Tampere University

Juho Torkkeli

ANALYZING SEQUENTIAL PATTERN MINING TO DETECT CALCIUM PEAKS IN CARDIOMYOCYTES DATA

Faculty of Information Technology and Communication Sciences
Master's thesis
May 2023

ABSTRACT

Juho Torkkeli: Analyzing Sequential Pattern Mining to Detect Calcium Peaks in Cardiomyocytes Data
Master's Thesis
Tampere University
Master's Programme in Computer Science
May 2023

This study examines sequential pattern mining and its applications in various fields. The previous research was conducted by examining signal data, from which calcium peaks were automatically detected and classified. Before the implementation of sequential pattern mining approach to find out patterns from a dataset of 102 signals, association rule mining, frequent itemsets, Apriori algorithm, and rule generation were explored.

Sequential pattern mining, including time constraints, are defined, before examining a knowledge-assisted sequential pattern analysis, from which certain points are considered, such as what is a sequential itemset.

The implementation phase consists of calculating what constitutes a candidate itemset. The findings are modified to work with a sequential rule mining algorithm, and the results are discussed afterwards.

Keywords: [Sequential pattern mining, sequential itemset, signal peak data, sequential rules]

Contents

1 Introduction

The purpose of this thesis is to analyze sequential pattern mining approach to a study previously conducted by Juhola *et al*. [2014 ;2015]. The previous study consisted of automatically detecting calcium cycling peaks from cardiomyocytes data, classifying peaks of signals as either normal or abnormal, and classifying the entire signal into either normal or abnormal class. This thesis continues the first phase of detecting the calcium peaks with sequential pattern mining approach, where we try to discover interesting sequential patterns from a dataset of 102 signals.

Sequential pattern mining is used to discover patterns in a sequence of items, events, or transactions. It is used in various different fields, such as market basket analysis to analyze customer shopping behavior, web usage mining to improve website usability, in security to identify patterns in network traffic to prevent potential threats, and in health care to identify patterns in patient data in order to improve diagnosis, treatment, and disease management.

Sequential pattern mining is a data mining technique that is often used in conjunction with machine learning algorithms. Data mining is used to extract useful information from large datasets, while machine learning involves the use of algorithms to make predictions and decisions from data. [Fournier-Viger *et al*., 2017; Zhao and Bhowmick, 2003]

Calcium participates in most of the vital processes in the body. It is the most abundant cation, and it can circulate in free or ionized form (Ca2+). Circulating calcium amounts to around half of the total calcium levels and they participate in neuromuscular conduction, intracellular regulation, blood coagulation, glandular secretion, and control of skeletal and cardiac muscle contractility. Calcium levels are better for measuring calcium metabolism compared to total calcium levels since they are not influenced by protein concentrations. Measuring calcium is useful while monitoring patients undergoing cardiothoracic surgery or organ transplantation or evaluating patients in cardiac arrest. [Ca2+, 2013]

Calcium (Ca2+) cycling is electrical signaling that links together the cardiomyocyte (heart muscle cells) and contraction. Calcium cycling is used to investigate cardiac disorders and dysfunctions. Failures in cardiac functionality can be observed with the changes in calcium signals and these signals can be recorded with the help of fluorescent calcium indicator dyes. Differentiating cardiomyocytes from human pluripotent stem cells can be used to study cardiac functionality. Differentiated cells can be reprogrammed into pluripotent stem cells with induced pluripotent stem cell (iPSC) technology. iPSC gives new insights into calcium handling in different cardiac diseases. [Kujala *et al*., 2012]

A signal analysis procedure for detecting calcium cycling ‘peaks’ in cardiomyocyte signal data was developed to empower cardiologic investigations. Classifying these peaks into either normal or abnormal classes using machine learning methods was done to separate them for later medical research. [Juhola *et al*., 2014; Juhola *et al*., 2015]

2 Previous research

Previous research objective was to automatically detect calcium cycling peaks from a dataset of Ca2+ transients of 138 signals, which was increased to 280 signals in later research, and to classify the peaks of signals as either normal or abnormal using a developed signal analysis procedure. Then afterwards the entire signals were classified with another procedure, based on previous results, into either normal or abnormal class. The dataset contains data from calcium transients of spontaneously beating human induced pluripotent stem cell-derived cardiomyocytes with the help of Ca2+ imaging using inverted IX70 microscope and recorded with ANDOR camera. According to previous research this kind of research has only been done subjectively and visually. Research procedures for automatic detection and classification of calcium peaks would help future medical research as computational methods are needed. [Juhola *et al*., 2014, pp. 1444; Juhola *et al*., 2015, pp. 1-2]

The data is preprocessed first by using the various sampling frequencies of 8, 10, 11 and 23 Hz to record the signal-data with the help of two different programs [Juhola *et al*., 2015. pp 2]. The signals recorded length varied from 11 to 24 seconds. Modified signals are created by removing present linear descending trend from all signals one by one to facilitate peak detection. Modified signals are used to gain feature data of peaks from the original signals. Higher frequency signals of 23 Hz are smoothed out with median filter and filtering window of three samples to resemble lower frequency signals. Amplitude values are computed from samples in order to explore the distribution of these values. [Juhola *et al*., 2015, pp. 2]

2.1 Preprocessing signal peaks

Peaks are determined by locating their beginning, maximum and end. Beginning and end are local minima (minimum values of the peak) while maximum, the top of the peak, is always a positive local maxima.

Peaks were first preprocessed by removing a linear trend from signals and calculating a rough lower bound estimate for amplitudes of large peaks (Figure 1) by means of the amplitude distribution of samples. Next the minimum (non-negative number) was computed from the signals and subtracted from all values. These operations were done only for detecting peaks and further computations were made for the original data. Histogram mapping, as described by Han *et al*. [2011, pp. 108], is used to capture dependencies between attributes, and it was formed from all samples to represent amplitude distribution. Similarly, linear regression as described by Han *et al*. [2011, pp. 90], is used to find the best 'line' to fit two attributes, and it was computed in the research through successive samples with window length in order to differentiate (approximate) first derivative signal from a preprocessed one. The first derivative of the original signal was approximated

from slope values, also with linear regression. Peak candidates were selected after pre-processing steps by linearly searching means of derivative values from the beginning of the signal to the end. A given threshold T1 was used to find the beginning of the peak when the derivative values were less than T1 and there were more values with at least one greater than or equal to T1. Current peak maximum was found when there was another derivative value less than T1. For finding the end of the peak, threshold T1 was lowered to the right side because of slope being less steep at the end of the peak in most signals. Threshold values were experimented based on given datasets. [Juhola *et al*., 2014, pp. 1445]

Peak candidates were discarded when they were regarded as erroneous or differing considerably from each other. Initially all waveforms were considered to be peak candidates. The following criterion was used to discard them: partial peaks where there was no beginning or end, if the peak was too small (noise), or if the left or right side of a larger peak was considered to be much higher than its other side. [Juhola *et al*., 2014, pp. 1445-1446]

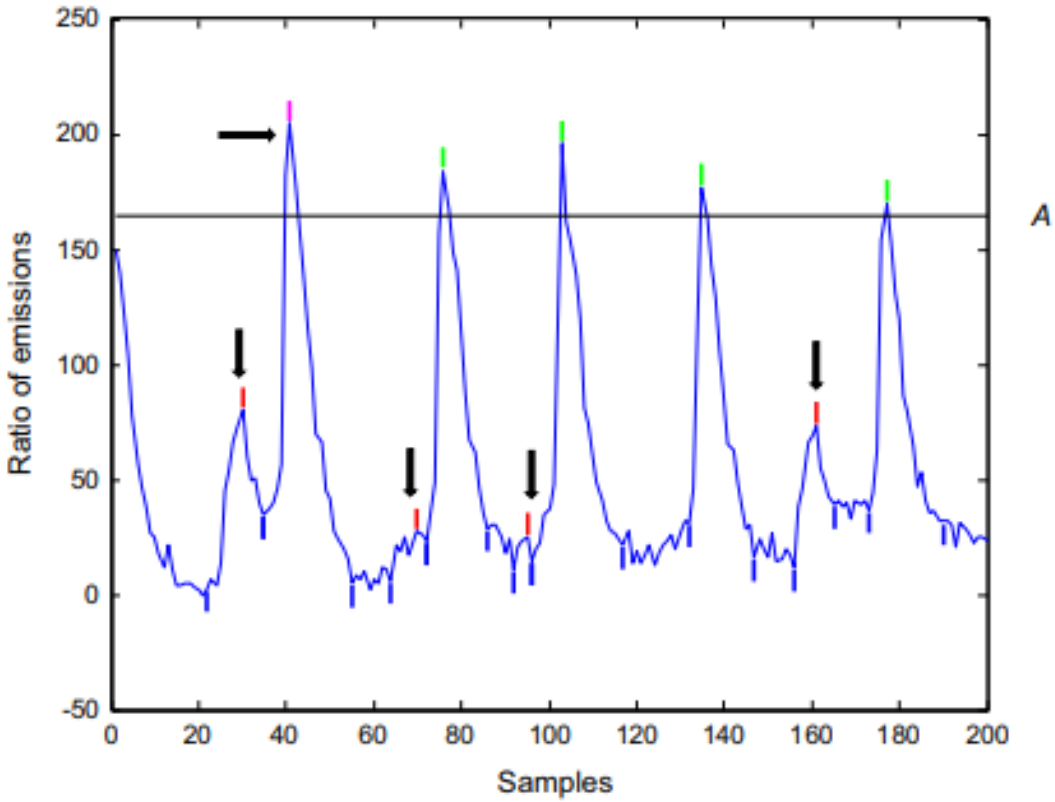

Figure 1: A signal of 19.2 seconds sampled at 10.4 Hz, after the removal of a linear trend. [Juhola *et al*., 2015, pp. 2]

In Figure 1 we have a signal length of 19.2 seconds with average amplitude A that was computed for an average amplitude of the large peaks. With distribution estimate A we can aid the subsequent peak detection. Normal calcium peaks are usually larger than

A however this is not always the case. The black arrows indicate abnormal peaks while the green arrows are normal peaks. The only abnormal peak that was above the average A was deemed abnormal because of asymmetry. The whole signal was determined abnormal because of these abnormal peaks. Vertical axis has no quantitative unit since abscissa values are ratios of two measured values. [Juhola *et al*., 2015, pp. 2]

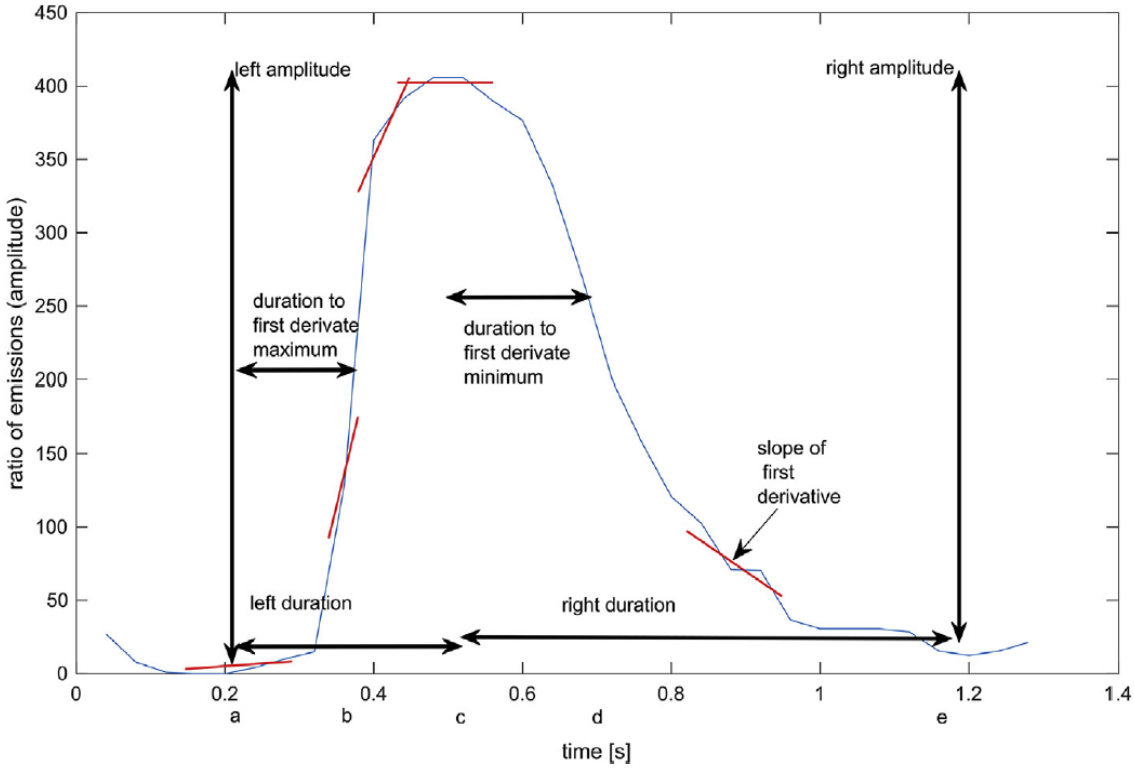

Figure 2: An example of a single peak from beginning to end. [Juhola *et al*., 2019, pp. 18]

