Diploma Thesis

Mining Associations Between Subsequences in Time-series

Submitted by:

Dominic Stange

August 11^{th}, 2011

First Thesis Advisor: Prof. Dr. habil. Rudolf Kruse

University of Magdeburg
School of Computer Science
Department of Knowledge and
Language Processing
Universitätsplatz 2, D-39106 Magdeburg

Second Thesis Advisor: Dr. Martin Spott

Business Modelling and Operational Transformation Practice
BT Research and Technology
Adastral Park, Martlesham
Ipswich, IP5 3RE, UK

Stange, Dominic:
Mining Associations Between Subsequences in Time-series,
University of Magdeburg, 2011.

Abstract

The discovery of patterns in a time-series is a popular and well-studied discipline of data mining. A time-series contains numeric measurements of a variable which are observed over a period of time. Association rule mining is a special technique of pattern discovery in time-series that focuses on uncovering relationships between measurements of the variable. In order to perform the association rule mining the time-series has to be discretised. Discretisation of time-series seeks to find a low-dimensional representation of the time-series in question. Discretisation of time-series is also an unsupervised process. Therefore, it is difficult to evaluate its outcome. The contributions of this thesis are mainly twofold. (1) The thesis introduces a new theoretical concept of subgroups in time-series which is used to carry out a novel approach of quality analysis of time-series discretisation. As a result of this analysis two criteria are identified that can be used to evaluate discretisation methods for their ability to provide useful inputs to a subsequent mining process. These criteria are based on two well-known classification errors which can now be interpreted in the context of time-series discretisation using the subgroups. (2) Based on the interpretation of these criteria an approach is proposed which relaxes a constraint that is commonly made in time-series discretisation. As to the best of the author's knowledge, for the first time, discretisation of a time-series allows measurements of a time-series to be represented by multiple discretised values at the same time. Due to this fact it is possible that valuable information is preserved for the mining process and a greater number of interesting patterns is found. The application of the new method is discussed in the context of association rule mining in time-series. Finally, an algorithm is implemented to proof the concept of the approach in a use case.

Zusammenfassung

Mustererkennung in Zeitreihen ist eine weithin angewendete und gut untersuchte Disziplin im Data Mining. Eine Zeitreihe ist eine Reihe numerischer Messdaten einer beobachteten Variablen über die Zeit. Die Suche nach Assoziationsregeln in Zeitreihen ist eine spezielle Art der Mustererkennung, die sich auf die Suche nach inhärenten Beziehungen zwischen Messwerten der Variable konzentriert. Um in kontinuierlichen Messreihen Assoziationsregeln zu finden, müssen die Zeitreihen diskretisiert werden. Diskretisierung von Zeitreihen zielt darauf ab, eine niedrigdimensionale Repräsentation der betrachteten Zeitreihe zu finden. Diskretisierung ist zudem ein in der Regel nicht überwachter Prozess und es ist daher schwierig dessen Ergebnisse zu bewerten. Die vorliegende Arbeit leistet in diesem Kontext in zweierlei Hinsicht einen Beitrag. (1) Einerseits schlägt die Arbeit das Konzept der Subgruppen vor, das genutzt werden kann, um die Qualität der Diskretisierung einer Zeitreihe zu bewerten. Hierzu führt die Arbeit theoretisch in das Konzept der Subgruppen in Zeitreihen ein. Am Ende werden zwei Kriterien identifiziert anhand derer eine Diskretisierungsmethode in Bezug auf ihre Zweckmäßigkeit für die Entdeckung von Assoziationsregeln in Zeitreihen evaluiert werden kann. Diese Kriterien basieren auf zwei gut verstandenen Klassifikationsfehlern, die unter Zuhilfenahme von Subgruppen im Kontext der Zeitreihendiskretisierung interpretiert werden. (2) Andererseits wird auf Basis dieser beiden Kriterien eine Herangehensweise vorgeschlagen, die eine gemeinhin postulierte Randbedingung bei der Diskretisierung von Zeitreihen entspannt. Im Anschluss wird nach bestem Wissen erstmalig ein Verfahren zur Diskretisierung von Zeitreihen vorgeschlagen, welches die Abbildung verschiedener diskreter Werte auf die Messwerte einer Zeitreihe erlaubt. Diese Besonderheit ermöglicht, dass wertvolle Informationen in der niedrigdimensionalen Repräsentation erhalten bleiben. Somit wird es möglich, weitere Muster und Regeln in den Zeitreihen zu finden. Angewendet wird das Verfahren im Kontext der Assoziationssuche in Zeitreihen. Abschließend wird das Verfahren im Rahmen eines Proof of Concept implementiert und angewendet.

Acknowledgement

I would like to thank Prof. Dr. Rudolf Kruse and PD. Dr. Detlef Nauck for making it possible for me to work for British Telecom (BT) in Ipswich. During my time in Ipswich I got to know a lot of interesting and very pleasant people. Moreover, I want to thank the whole team at BT for sharing a really great time with me. Particularly, I want to thank Dr. Martin Spott for being my supervisor and helping me out with valuable advice. Also, I want to thank Georg Ruß and Matthias Steinbrecher, my supervisors at the university.
Last but not least, I want to thank my family for supporting me in any imaginable situation - always. Also, I want to thank my girlfriend for her encouragement during the long hours of writing this thesis. Moreover, I am grateful for the helpful comments that my friend Christian Rößler provided me with.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur mit erlaubten Hilfsmitteln angefertigt habe.

Magdeburg, den 11. August 2011

Dominic Stange

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Time-series analysis is considered an important instrument to understand or influence systems in which sequences of data are monitored. The availability of large sets of time-series data has motivated numerous new pattern mining approaches that seek to deliver valuable and interesting information in a variety of domains, such as business, science, and engineering. The application of data mining approaches in time-series is paramount, because a lot of daily applications collect time-series data which are impossible to process manually. Such data represents a sequence of observations collected over time. By uncovering patterns in how these observations evolve, pattern mining in time-series can contribute to optimise processes, understand relationships, or predict future states of the underlying system.

1.1 Introduction to Mining Associations Between Subsequences in Time-series

Over the past two decades mining associations in highly complex and vast amounts of data has become an increasingly vital area of interest for scientists and practitioners in the field of data mining. Since the pioneering work of Agrawal et al. [1993], association mining has become a major tool for knowledge discovery in databases. As a result, association mining methods have been developed for a wide range of applications. These include databases where the data describes the current state of a system as well as temporal databases that contain data with an additional temporal dimension. The overall idea behind association mining is to find rules that describe patterns in the data.

Time-series analysis has been used for knowledge discovery in time-series for more than fifty years (see Laxman and Sastry [2006]). Typically used methods include modelling techniques that seek to find functions that best describe or predict the development of a time-series. While modelling strategies aim at finding global characteristics of the time-series in question, pattern mining in time-series seeks to uncover local patterns that describe only parts of the data. In the process of pattern mining in time-series difficulties arise due to the fact that a time-series can consist of a vast amount of observations. Common approaches to address this issue include methods to reduce the dimensionality of a time-series. These steps are carried out before the actual mining to preprocess the time-series. As a side-effect of the reduction, potentially useful information about the inherited patterns of a time-series may be lost. Most of the methods in literature dealing with association rule mining in time-series primarily focus on developing strategies to find new patterns or to make known mining algorithms more efficient. In contrast, the preprocessing step has only received little attention in the mining community. However, this first step plays an important role for the patterns that

can potentially be found during the mining. Although there is a variety of work in literature dealing with discretisation in general, few approaches are made to describe the impact of time-series preprocessing on the results of a mining process.
In this thesis the focus shall therefore be on the analysis of time-series preprocessing and its influence on the outcome of subsequent association rule mining.

1.2 Objectives of the Thesis

This thesis seeks to address the issue of discretising a time-series for a subsequent association rule mining process. In order to do that, a problem for association rule mining in time-series has to be defined. Based on this problem definition the effect of a commonly used discretisation method on the outcome of the association rule mining shall be examined. In particular, the ability of this method to create meaningful inputs for the mining process is to be investigated. In that respect, this thesis aims at introducing a new possibility to evaluate the quality of time-series discretisation. Using insights gained from interpreting the quality of the commonly used method a new approach to preprocess a time-series for association rule mining shall be developed. Then, an algorithm is to be created that is able to carry out the mining process and demonstrate the usefulness of the new approach.

1.3 Outline of the Thesis

The thesis is comprised of three parts. In the first part, in Chapter 2, the fundamentals of association rule mining are explained, based on the original case of mining co-occurrences in transactional data. In that respect, the need for a discretisation of numeric attributes is explained.

In the second part of the thesis, mining associations in temporal data is introduced in Chapter 3. Particularly, the mining problem for association rules in event sequences is formulated, and it is shown how this problem can be implemented in an algorithm. Pattern mining in time-series faces some issues which are covered in Chapter 4 with special emphasis on discretisation as a preprocessing step.
Both the rule mining problem formulated in Chapter 3 and the fundamentals of discretisation of time-series from Chapter 4 are then needed in Chapter 5. Chapter 5 consists of two parts. In the first part of Chapter 5, a novel problem statement for association rule mining in time-series is presented which is based on a new concept of subgroups of a time-series. Using this new concept the quality analysis of a common discretisation strategy is performed. In the second part of Chapter 5 a new approach for discretisation of numerical time-series data is proposed. At last, this new discretisation approach is applied to a sample time-series to demonstrate its functionality with respect to the discovery of patterns within the time-series. Chapter 6 summarises the results of the thesis and provides possibilities for future extensions for both the mining algorithm and the new discretisation process. A general overview of the major content of the thesis is illustrated in Figure 1.1.

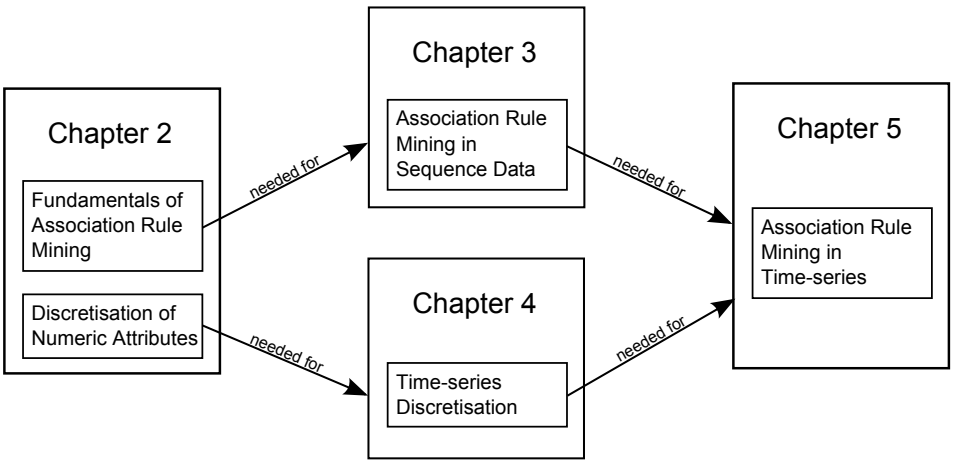

Figure 1.1: Chapter overview

Chapter 2

Fundamentals of Association Rule Mining

Association rule mining is one of the most popular pattern mining methods in knowledge discovery in databases (Hipp et al. [2000]). This chapter deals with the fundamentals of association rule mining as it was first proposed by Agrawal et al. [1993]. The basic concept of association rule mining is described in Section 2.1. There exist difficulties with association rule mining especially if the data contains numeric values. These difficulties are addressed in Section 2.2.

2.1 Basic Concept

The input to association rule mining is a database of records, where each record in the database is a *transaction* that contains *items* (see Agrawal et al. [1993]). Items are binary attributes (see Agrawal et al. [1993]). A typical application of association rule mining is in market basket data where the data consists of register transactions in a retail store (see Brin et al. [1997]). Here,

the items are the products that customers buy. Association rule mining has the goal to find patterns in these transactions so that interesting relationships between the items are revealed. These patterns are called *association rules* (see Agrawal et al. [1993]).
The following description of the basic concept of association rule mining is based on Agrawal et al. [1993], Tan et al. [2006], and Han et al. [2006].

The set of all items in a database is denoted by $I = \{i_1, i_2, ..., i_m\}$, where $i_j \in I$ is an item in the database. Each transaction T is a set of items such that $T \subseteq I$. A set of items is also called an *itemset*. The data on which association rule mining is carried out is a set of transactions D. The items in a transaction can belong to four categories of attributes. These categories are nominal, categorical, interval, and ratio attributes. Interval and ratio attributes are also called *numeric* attributes. Association rule mining requires items to be binary in order to perform the mining task. That is, an item is either present in a transaction or it is missing. Therefore, items of nominal, categorical, and numeric attributes have to be binarised. Methods that are used to perform the conversion into binary attributes are described in Section 2.2.

An important property of an itemset is its *support count*. The support count of an itemset is the number of transactions that contain a particular itemset. The support count, suppc(X), for an itemset X can be computed using the following equation:

$$\text{suppc}(X) = |\{T | X \subseteq T, T \in D\}|, \tag{2.1}$$

where the symbol $|\cdot|$ denotes the number of elements in a set. The goal of the mining task in association rule mining is to find association rules between items in a set of transactions. An association rule is an "implication of the form $(A \Rightarrow B)$"(Agrawal et al. [1993]), where A and B are itemsets, and $A \cap B = \emptyset$. The itemset A is also called *antecedent itemset* of an associ-

ation rule and the itemset B *consequent itemset*, respectively. In order to become an association rule, the implication $(A \Rightarrow B)$ has to fulfil a support and a confidence constraint. The support of an association rule determines the percentage of transactions that contain both the antecedent and consequent itemsets together. The confidence of an association rule determines the accuracy of the rule, the percentage of transactions that contain both antecedent and consequent itemset with respect to those that only contain the antecedent itemset. This approach of association rule mining is also called the *support-confidence framework* (see Adamo [2001]).

The support *supp* of an association rule $(A \Rightarrow B)$ can be calculated using the following equation:

$$\text{supp}(A \longrightarrow B) = \frac{\text{suppc}(A \cup B)}{|D|}, \tag{2.2}$$

The confidence of an association rule is also called a pruning measure, because it is used to filter itemsets that already fulfil the support constraint. The confidence constraint seeks to guarantee a certain accuracy of the rules. Given an association rule $(A \Rightarrow B)$, the confidence $\text{conf}(A \Rightarrow B)$ is computed using the following equation:

$$\text{conf}(A \longrightarrow B) = \frac{\text{suppc}(A \cup B)}{\text{suppc}(A)}. \tag{2.3}$$

The problem of mining associations in a database containing transactions can be stated as follows: *Given the set of transactions D, find all the rules having support $\geq$ minsup and confidence $\geq$ minconf, where minsup and minconf are the corresponding support and confidence thresholds.* These thresholds have to be specified by the user depending on the characteristics of the data and the special mining environment.

The mining process for association rules consists of two steps. In the first step, all itemsets that fulfil the support constraint are generated. These

itemsets are also called *frequent itemsets* and the step is referred to as *frequent itemset generation*. Based on the frequent itemsets the rules are generated using the confidence constraint to filter uninteresting rules. This step is called *rule generation*. There exist additional pruning methods based on interestingness measures explained in Wu et al. [2010].

There exists a variety of algorithms to mine association rules in a database (see Petersohn [2005]). Two popular algorithms are the *Apriori algorithm* (Agrawal and Srikant [1994]) and the *FP-Growth algorithm* (Han et al. [2000]). The Apriori algorithm takes advantage of the fact that support is downward closed. Downward closed means that if an itemset is frequent, then all of its subsets must also be frequent. This relationship is called the *apriori property* (see Han et al. [2006]). The Apriori algorithm is used to reduce the computational complexity of finding all itemsets that fulfil the support constraint. This is because the frequent itemset generation is generally considered computationally more expensive than rule generation (see Tan et al. [2006]).

The FP-Growth algorithm uses a prefix-tree that allows for the extraction of frequent itemsets without making repeated passes over the data. The algorithm uses a frequent pattern growth technique in the prefix-tree. An additional divide-and-conquer approach reduces the search space in the prefix-tree so that the algorithm can find association rules efficiently(Han et al. [2000]).

2.2 Association Rule Mining with Numeric Attributes

A precondition for the association rule mining is that transactions contain binary items (see Section 2.1). If the data contains nominal, categorical

or numeric attributes (i.e. interval or ratio attributes), these have to be binarised. In the nominal and categorical case, an approach is to convert the items into an attribute-value pair (Tan et al. [2006]), such that each value of the attribute is combined with the attribute. This combination is written as $attribute = value$. For example, a nominal attribute *color* can have multiple values, such as *green*, *blue*, or *red*. To create a binary attribute for an attribute *color*, it is possible to combine the attribute with the value so that the resulting binary attribute is "$color = green$". The same approach is possible for categorical attributes as well if they do not have a large number of values (categories).

For categorical attributes with a large number of values and for numeric attributes, however, the binarisation requires more processing steps (see Han et al. [2006]). These processing steps are needed to overcome a problem called the "*curse of dimensionality*" (Tan et al. [2006]). The curse of dimensionality refers to the phenomenon that many methods in data analysis become significantly harder as the dimensionality of the data increases. Specifically, the higher the dimensionality of the data, the sparser is the data in the space that it occupies (see Tan et al. [2006]). To approach the curse of dimensionality of numeric attributes for association rule mining, the attributes are transformed into categorical attributes first, so that they can be further binarised using the method described above. This transformation can be carried out using methods of *discretisation*.

Discretisation is a process that transforms a range of a numeric attributes into a finite number of intervals (see Cios et al. [1998]). Each interval is then associated with a discrete value. If the discretisation process uses class information about the attributes, then it is called *supervised* discretisation (see Han et al. [2006]). Otherwise, it is called *unsupervised* discretisation. Discretisation methods for numeric attributes include statistics-based methods, binning and entropy-based discretisation (see Han et al. [2006]). A survey on supervised and unsupervised discretisation methods can be found in

Dougherty et al. [1995].
For example, a numeric attribute *age* that contains integer values for the age of patients in a hospital database can be partitioned into intervals such as $[0, 8)$, $[8, 14)$, $[14, 18)$, etc. using a binning strategy so that each *age*-value is assigned to the interval to which it belongs. Then labels for each interval can be used to replace the respective value in the database.

Another example for discretisation is clustering (see Han et al. [2006]). In clustering the data tuples are treated as objects in a multidimensional space. These objects can be partitioned into groups or clusters so that the objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters (see Basu et al. [2008]). Similarity is expressed by the distance of the objects in the multidimensional space. The distance is calculated using a distance function. A frequently used distance function is the *Euclidean distance* (see Witten and Frank [2005]). Given two points x and y in a multidimensional space, the Euclidean distance d can be computed using the following formula (see Tan et al. [2006]):

$$d(x, y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}, \tag{2.4}$$

where n is the number of dimensions and x_k and y_k are the k-th components (attributes) of x and y. It should be noted that there exist special approaches to tackle the curse of dimensionality for association rule mining with numeric attributes. These approaches use discretisation to find "optimised confidence rules" (Fukuda et al. [1996]). In Fukuda et al. [1996], the authors divide the range of a numeric attribute into intervals so that the result is an interval in which the confidence of a rule is optimal. In that respect they use the characteristics of association rule mining and incorporate them into the discretisation process. Extensions of their approach can be found, for example, in Rastogi and Shim [2001] and Rastogi and Shim [2002].

