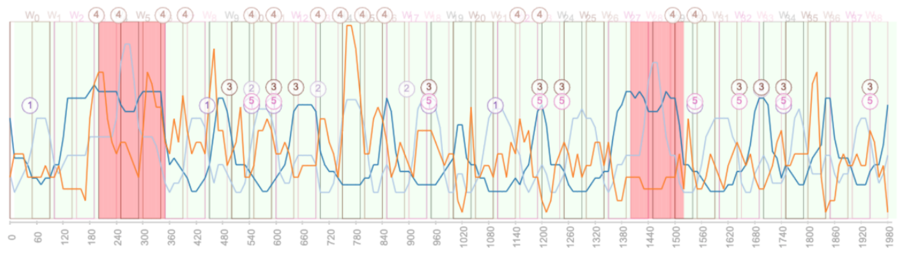


Fig. 4: Visualisation in Tipm of first minute of Kinect data. The blue line is *pc1*, the light blue line *pc2*, and the orange line *pc3*. We show the top-5 most frequent maximal itemsets, mined over all three dimensions.

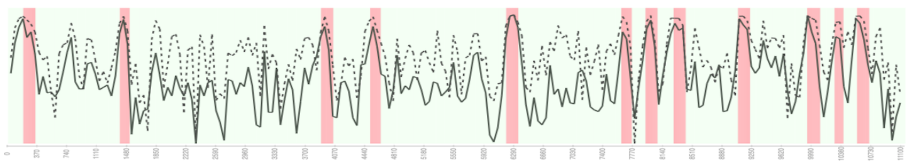


Fig. 5: Visualisation in Tipm of anomaly detection results. Segments with a red background are labelled anomalies, and the black line is the anomaly score predicted (unsupervised) using generic outlier factor using maximal sequential patterns. The dotted line is the results using maximal itemset patterns.

time series. Using the generic outlier factor anomaly detection method we can report an AUROC of 0.839 and average precision of 0.767 for maximal itemsets, and an AUROC of 0.884 and average precision of 0.833 for maximal sequential patterns.

4 Related work

Most general data mining libraries, such as Weka or Knime, are very immature when it comes to pattern mining. Tipm extends Spmf [5] by providing temporal constraints, multi-dimensional pattern mining, and pattern-based anomaly detection algorithms. In contrast to Spmf, and other libraries that implement time series transformations on consecutive numeric vectors, we support timestamped tabular data with multiple dimensions, and mixed-type attributes. Other tools for anomaly detection in time series uses either shapelets or motifs in single-dimensional continuous time series. Interactive pattern mining tools, such as Mime [6], are immature towards time series processing.

There exist algorithms for directly mining patterns with temporal constraints [11]. However, by providing temporal constraints as an external post-processing filter, we can apply them to any pattern mining algorithm. This is of interest for many efficient algorithms for mining closed, maximal or interesting patterns that do not support temporal constraints. Many more transformations for reducing the length of the time series exist [2]. We prefer Paa for two reasons.

First, different authors have confirmed that more advanced techniques are not necessarily more effective [2,10]. Second, many other representation techniques, i.e., transformation to spectral space, single value decomposition, or clustering, make interpretation much harder while patterns of binned values, are easy to interpret. Other transformations, such as differencing or smoothing the raw time series are not problematic regarding interpretation.

Two popular techniques for classification and anomaly detection in time series are the *matrix profile* [14], that computes an outlier score relative to the euclidean or dynamic time warping (DTW) distance to its nearest neighbour, and time series *shapelets*, which are subsequences from a continuous time series and are used in combination with the DTW distance to classify time series segments [13]. A key difference is that frequent patterns naturally handle both continuous time series and event logs [4]. If we compare sequential patterns to shapelets, we argue that on the one hand, sequential patterns generalise shapelets, because we use non-continuous subsequences with gaps. On the other hand, sequential patterns are more specific, because they consist of discretised values instead of continuous values. The latter argument against sequential patterns, however, can be relaxed by using a weighted distance [4]. Itemsets, however, are radically different from shapelets and of value for predicting anomalies [7]. Future work can look at an *ensemble* of representations. That is, we can compute itemset and sequential pattern distances, exact pattern matches, shapelet distances, motif distances, and combine those in one feature vector, as input for existing classification or anomaly detection algorithms.

5 Conclusion

Existing pattern-based anomaly detection algorithms focus on a particular combination of time series representation, pattern mining, and computation of the anomaly score [4,7]. In PBAD, the authors remarked that this method is a promising general framework for time series anomaly detection, where certain variations might be more effective in different applications. In this paper, we implement such a framework and discuss a wealth of general techniques, that can be composed using general transformations to create new variations. This allows data scientists, together with domain experts, to create novel unsupervised anomaly detection models. We also present TIPM, an interactive, easy-to-use, and open-source tool that implements our framework. TIPM is unique since we have a rich set of options for interactively preprocessing and mining patterns from mixed-type time series, supported by visualisation of (raw and transformed) time series, event logs, segments, patterns and anomaly scores. With our framework, we show how to discover interesting interpretable patterns and detect anomalies in multi-dimensional time series.

6 Acknowledgements

The authors would like to thank the VLAIO SBO HYMOP project for funding this research.

References

1. Decroos, T., Schütte, K., De Beéck, T.O., Vanwanseele, B., Davis, J.: AMIE: Automatic monitoring of indoor exercises. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 424–439. Springer (2018)
2. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. Proceedings of the VLDB Endowment **1**(2), 1542–1552 (2008)
3. Feremans, L., Cule, B., Goethals, B.: Mining top-k quantile-based cohesive sequential patterns. In: Proceedings of the 2018 SIAM International Conference on Data Mining. pp. 90–98. SIAM (2018)
4. Feremans, L., Vercruyssen, V., Cule, B., Meert, W., Goethals, B.: Pattern-based anomaly detection in mixed-type time series. Joint European Conference on Machine Learning and Knowledge Discovery in Databases (2019)
5. Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The spmf open-source data mining library version 2. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 36–40. Springer (2016)
6. Goethals, B., Moens, S., Vreeken, J.: Mime: a framework for interactive visual pattern mining. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 757–760. ACM (2011)
7. He, Z., Xu, X., Huang, Z.J., Deng, S.: FP-outlier: Frequent pattern based outlier detection. Computer Science and Information Systems **2**(1), 103–118 (2005)
8. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and information Systems **3**(3), 263–286 (2001)
9. Lam, H.T., Mörchen, F., Fradkin, D., Calders, T.: Mining compressing sequential patterns. Statistical Analysis and Data Mining: The ASA Data Science Journal **7**(1), 34–52 (2014)
10. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. pp. 2–11. ACM (2003)
11. Pei, J., Han, J., Wang, W.: Constraint-based sequential pattern mining: the pattern-growth methods. Journal of Intelligent Information Systems **28**(2), 133–160 (2007)
12. Petitjean, F., Li, T., Tatti, N., Webb, G.I.: Skopus: Mining top-k sequential patterns under leverage. Data Mining and Knowledge Discovery **30**(5), 1086–1111 (2016)
13. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 947–956. ACM (2009)
14. Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A., Keogh, E.: Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: 2016 IEEE 16th international conference on data mining (ICDM). pp. 1317–1322. IEEE (2016)
15. Zaki, M.J., Meira, W.: Data mining and analysis: fundamental concepts and algorithms. Cambridge University Press (2014)
16. Zimmermann, A.: Understanding episode mining techniques: Benchmarking on diverse, realistic, artificial data. Intelligent Data Analysis **18**(5), 761–791 (2014)