In Figure 2 there is a single peak with various variables: left and right amplitudes, left and right durations, duration to first derivate maximum and minimum (steepness of peak sides). Imaginary slopes are visualized with short red lines of the first derivative curve of the signal, in which upward is positive, downward is negative and horizontal is zero slope value. The peak begins from location a, maximum is at c, and ends at e. Locations b and d illustrate the maximum of the first derivative and its minimum, respectively. Beyond these six variables another variable was calculated as interval (in time) from the preceding peak maximum to the current peak maximum. Interval was used to detect if peaks appear regularly in subject to time since irregularity denoted abnormal peaks and signals. Afterwards individual peaks and entire signals were classified as either normal or abnormal with the help of these variables. [Juhola *et al*., 2014, pp. 1446; Juhola *et al*., 2019, pp. 18]

2.2 Signal peak classification

Two subsets of signals were classified depending on their sampling frequencies, number of samples and number of peaks. Algorithm findings were compared to an expert's findings who individually assessed each signal as either normal or abnormal. The idea of the algorithm was to automatically classify all accepted peaks as either normal or abnormal. The peak classification was performed with the seven variables to all accepted peaks. If the larger side of the peak was greater than or equal to the preceding normal peak with the given threshold T2 or greater than given threshold T3 of estimate A, it was classified as normal, and if not, abnormal. If the current peak was the first peak of the signal or no large predecessor peaks or normal peaks were found, estimate A was used as a comparison with threshold T3. If the other side of the peak was asymmetrical compared to the other, it was classified as abnormal. In Figure 1 there are lower than average amount of accepted peaks, only nine, however variability was large due to many signals being very small containing only few peaks. [Juhola *et al*., 2014, pp. 1446]

2.3 Results of the previous research

The two subsets of signal samples were used for classification with various different frequencies and afterwards with all the signal samples together in the first research. The individual signals varied greatly having peaks from 1 to 43, or 45 in later research, per signal. The average amount of peaks of a signal in the first research was 14.1, and 14.7 in later research. Subsets contained 43 and 93 signals, and later up to 280 signals combined, and 234 and 1690 peaks respectively, and up to 4120 peaks combined. Seven variables were standardized to have zero mean and unit variance. [Juhola *et al*., 2014, pp. 1446; Juhola *et al*., 2015, pp. 5]

Signal peaks were classified with leave-one-out validation where a single signal, one at a time, formed a test set while others were used as a learning set. This way all signals were run. Classification accuracy for peaks and whole signals were computed along with true positive and true negative rates. According to Han *et al*. [2011, pp. 327] classification is used to extract models describing important data classes, which are called classifiers that are used to predict categorical (discrete or unordered) class labels. K-nearest neighbor searching (K = 1, 2, … 21), linear and quadratic discriminant analysis, naïve Bayes rule and classification trees along with various support vector machines with kernel functions, were used for classification [Juhola *et al*., 2015, pp. 5-6]. These are described as follows: K-nearest neighbor searching is described by Han *et al*. [2011, pp. 423] as a classifier that learns by analogy by comparing a given test tuple with training tuples similar to it. Discriminant analysis is a technique that is used to predict a categorical response variable, where it assumes that the independent variable follows a multivariate normal distribution [Han *et al*., 2011, pp. 600]. Naïve Bayes classifier assumes a class-conditional independence, in which the effect of an attribute value on a given class is independent of the values

of other attributes [Han *et al*., 2011, pp. 385]. Classification trees or decision trees contain multiple different algorithms in which they are typically constructed in a top-down recursive divide-and-conquer manner, where a training set is recursively partitioned into smaller subsets as the tree is being built [Han *et al*., 2011, pp. 332]. Support vector machines uses a non-linear mapping to transform original data into a higher dimension [Han *et al*., 2011, pp. 408].

The smaller subset of forty-three was more difficult to classify than the bigger ninety-three signal set because of imbalanced class distribution. After peak classification, whole signals were classified as abnormal if even one peak in the signal was classified as abnormal. If a signal was deemed normal in the classification visually by the expert, the result was correct. Otherwise, it was classified as incorrect. Discriminant analysis, along with 3-nearest-neighbor searching in later research, proved to be the most accurate in peak and signal classification with all sets, approximating ~80% accuracy. [Juhola *et al*., 2014, pp. 1447; Juhola *et al*., 2015, pp. 5-6]

2.4 Further development of the previous research

The research has continued further in later studies, where healthy calcium transient signals (controls) were classified with different machine learning algorithms (K-nearest neighbor searching, random forests, least square support vector machine) from abnormal ones in order to tell them apart from different genetic cardiac diseases called catecholaminergic polymorphic ventricular tachycardia (CPVT), long QT syndrome (LQT1) and hypertrophic cardiomyopathy (HCM). Random forests are described by Han *et al.* [2011, pp. 378 & 383] as a type of ensemble method (combines models with the aim of creating an improved classification model) which combines different decision trees together, hence the name 'forest'. The number of variables used for signal peak detection increased from 7 to 10, and later up to twelve. New variables include maximum of the second derivative s″ on the right side of a peak, absolute minimum s″ of the second derivative on the right side of a peak, area R of the peak, duration from peak beginning [s] to left side maximum of first derivative and duration from peak maximum [s] to right side minimum of first derivative. Accuracy of 90% was acquired classification with the new variables with a total of 5290 recognized disease peaks and 2291 control peaks. [Juhola *et al*., 2018, pp. 2; Juhola *et al*., 2019, pp. 17-18]

Preliminary analysis was made to separate the three different diseases (along with controls) with one-way variance analysis (comparing means) as data classes from each other, in which almost all of the pairs with every variable (except two variables for LQT1 vs CPVT) differed significantly from each other ($p < 0.001$). A total of 394 signals from three different diseases along with added 133 controls (healthy subjects) to a total of 527 signals, or 593 in later research, were put into these four data classes. [Juhola *et al*., 2018, pp. 2 & 5; Juhola *et al*., 2019, pp. 15]

The main analysis consisted of classifying the four classes with different machine learning algorithms using leave-one-out principle where a model was constructed with n-1 training signals from the entire dataset of all of the signals and tested n times with each individual signal. The results indicate that separating the diseases with random forests yielded the best results with an accuracy of ~87% and with control data included ~79%. [Juhola *et al*., 2018, pp. 4-6; Juhola *et al*., 2019, pp. 19-20]

3 Association rule mining

Association rule mining is considered an important research technique of data mining and it was first introduced by Agrawal *et al*. [1993]. The idea is to discover frequent patterns, interesting correlations, and associations from among set of items in transaction databases. It is best explained by Zhao and Bhowmick [2003, pp. 3] as when purchasing a book called Data Mining Concepts and Techniques, you get related books which are Database System and Data Warehouse. The latter books are bought together with the first book 40% and 25% of the time. These discovered rules (if-then statements) from the database can be used to better market the said books together in order to make these books' associations stronger and thus sell more books. Association rules are widely used in stores to promote a specific product in order to sell related products, and they are also used in inventory control, market and risk management, telecommunications network etc.

3.1 Frequent itemsets and association rules

Mining association rules consists of finding frequent patterns, which are itemsets that frequently appear in each data set. An itemset is defined by Agrawal and Srikant [1995, pp. 1] as a non-empty set of items. An itemset I $\{I_1\ I_2 \ldots I_n\}$, contains items I_j. Large itemset (or frequent) must satisfy minimum support (minsup). Minimum support is described by Srikant and Agrawal [1996, pp. 2] as a user specified percentage of transactions that contain the pattern. [Agrawal and Srikant, 1995, pp. 4]

Association rules are constructed with if-then statements, which can be split into two parts: antecedent and consequent. This is called a rule-based classifier. The if-then rule is expressed as follows:

$$\text{IF condition THEN conclusion}$$

where a concrete example could be

$$\text{Rule1: IF age} = \text{youth AND student} = \text{yes THEN buys\_computer} = \text{yes.}$$

The "IF" part is called antecedent or precondition, while the "THEN" part is called consequent. The "IF" part can consist of one or more attributes, for example in this case age = youth and student = yes. With the above rule we are predicting whether, with the given attributes, a customer buys a computer or not. The rule is said to be satisfied, and therefore covers the tuple (ordered list of values), if the condition in a rule antecedent holds true for a given tuple. [Han *et al.,* 2011, pp. 355-356]

A given rule R can be assessed by its coverage and accuracy as follows:

$$\text{Coverage}(R) = \frac{n_{covers}}{|D|}$$

$$\text{Accuracy}(R) = \frac{n_{correct}}{n_{covers}}.$$

Given a tuple X, let n_{covers} be the number of tuples covered by R, $n_{correct}$ be the number of correctly classified by R, and |D| be the number of tuples in D, where D is a class labeled dataset. The percentage of tuples that are covered by the rule, where their attribute values hold true for the rule's antecedent, is called coverage. The percentage of correctly classified covered tuples is called accuracy. [Han *et al.,* 2011, pp. 356]

In order to find meaningful and interesting rules, association rule mining uses primarily two different constraints, support, and confidence. Let $I = \{I_1, I_2, \ldots, I_n\}$ be an itemset. Let A and B be a set of items. Let d be a transaction database that contains transactions T, where T is a non-empty itemset ($T \subseteq I$), which is said to contain A if $A \subseteq T$. Each T is associated with an identifier TID. An association rule is an implication of the form:

$$A \Longrightarrow B, \text{where } A \subset I, B \subset I, A \neq \emptyset, B \neq \emptyset, \text{and } A \cap B \neq \phi$$

$A \Longrightarrow B$ rule holds in the transaction set d with support s, where s is the percentage of transactions in d that contain the union of A and B. This is assumed to be the probability

$$P(A \cup B).$$

In the transaction set d, the rule $A \Longrightarrow B$ has confidence c, where it is the percentage of transactions in d, which contains A that also contains B. This is assumed to be conditional probability

$$P(B|A)$$

that is:

$$\text{Support}(A \Longrightarrow B) = P(A \cup B)$$

$$\text{Confidence}(A \Longrightarrow B) = P(B|A).$$

Rules are called strong when they satisfy both the user given minimum support threshold and minimum confidence threshold. [Han *et al.,* 2011, pp. 246-247]

When support and confidence are insufficient to filter out the uninteresting rules, a correlation measure can be used to augment said association rules. Lift is called a correlation measure and it is defined as:

$$\text{lift}(A, B) = \frac{P(A \cup B)}{P(A)P(B)}$$

where the occurrence of an itemset A is independent of B if $P(A \cup B) = P(A)P(B)$; otherwise itemsets A and B are correlated and dependent as transactions. Negatively correlated A with B is marked with resulting value less than 1, where one implies the absence of the other. That is to say positively correlated where the resulting value is more than one, implies that the occurrence of one is dependent of the other. If the resulting value is equal to 1, then there is no correlation between the itemsets A and B, thus them being independent. [Han *et al.,* 2011, pp. 265-266]