Chapter 3

Association Rule Mining in Event Sequences

This chapter emphasises on association rule mining in temporal data. In particular, a mining problem for event sequences is formulated. Moreover, an algorithm is proposed that uses a special preprocessing step that allows for the application of ordinary association rule mining algorithms. First, the fundamentals of temporal data are explained in Section 3.1 and the problem definition of the algorithm is given in Section 3.2. Related work for association rule mining in temporal data is described in Section 3.3. Finally, in Section 3.5, the algorithm is described and consequently extended in Section 3.6.
The descriptions of the mining problem and the explanations of the algorithm refer to the following example.

Example 1. *Imagine a farmer who usually plants wheat on his field after he prepared the field for cultivation the day before. Assuming that the farmers actions are restricted to one per day, there is a sequence of actions of the farmer in which preparing the field and planting wheat are just two of possibly many actions. Such a sequence of actions for twenty days*

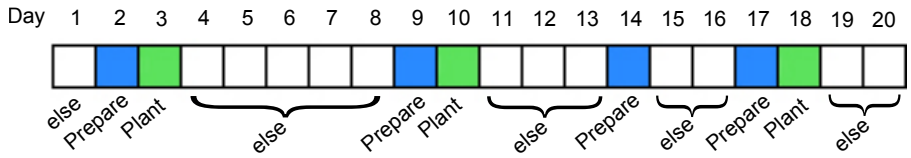

Figure 3.1: Actions of a farmer for 20 days. Each day is represented by a rectangle. Interesting actions are highlighted.

is shown in Figure 3.1. Each of the rectangles in the figure represents an action that the farmer performs. The blue highlighted rectangles show the "preparation of field" action, whereas the green highlighted rectangles show the "planting wheat" action. The sequence of actions can be written as ("Else", "Prepare", "Plant", "Else", "Else", "Else", "Else", "Else", "Prepare", "Plant", "Else", "Else", "Else", "Prepare", "Else", "Else", "Prepare", "Plant", "Else", "Else"), where "Prepare" stands for the preparation of the field, "Plant" stands for planting wheat and each "Else" in the sequence is a different, unspecified action. The task of the sequential rule mining algorithm is to find the pattern "Prepares the field and plants wheat on the next day". However, it can be seen that there are exceptions to this rule at days 14 and 15, which the mining algorithm has to take into account.

3.1 What is Temporal Data?

Temporal data refers to data that is collected over time. In temporal data, each data record has one or more dimensions of time (see Roddick and Spiliopoulou [1999]). For example, one dimension of time is the valid time of a record and another is the transaction time of the record (see Saraee and Theodoulidis [1995]). A record can contain one item or an itemset Garofalakis et al. [1999]. These records are sometimes also called *events* (e.g., see Mannila et al. [1995], Srikant and Agrawal [1996], Zaki [2001]). Temporal data can be divided into three categories (see Tan et al. [2006]). The first

category is *sequential data*. Sequential data is understood as an extension of transactional data from Chapter 2, so that each transactional record also has a time when it occurs. That is, an event in sequential data is associated with a timestamp.
The second category is *sequence data*. Sequence data is a "sequence of individual entities" Tan et al. [2006]. The difference between sequence and sequential data is that there is no timestamp associated with an event in sequence data. That is, only positions of events are considered.
According to Tan et al. [2006], the third category of temporal data is *time-series data*. A time-series is a special type of sequential data and consists of measurements of a variable taken over time. In this thesis, a time-series contains numeric values of an attribute. These measures can be discrete or continuous. Note that the term "time-series database" is sometimes used to describe a databse contain sequential data. For example, in Last et al. [2001] a time-series database corresponds to a transactional database where each record is associated with a timestamp. This interpretation of a time-series does not focus on measurements of a variable.

The algorithm that is proposed in this chapter is applied on sequence data. An event in sequence data is an instance of all possible event types which are called the *event class* (see Mannila et al. [1995] and Mannila et al. [1997]). The occurrence of an event is specified by its *position* in the sequence of events. The position of an event is determined by the number of events that occur before the particular event. An example of such a sequence is a nucleotide sequence in a gene, where each nucleotide is an event. A sequence of events is also called *event sequence* (e.g., in Baixeries et al. [2001] and Méger and Rigotti [2004]).

Definition 3.1 (Event Sequence)**.** *Given an event class* $\mathbf{E}$ *of event types and instances* e_i *of* $\mathbf{E}$*, with* $i = 1, 2, \ldots, n$*, let*

$$\mathcal{E} = ((e_1, 1),\ (e_2, 2)),\ \ldots,\ (e_n, n))$$

be an event sequence, where the event (e_i, i) *is event instance* e_i *occurring at position* i *in the event sequence* $\mathcal{E}$*.*

An event (e_i, i) occurs before another event (e_j, j), if $i < j$. Two events (e_k, k) and (e_l, l) are called *adjacent*, if either $k = l + 1$ or $l = k + 1$. The number of events n in an event sequence is also called *length* of an event sequence. The length of an event sequence $\mathcal{E} = ((e_1, 1), (e_2, 2)), \ldots, (e_n, n))$ is $length(\mathcal{E}) = n$. Thus, an event sequence can be understood as an ordered list of events, while the order is determined by the position of the events in the sequence. The corresponding event sequence for Example 1 can be written as $\mathcal{F} = ((else, 1), (prepare, 2), (plant, 3), \ldots, (plant, 18), (else, 19), (else, 20))$.

3.2 Related Work

Approaches for temporal pattern mining can be distinguished according to what type of patterns they find in the mining process. Without claiming to provide a complete overview, there exist at least four approaches for mining temporal patterns in literature. These approaches are briefly explained on the following pages.

3.2.1 Sequential Pattern Mining

Sequential pattern mining deals with finding patterns in sequential data. It was first introduced by Agrawal and Srikant [1995]. The sequential data is a set of sequences which are called data-sequences (see Srikant and Agrawal [1996]). These data-sequences contain transactions that arrive over time. A sequential pattern consists of an ordered list of itemsets which can be found in the data-sequences (see Srikant and Agrawal [1996]). The mining

problem is to find all sequential patterns that exceed a user-defined minimum support. The support of a sequential pattern is the percentage of data-sequences that contain the pattern (see Srikant and Agrawal [1996]). Thus, contrary to association rule mining as described in Chapter 2, sequential patterns describe relationships between transactions that arrive over time rather than relationships between items within transactions (see Zhao and Bhowmick [2003]). Furthermore, there exists no confidence constraint since the patterns found are only frequency-based. An example of a sequential pattern is that "8 % of customers of Amazon first buy 'Pride and Prejudice', then 'Persuasion', and then 'Emma"'. Each of the books is an item in a transaction that is created for the purchase of a customer. The pattern results if many customers buy these books within a given time-interval.

A selection of proposed algorithms for mining patterns in sequential data are GSP (Agrawal and Srikant [1995]), SPIRIT (Garofalakis et al. [1999]), SPADE (Zaki [2001]), and PrefixSpan (Pei et al. [2004]).

3.2.2 Frequent Episode Mining

An episode is a collection of events that occur in a certain order in an event sequence (see Mannila et al. [1995]). The mining of frequent episodes deals with finding all episodes in an event sequence that occur more often than a user-defined threshold (see Mannila et al. [1995]). An example of an episode in an event sequence containing events occurring in a software program is ("*open file*" $\Rightarrow$ "*close file*"). The order in which events have to occur is described by the arrow. Depending on what is done with the file, it is possible that between the events "*open file*" and "*close file*" additional actions are performed. Thus, the gap between events in a particular episode can vary in the event sequence.

There exist different approaches for frequent episode mining. For exam-

ple, some methods rely on a maximum window size to compute the frequency of an episode (e.g., see Mannila et al. [1995]). A window is a section of the event sequence. The frequency of an episode is the percentage of windows of a given size in which an episode occurs. Other methods allow the window size to be flexible so that the frequency computation is more accurate. In the latter case, a maximum window size is set by the user, but the frequency of an episode is computed based on the smallest window in which the episode actually occurs (Mannila and Toivonen [1996]). This way, depending on the "minimal occurrence" of an episode, the number of windows changes and, thus, does the frequency computation. Another approach is made, for example, in Baixeries et al. [2001] and Méger and Rigotti [2004]. They use a maximum gap constraint which determines the maximum time that can elapse between the occurrence of two consecutive events in an episode. Thus, these methods do not use the same window size for all episodes, but adopt the size depending on the number of events in an episode.

3.2.3 Partial Periodic Pattern Mining

Partial periodic pattern mining seeks to find sequences of events that occur repeatedly in an event sequence (see Han [1999]). The term 'partial' means that it is possible that these sequences contain gaps. Contrary to episodes, the gap between events in a sequence remains the same. Gaps in such a sequence are expressed by "don't care" events. The "don't care" events are used to allow partial matching, because any event matches with a "don't care" event. In order to become a pattern, the same sequence of events has to occur multiple times in an event sequence. A partial periodic pattern is characterised by its frequency, that is, the number of times that it occurs in an event sequence. Therefore, these patterns are called frequent partial periodic patterns (see Han et al. [1998]). An example of a partial periodic pattern is the pattern "$a**b$" in the sequence of events *acadebcbbaedb*, where the "don't care" events are represented by "$*$" in the pattern.

Methods that have been proposed for partial periodic pattern mining include Han et al. [1998], Han [1999], Yang et al. [2000], and Yang et al. [2001].

3.2.4 Temporal Association Rule Mining

Temporal association rule mining aims at finding association rules between items or itemsets in a transactional database that incorporate time information. An example of such an approach is the calendar-based mining of association rules described in Li et al. [2003]. The authors use the information about when transactions occur to find rules for specific intervals in time. That is, an itemset can be infrequent in the whole dataset, but frequent if the focus is restricted to certain periods of time. For example, an association rule ($apple \longrightarrow beer$) may be infrequent in the whole dataset, but becomes frequent if only Friday afternoons are considered. The same idea is also proposed by Ozden et al. [1998], where the result of a mining process is called "cyclic association rules".

3.2.5 Episodal Association Rule Mining

Episodal association rule mining has as its purpose the identification of association rules between events that occur within episodes. In order to become a rule, the events of an episode have to occur always in the same order in the event sequence. That is, the actual position of the events in the event sequence does not affect the mining process. The rule has an antecedent part and a consequent part. The antecedent part contains all events that must occur in order for the events contained in the consequent part to occur as well. Such a rule is characterised by a support and a confidence value. Similar to the frequency, the support is the number of times that all events in antecedent and consequent episodes can be observed. Additionally to the

frequency, the confidence describes how often the consequent events follow the antecedent events. To restrict the length of an episode, windows of a maximum size are used. An example of an episodal association rule for four events A, B, C, and D is "if A and B occur within 3 months, then within 2 months they will be followed by C and D occurring together within 4 months" (Harms et al. [2002]).

Methods proposed for episodal association rule mining include Mannila and Toivonen [1996], Mannila et al. [1997], Harms et al. [2001], Harms et al. [2002], and Harms and Deogun [2004].

3.3 Problem Definition

The problem definition for association rule mining in event sequences in this chapter is based on two approaches of pattern mining in temporal data. These are *partial periodic pattern mining* and *episodal association rule mining*. The problem definition uses the episodal association rule mining approach to describe patterns in an event sequence and uses the partial periodic approach to describe the sequence of events in the antecedent and consequent part of such an association rule. That is, contrary to partial periodic patterns, the resulting association rules are also characterised by a confidence. Furthermore, there exists an antecedent and a consequent sequence of events. On the other hand, in contrast to episodal association rules, the resulting association rules describe events that always occur at the same position in such a sequence. Since the result of the mining are association rules, they are called partial periodic association rules.

3.3.1 Basic Definitions

Many of the methods proposed for either approach outlined above restrict the maximum length of a pattern (e.g., in Mannila et al. [1995], Mannila et al. [1997], Harms et al. [2001], as well as Harms et al. [2002]). Restricting the length of patterns is done because it is assumed that interesting patterns occur only within a certain period of time (see Batal et al. [2009]). With respect to partial periodic pattern mining, the maximum length of a pattern determines how long a possible sequence of events that occurs in an event sequence can be. For episodal association rule mining, the maximum pattern length constraints the length of an episode.
There is no information about time in sequence data. Therefore, the assumption means that the patterns have to occur within a part of the event sequence that is called *window* on an event sequence (see, for example, Mannila et al. [1995] and Harms et al. [2002]). Using the notation of Mannila et al. [1995] and Harms et al. [2002], a window on an event sequence $\mathcal{E}$ is written as $win(\mathcal{E}, i, j)$ with i and j as integers and $i \leq j$. A window $win(\mathcal{E}, i, j)$ contains all events (e_k, k) of $\mathcal{E}$ for which $i \leq k \leq j$. The number of events that are contained in a window of an event sequence is called *window size*. The window size w of a window $win(\mathcal{E}, i, j)$ is $w = j - i + 1$. The sequence of events that belong to a window of an event sequence is also called *subsequence* of the event sequence (see Harms et al. [2001]).

Definition 3.2 (Subsequence)**.** *Given an event sequence $\mathcal{E}$ and a window $win(\mathcal{E}, i, j)$, $i \leq j$ with window size w, let a subsequence $\mathcal{E}[i,\ j]$ of $\mathcal{E}$ be an event sequence $\mathcal{E}[i,\ j] = ((e_i, 1),\ (e_{i+1}, 2),\ (e_{i+2}, 3),\ \ldots,\ (e_j, w))$.*

Thus, the first event instance of $\mathcal{E}[i,\ j]$ at position 1 is the event instance e_i at position i in $\mathcal{E}$. The integer i is, therefore, called *start* of the subsequences. Accordingly, the last event in $\mathcal{E}[i,\ j]$ at position w is the event instance e_j at position j in $\mathcal{E}$ and is called *end* of the subsequence.

In Example 1, a possible window of the event sequence $\mathcal{F}$ is $win(\mathcal{F}, 1, 4) =$

$\mathcal{F}[1,4] = ((else,1),\ (prepare,2),\ (plant,3),\ (else,4))$. The events belonging to the subsequence of the window correspond to the events at positions 1 to 4 in the event sequence. Another possible window is $win(\mathcal{F},9,11) = \mathcal{F}[9,11] = ((prepare,1),\ (plant,2),(\ else,3))$, where the events in the subsequence correspond to the events at positions 9 to 11 in the event sequence. In the example, the event instance *prepare* is often followed by *plant*. Thus, a possible pattern could be $(prepare \rightarrow plant)$, which can be read as "prepare before plant". To express this pattern a data structure is needed which is able to capture the temporal dimension between the events. This data structure is called *episode*. An episode is a particular sequence of events that can be observed in a window of an event sequence.

Definition 3.3 (Episode)**.** *Given a window size w and the class of events* $\mathbf{E}$ *of an event sequence $\mathcal{E}$, let an episode λ be a sequence of events*

$$\lambda = ((e_u,u),\ \ldots,\ (e_v,v)),\ 1 \leq u \leq v \leq w,$$

where each event instance in λ is an instance of event class $\mathbf{E}$*.*

Thus, there does not have to be an event for each position in an episode. Contrary to an event sequence, it is possible that episodes contain gaps. Gaps are necessary to describe patterns where not all events in a sequence of events are part of the pattern. As mentioned above, in partial periodic pattern mining these gaps are filled with "don't care" events. Since the current problem definition uses positions of events, the "don't care" events can be omitted by leaving out some positions in the sequence of events. The meaning of an episode in both definitions is the same.
Moreover, in episodal association rule mining, episodes do not specify the position of events that belong to an episode. Then, contrary to the current definition, an episode "a before b" where only the order is considered occurs in an event sequence $((a,1),\ (c,2),\ (b,3))$, but using the current definition, an episode $((a,1),\ (b,2))$ does not. In that respect, the definition of episodes

in the current problem definition is more closely related to the one in partial periodic pattern mining.

If an episode λ contains no events, it is said that λ is *empty*. Although the definition of an episode does not allow an episode do be empty, the notation of an empty episode is needed later for the description of a partial periodic association rule. Furthermore, two episodes α and β can be unified so that the resulting episode γ contains all events of α and all events of β. This is written as $\gamma = \text{union}(\alpha,\ \beta)$. If α and β contain different event instances at the same position, then both are contained in γ with the same position.

Episodes are ultimately used to detect patterns in an event sequence. Following the assumption that patterns can only occur within a window of the event sequence, episodes are said to *occur* in a window if each event instance of the episode occurs at the same position in the subsequence belonging to the window. Formally, an episode λ occurs in a window $win(\mathcal{E}, i, j)$, if all events (e_p, p) in λ are the same events $(e_p, p))$ in the subsequence $\mathcal{E}[i, j]$ belonging to the window. Since the subsequences start with position 1 and end with w the same episode can occur in different windows of an event sequence, given that the event instances match the ones of the respective subsequence. That means, it is possible to find recurring episodes in an event sequence. Therefore, all windows of size w have to be extracted from the event sequence (see Baixeries et al. [2001] and Harms et al. [2001]). Then, the window size w restricts the maximum length of the episodes. Therefore, the window size w can be understood as the maximum pattern length.

Definition 3.4 (Sliding Window)**.** *Given a window size w, let a sliding window be the process of extracting all windows $win(\mathcal{E}, i, j)$, with $i = 1,\ 2,\ \ldots,\ n-w+1$ of an event sequence $\mathcal{E} = ((e_1, 1),\ (e_2, 2)),\ \ldots,\ (e_n, n))$. Since the size of the windows is w, the end of each subsequence is $j = i+w-1$. The result of a sliding window with size w is a set of windows $WIN(\mathcal{E}, w)$ with $WIN(\mathcal{E}, w) = \{win(\mathcal{E}, i, j) \mid 1 \leq i \leq n - w + 1\}$ (see Harms et al. [2001]).*

For example, the set of subsequences of an event sequence $\mathcal{E} = ((e_1, 1), (e_2, 2)), (e_3, 3), (e_4, 4))$ with window size $w = 2$ is $WIN(\mathcal{E}, 2) = \{win(\mathcal{E}, 1, 2), win(\mathcal{E}, 2, 3), win(\mathcal{E}, 3, 4)\} = \{((e_1, 1), (e_2, 2)), ((e_2, 1), (e_3, 2)), ((e_3, 1), (e_4, 2))\}$. An excerpt of the sliding window with window size $w = 4$ for the event sequence from Example 1 is shown in Figure 3.2.

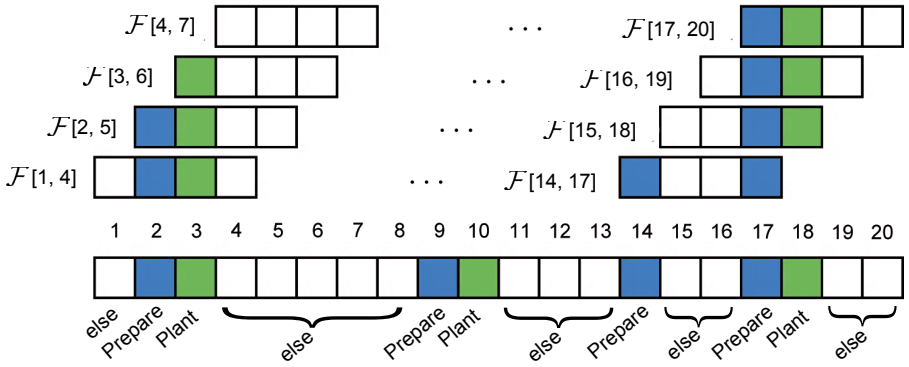

Figure 3.2: Excerpt of a sliding window with window size $w = 4$ for Example 1.

An episode can occur in several windows of an event sequence. The number of times that an episode occurs in a set of windows created by a sliding window is called the *support count* of an episode.