Finding frequent patterns helps in generating rules. A frequent itemset is typically referred to as a set of items which usually appear together, such as bread and milk in a shopping basket [Han *et al.,* 2011, pp. 17]. Frequent itemset Z is closed when there exists no super-itemset Y in a dataset D. Itemset Z is closed frequent itemset in dataset D if Z is both closed and frequent. Itemset Z is maximal frequent itemset (max-itemset) in dataset D if Z is both frequent and no super-itemset Y exists such that $Z \subset Y$ and Y is frequent in dataset D. [Han *et al.,* 2011, pp. 247]

Frequent itemsets are typically mined with Apriori algorithm. In this analysis customers shopping habits are analyzed by examining the contents of their shopping baskets. Discovering frequently appearing items helps to gain insight into the shopping habits of the customer, such as in the previous example of how likely a customer will buy a secondary book if they have bought the first one. A typical association rule in this case could be:

$$\text{Data Mining Concepts and Techniques} \rightarrow \text{Database System [support} = 2\%, \text{ confidence} = 40\%]$$

where the interestingness of a rule is examined with the previously mentioned two measures: support and confidence. These measures reflect the usefulness and certainty of discovered rules. Support in this case would mean that 2% of all the transactions under analysis show that Data Mining Concepts and Techniques and Database System are purchased together. Confidence would mean that 40% of those who purchased the first book bought the second book as well. The association rule is considered interesting if both the user given minimum support threshold and minimum confidence threshold (minconf) are satisfied. [Han *et al.,* 2011, pp. 243-245]

3.2 Apriori algorithm

Apriori algorithm is used to mine frequent itemsets from a database and it was first proposed by Agrawal and Srikant [1994]. The algorithm uses prior knowledge of itemset properties, and it employs an iterative approach known as level-wise search, where frequent k-itemsets are used to explore (k+1)-itemsets. Frequent 1-itemsets (I_1), which satisfy the minimum support, are counted first. Then I_1 are used to find frequent 2-itemsets (I_2), which are used to find frequent 3-itemsets (I_3) up until no more frequent k-itemsets (I_k) are found. Because the algorithm does a full scan with every single k-itemset search, the efficiency of the level-wise generation is improved with Apriori property in order to reduce search space. Apriori property is defined as follows: All non-empty subsets of a frequent itemset must also be frequent. This means that itemset I, which does not satisfy minimum support threshold:

$$\mathrm{P(I)} < \mathrm{minsup}$$

is not frequent. If an item a is added to the itemset I, then the resulting itemset union of I and a cannot occur more frequently than I. This means that:

$$\mathrm{P(I \cup a)} < \mathrm{minsup}$$

is not frequent either. If a set cannot pass a test, it is called antimonotonicity, because all its supersets fail the same test as well. [Han *et al.,* 2011, pp. 248-249]

Apriori algorithm consists of a two-step process, join and prune actions. In the join step in order to find frequent itemset I_k, a set of candidates, k-itemsets, are generated by joining the collection I_{k-1} of frequent itemsets with itself. This collection of candidate sets is denoted as C_k. Next let L_1 and L_2 be itemsets in I_{k-1}. Here the notation $L_i[j]$ refers to the jth item in L_i. Apriori assumes, for efficient implementations sake, that items within a transaction or itemset are sorted in lexicographic order. This means that for (k-1)-itemset, L_i, the items are sorted such that

$$\mathrm{L_i[1] < L_i[2] < \ \ldots L_i[k-1].}$$

The join step is performed when the members of $I_{k-1} \bowtie I_{k-1}$, are joinable if their first (k-2) items are in common. This means that L_1 and L_2 of I_{k-1} are joined if

$$\mathrm{(L_1[1] = L_2[1])\hat{}(L_1[2] = L_2[2])\hat{} \ldots \hat{}(L_1[k-2] = L_2[k-2])\hat{}(L_1[k-1] < L_2[k-1]).}$$

To ensure that no duplicates are generated, the condition:

$$L_1[k-1] < L_2[k-1]$$

is in place. The resulting itemset from the joining of L_1 and L_2 is

$$\{L_1[1], L_1[2], \dots, L_1[k-2], L_1[k-1], L_2[k-1]\}.$$

In the pruning step, candidate set C_k is a superset of frequent itemset I_k, which implies its members might not be frequent, but all the frequent k-itemsets are included in C_k. Counting frequency for each candidate of C_k would result in the determination of I_k, since all candidates which have a count no less than the minimum support count are frequent by definition, and thus belong to I_k. To lessen the burden on computation from a database scan, and reduce the size of C_k, Apriori is used. Non-frequent (k-1)-itemsets cannot be a subset of a frequent k-itemset. If any (k-1)-subset of a candidate k-itemset is not in I_{k-1}, then the candidate cannot be frequent either and is removed from C_k. [Han *et al.,* 2011, pp. 249-250]

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Table 1: Transactional Database example. [Han *et al*., 2011, pp. 250]

In Table 1 we have a transactional database (d), in which there are nine transactions from T100 to T900 (d=9) and list of item_IDs or itemsets. In the first iteration every single item is a candidate 1-itemset, C1, and their occurrences are counted. If the user given minimum support threshold is 2 (minsup = 2), all of the candidates of C1 in L1 satisfy the minimum support threshold. In order to discover frequent 2-itemsets, L2, join step of $L1 \bowtie L1$ generates a set of 2-itemset candidates, C2. Next, the support count of each candidate itemset in C2 are counted. Itemsets of L2:

$$(\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\})$$

are deemed frequent since their minimum support threshold ≥ 2. The generation of candidate 3-itemsets, C3 = L2⋈L2, yields us itemsets:

$$(\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I3, I5\}).$$

Apriori property demands that all subsets of a frequent itemset must also be frequent. With this we can determine that the last four candidate itemsets of C3, namely:

$$(\{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\})$$

cannot be frequent. For example {I3, I5} is not a member of L2 and as a subset of {I1, I3, I5} can be removed. Therefore, C3 after pruning yields us itemsets:

$$(\{I1, I2, I3\}, \{I1, I2, I5\}).$$

Lastly, L3⋈L3 are used to generate a candidate set of 4-itemsets, C4. The subsequent joining generates an itemset {I1, I2, I3, I5}, but the subset {I2, I3, I5} is not frequent, and the algorithm stops having generated all the frequent itemsets. [Han *et al*., 2011, pp. 250-252]

3.3 Generating association rules

After frequent itemsets have been found, it is time to generate strong association rules. These must satisfy both minimum support and minimum confidence, although frequent itemsets by definition already satisfy minimum support. Association rules are generated, according to Han *et al.,* [2011, pp. 254] as follows:

$for\ each\ frequent\ itemset\ I, generate\ all\ nonempty\ subsets\ of\ I.$

$For\ every\ nonempty\ subset\ of\ sb\ of\ I, output\ the\ rule$ sb→(I-sb) $if\ \dfrac{support\_count(I)}{support\_count(sb)} > \min\_conf, where\ min\_conf\ is\ the\ minimum\ confidence$

This means that based on the previous chapters frequent itemsets, which were calculated from Table 1, the nonempty subsets of the two frequent itemsets ({I1,I2,I3},{I1,I2,I5}) are: {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, and {I5}. The resulting association rules are shown with confidence:

$$\{I1, I2\} \rightarrow \{I5\}, confidence = \frac{2}{4} = 50\%$$
$$\{I1, I5\} \rightarrow \{I2\}, confidence = \frac{2}{2} = 100\%$$
$$\{I2, I5\} \rightarrow \{I1\}, confidence = \frac{2}{2} = 100\%$$
$$I1 \rightarrow \{I2, I5\}, confidence = \frac{2}{6} = 33\%$$
$$I2 \rightarrow \{I1, I5\}, confidence = \frac{2}{7} = 29\%$$
$$\{I5\} \rightarrow \{I1, I2\}, confidence = \frac{2}{2} = 100\%.$$

With minimum confidence of 70% only the second, third and last outputs are considered strong. [Han *et al.,* 2011, pp. 254]

4 Sequential pattern mining

Sequential pattern mining was first introduced by Agrawal and Srikant [1995] who applied it with different algorithms to a database filled with customer transactions. The database consists of Customer Id, Transaction Time, and Items Bought attributes. These are sorted with Customer Id and Transaction Time in Table 2. Customer Id represents a single customer, Transaction Time when the items were bought and Items Bought what those items are. A sequence is defined by Agrawal and Srikant [1995, pp. 1] as an ordered list of itemsets. A frequent sequence is constructed from a sequence that satisfies a minimum support constraint. Formally a sequence S contains (S1 S2 … Sn) itemsets Sj. A formal definition of a subsequence according to Srikant and Agrawal [1996, pp. 5] is that a sequence (a1 a2 ... an) is a subsequence of another sequence (b1 b2 ... bm) if there exist integers i1 < i2 < ... < in such that a1 ⊆ bi1, a2 ⊆ bi2 , ..., an ⊆ bin. All transactions bound to a single customer can be viewed as a customer-sequence where each transaction corresponds to a set of items, and the list of transactions, ordered by increasing transaction time, corresponds to a sequence. [Agrawal and Srikant, 1995, pp. 1]

Customer Id	Transaction Time	Items Bought
1	June 25 '93	30
1	June 30 '93	90
2	June 10 '93	10, 20
2	June 15 '93	30
2	June 20 '93	40, 60, 70
3	June 25 '93	30, 50, 70
4	June 25 '93	30
4	June 30 '93	40, 70
4	June 25 '93	90
5	June 12 '93	90

Table 2: Database Sorted by Customer Id and Transaction Time. [Agrawal and Srikant, 1995, pp. 2]

In Table 2 the problem is to find all desired maximal user specified minimum support sequences, from which sequential patterns are formed. With a minimum support of 25% i.e., at least two customers, we get sequences ({30}{90}) and ({30}{40,70}). Customers 1 and 4 ({30}{90}) and 2 and 4 have ({30}{40,70}) sequences. The subsequences of these sequences are {30}, {40}, {70}, {90}, ({30}{40}), ({30}{70}) and ({40,70}). These subsequences satisfy the minsup condition, but they are not what we are looking for since maximal sequence S in a set of sequences is not contained in any other sequence. [Agrawal and Srikant, 1995, pp. 1]

4.1 Time constraints

A time constraint contains a user specified minimum and maximum time gaps (min-gap & max-gap), window size (ws) and maximum span (maxspan) and are defined by Mooney and Roddick [2013, pp. 25] as follows; min-gap is a minimum required time difference between the earliest occurrence of an item in an itemset and the latest occurrence of an item in its immediately preceding itemset (transaction). It is formally defined by Masseglia *et al*. [2009, pp. 5] as:

$$\text{transactiontime } (ds_{l_i}) - \text{transactiontime } (ds_{u_{i-1}}) > \text{ mingap}, 2 \leq i \leq n$$

where ds is a data sequence = (ds1 … dsm) that supports a sequence S = (S1 … Sn) if there exists integers l1 ≤ u1 < l2 ≤ u2 < … < ln ≤ un such that:

$$S_i \text{ is contained in } \cup_{k=l_I}^{u_I} ds_k, 1 \leq i \leq n.$$

Max-gap is the maximum allowed time difference between the latest occurrence of an item in an itemset and the earliest occurrence of an item in its immediately preceding itemset. It is formally defined as:

$$\text{transactiontime } (ds_{u_i}) - \text{transactiontime } (ds_{l_{i-1}}) \leq \text{maxgap}, 2 \leq i \leq n.$$

Window size is the maximum allowed time difference between the latest and earliest occurrences of items in any itemset. It is formally defined as:

$$\text{transactiontime } (ds_{u_i}) - \text{transactiontime } (ds_{l_i}) \leq \text{ ws}, 1 \leq i \leq n.$$

Maxspan is the maximum allowed time difference between the latest and earliest occurrences of items in the entire sequence. It is formally defined as:

$$\text{transactiontime } \left(ds_{u_m}\right) - \text{transactiontime } \left(ds_{l_1}\right) \leq \text{maxspan}, 2 \leq m \leq n.$$

[Masseglia *et al*., 2009, pp. 5; Mooney and Roddick, 2013, pp. 25]

In order to better illustrate the above time constraints, we have a data sequence:

$$ds = ({}^{1}\{a\} \, {}^{2}\{b, c\} \, {}^{3}\{d\} \, {}^{4}\{e, f\} \, {}^{5}\{g\})$$

containing five itemsets or transactions which are timestamped from day one to day five. These itemsets contain seven items from a to g. Let us consider the following sub-sequence:

$$ds1 = (\{a, b, c, d\}\{e, f, g\})$$

with time constraints min-gap = 0, max-gap = 4, window size = 2 and maxspan = 6 in order to see how they are handled with the above data sequence ds. Window size for each itemset is marked with [l,u]:

$$ds = ({}^{l1}[\,{}^{1}\{a\}\,{}^{2}\{b, c\}\,{}^{3}\{d\}\,{}^{u1}]\,{}^{l2}[\,{}^{4}\{e, f\}\,{}^{5}\{g\}\,{}^{u2}])$$

where l is the earliest occurrence of an item, while u is the latest occurrence in the itemset. Window size = 3 allows us to group itemsets 1{a} & 2{b,c} with 3{d} and 4{e,f} with 5{g}. Since the earliest occurring item 1{a} of the first itemset of ds1 in data sequence ds is l1 = 1 and latest occurring item 3{d} u1 = 3 and the earliest 4{e} in the second itemset l2 = 4 and latest 5{g} u2 = 5, window size = 3 time constraint holds. With the above window size formula, we get to group together the first itemset:

$$\text{transactiontime}\ (3) - \text{transactiontime}\ (1) \leq \text{ws}\ (3), 1 \leq i \leq n$$

and consequently, group together the second itemset:

$$\text{transactiontime}\ (5) - \text{transactiontime}\ (4) \leq \text{ws}\ (3), 1 \leq i \leq n.$$

If window size were zero, itemsets {a,b,c,d} and {e,f,g} would have to occur simultaneously and using items from different itemsets would not be allowed.

The sequence ds1 is included in the data sequence ds since itemsets 3{d} = u1 and 4{e,f} = l2 can occur in consecutive days since with the above min-gap formula we get:

$$\text{transactiontime}\ (4) - \text{transactiontime}\ (3) > \text{mingap}\ (0), 2 \leq i \leq n.$$

If min-gap were increased to 1, the time constraint would not hold since 4{e,f} occurs immediately after 3{d}. With max-gap = 4, itemsets 1{a} = l1 and 5{g} = u2 are 4 days apart. With the above max-gap formula we get:

$$\text{transactiontime}\ (5) - \text{transactiontime}\ (1) \leq \text{maxgap}\ (4), 2 \leq i \leq n.$$

If max-gap were decreased to less than 4, the time constraint would not hold with ds1. With another subsequence:

$$ds2 = (\{a, b, c\}\{d\}\{e, f, g\})$$

max-gap = 2 would hold since 5{g} = u3 and 3{d} = l2 with the above max-gap formula would give us:

$$\text{transactiontime } (5) - \text{transactiontime } (3) \leq \text{maxgap } (2), 2 \leq i \leq n$$

and the preceding itemset 3{d} = u2 and 1{a} = l1 we get:

$$\text{transactiontime } (3) - \text{transactiontime } (1) \leq \text{maxgap } (2), 2 \leq i \leq n.$$

With maxspan being 6 the earliest occurrence of an item in the entire data sequence is l1 = 1 and latest u2 = 7, which means with the above maxspan formula:

$$\text{transactiontime } (7) - \text{transactiontime } (1) \leq \text{maxspan } (6), 2 \leq m \leq n$$

the time constraint holds. If we add another itemset {h} into the data sequence:

$$ds3 = (\{a, b, , c\}\{d\}\{e, f, g\}\{h\})$$

maxspan value of 6 would not hold since:

$$\text{transactiontime } (8) - \text{transactiontime } (1) \leq \text{maxspan } (6), 2 \leq m \leq n.$$

[Masseglia *et al*., 2009, pp. 6-7]

4.2 Time series mining

Database that consists of transactions with an added timestamp attribute is called time-series database. Mining time series can be divided into four different kind of pattern mining: trend analysis, similarity search, sequential patterns, and periodical patterns. Trend analysis consists of long term or cyclic movements, and it is widely used in predicting prices in a stock market such as if business stock goes up earlier in the week and down similarly later in the week. Similarity search consists of matching slightly different sequences. This can be divided into whole sequence matching or subsequence matching. Similarity search can also be used in stock markets to find similar ups and downs between different businesses. [Zhao and Bhowmick, 2003, pp. 6-7]

Sequential patterns are used to find relationships between ordered transactions from data. A transaction can be a set of items in a transaction between a customer and a provider, such as goods or services. Finding if there is an occurring specific pattern between items from data is sequential pattern mining. Such pattern in a sequence can be if a customer bought an item A and then an item B within a week. If a sequence happens regularly, it is called a periodical pattern. These periodical patterns can be viewed as sequential patterns if they are taken as a set of sequences. Periodical patterns can be mined from data that consists of a long period of time. These periods can be from days to years of worth of transactions. Splitting periods to smaller partitions can also be taken as a set of sequences. Timestamp is an essential attribute of sequential pattern mining when used on time-series databases, but the notion of time can be non-concrete in sequential databases if it is implied as an ordered sequence. These databases without an added timestamp can still be used to find these frequently occurring transactions to describe or predict data. With more accurate rules, such as adding timestamp attribute into an association rule mining, businesses can make better predictions and gain more value from data. [Zhao and Bhowmick, 2003, pp. 7]

4.3 Symbolic sequence mining

Sequence data is defined by [Han *et al*., 2011, pp. 587] as long sequences of transaction or nominal data, which typically are not observed at equal time intervals. An ordered list of numbers can have long lapses of time between them and thus the notion of time can be non-concrete. A string of letters ({A}{B}{C}) can be taken as a sequence as long it is read as {A} comes before {B} comes before {C}. Symbolic sequences consist of activities such as biological sequences, web click streams, customer shopping sequences, program execution sequences, and sequences of transactions in science and engineering and in natural and social developments. [Han *et al*., 2011. pp. 589]

4.4 Biological sequence mining

Biological sequences are generally referred to as nucleotides (DNA/RNA) and amino acids (proteins) sequences and are conducted in the field of bioinformatics and modern biology. These sequences are typically very long and carry complicated semantic meaning while posing many challenging research issues. Biological sequence analyses are used to compare, index, align and analyze these above sequences in order to find out significant patterns in genomic data. [Han *et al*., 2011. pp. 586-591]

An example of a biological sequence could be function n(s) where n represents nucleotides (Adenine, Cytosine, Guanine and Thymine) in a DNA sequence S. Because there are only 4 possible nucleotides A C G T, we know n(S) is ≤ 4. This is called an invariant sequence. [Wang *et al*., 2006. pp. 111-112]

5 Other sequential pattern mining applications

Sequential pattern mining has been used in several other researches such as mining negative sequential patterns (absent itemsets) in e-commerce for practical use [Hsueh *et al.,* 2008], predicting next prescribed medications for diabetes patients [Wright *et al.,* 2014], creating an automatic classifier for predicting occurring heart diseases in patients with congestive heart failure [Gladence *et al.,* 2014], analyzing historical records for recommending new optimal clinical pathways [Uragaki *et al.,* 2016], using various algorithms for extracting information from databases, and developing models for predicting uterine contractions to achieve maximal efficacy during labor [Huang *et al.,* 2013 & 2014].

5.1 Knowledge-assisted sequential pattern analysis

Sequential pattern analysis framework was proposed by Huang *et al.* [2013] in their research for predicting uterine contractions in order to maximize analgesia efficacy during contractions and minimize the impact of medications between those contractions. The proposed framework consists of predicting intrauterine pressure in real time, anticipating the next contraction, and using the developed sequential pattern mining approach to identify the patterns of the contractions from historical patient tracings. A sequential association rule-based collaborative training dataset selection component was developed to dynamically select a training dataset from historical patient tracings (HT), and from the current patient's most recent training time series (d). Collaborative filtering [Linden *et al.,* 2003] is used to filter information from multiple data sources in order to predict the user's preferences, interests, or behaviors from similar users, and in this case, it is used to solve the contraction prediction problem. In the proposed framework collaborative filtering is used with the assumption that the current patient's contraction is similar to previous patient's contractions in the database, and thus can be predicted. The database contains data (in the form of sequential association rules) from historical patient's labor tracings. The current patient's contraction pattern is searched from the database in order to predict the current patient's next contraction pattern. The prediction models are trained through the matching process by selectively utilizing the available data. Afterwards a long-term time series prediction is employed by using k-nearest neighbors least squares support vector machine (LS-SVM) approach with heuristic parameter tuning. Prediction results are further enhanced in the post prediction process. [Huang *et al.,* 2013, pp. 1290-1291]

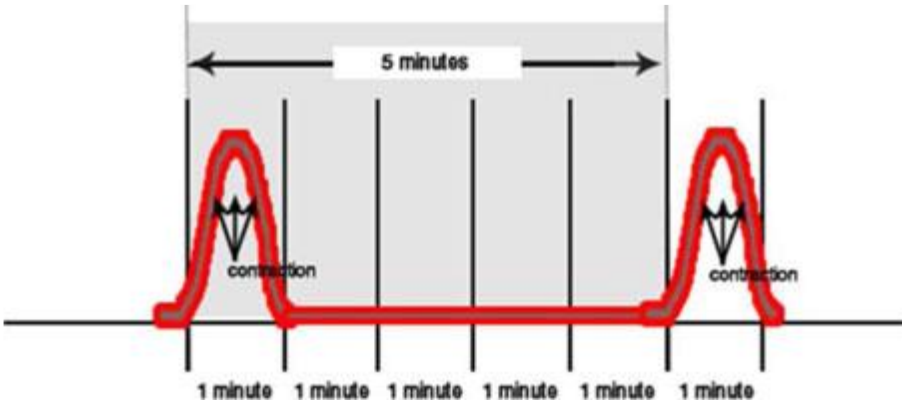

Figure 3: Intrauterine pressure time series. [Huang *et al.,* 2013, pp. 1291]

In Figure 3 is an example of a contraction cycle. The period is approximately 5 minutes, and the individual contractions are shown as a 'peak', with the contraction beginning at the start of the peak and subsiding towards the end after reaching the peak maximum. The values of the contractions were discretized, which according to Han *et al.* [2011, pp. 112] is used to convert raw values of numeric attributes into interval labels, into two tables consisting of the time (period) between the contractions (pd), and the height of the peak (ht) in mmHg (intrauterine pressure). The period varied between pd < 25 seconds up to pd ≥ 385 seconds long periods, and the height of the peak from ht < 40 up to ht ≥ 100. Equal-width discretization was used to create nominal markers for both values from a to f for height, and A to H for period. In the Figure 3 is a contraction with a height of 75, which was marked with nominal marker d, and a period of 300 seconds, which was marked with nominal marker F. Together the aforementioned contraction period was given a value dF. The original intrauterine pressure time series were sampled at each quarter second with a value range of [0, 100]. However due to loss of data and the consequent generated noise, preprocessing step was necessary to reduce noise as much as possible, by subsampling the data at each second in order to detect the peak points. The signal was rebuilt by applying the shape-preserving piecewise cubic Hermite interpolation method [Miranda, R. 2003], which is used to find polynomial functions with specified values. [Huang *et al.,* 2013, pp. 1291-1293]