Definition 3.5 (Support Count of an Episode)**.** *Given a set of windows $WIN(\mathcal{E}, w)$ extracted with a sliding window of size w, it can be said that the support count* $\text{suppc}_w(\lambda)$ *of an episode λ is the number of windows of size w in which the episode λ occurs. Formally, the support count* $\text{suppc}_w(\lambda)$ *can be computed using the following equation:*[1]

$$\text{suppc}_w(\lambda) = \Big| \{win \,|\, \lambda \; occurs \; in \; win, \; win \in WIN(\mathcal{E}, w)\} \Big|. \tag{3.1}$$

In the farmer example, for a window size of $w = 3$ the episode $\phi =$

[1] Compare with Equation 2.1 for association rule mining in Chapter 2

$((prepare, 1),\ (plant, 2))$ has a support count of $\text{suppc}(\phi) = 3$, because both events *prepare* and *plant* occur in three windows at position 1 and 2. These windows are $win(\mathcal{F}, 2, 4) = ((prepare, 1),\ (plant, 2),\ (else, 3))$, $win(\mathcal{F}, 9, 11) = ((prepare, 1),\ (plant, 2),\ (else, 3))$ and $win(\mathcal{F}, 17, 19) = ((prepare, 1),\ (plant, 2),\ (else, 3))$.

There exists an issue concerning the support count of episodes using equation 3.1. If the window size of a sliding window is larger than the length of an episode and if the episode occurs at the end of an event sequence, then the support count of this episode is not increased. For example, with a window size $w > 4$ the episode $\phi = ((prepare, 1),\ (plant, 2))$ would only occur two times or less in the event sequence. The positions of the event instances *prepare* and *plant* in the last window with window size $w = 5$, $win(\mathcal{F}, 16, 20) = ((else, 1),\ (prepare, 2),\ (plant, 3),\ (else, 4),\ (else, 5))$, is 2 and 3. Thus, the support count of $\phi = ((prepare, 1),\ (plant, 2))$ would be 2. This effect can be accounted for by allowing the sliding window to extract shorter windows of the event sequence. This is done to capture those events at the end of the event sequence that otherwise never occur at position 1 in a subsequence. These events are all events which occur at positions $n - w$ and higher. That is, all windows $win(\mathcal{E}, n - w, n])$, $win(\mathcal{E}, (n - w) + 1, n)$, ..., $win(\mathcal{E}, n, n)$ have to be extracted as well.
The set of windows that contains all windows $WIN(\mathcal{E}, w)$ extracted by the sliding window together with the windows extracted additionally is called the *extended window set* denoted by $WIN^*(\mathcal{E}, w)$. The new formula for the support count of an episode is then:

$$\text{suppc}^*_w(\lambda) = \Big| \{win \,|\, \lambda \; occurs \; in \; win, \quad win \in WIN^*(\mathcal{E}, w) \} \Big|.$$

In the example, the additional windows are $win(\mathcal{F}, 17, 20) = ((prepare, 1),\ (plant, 2),\ (else, 3),\ (else, 4))$, $win(\mathcal{F}, 18, 20) = ((plant, 1),\ (else, 2),\ (else, 3))$, etc. The extended window set con-

tains as many windows as there exist events in the event sequence. Due to the additional windows the support count of an episode in the extended window set corresponds to the support count of the episode in the whole event sequence, independent of the window size. That is, given that the window size is not smaller than the length of the episode.

If the events in an episode were not distinguished according to their positions in the event sequence, the support of smaller episodes would be higher, because the same episode could be found in adjacent subsequences, although it only occurs once in the event sequence.
For example, using a simplified notation of an event sequence *abbcebbdcbbaca*, where the first a is at position 1, the first b at position 2, etc. The pattern in this sequence is that b occurs always twice in a row. The support count of bb in the event sequence is $suppc(bb) = 3$. Using a sliding window of size $w = 3$, however, the support of the pattern becomes $suppc(bb) = 6$. This is due to the fact that bb occurs, for instance, in abb and in bbc, the first two windows on the event sequence. Adjacent subsequences are overlapping and, therefore, contain partly the same events.

The support count of episodes can be used to describe how often a certain episode occurs in an event sequence. This is the first criterion that episodes have to satisfy to describe a *partial periodic association rule*. A partial periodic association rule uses the concept of partial periodic patterns as explained in Section 3.1 and extends this concept with a confidence criterion. Such a pattern describes the relationship between events in an episode such that "if event A occurs at position X in a subsequence, event B is likely to occur at position Y". In a partial periodic pattern only the support count of episodes is considered. In the current approach, the relationship between events of an episode must also fulfil an accuracy criterion, which is the confidence of the rule. The confidence is needed since events that occur frequently together can do so only because they occur very often in an event sequence. This additional criterion is also present in episodal association rule mining. Therefore,

the following description of the mining problem is based on the formalisation for episodal association rule mining given in Harms et al. [2001] and Harms et al. [2002].

To constitute a partial periodic association rule, an episode is split into an antecedent and a consequent part. Both parts are episodes on their own. The antecedent and consequent episodes extracted from an episode λ are written as λ_{ant} for the antecedent episode and λ_{con} for the consequent episode. The interpretation of antecedent episode and consequent episode is the same as for antecedent itemset and consequent itemset from Chapter 2. The union of both episodes λ_{ant} and λ_{con} is the original episode λ. A partial periodic association rule R is written as $R = (\lambda_{ant} \longrightarrow \lambda_{con},\ \mathrm{supp}(R),\ \mathrm{conf}(R))$. The support $\mathrm{supp}(R)$ of a partial periodic association rule R is written as $\mathrm{supp}(R) = \mathrm{supp}(\lambda_{ant} \longrightarrow \lambda_{con})$, and the confidence $\mathrm{conf}(R)$ is written as $\mathrm{conf}(R) = \mathrm{conf}(\lambda_{ant} \longrightarrow \lambda_{con})$. The support and confidence of a partial periodic association rule R based on a sliding window with size w can be calculated using the following formulae[2]:

- Support: $\mathrm{supp}(R) = \dfrac{\mathrm{suppc}^*(\lambda)}{|WIN(\mathcal{E}, w)|}$,
- Confidence: $\mathrm{conf}(R) = \dfrac{\mathrm{suppc}^*(\lambda)}{\mathrm{suppc}^*(\lambda_{ant})}$.

The support of a partial periodic association rule is the percentage of windows in the set $WIN(\mathcal{E}, w)$ that contain episode λ, that is, the fraction of all windows in which both, λ_{ant} and λ_{con} occur in (see Mannila et al. [1997]). The confidence of a partial periodic association rule is the fraction between those windows in which both episodes occur and those in which only the antecedent episode occurs. Given two user-defined thresholds for a minimum support minsupp and a minimum confidence minconf, a partial periodic association rule can be defined as follows.

[2]Compare with Equation 2.2 and Equation 2.3 from Chapter 2.

Definition 3.6 (Partial Periodic Association Rule)**.** *Given two episodes* λ_{ant} *and* λ_{con} *of an event sequence* $\mathcal{E} = ((e_1, 1),\ (e_2, 2),\ \ldots,\ (e_n, n))$*, where* λ_{ant} *and* λ_{con} *are not empty, and the union of both episodes* $\lambda = \text{union}(\lambda_{ant}, \lambda_{con})$*; moreover given a minimum support threshold* minsupp*, a minimum confidence threshold* minconf *and a window size* w *for the maximum length of episode* λ*, let*

$$R = (\lambda_{ant} \longrightarrow \lambda_{con},\ \text{supp}(R),\ \text{conf}(R))$$

be the corresponding partial periodic association rule for the antecedent episode λ_{ant} *and consequent episode* λ_{con}*, with* $\text{supp}(R) \geq \text{minsupp}$ *and* $\text{conf}(R) \geq \text{minconf}$*.*

The partial periodic association rule R can be interpreted as follows: Given an episode λ_{ant} occurs in a window $win(\mathcal{E}, i, j)$, then the episode λ_{con} will also occur in $win(\mathcal{E}, i, j)$, with support $\text{supp}(R)$ and confidence $\text{conf}(R)$.

In the farmer example, the episode $\phi = ((prepare, 1),\ (plant, 2))$ can be split into two episodes $\phi_{prepare} = (prepare, 1)$ and $\phi_{plant} = (plant, 2)$. Given a sliding window with window size $w = 2$ is used, a partial periodic association rule between both episodes is $R = ((\phi_{prepare} \longrightarrow \phi_{plant}),\ .15,\ .75)$, assuming that the thresholds for support and confidence are set accordingly. The support results from the support count of the given window size $\text{suppc}^*_2(\phi)$ of episode ϕ, which is $\text{suppc}^*_2(\phi) = 3$ and the total number of windows which is $|WIN^*(\mathcal{F}, 2)| = 20$. Thus the support of the rule is $\text{supp}(R) = 3/20$. The confidence $\text{conf}(R)$ of the partial periodic association rule R is not 1, because on day 15 in the event sequence the farmer does not plant wheat although he prepared the field the day before. That is why the rule only holds three times out of four. Notice that it is possible to find the inverted rule $R' = ((\phi_{plant} \longrightarrow \phi_{prepare}),\ .15,\ 1)$ as well, which even has confidence of $\text{suppc}_2(R') = 1$, because there are no exceptions to this rule. The rule R' can be interpreted as: if the farmer plants wheat on a certain day, then he already prepared the field the day before. What can be read from both rules

is that the action "Prepare" is immediately followed by action "Plant" on the next day (with the given confidence).

3.3.2 Mining Problem

The interpretation of an episode in this problem definition has a side effect on the mining process. This side effect is that the same relationship between events within an event sequence can be expressed using different episodes, and thus different rules. For example, the partial periodic association rule $R_1 = ((prepare, 1) \longrightarrow (plant, 2)),\ .15,\ .75)$ expresses the same relationship between event instances *prepare* and *plant* as the partial periodic association rule $R_2 = ((prepare, 2) \longrightarrow (plant, 3)),\ .15,\ .75)$. Thus, in terms of the meaning of a partial periodic association rule, which is called the *semantics* of a partial periodic association rule, semantically redundant rules can be found.
To account for this redundancy of rules a new term is introduced. A partial periodic association rule $R = (\lambda_{ant} \longrightarrow \lambda_{con},\ \text{supp}(R),\ \text{conf}(R))$ is called *semantically unique* if the episode $\lambda = \text{union}(\lambda_{ant}, \lambda_{cons})$ contains an event that occurs at position 1. As a consequence all partial periodic association rules which do not contain such an event are considered semantically redundant. That is, the rule R_2 is semantically unique since it expresses a relationship between the event instances *prepare* and *plant* and contains an event $(prepare, 1)$ that has position 1. There exist a similar approach in partial periodic pattern mining where episodes are not allowed to start with a "don't care" event (see Yang et al. [2000]). This approach has the same effect on the mining process.
Using this clarification, the goal of partial periodic association rule mining in an event sequence can be defined as follows.

Definition 3.7 (Mining Problem for Partial Periodic Association Rules)**.** *Given an event sequence $\mathcal{E}$, a minimum support threshold* minsupp*, a min-*

imum confidence threshold minconf*, and the maximum length of a pattern* w*, find all semantically unique partial periodic association rules* R_i *of* $\mathcal{E}$ *that occur in windows of size* w *and for which* $\text{supp}(R_i) \geq \text{minsupp}$ *and* $\text{conf}(R_i) \geq \text{minconf}$*.*

3.4 Algorithm for Partial Periodic Association Rule Mining

The algorithm for partial periodic association rule mining in event sequences that is proposed in this thesis follows the overall idea of converting the problem of partial periodic association rule mining into a format where association rule mining algorithms designed for use in transactional data can be used. This approach has the advantage that already existing efficient algorithms can be employed. The strengths and shortcomings of these algorithms are also well-studied, which helps to interpret the quality and validity of the found rules.

What makes the problem of partial periodic association rule mining different from the one in transactional data as described in Chapter 2 is that there exist no transactions of records in which the co-occurrences of items can be examined. This is because an event sequence is a sequence of events occurring over time. Thus, only co-occurring events could be found. Furthermore, in the transactional case the order of items within a transaction is irrelevant for the outcome of the mining. That is, whether an item is first or second in a transaction does not influence the association rule as long as both belong to the same transaction. In an event sequence, however, the order of the events is part of the periodic association rule.
That is why, to account for the differences in the structure of both data types, in order to be able to apply association rule mining algorithms for transactional data, an additional preprocessing step is required. The purpose of this

preprocessing step is to convert the sequence data into transactional data. This preprocessing step is described in the next section.

3.4.1 Sequence Items and Sequence Transactions

As explained in Section 2.1, association rule mining finds co-occurrences between binary items. If attributes have multiple values (e.g. categorical and numeric attributes), they are binarised before the mining process (see Section 2.2).
The different event types of an event class can be understood as the values of an attribute. If more than one event instance exists, this attribute is non-binary as well. It is possible that the number of different values is very large. Then the same difficulties may occur as for mining association rules with categorical or numeric attributes.

To apply algorithms for association rule mining in sequence data, transactions are needed that preserve the sequence information of events in the event sequence and are able to find meaningful co-occurrences between them. The approach made in this thesis is to create transactions from windows of an event sequence extracted with a sliding window. These transactions contain items for each event in the subsequence belonging to the window. These items are called *sequence items*.

Definition 3.8 (Sequence Item)**.** *Given a subsequence* $\mathcal{E}[i,j] = ((e_i,1),\ (e_{i+1},2),\ \ldots,\ (e_j,w))$ *belonging to a window* $win(\mathcal{E},i,j)$ *of an event sequence* $\mathcal{E}$ *and an event* (e_k,k) *in* $\mathcal{E}[i,j]$*, let a sequence item seqitem of event* (e_k,k) *be a* $(position, value)$*-pair, where position is the position* k *of* (e_k,k) *and value is the event instance* e_k*. A sequence item for event* (e_k,k) *is written as* $seqitem_k = e_k$*.*

A set of sequence items is called a *sequence itemset*. What is needed to transform the sequence data into transactional data is to define the size of a

sliding window in which partial periodic association rules are expected. This is in accordance with the assumption that is made in the problem definition. Then, for each subsequence of a window extracted by the sliding window a *sequence transaction* is created that contains the sequence items for all events within that subsequence.

Definition 3.9 (Sequence Transaction)**.** *Given a subsequence* $\mathcal{E}[i,j] = ((e_i,1),\ (e_{i+1},2),\ \ldots,\ (e_j,w))$ *of an event sequence* $\mathcal{E}$*, let a sequence transaction seqtrans be the sequence itemset that can be created from all events in* $\mathcal{E}[i,j]$*. Formally, if* $seqitem_k$ *is a sequence item created for event* (e_k,k) *in* $\mathcal{E}[i,j]$*, let* $seqtrans = \{seqitem_k | 1 \leq k \leq w\}$ *be the sequence transaction for* $\mathcal{E}[i,j]$*.*

Examples of sequence transactions can be found in Table 3.3. The sequence transactions are created for the subsequences shown in Table 3.1, which are previously converted into sequence items shown in Table 3.2.

Index i	Subsequence
1	$\mathcal{F}[1,3] = ((else,1),\ (prepare,2),\ (plant,3))$
2	$\mathcal{F}[2,4] = ((prepare,1),\ (plant,2),\ (else,3))$
3	$\mathcal{F}[3,5] = ((plant,1),\ (else,2),\ (else,3))$
4	$\mathcal{F}[4,6] = ((else,1),\ (else,2),\ (else,3))$
⋮	⋮
17	$\mathcal{F}[17,19] = ((prepare,1),\ (plant,2),\ (else,3))$
18	$\mathcal{F}[18,20] = ((plant,1),\ (else,2),\ (else,3))$

Table 3.1: Subsequences belonging to all windows with size $w = 3$ of the event sequence of Example 1

If sequence transactions are created for all windows of an event sequence extracted by a sliding window, then the resulting set of sequence transactions is called a *set of sequence transactions*. Based on this set of sequence transactions, association rule mining algorithms such as the ones mentioned in Chapter 2 can be applied. This is briefly explained in the next section.

Index i	$seqitem_1$	$seqitem_2$	$seqitem_3$
1	*else*	*prepare*	*plant*
2	*prepare*	*plant*	*else*
3	*plant*	*else*	*else*
4	*else*	*else*	*else*
⋮		⋮	
17	*prepare*	*plant*	*else*
18	*plant*	*else*	*else*

Table 3.2: Sequence items corresponding to the subsequences in Table 3.1

Index i	sequence transaction
1	$\{seqitem_1 = else,\ seqitem_2 = prepare,\ seqitem_3 = plant\}$
2	$\{seqitem_1 = prepare,\ seqitem_2 = plant,\ seqitem_3 = else\}$
3	$\{seqitem_1 = plant,\ seqitem_2 = else,\ seqitem_3 = else\}$
4	$\{seqitem_1 = else,\ seqitem_2 = else,\ seqitem_3 = else\}$
⋮	⋮
17	$\{seqitem_1 = prepare,\ seqitem_2 = plant,\ seqitem_3 = else\}$
18	$\{seqitem_1 = plant,\ seqitem_2 = else,\ seqitem_3 = else\}$

Table 3.3: Sequence transaction for sequence items of Table 3.2

3.4.2 Mining Associations in a Set of Sequence Transactions

Partial periodic association rule mining in event sequences focuses on finding frequent episodes in a sequence of events that fulfil a support and a confidence constraint. This problem is now expressed by means of sequence items and sequence transactions. The goal of the partial periodic association rule mining is to find all *frequent sequence itemsets* that fulfil the confidence constraint. An episode can be expressed as a set of sequence items within a sequence transaction. To illustrate this, it is helpful to contrast the concept of episodes with a set of sequence items. An episode occurs within a window

of an event sequence. A window is a subsequence of the event sequence that corresponds to a sequence transaction. Thus, the combinations of sequence items in a sequence transaction contain the same information as episodes in the subsequence for which the sequence transaction is created. As a result both structures contain the same data. By finding all frequent sequence itemsets in the set of sequence transactions the corresponding frequent episodes in all subsequences are found. Based on these frequent sequence itemsets a rule generation step for association rule mining in transactional data can be carried out. This step is called *partial periodic rule generation*.

The conversion of events into sequence items serves two purposes. First, events that belong to the same subsequence of a window are now co-occurring in the same sequence transaction. This co-occurrence can be mined by association rule mining algorithms. Second, each of the sequence items within a transaction is automatically binarised due to the conversion process. This is because for each event a (position, value)-pair is created. In order to mine in an event sequence, the same steps as described in Chapter 2 can therefore be carried out, and hence are not discussed here further. For example, in the set of sequence transaction an association rule can be found that has the form $R = (seqitem_1 = prepare \longrightarrow seqitem_2 = plant)$ with support $\text{supp}(R) = .15$ and confidence $\text{conf}(R) = .75$. Since the sequence items $seqitem_1 = prepare$ and $seqitem_2 = plant$ correspond to the events $(prepare, 1)$ and $(plant, 2)$, respectively, this association rule corresponds to the partial periodic association rule described in the problem definition of this chapter ($R = ((prepare, 1) \longrightarrow (plant, 2)),\ .15,\ .75)$).

To sum up, the search for partial periodic association rules in an event sequence is delegated to a search of association rules between sequence items. The necessary steps to carry out the mining correspond to the ones in Chapter 2. Note that only semantically unique association rules shall be mined. Thus, this is another possibility to speed up computation time of the mining process. Only frequent sequence itemsets have to be computed which have a

sequence item at position 1.
In practice, it is likely that multiple events occur at the same time, that is, at the same position in an event sequence. In the next section a modification of the preprocessing step is introduced that allows partial periodic association rule mining in an event sequence with co-occurring events.

3.5 Extending the Algorithm for Co-occurring Events

In the problem definition at the beginning of this chapter, episodes can contain events with the same position. With respect to the farmer example, it is very likely that the farmer can perform several actions per day. In this section, a simple way is described how the mining algorithm can be modified in order to account for co-occurrences of events.