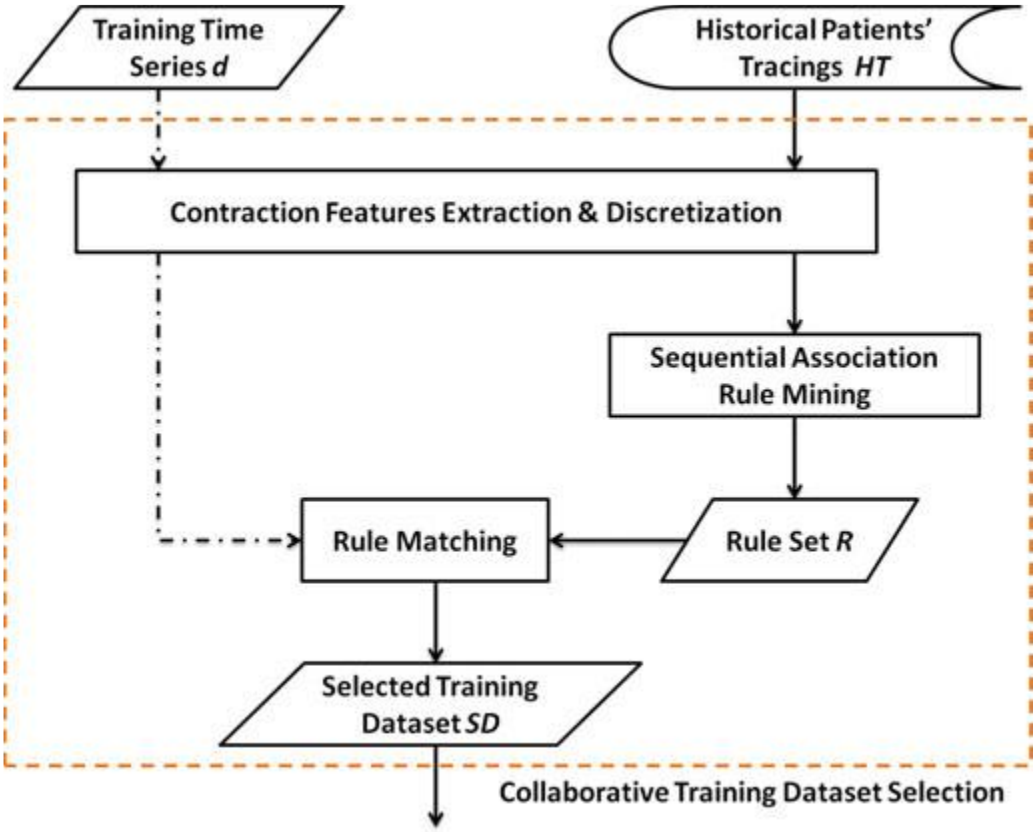

Figure 4: Collaborative training dataset selection component. [Huang *et al.,* 2013, pp. 1292]

In Figure 4 the collaborative training dataset selection consists of individual patients training time series (d) and historical patients tracings (HT). The numerical features are extracted from both of these groups of data and period and height are discretized into nominal values. In the process of mining sequential association rules, the generated rules should be meaningful and interesting. An item is one contraction, and it contains two letters, which represents its period and height. Two consecutive contractions as a sequential itemset are described as follows:

$$(wX \cup yZ)$$

where w and y are discretized heights and X and Z are discretized periods. The rule set process returns a set of rules (R) as the output in the form of

$$wX \rightarrow yZ.$$

Frequent sequential itemsets are searched for each individual patient separately (local frequent sequential itemsets). Multiple discretized contractions series are obtained from HT, which contains patients intrauterine pressure tracings. Afterwards overall global frequent itemsets are generated based on the local frequent sequential itemsets. In order to

evaluate the interestingness of the sequential association rule, three measurements and thresholds are defined. Definition 1 (supL): the local support value. Let N_c be the total number of contractions in one patient's intrauterine pressure tracing. Let σ(wX ∪ yZ) be the occurrences of the local sequential itemset wX ∪ yZ. SupL is defined as follows:

$$supL\left(wX \rightarrow yZ = \frac{\sigma\ (\mathrm{wX}\ \cup \mathrm{yZ})}{N_c}\right).$$

Definition 2 (minSupL): the minimum support value for deciding the local frequent sequential itemsets. The itemset is considered as a local frequent sequential itemset of the patient if the frequency of one itemset in one patient's discretized tracings, i.e., supL is no less than minSupL. Definition 3 (supG): the global support value. Let M be the total number of patients in the HT. Let N_p be the number of patients who have the sequential itemset as one of the local frequent itemsets. SupG is defined as follows:

$$supG\ (wX \rightarrow yZ) = \frac{N_p}{M}.$$

Definition 4 (minSupG): The minimum support value for deciding the global frequent sequential itemsets. The itemset is considered as a global frequent sequential itemset of the patient if the percentage of those patients who share one local frequent sequential itemset, i.e., supG, is no less than minSupG. Definition 5 (PS): Piatetsky-Shapiro, the interestingness measure. The rule:

$$dX \rightarrow yZ.$$

is derived from a global frequent sequential itemset. Let P(wX ∪ yZ) be the probability of wX ∪ yZ, and P(wX) and P(yZ) be the probabilities of wX and yZ. PS is defined as:

$$PS(wX \rightarrow yZ) = P(wX \cup \mathrm{yZ}) - \mathrm{P(wX)P(yZ)}.$$

Definition 6 (minPS): the minimum PS measure. The rule is considered interesting if the PS measure of a sequential association rule derived from a global frequent sequential itemset is no less than minPS. The proposed sequential association rule mining approach consists of three steps. At first, it generates local frequent sequential itemsets, which are items that occur no less than minSupL in each patient's tracing. Second, it generates global frequent sequential itemsets. Third, it generates sequential association rules from the frequent global itemsets. The first step is similar to Apriori algorithm, where a set of

candidate m-itemsets is generated connecting the frequent (m-1)-itemsets generated in the previous iteration. Concerning Apriori algorithm, the sequence is taken into consideration. In the second phase, the local frequent sequential itemsets derived from all the historical patient's tracings and generating candidate global frequent m-itemsets are combined based on local frequent m-itemsets. Whereas if global support of one candidate m-itemset is no less than minSupG, it is recognized as a frequent global m-itemset. In the third phase only one item consequent sequential association rules are generated, i.e., one contraction in the consequent part since the plan is to predict the occurrence of the next contraction. It is recognized as an interesting sequential association rule if the PS measure of the derived rule is no less than minPS. Rule set R is formed from all lengths of the interesting sequential association rules. The larger the PS value the more interesting the rule is i.e., when the condition part occurs the chance for the consequent part occurring is high. [Huang *et al.,* 2013, pp. 1293-1294]

Rule matching: by selectively employing available contractions from both HT and the current patient's most recent intrauterine pressure tracing to train the prediction model, is the purpose of the collaborative training dataset selection process. The prediction model is made more adaptive to the changing contraction of the patient with this dynamic selection. The inputs for this process are as follows: first, the current patient's most recent contraction pattern, and second, the rules set R extracted from HT. The rules ,in R, are sorted with PS values in descending order. The rule matching process is to compare the current patient's recent contraction pattern with the condition part of the generated sequential rules. The first matched rule is kept for instance selection. Sliding window technique, as described by [Koç, 1995], is an exponentiation algorithm used for partition strategies, and it is used to form the selected training dataset in the intrauterine pressure sequence of the current patient's most recent C_n contractions in d, if no matched rule is found by searching the entire rule set. If there is a matched rule, i.e., the current patient's contraction pattern matches with the condition part of one rule, the same number C_n of contraction tracings from the HT are selected. These correspond to the consequent part of the rule, and in turn are combined with the current patient's most recent tracing of C_n contractions in d to form the selected training dataset SD using the sliding window technique. Usually, the number of contraction tracings that follow one interesting sequential association rule is much larger than C_n. Euclidean distance, which measures the distance between two points [Han *et al.,* 2011, pp. 72], is calculated between the current patient's recent feature vector and each of the feature vectors in HT that follows the selected sequential association rule. The feature vector of one contraction contains the normalized period and normalized height of the corresponding peak. The smallest distance C_n contraction tracings are selected. The selected training dataset SD is one of the inputs of the k-NN based LS-SVM component for long-term time series prediction modelling.

This proposed CF approach selectively utilizes the data from HT and combines them with the current patient's most recent intrauterine contraction tracing. The sequential association rule-based dataset selection enables incorporating similar patients recommended patterns. However, using the current patient's most recent intrauterine contraction tracing helps to discover and preserve the current patient's own contraction patterns. These are different from the sequential contraction patterns learned from the database, and the combination of these two sources facilitates adaptive prediction and enhances the robustness of the obtained model and its prediction ability. [Huang *et al.,* 2013, pp. 1294]

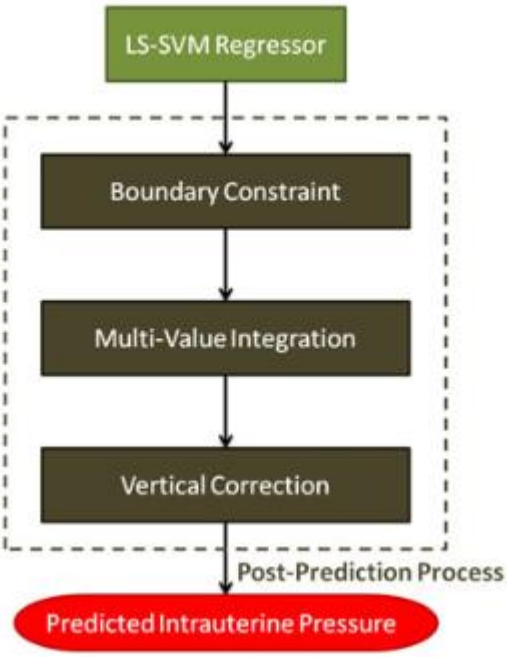

Figure 5: Postprediction process [Huang *et al.,* 2013, pp. 1294]

In the Figure 5 the postprediction process consists of boundary constraint, multivalue integration and vertical correction components. These are used to postprocess the prediction results, further improve the prediction precision, and ensure valid results. The boundary constraint is used to bound the predicted values rendered by LS-SVM regressors. The prediction model might not be able to capture the changing patterns of the time series very well when the prediction horizon is very large. Real world time series data should fall within a certain reasonable range. Multivalue integration component is used to combine prediction results from individual models. Models which are not suggested to be combined are very similar with each other by having access to the same information set and capturing similar patterns. However, it is also important to combine forecasts from very similar models. Utilizing multiple models for estimation and prediction could reduce uncertainty. Vertical correction component consists of two steps: first to smooth out the prediction results from the multivalue integration component. Second, to smooth out the sharp pulses and peaks with very low heights. The smoothing in the first step can reduce

inconsistency of the models, which combines data subsampling, peak detection, and interpolation processes. In the second step, three conditions are set to determine if a peak should be smoothed out or not.

Condition 1: if the height of a peak is lower than a given threshold, minHeight, it is considered as candidate noise.

Condition 2: if the width of a peak is smaller than a given threshold, minWidth, it is considered as candidate noise.

Condition 3: for the candidate noise that satisfies either one of the previous conditions, if the starting point value of the peak is smaller than a threshold, minSPV, it is considered as noise.

In order to smooth out the noise peak, the peak area is reset to be the value of the starting point value of the peak. The vertical correction component generates the final prediction results as an output. These are the sequences of the intrauterine pressure which are predicted multiple seconds ahead. [Huang *et al.,* 2013, pp. 1294-1295]

The prediction model was compared to two existing methods, LS-SVM and LL-MIMO (Long Term Multi-Input Multi-Output). The proposed model's purpose is to predict the coming contractions ahead of time in order to relieve labor pain with an analgesic injection. The dataset consists of samples from 2041 women. Two error measurements were used to measure the performance of the prediction model, RMSE (root mean squared error) and SMAPE (symmetric mean absolute percentage error). The experimental results indicate that the proposed prediction model is better on average compared to the above two methods (LS-SVM, LL-MIMO). In the postprediction process, prediction accuracy was 40.3% higher than LS-SVM and 76.8% higher than LL-MIMO. [Huang *et al.,* 2013, pp. 1295-1296]

6 Classifier performance metrics

Evaluating classifier performance can be measured in several ways: accuracy or recognition rate, sensitivity or recall, specificity, precision, F1 (harmonic mean of precision and recall) and Fβ (where β is a non-negative real number). Accuracy as defined by [Han *et al*., 2011, pp. 366] is used to measure a classifiers correctly classified percentage of a given test set tuples (samples), with the given formula:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}} 100\%$$

where true positive (TP) refers to the number of positive tuples which are correctly classified by the classifier, true negative (TN) refers to negative tuples which are correctly classified, and positive (P) and negative (N) refers to number of positive and negative tuples. Along with true positive and true negative there are opposite terms called false positive (FP) and false negative (FN) which both refers to incorrectly classified tuples.

Sensitivity or recall refers to true positive rate:

$$\frac{\text{TP}}{\text{P}} 100\%$$

which is calculated as true positive divided by number of positives, while specificity refers to true negative rate:

$$\frac{\text{TN}}{\text{N}} 100\%$$

in which true negative is divided by number of negatives.