In an event sequence with co-occurring events, the complexity of mining patterns is increased. Imagine an event sequence consisting of 20 events, similar to Example 1. If for each position three events can be observed, then 3^{20} different event sequences have to be analysed. Although it is possible to reduce the number of computations, for example, by accounting for the fact that there are duplicate subsequences between the 3^{20} event sequences, the complexity of the mining process is still very high.

The sequence items of co-occurring events of an episode possess the same position attribute. Thus, a subsequence of an event sequence in which events occur at the same position is converted into a sequence transaction that contains sequence items with the same position attribute. During the mining, all sequence items within a transaction are considered equally, so even patterns of events always occurring together can be found. If the event sequence of Example 1 is slightly modified so that

there exist another two events $(plant, 4)$ and $(prepare, 20)$ so that $\mathcal{F}' = (\ldots,\ (else, 4),\ (\mathbf{plant}, \mathbf{4}),\ \ldots,\ (else, 19),\ (else, 20),\ (\mathbf{prepare}, \mathbf{20}))$, then Table 3.4 shows an excerpt of the subsequences of the new event sequence, Table 3.5 shows the corresponding sequence items, and in Table 3.6 the sequence transactions for this event sequence can be seen, given the window size is still $w = 3$.

Index i	Subsequence
1	$\mathcal{E}[1,3] = ((else, 1),\ (prepare, 2),\ (plant, 3))$
2	$\mathcal{E}[2,4] = ((prepare, 1),\ (plant, 2),\ (else, 3),\ (\mathbf{plant}, \mathbf{3}))$
3	$\mathcal{E}[3,5] = ((plant, 1),\ (else, 2),\ (\mathbf{plant}, \mathbf{2}),\ (else, 3))$
4	$\mathcal{E}[4,6] = ((else, 1),\ (\mathbf{plant}, \mathbf{1}),\ (else, 2),\ (else, 3))$
⋮	⋮
17	$\mathcal{E}[17,19] = ((prepare,)1,\ (plant, 2),\ (else, 3))$
18	$\mathcal{E}[18,20] = ((plant, 1),\ (else, 2),\ (else, 3),\ (\mathbf{prepare}, \mathbf{3}))$

Table 3.4: Subsequences with window size $w = 3$ of the event sequence of Example 1, extended by two events $plant_4$ and $prepare_20$

Index i	$seqitem_1$	$seqitem_2$	$seqitem_3$
1	*else*	*prepare*	*plant*
2	*prepare*	*plant*	{*else*, **plant**}
3	*plant*	{*else*, **plant**}	*else*
4	{*else*, **plant**}	*else*	*else*
⋮		⋮	
17	*prepare*	*plant*	*else*
18	*plant*	*else*	{*else*, **prepare**}

Table 3.5: Sequence items corresponding to the subsequences in Table 3.4

It can be seen that the co-occurrence of events requires only a few changes in the preprocessing step of the algorithm. In Chapter 5, this modification will be needed to find partial periodic association rules in a time-series.

TID	sequence transaction
1	$\{seqitem_1 = else,\ seqitem_2 = prepare,\ seqitem_3 = plant\}$
2	$\{seqitem_1 = prepare,\ seqitem_2 = plant,\ seqitem_3 = else,$ $\mathbf{seqitem_3 = plant}\}$
3	$\{seqitem_1 = plant,\ seqitem_2 = else,\ \mathbf{seqitem_2 = plant},$ $seqitem_3 = else\}$
4	$\{seqitem_1 = else,\ \mathbf{seqitem_1 = plant},\ seqitem_2 = else,$ $seqitem_3 = else\}$
⋮	⋮
17	$\{seqitem_1 = prepare,\ seqitem_2 = plant,\ seqitem_3 = else\}$
18	$\{seqitem_1 = plant,\ seqitem_2 = else,\ seqitem_3 = else,$ $\mathbf{seqitem_3 = prepare}\}$

Table 3.6: Sequence transaction for sequence items of Table 3.5

3.6 Evaluation of the Algorithm

The algorithm proposed in this chapter allows partial periodic association rule mining based on a preprocessing step that converts sequence data into transactional data. This way, already developed algorithms such as Apriori and FP-Growth can be used to patterns in event sequences. Even more, the algorithm is able to process event sequences with co-occurring events.
There are shortcomings with this kind of conversion. Currently used association rule mining algorithms for transactional data are optimised for a large amount of rows that are to be processed as the data sets usually contain a lot of records. Compared to the rows, the number of columns can be rather small in these data sets. The converted set of sequence transactions used in the algorithm, on the other hand, can contain a large number of columns if the event sequence has a lot of events that occur at the same position. Moreover, in some applications the number of rows, that is the number of subsequences, can be considered to be rather small. This is the case if the event sequence is short. Thus, in order to improve mining performance, the implementations of the algorithm should be modified.

Furthermore, the algorithm described here is designed for an application in sequence data. Thus, the patterns that are found describe regularities between events occurring at certain positions in an episode. The exact time of the event is not considered. If only the positions are considered, the support count of episodes is influenced by the order of appearance of events in the event sequence. For example, even if an event always follows another after the same amount of time, if the number of events that occur between them changes in the event sequence, the support count will be low. Therefore, the current version of the algorithm can only be used in domains where events occur (nearly) equidistant so that the position is a good approximation of the time that an event occurs at.

Moreover, the parameter for the maximum length of an episode is used to reduce the number of candidates that have to be processed during the mining process. If it is set too small, then potentially interesting rules are not found. This is why domain knowledge should be used to specify a maximum pattern length which is reasonably high.

3.7 Conclusion

In Chapter 2 the fundamentals of association rules were explained. The curse of dimensionality was introduced as a special problem for association rule mining with numeric attributes. In Chapter 3 a problem of mining in sequence data was then formulated. In the course of the problem definition, partial periodic association rule mining was defined. An approach for mining partial periodic association rules in sequence data was hence proposed which employs a transformation process that converts the problem of mining in event sequences into a mining problem for transactional data so that existing algorithms can be used. From there an extension of this algorithm being able to cope with co-occurring events was developed.

Chapter 4

Knowledge Discovery in Time-series

A time-series is a special type of temporal data where measurements of a variable are carried out and each measurement is associated to a point in time. Knowledge discovery in such a time-series can be divided into two approaches. The first approach includes methods that seek to create models of the time-series. With the help of a time-series model "meaningful statistics and other properties of the time-series data" shall be extracted (see Ao [2010]). The second approach employs pattern recognition techniques. Pattern recognition in time-series can be divided into four directions (see Cotofrei and Stoffel [2002]). These are similarity querying, pattern finding, clustering/classification, and rule mining. Similarity querying has the goal to find all subsequences of a time-series that match a given input sequence. Pattern finding deals with periodic patterns in a time-series, similar to what is described for an event sequences in the previous chapter. Clustering in time-series has the goal to find groups of subsequences that exhibit a similar development in a time-series. Association rule mining in time-series is concerned with deriving rules from a time-series. This thesis focuses on rule

discovery in a time-series, which faces a variety of problems when applied to time-series. Some of them are described in Section 4.1. A particular problem for rule mining will be detailed in Section 4.2.

4.1 Problems with Pattern Recognition in Time-series

Pattern recognition in time-series is considered non-trivial for a variety of applications (see Hand et al. [2001]). An overview of some of the characteristics of time-series that influence the discovery of patterns is given next. The overview is based on the descriptions in Daw et al. [2003].

Aliasing is an effect that occurs when the chosen sampling rate for the observed variable of a time-series is too slow. Then, patterns within the time-series can be the result of the sampling instead of actual characteristics of the time-series. Therefore, antialiasing filters have been proposed (e.g. in) that can help to prevent aliasing.

Noise and artefacts in time-series describe errors in the measurement of the variable that can result from the measuring device or irregular "fluctuations" of the variable due to external influences. Noise is also described as a "random error or variance in a measured variable"(Han et al. [2006]). If noise is observed in a time-series, additional preprocessing steps have to be carried out which influence the results of a mining process.

Stationarity describes a systems stability towards its parameters. If the parameters of a system are stable over time, then it is called stationary. Nonstationary systems are demanding for a modelling and pattern mining tasks, because the statistical properties of the time-series can

vary over time and become more difficult to compare for different points in time.

Missing values can occur when the information of a value was not available during the time of its collection (see Tan et al. [2006]), for example, because the measuring device was not functioning. Missing values of a time-series leave gaps in the sequence of values which have to be dealt with. Common strategies include filling these gaps with expected values as a preprocessing step. Some methods to preprocess data with missing values are described in Han et al. [2006].

Dimensionality of a time-series describes the fact that a time-series, ordinarily, contains large amounts of numeric values. The problem caused by high dimensionality is already described as the "curse of dimensionality" in Section 2.2.

Especially the dimensionality of a time-series is difficult to cope with and has a significant effect on the outcome of a mining process. The next section provides a brief description of how dimensionality in a time-series can be reduced.

4.2 Reducing the Dimensionality of a Time-series

The reduction of the dimensionality of a time-series can be carried out in two ways. One of them is called *dimensionality reduction*. Dimensionality reduction aims at compressing the information contained in a time-series (see Tan et al. [2006]). This is done by reducing the length and the range of the time-series. The goal of dimensionality reduction is to "eliminate irrelevant features, reduce noise and produce a more understandable model

of the time-series" (see Tan et al. [2006]). Some methods used for dimensionality reduction are, for example, sampling and the piecewise aggregate approximation (see Fu [2011]).

Another method to reduce the dimensionality of a time-series is discretisation (see Keogh et al. [2002]; Rasheed et al. [2011]). Discretisation addresses the range of the numeric measurements of a time-series. How numeric values can be discretised is already discussed in Section 2.2. In addition to what is said there about numeric attributes, time-series contain information about the order (sequence) of the measured values. This information can be incorporated into the discretisation process. That is, the discretisation can be done on single values, which would be the same approach as applied to numeric attributes, such as the age of a patient. It can also be based on subsequences of a time-series. In the case of the discretisation of a time-series, subsequences play an important role because the values in a time-series are considered "meaningful only as a part of a time segment" (see Last et al. [2001]). That means the values of a time-series have to be interpreted as part of a (sub)sequence of a time-series. In general, it can be said that time-series are discretised based on subsequences of the time-series, and the length of the subsequences can vary between $1, 2, \ldots$, where 1 corresponds to the discretisation of single values.
The process of dividing the time-series into subsequences is also referred to as *segmentation* (see Keogh et al. [2001]; Fu [2011]). The segmentation of the time-series can be carried out according to three strategies (see Keogh et al. [2001]). Besides the sliding window, which was already explained in Section 3.3.1, there exist a bottom-up and a top-down approach. For completeness, the bottom-up approach starts from the finest possible segmentation of the time-series and merges the subsequences until a criterion is met. The top-down approach, on the other hand, partitions the time-series recursively until a stopping criterion is met as well. Both will not be required for the remainder of this thesis. In the context of time-series segmentation, the parameter for the size of a sliding window is denoted by l. In the context of

time-series segmentation, the parameter for the size of a sliding window is denoted by l. This is done to distinguish the sliding window used in segmentation form the one used for the association rule mining in event sequences. A common approach for time-series discretisation is to use all subsequences of a time-series extracted with a sliding window and label them with a primitive (discretised) value (e.g. see Das et al. [1998]; Daw et al. [2003]; Warren Liao [2005]). Such a primitive value is also called a *symbol* (see Das et al. [1998]). A symbol becomes the low-dimensional representation of a subsequence. Each of the thus extracted subsequences can be replaced by such a symbol in order to create a sequence of symbols that represents the time-series (see Das et al. [1998]). This sequence of symbols is called *symbol sequence*. The set of all possible symbols which is created during a discretisation process is also called an *alphabet* (see Das et al. [1998]; Keogh et al. [2002]). With respect to the definitions and notations given in Chapter 3, a symbol sequence is an event sequence where each event is a symbol that has a specific position. This position corresponds to the start of the subsequence which the symbol represents. The alphabet of the discretisation is then set of all event types of a symbol sequence. Thus, a symbol sequence $\mathcal{C}$ can be written as $\mathcal{C} = ((c_1, 1),\ (c_2, 2),\ \ldots,\ (c_m, m))$, where m is the number of events in the symbol sequence and the event (c_k, k) is the symbol c_k of the alphabet $\mathbf{C}$ occurring at position k.

One of the methods used to carry out the discretisation of a time-series is clustering (see Das et al. [1998]; Harms et al. [2002]). In order to apply clustering methods to identify groups of similar subsequences, as a preprocessing step the time-series has to be segmented. The clustering then groups the resulting subsequences, for instance, by minimising an objective function (partitional clustering) or by using techniques of hierarchical clustering (see Jain et al. [1999]; Warren Liao [2005]). In the discretisation a subsequence is then labelled with a symbol belonging to the cluster to which the same subsequence is assigned to. In that respect, symbols are simply labels of clusters. After the clustering process is carried out it is possible to create a

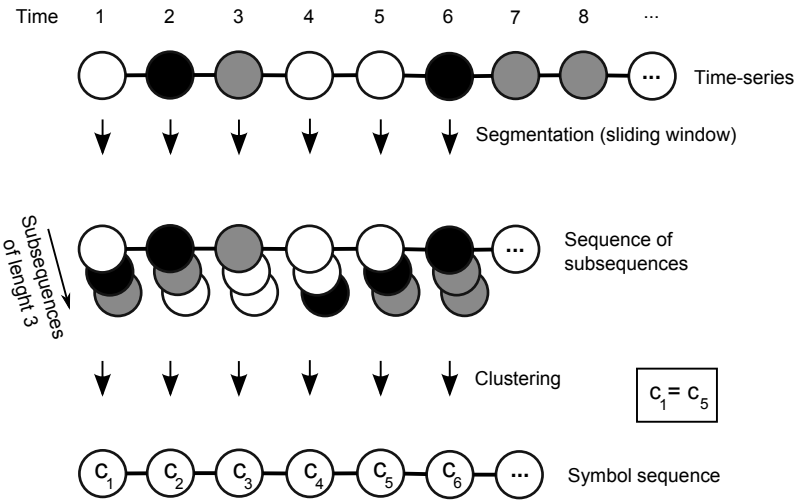

Figure 4.1: Overview of the symbolisation of a time-series using a sliding-window segmentation and clustering discretization for a random time-series

symbol sequence as described above using the symbols of each cluster. For the remainder of this thesis, if the focus of a particular description is on the set of subsequences itself, the term 'cluster' is used. If the description relates to a cluster as part of the symbol sequence the term 'symbol' is used. Figure4.1 shows the discretisation of a time-series. The circles represent measures of the observed variable of the time-series. In the figure, circles with the same shade represent the same value in the time-series. The circles are highlighted to distinguish between different measurements. It can be seen that the time-series is first segmented into subsequences. A sliding window approach is used with a window size $l = 3$. Then, an alphabet of symbols is created using a clustering method. The clustering method partitions the subsequences into clusters and labels them with the corresponding symbols. Each symbol is then used to replace the respective subsequence in the time-series. The symbol $c_\mathbf{1}$ is the symbol belonging to subsequence starting at time 1, $c_\mathbf{2}$ belongs to subsequence starting at time 2, $c_\mathbf{3}$ to the subsequence starting at time 3 etc. The subsequences starting at time 1 and time 5 contain the same measurements, so they belong to the same cluster. Thus, they

are labelled with the same symbol. That is, the symbols c_1 and c_5 are the same.

In order to visualise clustering of time-series subsequences, it is possible to think of a time-series as a multidimensional vector (see Cios et al. [1998]). The dimensionality of this vector corresponds to the length of the time-series. Hence, if a sliding window of size l is used to create subsequences of a time-series these subsequences are vectors of size l. It is possible to project these subsequences into l-dimensional space, where each value contained in the subsequence has its own dimension. This is shown in Figure 4.2 where two 2-dimensional subsequences of a time-series are projected into the same 2-dimensional space. Each of the subsequences is a point in this projection. The multidimensional space is called *projection space*.

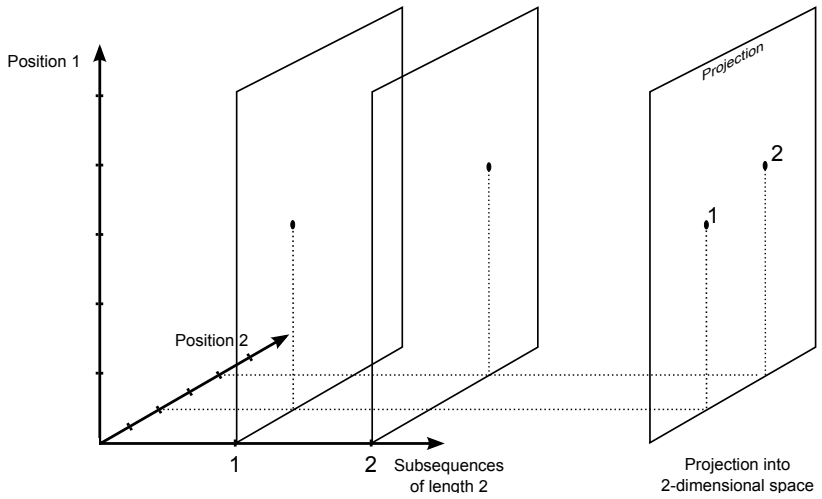

Figure 4.2: Projection of subsequences into 2-dimensional space

An important aspect of clustering is the specification of a distance function. A frequently used distance function in time-series is the Euclidean distance (see Fu [2011]). Using the Euclidean Distance computation from Section 2.2, the distance $d(x, y)$ between two subsequences $x = ((x_1, 1),\ (x_2, 2),\ \ldots,\ (x_l, l))$ and $y = ((y_1, 1),\ (y_2, 2),\ \ldots,\ (y_l, l))$ is

calculated by

$$d(x, y) = \sum_i \sqrt{(x_i - y_i)^2}^1. \tag{4.1}$$

To sum up this section, the discretisation of a time-series follows a similar approach as the discretisation of numeric attributes. This is because time-series usually contain numeric values. In addition to numeric attributes for which exists no temporal dimension, however, a time-series contains numeric values that have a temporal context with each other. That is why it is possible to interpret the values of a time-series with respect to time. That allows to discretise subsequences of time-series rather than single measurements. If the discretisation is based on subsequences then discretisation methods are employed which can handle multidimensional objects. One such method is clustering.

[1] Compare with Equation 2.4

Chapter 5

Association Rule Mining in Time-series Data

This chapter deals with association rule mining in time-series. Based on the theoretical background on knowledge discovery in time-series given in the previous chapter and association rule mining in event sequences described in Chapter 3, the problem of association rule mining in time-series is formulated in Section 5.1. The quality of discretisation with respect to the mining process is analysed in Section 5.2. In order to perform the quality analysis, a new concept of subgroups in a time-series is introduced. An interpretation of a subgroup is given in Section 5.3. Based on the results of this quality analysis, a novel discretisation method is proposed and subsequently explained in Section 5.4. It is then evaluated in Section 5.5 and applied in a mining process, which is described in Section 5.6.
The descriptions and explanations in the following sections are guided by an example of an inventory management system which is described next.

Example 2. *Imagine an inventory management system of a logistics service provider. The system collects information of all items in stock. Over the day these items are sold to customers (outflow) and re-ordered by the warehouse*

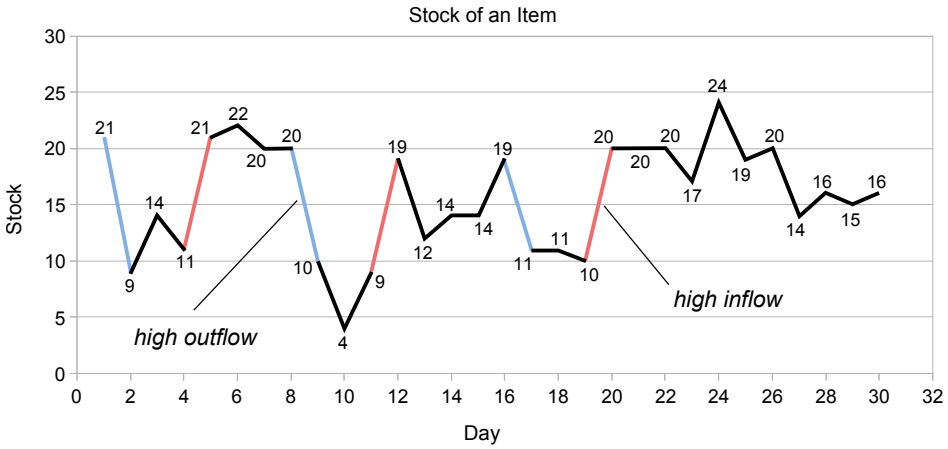

Figure 5.1: A time-series showing the stock of an item in a warehouse, monitored over 30 days. Subsequences highlighted in blue describe a 'high outflow' in the closing stock due to high demand. Those highlighted in red describe a 'high inflow' in the closing stock due to arriving orders

manager to replenish the stock (inflow). After each day of sale the closing stock of an item is recorded. The time-series in Figure 5.1 describes the development of the closing stocks of an item over 30 *days. The objective of the association mining is to discover interesting patterns in the development of a closing stock of an item. Consolidated findings shall assist the manager in optimizing their order strategy. Although demand and reorders may vary over time, there are times of an even stronger degree of increase or decrease between adjacent days. In the example, from day 1 to day 2 the stock of the item drops stronger than usual. The same is true from days 8 to 9 and from days 16 to 17. Contrary, there is a stronger increase of the stock from days 4 to 5, 11 to 12, and 19 to 20. In the sample period the time between a strong decrease and strong increase is always three days. Thus, one might expect a pattern that reflects this regularity. Such a pattern could be described as whenever the stock of the item drops by a larger amount between two days, then three days later the stock rises by a similarly high value.*

For instance, it is possible that the manager follows the strategy that whenever a large demand of items occurs to always (re-) order the same amount of

items which need three days to arrive at the warehouse. If lower demand is observed, the manager places orders of the product depending on price fluctuations on the market.

There are subsequences within the time-series that describe the "strong decrease" and "strong increase" of the item stock. A strong decrease in the stock is highlighted in blue and called high outflow (**out**). *A strong increase in the stock is highlighted in red and called* high inflow (**in**). *In Example 2, the subsequences that exhibit the pattern have a length of two days since a decrease or increase is described by two values of the time-series.*
The corresponding event sequence for the described time-series with subsequences of length 2 can be seen in Figure 5.2. The figure also shows the corresponding sequence of subsequenes for a subsequence length $l = 2$. The blue and red highlighted subsequences belong to high outflow *and* high inflow. *The goal of association rule mining in time-series is to find the described pattern between high outflow and high inflow.*

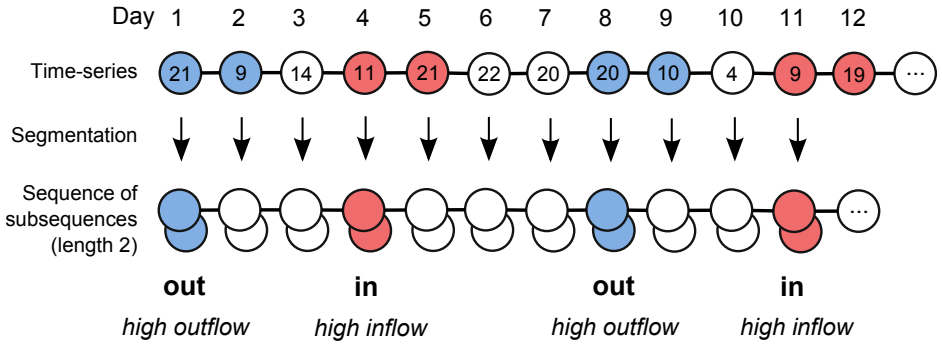

Figure 5.2: Event sequence for the time-series of Example 2 and the corresponding sequence of subsequences with $l = 2$. The values belonging to subsequences that describe *high outflow* and *high inflow* are highlighted with the respective colours

5.1 Problem Definition

Pattern mining in time-series is difficult because the characteristics of a time-series can influence the mining process (see Section 4.1). Therefore, some assumptions are made for the following mining process.

5.1.1 Assumptions

It is assumed that the time-series for which association rules are mined does not suffer from problems that typically occur in them, such as aliasing, missing values, noise and instationarity. This assumption is made to reduce the unpredictability in the values of a time-series. A more practical application of the following mining problem has to also account for aspects of data quality, which is done using methods of data cleaning (Han et al. [2006]).
Measurements of a time-series are also often made at fixed time intervals (see Hand et al. [2001]). Therefore, in order to simplify the problem of mining association rules in time-series, it is assumed that the measurements of a time-series are equidistant. That is, the time of the occurrence of each measurement determines its position in the event sequence. This assumption is also made in Udechukwu et al. [2004]. As a result of this assumption, the mining approach described in Chapter 3 can be used to find association rules in a time-series. Hence, the time-series $\mathcal{T}$ can be interpreted as an event sequence

$$\mathcal{T} = ((t_1, 1),\ (t_2, 2),\ \ldots,\ (t_n, n)),$$

with n as the number of events (t_k, k), $1 \leq k \leq n$ in the time-series (see Definition 3.1 of an event sequence). Each event corresponds to a measurement. The definitions for subsequences, episodes and sliding windows also apply on time-series.

5.1.2 Mining Problem using Symbols

Pattern mining in time-series has to take into account that time-series contain numeric values (see Das et al. [1998]). Numeric values have an effect on association rule mining in general (see Section 2.2). Therefore, an additional preprocessing step has to be carried out to enable association rule mining in time-series. Similar to association rule mining with numeric attributes, the time-series is typically discretised (see Das et al. [1998], Cotofrei and Stoffel [2002], and Hetland and Sætrom [2005]). A frequently used discretisation method is clustering, where the time-series is converted into a symbol sequence and each symbol is used to represent subsequences of the time-series (see Section 4.2). Recall that the set of all possible events of a symbol sequence is called an alphabet. Association rule mining in time-series is performed on the symbol sequences as a representation of the time-series.

Thus, if the approach for association rule mining from Chapter 3 is to be applied on a time-series, a discretised version of the time-series has to be created first. Then, the mining process is carried out on the symbol sequence $\mathcal{C} = ((c_1, 1),\ (c_2, 2),\ \ldots,\ (c_m,\ m))$ as the representation of $\mathcal{T}$, where the event (c_k, k) is the symbol c_k of the alphabet $\mathbf{C}$ at position k in the symbol sequence $\mathcal{C}$. The symbol $c_\mathbf{1}$ is the symbol belonging to subsequence $\mathcal{T}[\mathbf{1}, 2]$, $c_\mathbf{2}$ belongs to subsequence $\mathcal{T}[\mathbf{2}, 3]$, $c_\mathbf{3}$ to $\mathcal{T}[\mathbf{3}, 4]$ etc. The discretisation is carried out using a sliding window as the segmentation strategy with window size l. Thus, the last event in $\mathcal{C}$ occurs at position $m = n - l + 1$.

The problem of finding partial periodic association rules in a time-series using the discretised symbol sequence of the time-series can be stated as follows (based on the general mining problem for event sequences of Chapter 3):

Definition 5.1 (Mining problem in a symbol sequence)**.** *Given a symbol sequence* $\mathcal{C} = ((c_1, 1),\ (c_2, 2),\ \ldots,\ (c_m,\ m))$ *of a time-series* $\mathcal{T}$*, a minimum support threshold* minsupp*, a minimum confidence threshold* minconf*, and the maximum length of a pattern* w*, find all semantically unique partial periodic*

association rules R_i *of* $\mathcal{C}$ *that occur in windows of size* w *and for which* $\text{supp}(R_i) \geq \text{minsupp}$ *and* $\text{conf}(R_i) \geq \text{minconf}$.

The general approach for mining in time-series is to convert the time-series into a "suitable" symbol sequence and follow the general mining strategy for event sequences (sequence data). The term "suitable" indicates that the results of the discretisation can change depending on what discretisation method is used and on the parameter settings of the discretisation (see Das et al. [1998]). Therefore, the choice of the discretisation method influences the outcome of the mining process.
For example, for the same time-series different alphabets can be created which result in different symbol sequences on which the mining is performed. That is, the applied pattern mining process can only find patterns between those symbols that are contained in the alphabet that the discretisation provides.

To illustrate this issue, one can think of the results of two different clustering methods for the same time-series. In order to simplify matters assume that both use the same set of subsequences, created with a sliding window. It is possible (and very likely) that both methods do not lead to the same clusters. As a consequence, both symbol sequences are not identical. Thus, a pattern mining algorithm applied on these sequences is likely to also yield different results. The question that arises is how to find out whether there exist results of other clusterings that return additional patterns that could not be found with both symbol sequences alone. In Das et al. [1998], they propose multiple runs of the mining process using different parameter settings and discretisation methods. Then, another question would be whether there also exist patterns between the results of different clusterings. To deal with these problems, a new approach is made in this thesis to describe the pattern mining problem in time-series. As a mining problem the partial periodic association rule mining from Chapter 3 is used. Furthermore, the new approach requires the definition of an additional data structure.

5.1.3 Mining Problem using Subgroups

For another problem statement of mining associations in time-series in this thesis it is assumed that there exist groups of subsequences in a time-series and, furthermore, there exist patterns between these groups. The patterns can be found using a mining process. The term 'subgroup' is borrowed from subgroup discovery in data mining, introduced by Wrobel [1997]. In this thesis, subgroups can be regarded as the result of a supervised clustering process. That means, it is possible to think of a subgroup as a cluster of subsequences where each subsequence is assigned to the cluster based on the knowledge of a pattern that it exhibits in the time-series. In that respect, supervised means that for each of the subsequences of a time-series it is known what pattern they exhibit (if they do). These patterns always relate subsequences to other subsequences. Thus, a subgroup cannot constitute a pattern by itself. For instance, the subsequences that describe the *high outflow* behaviour in the time-series of Example 2 constitute a subgroup (**out**) and the subsequences describing *high inflow* behaviour belong to another subgroup (**in**). The pattern between these subgroups is that if a subsequence of **out** occurs in the time-series, after three time steps a subsequence of **in** can be observed in the time-series. A subsequence representing *high outflow* has a high value at position 1 and a low value at position 2, whereas a subsequence representing *high inflow* has a low value at position 1 and high a value at position 2. Notice that it is only possible to describe this pattern because it is assumed that both subgroups are known.

In order to be able to apply the concept of subgroups to the mining problem for association rules proposed in this thesis, it is assumed that the pattern between subgroups is a partial periodic association rule. Subgroups can then be used for a *comparison* between the rules that are identified as the result of the rule mining algorithm and those rules that could actually be found. However, in theory, subgroups may exhibit any kind of pattern with each other. Thus, they can also be used for other mining problems.

The idea of subgroups is illustrated in Figure 5.3. The subsequences of length 2 are projected in 2-dimensional space. Recall that time-series and subsequences of time-series can be considered multidimensional vectors and, thus, be projected into a multidimensional space which is called projection space (see Section 4.2). The blue points are subsequences that describe a "high outflow" (**out**) in the stock of the item. The red points describe a "high inflow" (**in**) in the stock.

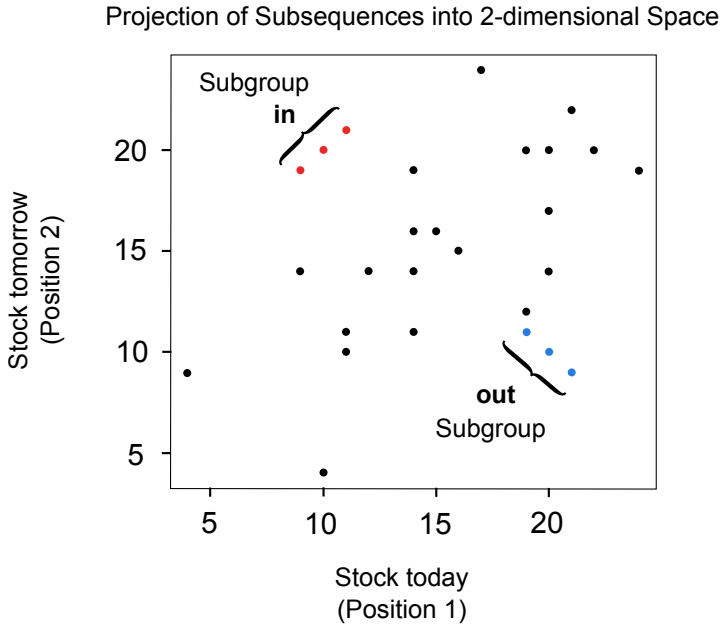

Figure 5.3: Projection of all subsequences of length 2 contained in the time-series of Example 2 into 2-dimensional space. Subsequences are highlighted according to their membership to a subgroup as high outflow (blue points), high inflow (red points), and no membership (black points)

Similar to the use of a symbol sequence as a representation of a time-series, labels for subgroups can be used to convert the time-series into a sequence of subgroup labels. That is, instead of symbols which are the result of a discretisation step, subgroup labels can be used to represent subsequences of the time-series. A sequence of subgroup labels is called a *subgroup sequence*.

Note that according to the concept of subgroups introduced here, it is possible that a time-series contains subsequences that are not a member of a subgroup. These subsequences are not of interest for the particular mining objective and hence are considered "noise". From the perspective of event sequences these unassigned subsequences are marked "else" (see Chapter 3). Therefore, noise in a subgroup sequence is represented as an event which is denoted by " $(else, p)$ ", where p is the position at which the noise occurs in the subgroup sequence.
In Example 2, only the subgroups **out** and **in** of the time-series are considered for the mining process. In practice, it is possible that other subsequences of the time-series are members of a subgroup as well. Then, the mining algorithm would have to find the rules belonging to these subgroups as well. The limitation is done to simplify the mining scenario.

Following the same notation for subgroup labels as for symbols, a subgroup sequence $\mathcal{G}$ can be written as $\mathcal{G} = ((g_1, 1),\ (g_2, 2),\ \ldots,\ (g_m,\ m))$, where m is the number of events in $\mathcal{G}$ and (g_k, k) is the subgroup label g_k occurring at position k in $\mathcal{G}$. The alphabet of a subgroup sequence is $\mathbf{G}$, which is the set of all existing subgroup labels of the time-series $G \cup \{else\}$ and $else$ is the label of an event in $\mathcal{G}$ which is considered noise.
For Example 2, the corresponding subgroup sequence can be written as $\mathcal{G} = ((out, 1),\ (else, 2),\ (else, 3),\ (in, 4),\ (else, 5),\ \ldots,\ (else,\ 29))$, where **out** is the subgroup representing *high outflow* and **in** is the subgroup representing *high inflow*. Using the concept of subgroups, the partial periodic association rule mining problem can be formulated based on a subgroup sequence.

Definition 5.2 (Mining Problem in a Subgroup Sequence)**.** *Given a subgroup sequence* $\mathcal{G} = ((g_1, 1),\ (g_2, 2),\ \ldots,\ (g_m,\ m))$ *of a time-series* $\mathcal{T}$*, a minimum support threshold* minsupp*, a minimum confidence threshold* minconf*, and the maximum length of a pattern* w*, find all semantically unique partial periodic association rules* R_i *of* $\mathcal{G}$ *that occur in windows of size* w *and for which* $\text{supp}(R_i) \geq \text{minsupp}$ *and* $\text{conf}(R_i) \geq \text{minconf}$*.*

It is clear that this problem definition can only be understood as a theoretical description of what partial periodic association rule mining in time-series is supposed to do in general. Since it is assumed that the patterns between the subgroups are known beforehand, such a mining process would not have to be carried out. However, this definition provides a way to contrast the actual situation in the mining between discretised symbols of a time-series with the theoretical situation in the mining between subgroup labels.
Both sequences vary only in the used alphabet of the event sequence. In the symbol sequence, partial periodic association rules are found for symbols. In the subgroup sequence, partial periodic association rules are found for subgroup labels. Recall that a symbol in the symbol sequence is the label of the cluster to which the subsequence is assigned to. Clusters are the result of the discretisation step. Thus, one can come to the conclusion that the goal of the discretisation of a time-series is not only to find a low-dimensional representation for the measurements of the time-series, but also to find clusters that are similar to existing subgroups in the time-series. That means, subsequences that are members of a subgroup shall be members of the same cluster after the discretisation as well. In that respect, the task for the discretisation of a time-series is to create a symbol sequence so that the patterns between subgroups of the time-series are preserved in the clusters. This way, the existing patterns between the subgroups are found if the mining process is applied on the symbol sequence.

Using these theoretical underpinnings, the goal of the partial periodic association rule mining in time-series can be stated as follows: *Create a symbol sequence that preserves the pattern structure between the subgroups of a time-series and carry out the partial periodic association rule mining to find rules between these symbols.*

As to the best of the author's knowledge, no such approach to describe pattern mining in time-series has been undertaken, so far. Based on this general goal, in Section 5.2 the quality of a frequently used discretisation

method is analysed using the theoretical concept of subgroups, and the insights gained from this analysis are then used to propose a new discretisation method in Section 5.4.

5.2 Analysing the Quality of Time-series Discretisation

This section seeks to address the problem of finding a symbol sequence that preserves the pattern structure between subgroups of a time-series. In order not to overboard the scope of this thesis the quality of clustering as one discretisation strategy is analysed. First, some problems related to clustering in time-series are discussed based on previous work in that field. Then, a novel quality analysis is carried out using the concept of subgroups of a time-series.

5.2.1 Problems in Time-series Subsequence Clustering

Clustering is part of the time-series discretisation process. This process is also described as data reduction (see Section 2.2). "The effectiveness of this technique [data reduction] depends on the nature of the data. It is much more effective for data that can be organized into distinct clusters than for smeared data" (Han et al. [2006]). The problem is that it is difficult to foresee what kind of clusters can be found in a time-series, because they depend on its characteristics, such as the data range, noise and missing values, and also on the segmentation method. In Lin and Keogh [2003], time-series clustering is said to be meaningless. Meaningless means that the outcome of a clustering method applied on time-series subsequences is random. The general idea of this observation is that the result of a clustering for random time-series data cannot be distinguished from one performed on the data of interest. The

authors in Lin and Keogh [2003] have examined both partitional and hierarchical clustering. Their observation is confirmed in a subsequent examination in Fujimaki et al. [2008]. Thus, if clustering can lead to meaningless clusters which are used to create a symbol sequence, how can patterns between the symbols of such a sequence be meaningful in association rule mining? Although it is out of the scope of this thesis to definitely answer this question the following quality analysis seeks to provide useful insights into the effect of subsequence clustering on the mining process.

5.2.2 Supervised Quality Analysis Based on Subgroups

The clustering of time-series subsequences can be considered an unsupervised data mining task (see Mörchen and Ultsch [2005]). Therefore, to evaluate the quality of clustering as part of the time-series discretisation, typically, only quality measures for unsupervised clustering can be used. The disadvantage of such measures is that it is difficult to compare the actual outcome of the clustering with a desired outcome since no information about the ground truth of the objects that are clustered is available. In this section an approach is described to compare clusters with subgroups and two criteria for measuring the quality of a discretisation method are introduced.