Precision is calculated as:

$$\frac{\text{TP}}{\text{TP} + \text{FP}} 100\%$$

where true positive is divided with the sum of true positive and false positive.

For measuring a given classifiers error rate or misclassification rate:

$$\frac{\text{FP} + \text{FN}}{\text{P} + \text{N}} 100\%$$

can be calculated by dividing the sum of false positive and false negative with the sum of positive and negatives. [Han *et al*., 2011, pp. 364-366]

Classes	Predicted buys_computer = yes	Predicted buys_computer = no	Total	Recognition (%)
Actual buys_computer = yes	6954	46	7000	99.34
Actual buys_computer = no	412	2588	3000	86.27
Total	7366	2634	10000	95.42

Table 3: Confusion matrix for the classes buys_computer = yes and buys_computer = no [Han *et al*., 2011, pp. 366]

The terms of true positive, false negative (first row), false positive and true negative (second row) are summarized in Table 3 as a confusion matrix for classes buys_computer = yes and buys_computer = no. The idea is that with a quick glance a researcher can analyze how well a classifier is getting things right with true positive and true negative, and wrong with false positive and false negative. Buys_computer = yes with 6954 tuples determines true positive and buys_computer = no with 2588 tuples determines true negative. False positive happens when a classifier classifies a given tuple incorrectly as buys_computer = yes when it is a false positive as buys_computer = no, which is marked in the confusion matrix with 412 tuples. The same goes with false negative when classifier incorrectly classifies something as false when it is true, which is marked in the confusion matrix with 46 tuples. Ideally false positives and false negatives are around zero for the classifier to be considered accurate. Recognition consists of accuracy, which is shown as 95.42%, sensitivity with 99.34% and specificity with 86.27%.

Estimating the accuracy of a given classifier is better suited for test sets rather than tuples which were used to train the model in order to avoid misleading estimates due to overspecialization. Confusion matrixes can be drawn for multiple classes rather than for only two in the example shown in Table 3. [Han *et al*., 2011, pp. 364]

7 Implementation

7.1 Methodology

This research uses quantitative research method to mine sequential patterns from calcium peak data by using a study "Knowledge-assisted sequential pattern analysis framework" proposed by Huang *et al*. [2013] as a rough example.

7.2 Calcium peak data

According to the previous study the calcium signal data was recorded in fall of 2013 from various individuals, from which the region of interest was collected from between 1 and 5 different areas where cells were beating spontaneously. Two different groups were tested with different intervals: one with every recording happening between 0.043-0.044 seconds and the second with 0.088-0.089 seconds. Signal length varied from 11 to 24 seconds between signals and there are in total 102 different signals from three subjects. Each signal is measured along with time in seconds, and as amplitude values called ratio values. [Juhola *et al*., 2014; Juhola *et al*., 2015]

7.3 Preprocessing phase

This research is conducted with MATLAB R2020 version. MATLAB was chosen from among a few different programming platforms, mainly because of previous experience, although R and Python were considered as alternatives. Afterwards during the research, it was concluded that MATLAB is not the most viable programming language for this kind of study, since it lacks built-in algorithms for sequential pattern mining and sequential rule mining, which were necessary to conduct this study. A viable alternative was found "The SPMF Open-Source Data Mining Library", which was developed by Fournier-Viger *et al.* [2016]. The library contains 254 data mining algorithms, from which we found TRuleGrowth algorithm that was compatible with our dataset.

The first objective was to extract data from multiple excel files, each containing a single signal, into raw data containing the previously mentioned two variable columns: time and (ratio) value. The signals were detrended to remove a linear trend, like in the previous study, by using MATLAB's detrend function. Next, the detrended signals were further processed by removing negative values caused by the detrend function, by raising every signal individually by its minimum value up to zero (zero minimum). Afterwards the two variables were further processed into sequential pairs of letters, which is based on previously reviewed study by Huang *et al*. [2013]. The letters assigned for time values are small letters (a, b, c, d, etc.) and for ratio values as capital letters (A, B, C, D, etc.).

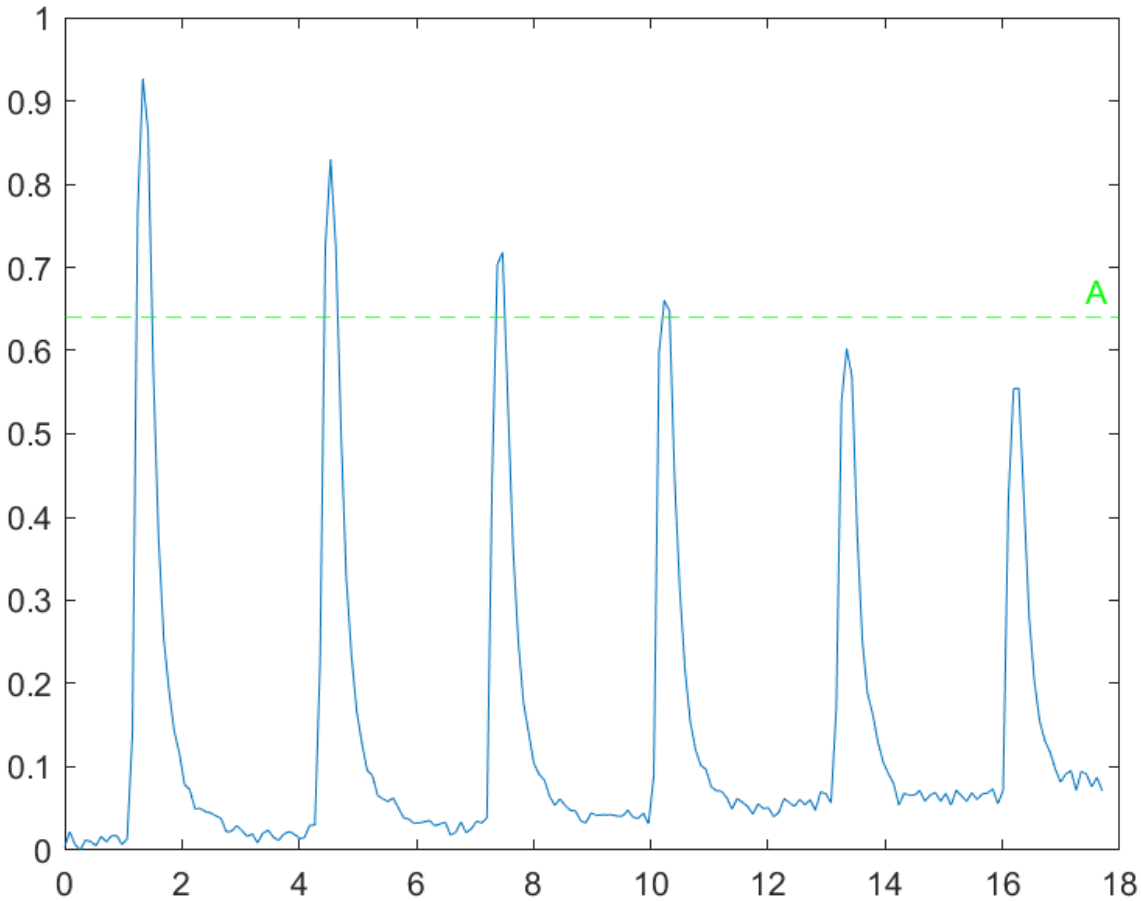

Figure 6: Rough estimate A for peaks in a detrended, zero minimum signal

In Figure 6 an amplitude A was calculated for every signal individually. The values were sorted and the means of the lowest 10% of values were subtracted from the highest 10% in order to calculate the average amplitude A. The signals were first pre-treated by removing a linear trend by using the MATLAB's detrend function and furthermore raising the minimum value to zero (zero minimum).

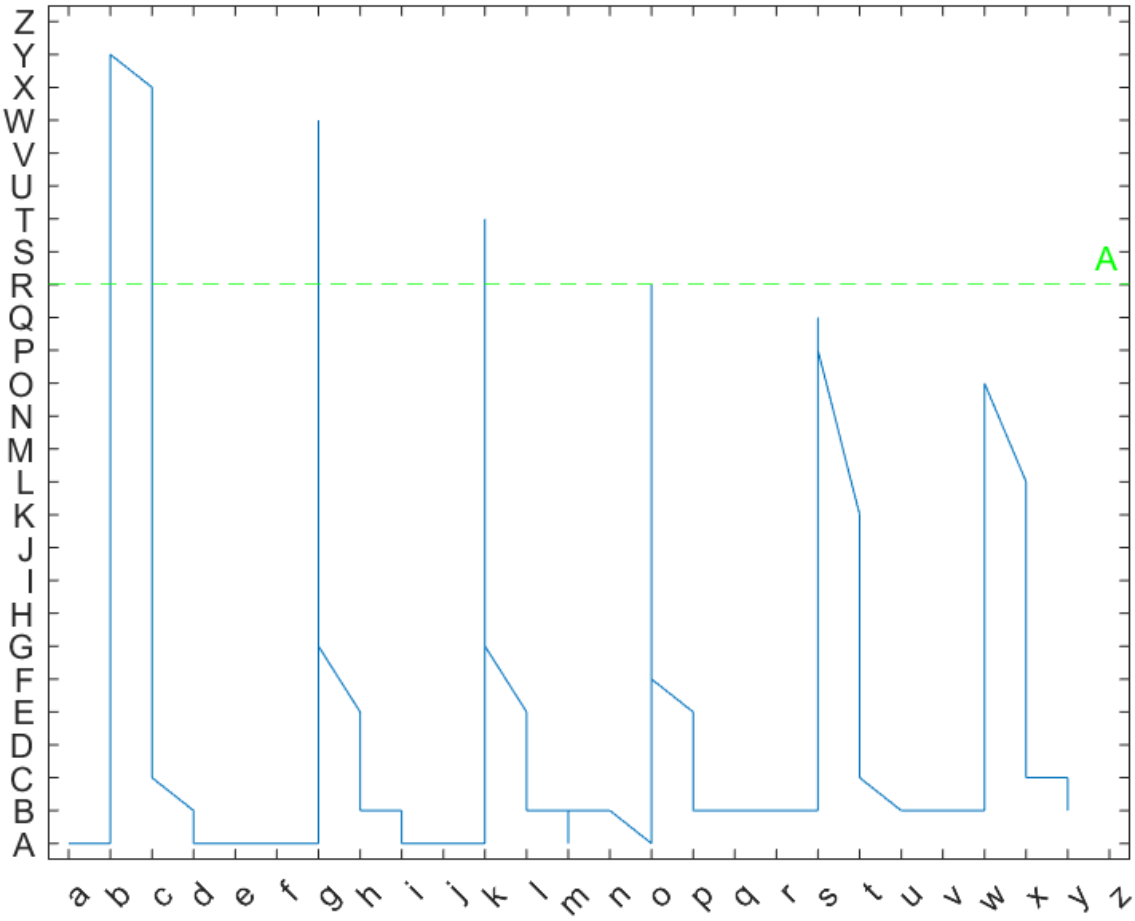

Figure 7: Discretized time and ratio values into categorical values

Figure 7 shows the same signal as illustrated in Figure 6 with the exception in which the time and ratio values have been categorized into small and capital letters. Small letters in the x-axis are categorized from a to z, and capital letters in the y-axis from A to Z. These two values together create categorical value pairs from which we can mine sequential patterns. The problem is to find what is classified as a 'peak'. When does it start, reaches its maximum height, and when does it end, and when does the next peak start? Two adjacent calcium peaks can be called a sequential itemset, from which we can calculate, with the help of rough estimate A, when the previous peak ends and a new one starts.

The categorical values were split evenly between the time and ratio as in both having twenty-six letters from a to z and A to Z, but this was purely experimental decision based on the calculations between the various signal lengths in between time and ratio. Some signals were over twice as long as others and the height of the peaks varied greatly as well. A constant time and ratio were calculated from the longest and largest signal ratio wise so that the categories would not run out with varying signal lengths.

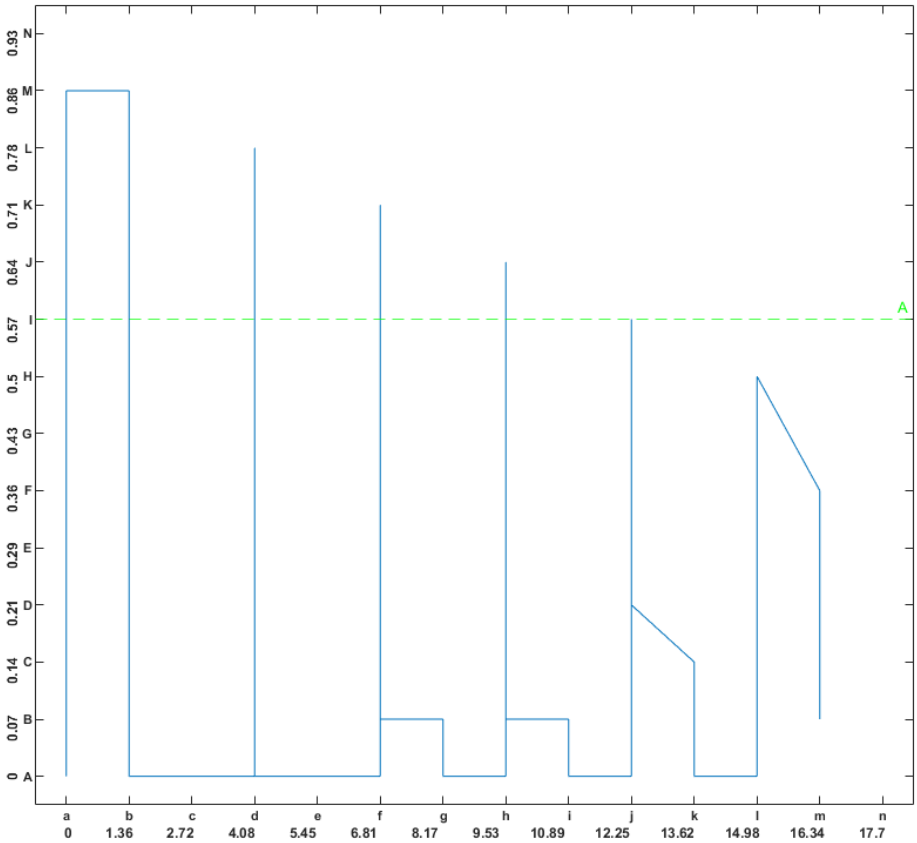

Figure 8: Discretized time and ratio values into categorical values (14 by 14 categories)

Later during the research, the twenty-six categories were reduced to the smaller quantity of fourteen categories. The research problem of considering how many categories is too few or too many was done at first by a rough estimate. It was concluded that there have to be enough discretized categories so the varying signal peaks can be found, yet too many feel unnecessary to capture all the signal peaks needed. The maximum ratio value was calculated as 0.9268, while the lowest was calculated as 0.1225. The highest maximum ratio in a signal was over seven times bigger compared to the maximum ratio in the lowest signal. With this calculation, fourteen categories is the bare minimum required to detect peaks in the smallest signal. The values in the x- and y-axis in Figure 8 are shown with two decimal places along with their category letter.

Time range	Categories
Time < 1.3615	a
1.3615 ≤ Time < 2.7231	b
2.7231 ≤ Time < 4.0846	c
4.0846 ≤ Time < 5.4462	d
5.4462 ≤ Time < 6.8077	e
6.8077 ≤ Time < 8.1692	f
8.1692 ≤ Time < 9.5308	g
9.5308 ≤ Time < 10.8923	h
10.8923 ≤ Time < 12.2538	i
12.2538 ≤ Time < 13.6154	j
13.6154 ≤ Time < 14.9769	k
14.9769 ≤ Time < 16.3385	l
16.3385 ≤ Time < 17.7	m
Time ≥ 17.7	n

Table 4: Time categorized into fourteen categories

In Table 4 time is categorized into fourteen categories, from a to n. The value range is calculated by dividing the longest signal, with the number of categories.

Ratio range	Categories
Ratio < 0.0713	A
0.0713 ≤ Ratio < 0.1426	B
0.1426 ≤ Ratio < 0.2139	C
0.2139 ≤ Ratio < 0.2852	D
0.2852 ≤ Ratio < 0.3565	E
0.3565 ≤ Ratio < 0.4278	F
0.4278 ≤ Ratio < 0.4991	G
0.4991 ≤ Ratio < 0.5704	H
0.5704 ≤ Ratio < 0.6417	I
0.6417 ≤ Ratio < 0.713	J
0.713 ≤ Ratio < 0.7843	K
0.7843 ≤ Ratio < 0.8556	L
0.8556 ≤ Ratio < 0.9269	M
Ratio ≥ 0.9269	N

Table 5: Ratio categorized into fourteen categories

In Table 5 the ratio is categorized in the same way as time is in Table 4 by having the same number of categories. This is done in order for the smallest signal to have enough categories to detect peaks. Fourteen categories were the lowest number of categories possible to detect peaks in the smallest of signals, ratio wise.

In Figure 8 we can see six peaks. The peak maxima can be seen at: bM, dL, fK, hJ, jI and lH. In this case the signals first 'peak' can be seen as starting at: aA, rising all the way towards the maximum height at bM, and ending at bA. The resulting peak yields us a subsequence:

$$ss1 = (\{aA\}, \{bM\}, \{bA\})$$

The resulting subsequence is a simplified version, the end result, and there are many more categories from which we have to calculate what constitutes a peak from between the peak start to its end. The next peak would yield us a subsequence:

$$ss2 = (\{dA\}, \{dL\}, \{dA\})$$

where the peak starts at dA, reaches its maximum height at dL, and ends at dA. A sequential itemset would be the union of these two subsequences:

$$ss3 = (\{aA\}, \{bM\}, \{bA\}, \{dA\}, \{dL\}, \{dA\})$$

Itemset	Amount
aA	13
aB	1
aK	1
aM	1
bM	1
bI	1
bF	1
bD	1
bC	2
bB	3
bA	6
cA	15
dA	3
dD	1
dK	1
dL	1
dK	1
dH	1
dE	1
dD	1
dC	1
dB	3
dA	2

Table 6: Complete itemsets of the first two peaks of a signal

In Table 6 we have all the itemsets of Figure 8 signal for the first two peaks from the beginning of the signal to the end of the second peak. We can see that before the first peak starts there are thirteen {aA} itemsets, from which the peak gradually rises towards the maxima at {bM} and ends at {bA}. The second peak similarly starts at {dA} having three itemsets, rises towards the maxima at {dL} and ends at {dA}. The entire subsequence would look like:

$$ss4 = (({aA}^{13}, {aB}, {ak}, {aM}, {bI}, {bF}, {bD}, {bC}^{2}, {bB}^{3}, {bA}^{6}),$$
$$({dA}^{3}, {dD}, {dK}, {dL}, {dK}, {dH}, {dE}, {dD}, {dC}, {dB}^{3}, {dA}^{2}))$$

where the number after itemset, for example $\{aA\}^{13}$, means how many itemsets, recording intervals, there are in a row. The recording intervals of the calcium peak data are mentioned in 7.2.

7.4 Candidate sequential itemsets

In order to find out meaningful sequential patterns from the data, we must first generate candidate sequential itemsets. We know that the data consists of various amounts of calcium peaks, and those peaks have a beginning, peak maxima, and peak end. With this information we first generate every single peak candidate that follows the following rules:

1. Starting from the beginning of the signal, find out the beginning of a peak before upwards trend begins
2. Find out the current peak maxima, by following the upwards trend towards the maximum ratio value
3. Find out the current peak end, by following the downward trend, from peak maxima, until the first lowest possible value is found between current peak end and next peak start
4. Repeat to the end of the signal until all peak candidates have been found

ID	Peak candidate
1	({aA}, {bM}, {bA})
2	({dA}, {dL}, {dA})
3	({fA}, {fK}, {gA})
4	({hA}, {hJ}, {iA})
5	({jA}, {jI}, {kA})
6	({kA}, {kB}, {lA})
7	({lA}, {lB}, {lA})
8	({lA}, {lB}, {lA})
9	({lA}, {lH}, {mB})

Table 7: Candidate itemsets of a signal

In Table 7 we have nine different peak candidates of a signal, which is shown in Figure 8. Each valid peak candidate calculated thus far must have had a peak beginning, which is a value right before an upwards trend starts, a peak maxima which is the highest ratio value before a downwards trend starts, and a peak end which is the first lowest possible value found. With this criteria ratio values that cannot get past category ‘A’, which is shown in Table 5, are automatically considered as noise. However, we can see that there are still unnecessary peak candidates, which has to be removed. Namely peak candidates 6, 7, and 8 are considered noise since those peak candidates hover between two

ratio categories ‘A’ and ‘B’. In order to remove these candidates, threshold T was calculated for each signal individually based on the average amplitude A, which is shown in Figure 6. For this particular signal in Table 7, threshold T included only those peak candidates where the peak maxima were greater than ‘B’.

$$peak\ max > (\mathrm{T} = A * 0.2)$$

where amplitude A is ‘I’, which is shown in Figure 8. The letters were mapped into positive whole numbers, starting from one, and threshold T was rounded towards the nearest greater integer value, which was in this case ‘2’. Threshold T was found by experimenting with different threshold factors, where the intent was to remove only particular noise, which did not correspond to a signal peak.

ID	Peak candidate
1	({aA}, {bM}, {bA})
2	({dA}, {dL}, {dA})
3	({fA}, {fK}, {gA})
4	({hA}, {hJ}, {iA})
5	({jA}, {jI}, {kA})
6	({iA}, {iH}, {mB})

Table 8: Candidate itemsets of a signal after removing further noise

In Table 8 we have the updated peak candidates, where the noise between small ratio values, which were first treated as peak candidates, have been removed.

In order to develop Sequential Association Rule Mining Algorithm approach [Huang et *al*., 2013] to work with our data, which is explained in 5.1, we need to further preprocess our candidates by calculating the time range of each candidate peak beginning from the next peak beginning, and denoting a new mapping table for time that is appropriate for the sequential itemset definition:

$$(\mathrm{wX} \cup \mathrm{yZ})$$

where, in our case, w and y are discretized time values and X and Z are discretized ratio values. In our data the above description corresponds to two consecutive signal peaks.

Time range	Categories
Time < 1.3615	a
1.3615 ≤ Time < 2.7231	b
2.7231 ≤ Time < 4.0846	c
4.0846 ≤ Time < 5.4462	d
5.4462 ≤ Time < 6.8077	e

Table 9: New table for peak beginning time difference

In Table 9 we calculated the largest found candidate peak beginning time difference and denoted small letters from 'a' to 'e' as our new categories. With this calculation, the old calculated constant time (1.3615) denoted for our categories is still valid. There were no time differences larger than 6.8077 found in our dataset, and thus the rest of the categories are no longer needed.

ID	Peak candidate
1	{dM}
2	{cL}
3	{cK}
4	{cJ}
5	{cI}
6	{bH}

Table 10: Further discretized candidate itemsets

In Table 10 the candidate itemsets of a signal from Table 8 have been further modified, in which the first small letter of the itemset signifies the time difference of the current candidate peak beginning from the consequent peak candidate beginning. The capital letter definition remains the same as before, and the value range is shown in Table 5. The entire signal with the candidate peaks would be:

$$sequence = (\{dM\}, \{cL\}, \{cK\}, \{cJ\}, \{cI\}, \{bH\}).$$

After all the peak candidates have been calculated from our signal's dataset, the final phase of this research is to find out how to calculate local (within a signal) sequential itemsets, and afterwards the global (from all signals) sequential itemsets and find out frequent sequential itemsets which satisfy the minimum, user given, support and confidence.

7.5 Generating sequential association rules

In order to start generating sequential association rules, we must decide which algorithm to use in our research. Apriori algorithm in 3.2 cannot be used since it does not consider

the sequence of events within our dataset. MATLAB unfortunately does not contain built-in sequential rule mining algorithms, and the Sequential Association Rule Mining Algorithm proposed by Huang *et al.* [2013] is not very well explained. Creating an entire algorithm in MATLAB from beginning is not feasible within the scope of this thesis.

From various different sequential rule mining algorithms, the most suitable for our candidate peak dataset is TRuleGrowth algorithm [Fournier-Viger *et al.,* 2012].