Due to the introduction of subgroups in this thesis it is possible to use supervised quality measures to assess the quality of subsequence clustering. On a theoretical level, the outcome of a clustering can be compared to the existing subgroups of a time-series. This analysis is theoretical, because in practice the subgroups are not known. A quality measure for supervised clustering is the accuracy, which computes the similarity between two sets (see Han et al. [2006]). In the current situation of subgroups and clusters, accuracy can be used to measure the similarity between a subgroup and a cluster. As noted earlier, subgroups and clusters are sets of subsequences of

a time-series. Thus, both can be compared according to what subsequences they contain. The advantage of using accuracy compared to precision, for example, is that all subsequences of the time-series are taken into account. That includes those subsequences which both do not share. An advantage of using quality measures for supervised clustering in general is that the quality of each cluster can be analysed with respect to all subgroups that the cluster may represent in the mining process. This enables an evaluation of how the accuracy of a cluster may influence the results of the mining process. Depending on how accurate a cluster is, possible rules for the cluster may vary compared to the rules existing for the subgroup.

The accuracy is computed for a cluster represented by a symbol c with respect to a subgroup represented by the subgroup label g and is denoted as $\text{acc}_g(c)$. Please note that for reasons of simplicity a cluster and symbol will be used in a synonym manner as well as the subgroup and subgroup label. Given a subgroup g and a cluster c, the accuracy of the cluster describes the relationship between the number of subsequences it shares with the subgroup TP (true positives), the number of subsequences both do not contain TN (true negatives) and the respective subsequences that either belong only to the cluster FP (false positives) or to the subgroup FN (false negatives). The accuracy is defined as follows:

$$\text{acc}_g(c) = \frac{TP + TN}{TP + TN + FP + FN}. \tag{5.1}$$

The accuracy is a function with domain $acc_g(c) : [0, 1]$. That is, if a cluster and a subgroup contain the same subsequences, the accuracy of the cluster with respect to this subgroup is 1. On the other hand, if the cluster contains a lot of subsequences that do not belong to the subgroup or the other way around, then the value is close to 0. The accuracy of a cluster can vary for different subgroups. Therefore, the same cluster can have a high accuracy for one subgroup but a low accuracy for another.

With the help of the accuracy of clusters and the concept of subgroups it is possible to identify the following three categories of errors that can occur in the discretisation step of a time-series. These three categories result from two sources of errors that are part of the accuracy measure. The first source of error is that a cluster can contain subsequences that do not belong to the subgroup (FP). The second is that a symbol can also miss some subsequences that belong to the subgroup (FN). Both sources of errors are common problems for supervised clustering which can now be interpreted in the context of the discretisation of a time-series as part of the preprocessing for a pattern mining approach.

The three categories for errors in a time-series discretisation are:

Too general clusters. $FP \gg 0, FN = 0$

A cluster contains the same subsequences as a subgroup, but too many additional subsequences that do not belong to the subgroup.

Too special clusters. $FN \gg 0, FP = 0$

A cluster contains only subsequences of a subgroup, but misses too many of them.

Displaced clusters. $FP > 0, FN > 0$

A cluster contains some of the subsequences of a subgroup, but also some that do not belong to the subgroup.

Each category describes an error that influences the ability of the association rule mining process to find the actual patterns between the subgroups. The effect on the association rule mining can be distinguished according to whether the subgroup for which a pattern shall be found is part of the antecedent or consequent of the association rule. Too general clusters represent too many additional subsequences. If a cluster in the antecedent is too general, the rule will not be meet the confidence threshold. That is, if too many subsequences belong to a cluster that do no exhibit the specific pattern, the

consequent will not occur and the confidence of the rule is reduced. This also means that the pattern of the corresponding subgroup will not be found. On the other hand, a too general cluster does not influence the rule mining if it occurs in the consequent part of the rule because it does not change the rule's characteristics. This is because the confidence measure does not consider the times that the consequent occurs without the antecedent occurring before. This is an issue of the support-confidence framework in general which is, for instance, discussed in Adamo [2001].
If either the antecedent or consequent part of the rule contain a cluster which is too special, then the support threshold will not be met, because the cluster occurs too infrequently. The problem of a displaced cluster is a combination of both mentioned before. If one assumes that the displacement between cluster and subgroup is fairly high, a displaced cluster in antecedent or consequent reduces both support and confidence of the rule.

The problem of clustering strategies which are used for discretisation as preprocessing for association rule mining in time-series is that they cannot distinguish between relevant and noisy subsequences. Therefore, the creation of the alphabet in the discretisation process is based on all subsequences of the time-series including those that do not exhibit any pattern. In case of k-means, for example, the same method can potentially lead to different results if applied multiple times, due to the random initialisation of the cluster centres. Since the underlying pattern structure between the subgroups is not known, it is not possible to pick the best result and continue with the mining.

Another issue for clustering strategies is that several subgroups can contain the same subsequences. That is, a specific development in a time-series can actually belong to different subgroups. Thus, these subgroups are overlapping. Currently used clustering strategies for time-series discretisation employ a hard clustering as it is described in Jain et al. [1999]. Hard clustering means that each subsequence is assigned to a single cluster. If that is the case, the discretisation can for example assign each of the "shared"

subsequences to a single cluster. Then, the accuracy of the other clusters is reduced, because the number of false negatives (FN) for them increases. On the other hand, if those subsequences belonging to multiple subgroups are split into several clusters, the accuracy for all clusters is reduced. In clustering terminology, there exists an effect called information loss that is used to express this problem (see Jain et al. [1999]). With respect to subsequence clustering, information loss means that possibly useful information about the similarity of subsequences is lost due to the hard allocation of subsequences. However, if the discretisation allows a subsequence to belong to multiple clusters, the accuracy for all subgroups can be high.
These theoretical considerations lead to two criteria which a discretisation method should suffice in order to achieve a high accuracy for the created clusters. These criteria are the completeness and the accuracy of a discretisation method.

1. **Completeness.** Completeness means that all subgroups of a time-series have to be considered during the mining process. It is, therefore, required that there exists a cluster for each subgroup. If there exist subgroups that are not represented by a cluster the corresponding pattern(s) for the subgroup cannot be found.

2. **Accuracy.** Accuracy of the discretisation describes its ability to create clusters with a high value for the accuracy. Since the accuracy varies for different subgroups, the criterion means that each cluster should have a high accuracy for the subgroup that it represents.

It is difficult to specify a minimum accuracy for clusters, because it may depend on the characteristics of the time-series, such as the range of the values, noise, as well as missing values. The more influences exist which affect the values of a time-series, the more difficult it is to achieve a high accuracy. Moreover, the considerations of a user on what is considered high and low can vary. Thus, there can be no overall consent on what minimum accuracy

of the clusters must be obtained with the discretisation step.
If both completeness and accuracy are achieved, then the influence of discretisation as an additional preprocessing step on the results of the mining process can be reduced. Although these criteria are based on subgroups for partial periodic association rules, it is possible to apply similar criteria on episodal association rules, for example, as well if a symbol sequence is used as representation of a time-series.

5.3 An Interpretation of a Subgroup

So far the concept of subgroups in a time-series is used to evaluate the quality of clusters created with a discretisation method. To make subgroups more feasible for the remainder of this chapter, a concrete interpretation of a subgroup is described in this section. Based on this interpretation a new discretisation approach will be explained in the next section.

Subgroups are introduced as sets of subsequences that exhibit interesting patterns of a time-series. It is also said that these sets are not known prior to the mining process. Therefore, subgroups are now approximated as regions in the projection space in which subsequences are projected. In such a region all subsequences exhibit the same pattern.
This approximation follows the assumption that similar subsequences, that is, subsequences that are close to each other in the projection space, exhibit the same pattern. Due to the lack of knowledge about the subgroups all dimensions of the projection space are considered equally. That is, it is assumed that the region of a subgroup is a sphere. A sphere has a centre, which can be interpreted as the *prototype* subsequence of a subgroup, a *location* of the prototype in the projection space, and a *radius*, which determines the maximum distance between prototype and a subsequence so that the subsequence still belongs to the subgroup. In other words, the radius also

determines the minimum similarity between the prototype and each subsequence in the subgroup. Since the radius determines the neighbourhood of a subgroup prototype, it is also called *neighbourhood threshold*. Following this approximation, (the region of) a subgroup can be interpreted as the *neighbourhood* of a prototype subsequence.

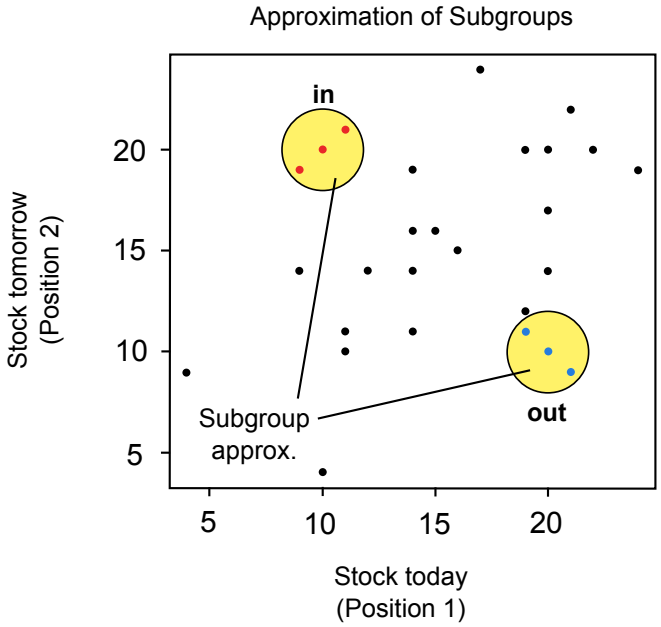

Figure 5.4: Possible approximation of subgroups for Example 2

An example of an approximated subgroup for the subsequences of length $l = 2$ of Example 2 is shown in Figure 5.4. The subsequences belonging to the subgroups are highlighted in their respective colour. The subgroup approximations are illustrated by the yellow circles. The radius of both circles is set arbitrarily. Since the subgroups are not known, it is not possible to specify the exact radius of both regions for the approximation. This circumstance is explained in more detail in the next section.

5.4 A New Discretisation Approach

After the quality of clusters with respect to subgroups in a time-series was explored in Section 5.2.2, this part of the thesis will introduce a new discretisation approach. The approach is guided by two criteria that were identified during the quality analysis. These criteria are the completeness and the accuracy of a discretisation.

5.4.1 Objective of the Discretisation

If a user has perfect knowledge about the subgroups and lacks only the patterns between them, a corresponding cluster could be created for each of the subgroups and labelled with a symbol. Based on these symbols a symbol sequence can be created using the notation described in Section 4.2. Applying a pattern mining method on the symbol sequence would then return the missing patterns. That is, given that the mining method is able to find the type of pattern between the subgroups. Unfortunately, subgroups are unknown. Therefore, the patterns between the subgroups can only be found if the discretisation strategy returns highly accurate clusters for each of the subgroups (see the two criteria for discretisation provided in Section 5.2.2).

The following discretisation seeks to primarily address two of the described problems related to the discretisation process. One of them concerns the fact that clusters can be displaced. In terms of the interpretation of a subgroup, as presented in the previous section, the displacement can be understood as a false location of the centre of a cluster with respect to the centre of the corresponding subgroup.
The second problem is that subsequences of a time-series can belong to multiple subgroups. Commonly used discretisation strategies for temporal pattern mining in time-series label subsequences with one symbol. If subsequences are only represented by a single symbol, the criteria for completeness and accu-

racy of a discretisation are not fulfilled (see Section 5.2.2.). Some subgroups may only be represented with a low accuracy, if there exist no respective clusters that contain the same subsequences. Thus, the completeness criterion is violated, because these subgroups are actually missed. Furthermore, if the assignment is hard, clusters cannot be overlapping. That means that subsequences that belong to different subgroups can only be assigned to one cluster. Therefore, the accuracy criterion is also not met.
For this reason, a new discretisation strategy is proposed in this section that attempts to fulfil the completeness criterion. Note that due to the already mentioned problems with defining the minimum accuracy of clusters (symbols), it is not the aim here to present an approach in which to create complete and highly accurate symbols at the same time. The accuracy considerations are dealt with via an additional parameter in the mining process. As for the method proposed here, it remains future work to develop a strategy to cope with accuracy considerations without using a parameter. One possible strategy will be presented at the end of this thesis.

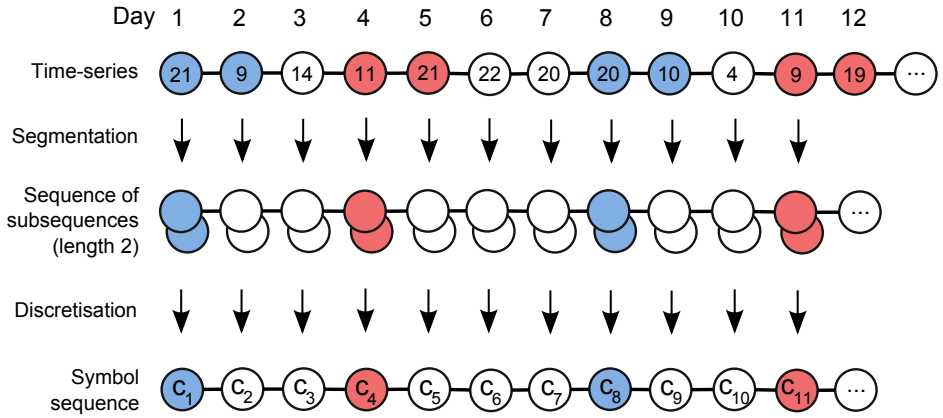

Figure 5.5: Goal of discretisation: Convert the time-series into a symbol sequence in which the interesting patterns are maintained between the symbols

The goal of the discretisation is illustrated for Example 2. In Figure 5.5 two sequences are shown. The upper sequence is the sequence of subse-

quences of the time-series. The subsequences have a length of $l = 2$. The lower sequence is the symbol sequence created by the discretisation. The presentation of both sequences corresponds to the one used in Section 4.2. At the end of the discretisation all subsequences of a subgroup should be labelled with the same symbol. (That is, all subsequences of a subgroup shall be represented by the same symbol.) In Figure 5.5 this is indicated by the colour of the subsequences in the symbol sequence. Subsequences that belong to subgroup **out** are highlighted in blue, whereas subsequences belonging to subgroup **in** are highlighted in red. In Example 2 it can be said that the subsequences of subgroup **out** should be labelled with the same symbol. That means the following condition should be satisfied: $c_1 \stackrel{!}{=} c_8 \stackrel{!}{=} c_{16}$. This aim is indicated by highlighting these symbols blue as well. Accordingly, the symbols for the subsequences of subgroup **in** have to be the same as well, which means $c_4 \stackrel{!}{=} c_{11} \stackrel{!}{=} c_{19}$. Similarly, they are highlighted in red. The choice of the sliding window is made after prior examinations of the data. Choosing a window size of $l = 2$ is only one possible way to set the parameter. In order to find potentially interesting patterns, different lengths can be tried as well. Recall that each symbol c_i is the label of subsequence $\mathcal{T}[i, j]$. For example, the symbol belonging to subsequence $\mathcal{T}[3, 4]$ is c_3, for subsequence $\mathcal{T}[8, 9]$ it is c_8, and so forth. That means that each symbol occurs when the subsequence which the symbol represents starts.

5.4.2 Basic Concept of the Discretisation

A suggestion made, for instance, by Das et al. [1998] is to perform different discretisations of the same time-series, repeat the mining for each discretisation, and compare the results. This approach has two shortcomings. First, patterns between symbols of different discretisation results are not found. Second, depending on how often this process is repeated the amount of rules can become very large. Therefore, it might be impractical to process them manually. Even the application of additional pruning measures does not

overcome the first downside. Hence, the following approach tries to achieve completeness of the discretisation in a different way. In clustering literature, there exists a concept called "overlapping clustering" (see Banerjee et al. [2005]). Overlapping clustering means that "some items are allowed to be members of two or more discovered clusters" (Banerjee et al. [2005]). In that respect, the following discretisation method performs an overlapping clustering since the items (subsequences) are assigned to multiple clusters.
The criterion of completeness of the discretisation establishes that each subgroup be considered in the mining process. Since there is no prior knowledge of subgroups of a time-series, this can be understood in a way that each *possible* subgroup has to be considered. Using the approximation of a subgroup as a sphere in the projection space (see Section 5.4), this means that any possible location of such a sphere should be considered. However, this can become computationally expensive for the mining process, because there are numerous possible locations available. Considering the second criterion for the discretisation, accuracy, means - in the same interpretation of a subgroup - that each possible radius has to be considered for the mining process. In its most extreme fashion, both criteria are considered simultaneously, and, in theory, all possible combinations between location and radius of a subgroup have to be considered. To make this combinatorial problem more manageable, an approximation is made towards the location of subgroups.

The basic approach of the discretisation process is to create a cluster for each subsequence of the time-series. That means, each subsequence of a time-series becomes the prototype subsequence of a subgroup. Due to this approach only some of the possibly many locations for subgroups are considered. In particular, the number of the resulting clusters corresponds to the number of subsequences in the time-series. This relationship can be stated as follows: Given a time-series with n measurements and a sliding window with size l, using this sliding window a total of $m = n - l + 1$ subsequences can be extracted (see Section 3.3.1). Since every cluster is created based on one of the subsequences, there are m distinct symbols in

the alphabet of the discretisation as labels for the clusters. In that respect, the number of different values in the time-series and the number of different symbols in the symbol sequence deviates only by $l + 1$. If the size of the sliding window is relatively small compared to the number of measurements in the time-series, then the range of the values in the symbol sequence and the range of the values in the time-series are similar.

The steps to carry out the discretisation correspond to the ones for the discretisation described in Section 4.2. Except that instead of applying a clustering technique to the set of subsequences, clusters are created based on the neighbourhood of each of the subsequences of the time-series. Therefore, a neighbourhood threshold has to be specified. The step of creating the clusters is called *alphabet creation*, because as a result of the clustering the alphabet of symbols is obtained. In the next step, the symbol sequence can be generated using the symbols of each cluster to represent the subsequences of the time-series. This step is called *symbol generation*. Both steps are detailed in the following sections.

5.4.3 Alphabet Creation

Discretisation requires the segmentation of the time-series. As segmentation strategy a sliding window is used to create subsequences of the same length l. The length of the subsequences determines the kind of subgroups that are represented in the mining process. To create the distance matrix the Euclidean distance between all pairs of subsequences is computed. For the distance computation between two subsequences Equation 4.1 is used. Depending on the characteristics of the time-series, other distance measures may be used as well. The choice of the measure influences the shape of the neighbourhood of the subsequences. The Euclidean distance leads to a sphere which is in accordance with the interpretation of a subgroup of Section 5.4.