TRuleGrowth algorithm is a variation of RuleGrowth, in which it accepts window size parameter as a constraint. RuleGrowth works by first finding 1*1 rules, recursively growing them by scanning sequences containing said rules to find single items that can expand their left or right parts. The two processes for expanding rules are called left and right expansion where adding item to the left side is formally defined as:

$$X \cup \{i\} \Rightarrow Y$$

and right side:

$$X \Rightarrow Y \cup \{i\}.$$

Any rule obtained by an expansion has a support lower or equal of the original rule. All the rules where support is at least minimum support can be found recursively. The parameters accepted by the algorithm are minimum support, minimum confidence, window size time constraint, and optional parameters are max antecedent size and max consequent size. In our case we only care about two consecutive peaks happening in our dataset, and we can leave these optional parameters unused. In order to find out two consecutive peaks locally within a single signal ($wX \cup yZ$) with window size $= 1$, our sequences would be as follows:

ID	($wX \cup yZ$)
1	({dM}, {cL})
2	({cL}, {cK})
3	({cK}, {cJ})
4	({cJ}, {cI})
5	({cI}, {bH})

Table 11: All sequential itemsets of a signal

In Table 11 we have all the sequential itemsets of a signal, which is shown in Figure 8. However, the rules generated from this particular signal are not very interesting.

Rule: (wX ==> yZ)	Support	Confidence
cI ==> bH	1	0.5
cJ ==> cI	1	0.5
cK ==> cJ	1	0.5
cL ==> cK	1	0.5
dM ==> cL	1	1

Table 12: Rules of a signal with support 0.2 and confidence 0.5

In Table 12 we have generated every single rule (wX==>yZ) , where our user defined minimum support is 0.2, and minimum confidence is 0.5. The definitions of minimum support and confidence are explained in 3.1. This signal does not have many peaks from which to draw meaningful rules.

We have another sequence:

$$sequence = (\{aE\},\{aE\},\{aE\},\{bE\},\{aE\},\{aD\},\{bE\},\{aE\},\{aD\},\{bD\},\{aD\},\{aD\},\{bD\},\{aD\},\{aD\},\{aD\},\{bD\},\{aD\},\{aD\},\{aC\},\{bC\},\{aC\},\{aD\},\{bD\},\{aC\},\{aC\},\{aC\},\{bC\},\{aC\},\{aC\},\{aC\},\{bC\},\{aC\},\{aC\},\{aC\},\{bC\},\{aB\},\{aC\},\{aC\},\{aC\},\{bC\},\{aC\})$$

which have the most amount of candidate peaks, 42, in the entire dataset. We generate the following rules:

Rule: (wX ==> yZ)	Support	Confidence
aC ==> bC	5	0.238
aD ==> bD	4	0.25
bC ==> aC	4	0.4
bD ==> aD	3	0.375

Table 13: Rules of a signal with support 0.05 and confidence 0.05

In Table 13 we have a sequence with the most amount of candidate itemsets found in the entire dataset of 102 signals.

Generating the global sequential itemsets from all signals yields us the following results:

Rule: (wX ==> yZ)	Support	Confidence
aB ==> bB	27	0.574
bB ==> aB	26	0.722
aC ==> bC	37	0.685
bC ==> aC	39	0.75
aD ==> bD	30	0.698
bD ==> aD	34	0.642
aE ==> bE	21	0.778

Table 14: Global rules of all 102 signals with 0.2 support and 0.5 confidence

In Table 14 we have calculated with support of 0.2 and confidence of 0.5 frequent sequential itemsets from 102 signals in total. We can see that the overall most frequent and meaningful rule with support of 39 and confidence of 0.75 is when the antecedent bC is followed by the consequent aC, which would give us:

$$IF\ 1.3615 \leq \text{Time} < 2.7231\ \&\ 0.1426 \leq \text{Ratio} < 0.2139\ \text{THEN Time} < 1.3615\ \&\ 0.1426 \leq \text{Ratio} < 0.2139.$$

8 Conclusions

The rule wX ==> yZ, where we want to find two consecutive peaks (wX ∪ yZ), which makes a sequential itemset, is not generating meaningful and strong sequential association rules. Some signals have too low of a count of candidate itemsets to calculate anything beyond 0.2 support or 0.5 confidence at most. The reason for this could be if our candidates are generated poorly with not narrow enough criteria for either time or ratio. The time was narrowed into five categories, where the current peak beginning was calculated from the consequent peak beginning. Ratio was kept at 14 different categories, since the ratio values were over seven times bigger in the tallest signals compared to the smallest signals. The generated sequential rules might have generated more meaningful rules if these discretized heights were narrowed down from 14 closer to five.

More kinds of rules could have been tried to mine from our dataset of 102 different signals, for example, by using more time constraints with the generated candidate peaks. The research came to halt when the association rules package for MATLAB was not viable for sequential data. Fortunately, a viable replacement algorithm program package was found. This study would probably have been concluded more quickly if the preprocessing phase were done with something other than MATLAB. For example, Python has several popular packages for mining sequential patterns or sequential rules.

Conducting this kind of study for the first time, when there was no prior knowledge or experience about sequential pattern mining, made this endeavor challenging.

The topics until the implementation phase were done well, but due to time constraints the actual results could have been studied much more than what was accomplished in the end. For example, implementing and using more sequential pattern algorithms, to mine frequent patterns.

Over 1400 lines of MATLAB code, along with comments, was created all the way up to the point where all the candidate peaks were created, according to the criteria mentioned by Huang *et al.* [2013]. Afterwards the dataset was transformed to fit the criteria required by the SPMF-program [Fournier-Viger *et al.,* 2016].

9 References

Agrawal, R. Imieliński, T. and Swami, A. 1993. *Mining association rules between sets of items in large databases*. In Proceedings of the 1993 ACM SIGMOD international conference on Management of data, pp. 207-216.

Agrawal, R. and Srikant, R. 1994. *Fast algorithms for mining association rules*. In Proc. 20th int. conf. very large data bases, VLDB Vol. 1215, pp. 487-499.

Agrawal, R. and Srikant, R. 1995. *Mining Sequential Patterns*. In Proceedings of the Eleventh International Conference on Data Engineering IEEE, pp. 1-22.

Ca2+. 2013. *Handbook of Laboratory and Diagnostic Tests*. Retrieved September 28, 2017, from *http://medical-dictionary.thefreedictionary.com/Ca2%2b*

Fournier-Viger, P. Wu, C. W. Tseng, V. S. & Nkambou, R. 2012. *Mining sequential rules common to several sequences with the window size constraint*. In Advances in Artificial Intelligence: 25th Canadian Conference on Artificial Intelligence, Canadian AI 2012, Toronto, ON, Canada, May 28-30. Proceedings 25, pp. 299-304.

Fournier-Viger, P. Lin, C.W. Gomariz, A. Gueniche, T. Soltani, A. Deng, Z. Lam, H. T. 2016. *The SPMF Open-Source Data Mining Library Version 2*. Proc. 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016) Part III, Springer LNCS 9853, pp. 36-40.

Fournier-Viger, P. Lin, J. C. W. Kiran, R. U. Koh, Y. S. and Thomas, R. 2017. *A survey of sequential pattern mining*. Data Science and Pattern Recognition, 1(1), pp. 54-77.

Gladence, L. M. Ravi, T. and Karthi, M. 2014. *Heart Disease Prediction using Naive Bayes Classifier–Sequential Pattern Mining*. International Journal of Applied Engineering Research (IJAER).

Han, J. Kamber, M. and Pei, J. 2011. *Data Mining: Concepts and Techniques*. Burlington, MA, Elsevier, 3rd ed.

Hsueh, S. C. Lin, M. Y. and Chen, C. L. 2008. *Mining negative sequential patterns for e-commerce recommendations*. In 2008 IEEE Asia-Pacific Services Computing Conference, pp. 1213-1218.

Huang, Z. Shyu, M. L. Tien, J. M. Vigoda, M. M. and Birnbach, D. J. 2013. *Prediction of Uterine Contractions Using Knowledge-Assisted Sequential Pattern Analysis*. IEEE Transactions on Biomedical Engineering, 60(5), pp. 1290-1297.

Huang, Z. Shyu, M. L. Tien, J. M. Vigoda, M. M. and Birnbach, D. J. 2014. *Knowledge-Assisted Sequential Pattern Analysis with Heuristic Parameter Tuning for Labor Contraction Prediction*. IEEE Journal of Biomedical and Health Informatics, 18(2), pp. 492–499.

Juhola, M. Joutsijoki, H. Varpa, K. Saarikoski, J. Rasku, J. Iltanen, K. Laurikkala, J. Hyyrö, H. Avalos-Salguero, J. Siirtola, H. Penttinen, K. and Aalto-Setälä, K. 2014. *On Computation of Calcium Cycling Anomalies in Cardiomyocytes Data*. 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 1444-1447.

Juhola, M. Penttinen, K. Joutsijoki, H. Varpa, K. Saarikoski, J. Rasku, J. Siirtola, H. Iltanen, K. Laurikkala, J. Hyyrö, H. Hyttinen, J. and Aalto-Setälä, K. 2015. *Signal Analysis and Classification Methods for the Calcium Transient Data of Stem Cell-Derived Cardiomyocytes*. Computers in Biology and Medicine, 61, pp. 1-7.

Juhola, M. Joutsijoki, H. Penttinen, K. and Aalto-Setälä, K. 2018. *Detection of genetic cardiac diseases by Ca 2+ transient profiles using machine learning methods*. Scientific reports, 8(1), pp. 1-10.

Juhola, M. Joutsijoki, H. Penttinen, K. and Aalto-Setälä, K. 2019. *Machine learning to differentiate diseased cardiomyocytes from healthy control cells*. Informatics in Medicine Unlocked, 14, pp. 15–22.

Koç, C. K. 1995. *Analysis of sliding window techniques for exponentiation*. Computers & Mathematics with Applications, pp. 17–24.

Kujala, K. Paavola, J. Lahti, A. Larsson, K. Pekkanen-Mattila, M. Viitesalo, M. Lahtinen, A. Toivonen, L. Kontula, K. Swan, H. Laine, M. Silvennoinen, O. and Aalto-Setälä, K. 2012. *Cell Model of Catecholaminergic Polymorphic Ventricular Tachycardia Reveals Early and Delayed Afterdepolarizations*. PloS One, 7(9).

Linden, G. Smith, B. and York, J. 2003. *Amazon. com recommendations: Item-to-item collaborative filtering*. IEEE Internet computing, 7(1), pp. 76–80.

Masseglia, F. Poncelet, P. and Teisseire, M. 2009. *Efficient Mining of Sequential Patterns with Time Constraints: Reducing the Combinations*. Expert Systems with Applications: An International Journal, 36(2), pp. 2677-2690.

Miranda, R. 2003. Encyclopedia of Physical Science and Technology. Academic Press, Third Edition, pp. 465-475.

Mooney, C. H. and Roddick, J. F. 2013. *Sequential pattern mining--approaches and algorithms*. ACM Computing Surveys (CSUR), 45(2), pp. 1-39.

Srikant, R. and Agrawal, R. 1996. *Mining Sequential Patterns: Generalizations and Performance Improvements*. In: International Conference on Extending Database Technology, Springer, Berlin, Heidelberg, pp. 1-17.

Uragaki, K. Hosaka, T. Arahori, Y. Kushima, M. Yamazaki, T. Araki, K. and Yokota, H. 2016. *Sequential pattern mining on electronic medical records with handling time intervals and the efficacy of medicines.* In 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 20-25, IEEE.

Wang, Y. Hale, L. Hill, K. and Singh, S. 2006. *Mining Invariants in Biological Sequences*. In: Industrial Conference on Data Mining-Workshops, pp. 111-118.

Wright, A. P. Wright, A. T. McCoy, A. B. and Sittig, D. F. 2015. *The use of sequential pattern mining to predict next prescribed medications*. Journal of biomedical informatics, *53*, pp. 73-80.

Zhao, Q. and Bhowmick, S. S. 2003. *Sequential Pattern Mining: A Survey*. ITechnical Report CAIS Nayang Technological University Singapore, 1, pp. 1-26.