	$\mathcal{T}[1,2]$	$\mathcal{T}[2,3]$	$\mathcal{T}[3,4]$	$\cdots$	$\mathcal{T}[8,9]$	$\cdots$	$\mathcal{T}[29,30]$
$\mathcal{T}[1,2]$	0.00	13.00	7.28	$\cdots$	1.41	$\cdots$	9.22
$\mathcal{T}[2,3]$	13.00	0.00	5.83	$\cdots$	11.70	$\cdots$	6.32
$\mathcal{T}[3,4]$	7.28	5.83	0.00	$\cdots$	6.08	$\cdots$	5.10
$\mathcal{T}[4,5]$	15.62	7.28	10.44	$\cdots$	14.21	$\cdots$	8.49
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$\mathcal{T}[16,17]$	2.83	10.44	5.00	$\cdots$	1.41	$\cdots$	6.40
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$\mathcal{T}[29,30]$	9.22	6.32	5.10	$\cdots$	7.81	$\cdots$	0.00

Table 5.1: Distance matrix for subsequences of length $l = 2$ of Example 2

An example of a distance matrix is shown in Table 5.1, where the subsequences of length $l = 2$ are extracted from the time-series in Example 2. Note that different sizes for the sliding window also lead to different distances and therefore to different clusters. In that sense, only subgroups that contain subsequences of length $l = 2$ can be represented with a single discretisation process. This drawback is shared with all discretisation approaches which employ a sliding window. As a consequence, the approach cannot be used to find all possible subgroups of a time-series. In order to consider subgroups of subsequences with different lengths, multiple discretisation steps have to be carried out to create the corresponding clusters.
The columns and rows of Table 5.1 contain the subsequences of the time-series. The distances vary between 0 and 21.4. It can be seen that the distances between subsequences belonging to one of the subgroups (for example, the subsequences $\mathcal{T}[1,2]$, $\mathcal{T}[8,9]$, and $\mathcal{T}[16,17]$ of subgroup **out**) are relatively small.

There are various modifications that can be made in the distance computation. It is, for instance, possible to normalise the distances by dividing them by the largest distance in the matrix. In this case, however, the resulting values are influenced by outliers in the data, which have to be taken into account. Furthermore, there exist variations of the distance measure that

may be more suitable for time-series subsequences than the Euclidean distance. For example, in Höppner and Klawonn [2009] a measure is proposed that uses a cross correlation distance between subsequences. The authors also suggest that this distance may solve the issue related with meaningless results of clustering in time-series. Then again, the problem with subsequences that belong to multiple subgroups still remains.

In order to create clusters for each subsequence a size of the neighbourhood of each subsequence is required. As mentioned before, this is done by specifying a neighbourhood threshold. The threshold is denoted by θ. With the help of the neighbourhood threshold, the distance matrix can be converted into an adjacency matrix, where each subsequence lying within the neighbourhood of another subsequence gets assigned a value of 1 and 0 otherwise for this subsequence. All subsequences within the neighbourhood of a subsequence belong to its cluster and receive the corresponding label.

	$\mathcal{T}[1,2]$	$\mathcal{T}[2,3]$	$\mathcal{T}[3,4]$	$\cdots$	$\mathcal{T}[8,9]$	$\cdots$	$\mathcal{T}[29,30]$
$\mathcal{T}[1,2]$	1	0	0	$\cdots$	1	$\cdots$	0
$\mathcal{T}[2,3]$	0	1	0	$\cdots$	0	$\cdots$	0
$\mathcal{T}[3,4]$	0	0	1	$\cdots$	0	$\cdots$	0
$\mathcal{T}[4,5]$	0	0	0	$\cdots$	0	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$\mathcal{T}[16,17]$	0	0	0	$\cdots$	1	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$\mathcal{T}[29,30]$	0	0	0	$\cdots$	0	$\cdots$	1

Table 5.2: Corresponding adjacency matrix for the distance matrix in Table 5.1 with neighbourhood threshold $\theta = 1.5$

An adjacency matrix for the distance matrix of Table 5.1 with a neighbourhood threshold $\theta = 1.5$ is shown in Table 5.2. For example, the subsequences $\mathcal{T}[1,2]$ and $\mathcal{T}[16,17]$ are both adjacent to subsequence $\mathcal{T}[8,9]$. It is important to note that $\mathcal{T}[1,2]$ and $\mathcal{T}[16,17]$ are not adjacent to each other due to the intransitivity of the distance measure. This circumstance has an effect on the resulting symbol sequence.

If there is no prior knowledge about the data, it is assumed that the same neighbourhood threshold is used for each of the created clusters. In practice, it might be interesting to set different thresholds depending on how dense the subsequences in the projection space are. An interpretation of the neighbourhood threshold is that similar subsequences shall be represented by the same symbol.

Another reason to use a flexible parameter to determine the neighbourhood is because the distances between subsequences depend on the values of the time-series. For example, a time-series that contains values that range from 0 to 1 exhibits different distances than a time-series that contains values that range from 0 to 100. The neighbourhood threshold changes the size of a cluster. As could be seen from the quality examinations in Section 5.2.2, the size of such a cluster influences its accuracy. Thus, the specification of the threshold has an impact on the accuracy of the discretisation. As a parameter the threshold can be used to adjusted the mining process to user considerations and the characteristics of the time-series.

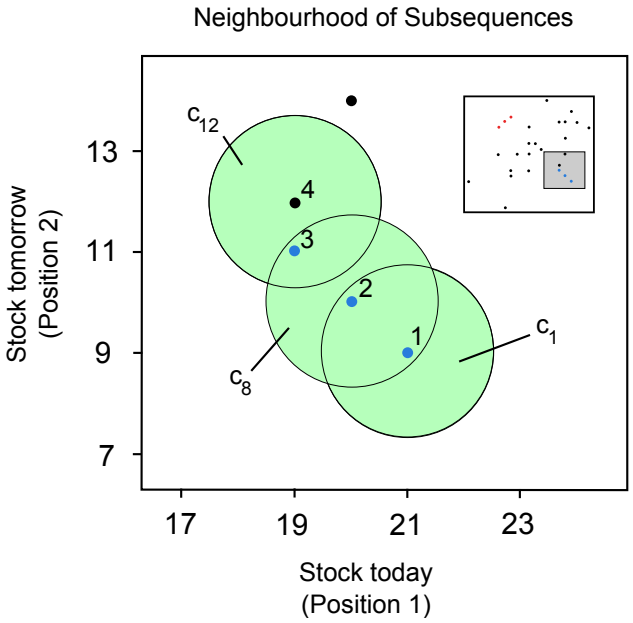

Figure 5.6: Neighbourhoods of subsequences with $\theta = 1.50$

Day	Subsequence	Labelled by Symbols	Observed Phenomenon in Time-series
1	$\mathcal{T}[1,2]$	$\{c_1,\ \underline{c_8}\}$	*high outflow*
2	$\mathcal{T}[2,3]$	c_2	
3	$\mathcal{T}[3,4]$	c_3	
4	$\mathcal{T}[4,5]$	$\{c_4,\ \underline{c_{19}}\}$	*high inflow*
8	$\mathcal{T}[8,9]$	$\{c_1,\ \underline{c_8},\ c_{16}\}$	*high outflow*
11	$\mathcal{T}[11,12]$	$\{c_{11},\ \underline{c_{19}}\}$	*high inflow*
16	$\mathcal{T}[16,17]$	$\{\underline{c_8},\ c_{12},\ c_{16}\}$	*high outflow*
19	$\mathcal{T}[19,20]$	$\{c_4,\ c_{11},\ \underline{c_{19}}\}$	*high inflow*

Table 5.3: Symbols representing subsequences in the symbol sequence based on a neighbourhood threshold $\theta = 1.5$

Examples for clusters created with the new discretisation are shown in Figure 5.6. The points in the figure are the projections of five subsequences into 2-dimensional space. The neighbourhoods of three of these subsequences are depicted as circles. Two of the clusters belong to subsequences that are members of subgroup **out**, $\mathcal{T}[1,2]$ (=1) and $\mathcal{T}[16,17]$ (=2), the third neighbourhood belongs to subsequence $\mathcal{T}[12,13]$ (=4). It is clear to see that the neighbourhoods are overlapping. Notice that all three subsequences of subgroup **out** are contained in the neighbourhood of subsequence $\mathcal{T}[8,9]$ (=3). The used neighbourhood threshold is $\theta = 1.50$.

Based on the adjacency matrix the clusters for each prototype subsequence can be created. Therefore, the set of all adjacent subsequences of a prototype subsequence becomes the cluster of this subsequence. Hence, a total of m (overlapping) clusters are created during the discretisation process. The more clusters a subsequence belongs to, that is the more subsequences it is adjacent to, the more labels it gets. Thus, for each subsequence exists a set of symbols by which it is labelled. Such a set of symbols is shown in Table 5.3. It can be seen that some subsequences are represented by multi-

ple symbols. Recall that symbol c_1 is the symbol belonging to subsequence $\mathcal{T}[1,2]$, c_2 belongs to subsequence $\mathcal{T}[2,3]$ etc. For example, subsequence $\mathcal{T}[1,2]$ is represented by two symbols, c_1 and c_8. Subsequence $\mathcal{T}[2,3]$ is only represented by symbol c_2. This happens because no other subsequence is adjacent to $\mathcal{T}[2,3]$ using a neighbourhood threshold of $\theta = 1.50$.

5.4.4 Symbol Sequence Generation

For each subsequence of a time-series there exists a set of symbols as shown in Table 5.3. Typically used discretisation methods only have a single symbol for each subsequence. That is, each subsequence is represented by one symbol only. Due to the overlapping nature of neighbourhoods used in the new discretisation approach, subsequences can be represented by multiple symbols depending on the location of a subsequence in the projection space. The sets of symbols are used to create a (more complex) symbol sequence following the basic approach described in Section 4.2. For each symbol representing a subsequence an event is created. The event contains the symbol as a value and the position is determined by the integer for the start of the represented subsequence in the time-series. The resulting symbol sequence for the subsequences of length $l = 2$ of Example 2 with neighbourhood threshold $\theta = 1.50$ is shown in Figure 5.7. Contrary to symbol sequences typically created with discretisation, the symbol sequence shown in Figure 5.7 is a bipartite graph. The bipartite graph allows one to consider different paths from the start of the symbol sequence to the end and each path accounts for a specific representation of all subsequences. Due to this, it is possible to maintain more information within the symbol sequence, because each subsequence preserves its uniqueness for the mining process. This is due to the fact that each subsequence is considered individually with its neighbourhood. The original aim of the discretisation is to satisfy the following two statements: $c_1 \stackrel{!}{=} c_8 \stackrel{!}{=} c_{16}$ and $c_4 \stackrel{!}{=} c_{11} \stackrel{!}{=} c_{19}$. Both statements mean that each of the symbols representing the subsequences of either subgroup have to be

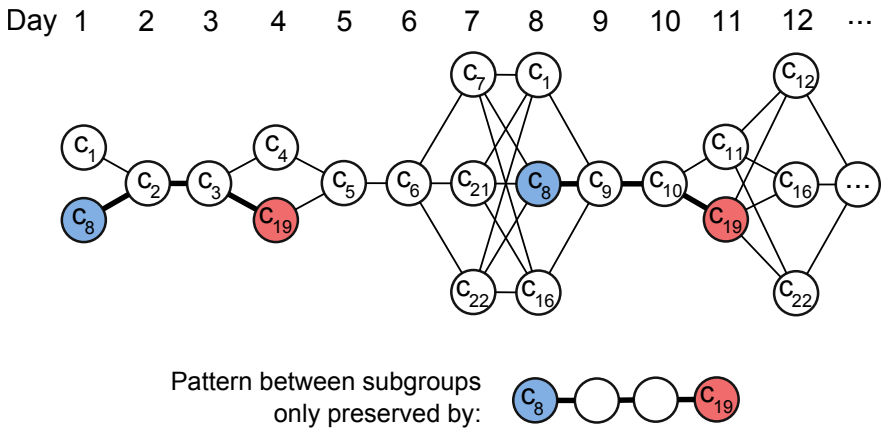

Figure 5.7: Symbol sequence for Example 2 using the proposed discretisation method with a neighbourhood threshold $\theta = 1.50$

the same. That is, the symbol representing subsequence $\mathcal{T}[1, 2]$, the symbol representing subsequence $\mathcal{T}[8, 9]$, and the symbol representing $\mathcal{T}[16, 17]$ have to be the same. (And this symbol may not occur elsewhere.) As can be seen in Figure 5.7, this aim is achieved not by making the symbols equal but by allowing each symbol to occur multiple times in the symbol sequence parallel to others. Although it actually remains to be seen until after the mining process has been applied to the symbol sequence, it can already be noticed that the pattern structure between subgroup **out** and subgroup **in** is maintained between symbol c_8 and symbol c_{19}, because both represent all of the subsequences belonging to the respective subgroups. Particularly, c_8 represents all subsequences belonging to subgroup **out**, and c_{19} every subsequence belonging to subgroup **in**. The symbol sequence in the figure does not contain events after day 12, but the symbols for the subsequences of both subgroups not shown in the figure can be found in Table 5.3. It is important to note that other symbols do not preserve the pattern quite as well. For example, symbol c_{16} does not represent $\mathcal{T}[1, 2]$, and symbol c_{11} does not represent subsequence $\mathcal{T}[4, 5]$. This is because the distance between $\mathcal{T}[16, 17]$ and $\mathcal{T}[1, 2]$ or between $\mathcal{T}[11, 12]$ and $\mathcal{T}[4, 5]$ is greater than the neighbourhood threshold. Therefore, the support of the patterns described by these

symbols is lower than for the rule described by c_8 and c_{19}.

5.4.5 Evaluation of the Approach

The idea of the new discretisation method is to keep the similarity information between the subsequences of a time-series in the mining process for as long as possible so that potentially interesting patterns in a time-series are still contained in its discretised representation. The discretisation approach proposed here follows the same complexity reduction goals as other discretisation methods (e.g. Das et al. [1998]; Höppner [2001]; Harms et al. [2002]; Mörchen and Ultsch [2005]), but at the same time allows subsequences to be represented by multiple symbols in the symbol sequence. This way the number of distinct values of a time-series and of the resulting symbol sequence deviate only by the length of the sliding window used to segment the time-series. By creating a symbol for each subsequence in the time-series, the uniqueness of the respective subsequence is preserved during the mining process. That also means that the number of symbols in the symbol sequence adapts automatically to the length of the time-series without requiring any user input. Although this is an interesting feature of the discretisation, since the specification of clusters in k-means clustering, for instance, is difficult without prior knowledge of the subgroups. It may become an issue if too many subsequences exist in a time-series that are reasonably similar. Then the complexity of the symbol sequence increases despite the fact that only little "useful" additional information is maintained[1]. In such a case, it might become necessary to preprocess these subsequences in an additional step between time-series segmentation and alphabet creation. For example, it is possible to create an artificial subsequence based on the mean values of similar subsequences. This artificial subsequence is then used as a substitute for these subsequences in the discretisation. An additional parameter would be

[1] A similar problem is also discussed in literature on partial matching/query matching in time-series. It is called "trivial matching" (e.g. see Keogh et al. [2005])

required that specifies the minimum dissimilarity between subsequences in order for them to be considered unique. A drawback of this additional preprocessing, however, is that due to slight variations between the subequences which are to be "merged" together, a possible pattern may not be found by an artificial subsequence, but it may be found for one of the individual subsequences. In a practical case a user would have to contemplate using such an additional preprocessing step or just the raw set of subsequences.

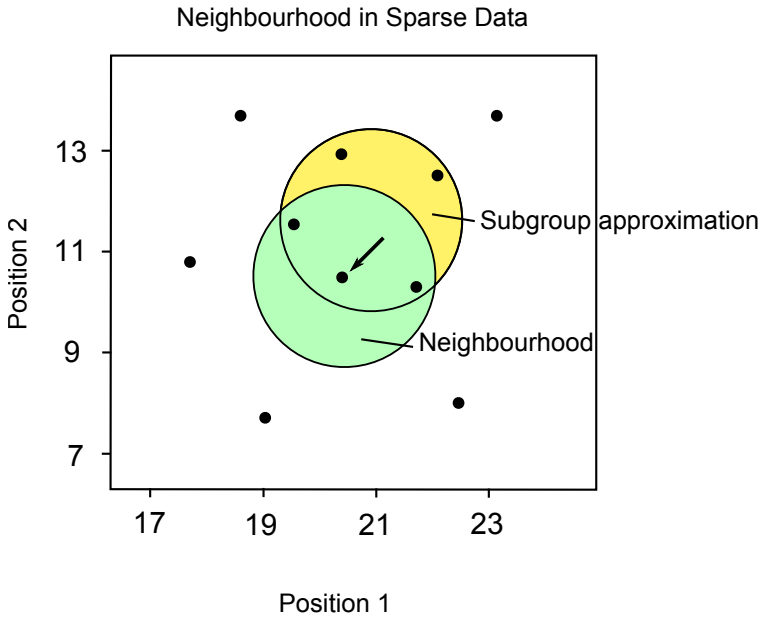

Figure 5.8: Difficulties for the discretisation: Sparse data may invalidate the goal to achieve a high accuracy

Furthermore, the discretisation creates clusters based solely on the subsequences of a time-series. This is done to limit the number of possible locations of clusters. With this limitation in place, however, situations like the one shown in Figure 5.8 are not well captured. As can be seen in the figure, there is no subsequence close to the centre of the subgroup, which is represented by the yellow circle. Therefore, a possible cluster represented by the green circle misses some of the subsequences of the subgroup and hence exhibits a low accuracy. The cluster belongs to the subsequence indicated

by the black arrow. This example shall only illustrate the problem of sparse data. Therefore, the neighbourhood threshold of the cluster is not important for this consideration.

Another downside of the new discretisation approach is the sliding window as segmentation strategy for the time-series. Sliding windows are often considered inappropriate as preprocessing for a subsequent pattern mining process (see Chiu et al. [2003]; Fu [2011]). This is because they extract only subsequences of the same length, whereas patterns in a time-series are likely to be found between subsequences with different lengths as well. In that respect, there are two aspects that have to be mentioned for the current approach. The first is that the new discretisation method does not seek to be complete in the sense that *all* existing subgroups with varying lengths are represented as the result of the discretisation. To consider subsequences with different lengths, the discretisation has to be performed multiple times for different sizes of the sliding window. The result is a number of symbol sequences. These symbol sequences can then be used as input to the mining algorithm described in Chapter 3. Notice that the algorithm is able to process co-occurring events which also includes co-occurring event sequences.
The second aspect is that the new discretisation method is to be understood as a proof of concept for the novel approach of allowing subsequences to belong to multiple clusters. Thus, the method itself does not claim to revolutionise time-series discretisation in general.

5.5 Mining Partial Periodic Association Rules in a Time-series

The discretisation approach described in the previous section creates a symbol sequence that is likely to contain co-occurring events. Therefore, the extended version of the mining algorithm described in Section 3.5 is used to

find association rules between these events.

5.5.1 Basic Concept of the Mining Process

The mining approach for association rules in time-series proposed in this thesis consists of three stages. Each of the stages has subordinate tasks. The first stage addresses the *discretisation* of a time-series to create a symbol sequence. The second stage deals with the *conversion* of the symbol sequence into a set of transactions. Finally, the third stage is the application of an association rule *mining* algorithm on the previously created transactions. An overview of the mining process is given in Figure 5.9.

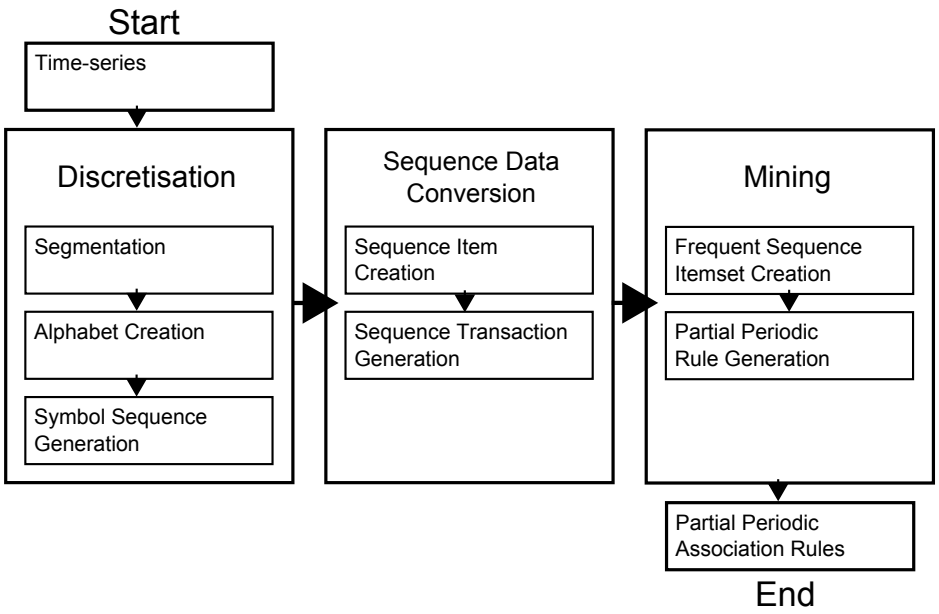

Figure 5.9: Overview of the mining process and its subordinate tasks

The basic concept of the mining approach is briefly outlined next. A *time-series* is processed with a sliding window to create a set of subsequences in the *segmentation* task (see Section 4.2). These subsequences are used to create an alphabet of symbols in the *symbol creation* (see Section 5.4.3). Based

on the created symbols a symbol sequence is generated as the result of the *symbol sequence generation* (see Section 5.4.4). The symbol sequence is then converted into a set of sequence transactions by first creating all sequence items (*sequence item creation*), followed by all sequence transactions that are contained in the symbol sequence (*sequence transaction generation*) (see Section 3.4.1). The set of sequence transactions is used as input to a conventional association rule mining algorithm in the next step in order to create all *frequent sequence itemsets*. By pruning these frequent sequence itemsets, then, all *partial periodic association rules* are discovered and displayed to a user (see Section 3.4.2 and Section 3.5).

In the course of the mining process, the user is allowed to specify six parameters to incorporate user considerations. Three of them are needed in the first stage for the discretisation method to preprocess a time-series. These are the size, $\mathbf{l}$, of a sliding window for the segmentation task, a neighbourhood threshold, θ, to determine the size of clusters in the symbol creation, and the choice of a distance function, $\mathbf{d}(\cdot, \cdot)$, which is required to create clusters of subsequences.
In the second stage, the sequence data conversion, a single parameter has to be set by the user. This parameter is the maximum pattern length, $\mathbf{w}$, specifying the maximum gap between two measurements of a time-series in order to still be considered an interesting pattern. The rule mining itself has the two common parameters for minimum support, minsupp, and minimum confidence, minconf.

5.5.2 Results of the Mining Process

The discretisation method proposed in this chapter and the sequence mining algorithm from Chapter 3 are implemented in a data mining tool called 'EARA' (partial pEriodic Association Rule mining Algorithm). With the help of EARA the introductory example of this chapter can be mined for

association rules. The core mining part of EARA is performed by an implementation of the Apriori algorithm written by Christian Borgelt [2].

Day	sequence transaction
1	$\{seqitem_1 = c_1,\ \mathbf{seqitem_1 = c_8},\ seqitem_2 = c_2,$ $seqitem_3 = c_3,\ seqitem_4 = c_4,\ \mathbf{seqitem_4 = c_{19}}\}$
⋮	⋮
8	$\{seqitem_1 = c_1,\ \mathbf{seqitem_1 = c_8},\ seqitem_1 = c_{16},$ $seqitem_2 = c_9,\ seqitem_3 = c_{10},\ seqitem_4 = c_{11}$ $\mathbf{seqitem_4 = c_{19}}\}$
⋮	⋮
16	$\{\mathbf{seqitem_1 = c_8},\ seqitem_1 = c_{12},\ seqitem_1 = c_{16},$ $seqitem_2 = c_{17},\ seqitem_2 = c_{18},\ seqitem_3 = c_{17}$ $seqitem_3 = c_{18},\ seqitem_4 = c_4,\ seqitem_4 = c_{11}$ $\mathbf{seqitem_4 = c_{19}}\}$
⋮	⋮

Table 5.4: Sequence transaction for sequence items of Table 3.2. The highlighted sequence items belong to the symbols c_8 and c_{19} that preserve the pattern between the subgroups

For the following description of mining results of the algorithm the time-series of Example 2 is used. Therefore, the time-series is first discretised with the help of the new discretisation method. The parameters which are used for the discretisation are $l = 2$ for the segmentation and $d(\cdot, \cdot)$ as the Euclidean distance. In order to show the influence of different neighbourhood thresholds on the mining results, the algorithm is run with different settings for the neighbourhood threshold. Based on the resulting symbol sequence of such a run, the set of sequence transactions is created with a maximum pattern length of $w = 4$. An excerpt of the set of sequence transactions with $\theta = 1.50$ is shown in Table 5.4.
The steps for applying the mining algorithm to a symbol sequence are es-

[2]PD Dr. Christian Borgelt is a principal researcher at the European Centre for Soft Computing in Mieres (Spain). See http://www.borgelt.net/apriori.html for the source code of the algorithm.

sentially the same as for the sample event sequence described in Section 3.5. Therefore, they shall not be described in more detail. Briefly, all frequent sequence itemsets are created, from which in turn all rules are generated using the confidence as pruning measure.
For each run of EARA a minimum support of minsupp = .10 and a minimum confidence of minconf = .75 is used. The different neighbourhood thresholds are chosen in order to demonstrate three features of the discretisation method. The results of the three runs are shown in Table 5.5.

In the first run a neighbourhood threshold of $\theta = 1.50$ is used. As a result of the mining only one rule is found. This rule is shown in the first row of Table 5.5. It contains a single antecedent item with value c_8 and a single consequent item with value c_{19}. The value c_8 is the symbol that belongs to the cluster created for subsequence $\mathcal{T}[8, 9]$. This subsequence has a high closing stock (20) at position 1 and a low closing stock (10) at position 2. The value c_{19} is the symbol that belongs to the cluster of subsequence $\mathcal{T}[19, 20]$. This subsequence has a low closing stock (10) at position 1 and a high closing stock (20) at position 2.
Since the position of symbol c_8 is 1 and of symbol c_{19} is 4, the temporal gap between both symbols is 3. The support of the rule is 10% and the confidence is 100%. In other words, this rule can be stated as follows:
In 10% of all measurements of the closing stock of the item, the stock will describe a strong decrease until the next day and if that is the case, then three days later the stock is guaranteed to increase strongly until the following day.
In fact, this rule expresses the same relationship that is also described in the introduction of Example 2.
In that respect, the symbol c_8 represents the *high outflow* in the stock of the observed item in the warehouse, whereas symbol c_{19} represents the *high inflow*. That means, the algorithm is successful in creating a cluster that appropriately represents subgroup **out** as well as subgroup **in**. Consequently, neither of the other clusters also used to represent the subgroups satisfy the support and/or confidence constraints. However, given the specified settings

of the mining algorithm, the existing pattern is identified.

	Row	Partial periodic association rule	Supp.	Conf.
First run: $\theta = 1.50$	1	$(seqitem_1 = c_8 \rightarrow seqitem_4 = c_{19})$	.10	1.00
Second run: $\theta = 2.50$	2	$(seqitem_1 = c_8 \rightarrow seqitem_4 = c_{19})$	.10	.75
Third run: $\theta = 3.00$	3	$(seqitem_1 = c_1 \rightarrow seqitem_4 = c_4)$	.10	1.00
	4	$(seqitem_1 = c_1 \rightarrow seqitem_4 = c_{11})$	.10	1.00
	5	$(seqitem_1 = c_1 \rightarrow seqitem_4 = c_{19})$	.10	1.00
	6	$(seqitem_1 = c_8 \rightarrow seqitem_4 = c_4)$	.10	.75
	7	$(seqitem_1 = c_8 \rightarrow seqitem_4 = c_{11})$	.10	.75
	8	$(seqitem_1 = c_8 \rightarrow seqitem_4 = c_{19})$	.10	.75
	9	$(seqitem_1 = c_{16} \rightarrow seqitem_4 = c_4)$	.10	.75
	10	$(seqitem_1 = c_{16} \rightarrow seqitem_4 = c_{11})$	.10	.75
	11	$(seqitem_1 = c_{16} \rightarrow seqitem_4 = c_{19})$	.10	.75

Table 5.5: Association rules for sequence transactions of Table 5.4 with neighbourhood thresholds $\theta = 1.50$, $\theta = 2.50$, and $\theta = 3.00$. The minimum support is minsupp = .10 and the minimum confidence is minconf = .75

The second row of Table 5.5 shows the result of the mining using a neighbourhood threshold of $\theta = 2.50$. As can be seen, still only one rule is found. Although this rule appears to be similar to the one of the previous run, it has a different confidence which now is 75%. This decrease in the confidence is a result of the neighbourhood of the cluster belonging to symbol c_8 becoming larger. In particular, as the neighbourhood increases an additional subsequence ($\mathcal{T}(12, 13)$) of the time-series is assigned to it[3]. This then affects the confidence of the rule. This circumstance demonstrates the effect of discretisation on rule mining as well as the difficulty to set the neighbourhood threshold appropriately.

[3]For a visualisation of this issue, Figure 5.6 shows the projection of the relevent subsequences.

The third run of the algorithm shown in Table 5.5 constitutes a special situation. This time a neighbourhood threshold of $\theta = 3.00$ is used and the algorithm returns several association rules for the time-series. At first sight, one may assume that there exist additional patterns within the time-series that could not be discovered using the configurations before. A closer look, however, reveals that each of the association rules describes the same relationship in the time-series. Without going into detail what each association rule means, it is important to understand that this effect is due to the design of the discretisation process. Since overlapping clusters are allowed to occur, it is possible to find association rules that contain different symbols but describe the same relationship. In order to clarify this point, the discretisation is able to create clusters of subsequences that share the majority of these subsequences. This is especially the case if subsequences are very dense in the multidimensional space and each of them is kept for the mining process. In order to reduce the number of such rules presented to a user, an additional postpruning step can be carried out. This postpruning reduces the actual set of association rules to those that are considered genuine by a user. With the help of this postpruning, unwanted association rules that describe the same pattern of a time-series are filtered out. The pruning process is described in the next section.

Before the postpruning of the rules is detailed, however, a brief comparison of the results of the new algorithm compared to another clustering strategy shall be described. For the same sample time-series of Example 2, a mining approach using a k-means clustering[4] with the Euclidean distance has been undertaken to compare the applicability of the new discretisation method. After extensive testing, no optimal partition for the input data could be achieved. That circumstance was mainly due to k-means being unable to create an accurate cluster for subgroup **out**. Either the subsequences that belong to the subgroup were split or too many additional subsequence were included, depending on the specified number of clusters. That is, the

[4]As implementation the k-means clustering in R was used.

best result that could be achieved receives a support of 10% and a confidence of 75%. The rule that could be found corresponds to the one in row 2. This is because the cluster representing subgroup **out** includes subsequence $\mathcal{T}(12, 13)$ as well. Similar considerations also apply to hierarchical clustering strategies, since the stopping criterion that has to be set beforehand also influences the number of created clusters. It is important to note that the accuracy of these discretisations is dependent on the chosen distance measure. Therefore, it is possible that clusterings using other distance measures than the Euclidean may perform better.
In that sense, this comparison cannot be considered exhaustive. More extensive studies on the performance of the new algorithm and its benchmark against other approaches still have to be carried out to make reliable statements about its ability to find patterns in time-series that alternative strategies fail to deliver. Due to the fact that such an endaviour encompasses a vast amount of data, studies of this kind go beyond the scope of this thesis.

The key differences between the results of typically used clusterings and the new approach proposed here are twofold. These differences are independent of particular clustering strategies. On the one hand, the new approach uses the neighbourhood of subsequences to create clusters. The neighbourhood is considered more meaningful for a mining process because it does not lead to irregular shapes of clusters that may lack the possibility of a meaningful interpretation. On the other hand, the new approach enables subsequences to be part of different subgroups of a time-series. This specifically allows the mining process to find a more complete representation of subgroups of a time-series within a single discretisation process.

5.5.3 Additional Postpruning

The options of pruning partial periodic association rules describing patterns in time-series are twofold. First, rules can be pruned using interestingness measures as they are proposed, for instance, in Wu et al. [2010]. Second, rules can be pruned if they describe the same or a similar relationship between subsequences of a time-series. While interestingness measures are commonly used to filter uninteresting rules, the second pruning option is special for the association rules described here and becomes available as a result of the new discretisation method. As could be seen in the example of the previous section, if an association rule mining algorithm is applied to a discretised time-series using the new approach, the same relationship between subgroups of a time-series can be expressed by different symbols equally well. This is because neighbouring clusters can be overlapping and thus contain a similar set of subsequences. Filtering these clusters before the actual mining process can reduce the number of such "redundant" rules, but, as mentioned before, filtering similar clusters before the mining process can also lead to some interesting patterns not being found. Therefore, even similar clusters are kept for the mining process. In order not to flood the user with redundant association rules, an additional postpruning step is therefore carried out. This postpruning takes advantage of the quality considerations described in Section 5.3.2. If different clusters are similar, their accuracy for each other is high as well. That is, if they contain a similar set of subsequences they can be used to represent each other. This is straightforward because it is the very reason why rules are redundant. In order to postprune such rules, the accuracy between symbols contained in different association rules can be computed. If association rules contain symbols where for each symbol in one association rule exists a symbol in the other association rule, the one to which it is similar, and both have the same position attribute, then only one of these rules has to be displayed to a user. This postpruning requires an additional parameter because whether two symbols are considered similar is

up to the user to determine.

5.6 General Remarks on Used Parameters

The parameters of algorithms for association rule mining, in general, need to be specified according to the purpose of the application (see Das et al. [1998]). In the particular case of mining in time-series, that includes considerations of the environment where the time-series is collected. The characteristics of the time-series have to be accounted for by defining the distance measure and neighbourhood threshold for the discretisation step of the mining process accordingly. The choice of the size of the neighbourhood threshold determines how general or special a symbol is. In order to prevent problems related to too general and too special symbols, domain expert information should be incorporated to set the neighbourhood threshold. A major advantage of the proposed discretisation method is that a neighbourhood of a subsequence is easy to grasp. Therefore, the results of the discretisation are very intuitive and the impact of adjustments on the mining process are easier to understand. For instance, using typical clustering methods, such as k-means, the resulting clusters can change entirely if a parameter is modified, so that cluster centres as well as the shape of clusters are difficult to foresee.

There is also a possibility of a trade-off between the size of the neighbourhood threshold in the discretisation step and the support threshold in the mining step. That is, the choice of a relatively small neighbourhood for each symbol will reduce the support of the corresponding rule in the mining process because fewer subsequences contribute to the rule. By adjusting the support threshold accordingly the rule can still be found. However, finding a good trade-off between neighbourhood and support threshold can require several runs of the mining algorithm or substantial domain knowledge.

Chapter 6

Conclusion

6.1 Summary

It is well known that association rule mining in time-series requires the reduction of the dimensionality of a time-series in order to perform the mining task. A basic technique for reducing dimensionality is preprocessing the time-series using a discretisation method. The result is a reduced representation of the time-series, which is then used in the association rule mining instead of the original time-series. Examinations in this thesis show that the ability of a mining process to find interesting patterns within a time-series depends on the quality of the representation of the time-series in question. These examinations are based on a mining problem for association rules in time-series which is defined in the course of the thesis. In order to evaluate the quality of such a representation, a novel concept utilising subgroups in time-series is introduced. A subgroup represents a set of subsequences of the same time-series, with each subsequence exhibiting the same temporal pattern. Based on these subgroups, a supervised quality measure is employed to determine the influence of discretisation on the outcome of the defined association rule

mining process. As a result, three categories of discretisation errors have been identified. Based on these categories, two criteria, completeness and accuracy, are proposed which can be used to evaluate time-series discretisation methods. The first criterion describes the ability of a discretisation method to represent all subgroups of a time-series in the mining process, while the latter is a measure for the similarity of a subgroup and its respective representation in the mining process.
A new type of discretisation is proposed. While conventional methods do not allow for one and the same subsequence to be represented by multiple discretised values, this new method relaxes this constraint and creates a representation of the whole time-series that has an increased complexity and is able to maintain more information for the actual mining process. This way mining in a thus discretised time-series allows for finding a greater set of interesting patterns.
The mining process is implemented in a data mining tool called EARA. The implementation of the algorithm is also able to cope with the increased complexity of the new discretised representation of a time-series. With the help of EARA, the new discretisation approach could successfully be applied to a use case and show its potential for practical considerations.
In summary, the thesis introduced a completely new approach to association rule mining in time-series. The concept of subgroups emphasises the importance of time-series discretisation as a preprocessing step in the association rule mining in time-series. This insight in turn opens up new possibilities for discovering patterns in time series analysis.

6.2 Future Work

The algorithm for association rule mining in time-series as it is proposed in this thesis has shortcomings which can partly be explained by parameters that have to be set prior to the mining. One of these parameters is a

neighbourhood threshold which is used to convert a distance matrix into an adjacency matrix during the discretisation of the time-series. If the threshold is not set appropriately it will prevent interesting rules from being found. Thus, a possibility to eliminate this parameter could improve the mining process.
One possible way to resolve this issue is an idea of fuzzy association rule mining proposed by Delgado et al. [2003]. In their work the authors introduce fuzzy items and fuzzy transactions that can be used to run association rule mining on data where each item is only present in the transaction by a certain "degree". Instead of performing association rule mining based on the (converted) adjacency matrix, this approach would make it possible to perform the mining task on the distance matrix, where the distance becomes the degree by which an item is present in a transaction. Since no conversion of the distance matrix is required, this approach would avoid the neighbourhood threshold.

Besides the efficiency of the mining process, there are at least two options for an extension to improve the ability of the mining process to find more interesting patterns in a time-series. Both address the currently made assumption that a time-series contains equidistant measurements. The first extension can allow for different intervals between adjacent measurements in a time-series. This would make it possible to apply the algorithm even on time-series where measurements are taken on an irregular basis. The second extension may consider uncertainty in the order of measurements of a time-series. In the current state, the algorithm is only able to find patterns between multiple measurements of a time-series if the measurements always occur with the same temporal gap. However, in practice it is likely that measurements also occur slightly earlier or later which may change the order of adjacent measurements. A possible approach to extend the current algorithm is described in Sun et al. [2003]. The authors consider uncertainty in an event sequence which enables pattern mining in the presence of slight variations in the occurrence of events.

Bibliography

Adamo, J.-M. (2001). *Data Mining for Association Rules and Sequential Patterns: Sequential and Parallel Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

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*, SIGMOD '93, pages 207–216.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499.

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering*, pages 3–14.

Ao, S.-I. (2010). *Applied Time Series Analysis and Innovative Computing*. 1st edition.

Baixeries, J., Casas-Garriga, G., Balcázar, J., and Lsi, D. D. (2001). Mining unbounded episodes from sequential data.

Banerjee, A., Krumpelman, C., Basu, S., Mooney, R. J., and Ghosh, J. (2005). Model-based overlapping clustering. In *Proceedings of the Eleventh*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-05).

Basu, S., Davidson, I., and Wagstaff, K. (2008). *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. 1 edition.

Batal, I., Sacchi, L., Bellazzi, R., and Hauskrecht, M. (2009). Multivariate time series classification with temporal abstractions. In *The Florida AI Research Society Conference*.

Brin, S., Motwani, R., and Silverstein, C. (1997). Beyond Market Baskets: Generalizing Association Rules to Correlations. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 265–276.

Chiu, B., Keogh, E., and Lonardi, S. (2003). Probabilistic Discovery of Time Series Motifs.

Cios, K., Pedrycz, W., and Swiniarski, R. (1998). *Data Mining Methods for Knowledge Discovery*. Kluwer International Series in Engineering and Computer Science.

Cotofrei, P. and Stoffel, K. (2002). Classification rules + time = temporal rules. In *Proceedings of the International Conference on Computational Science-Part I*, ICCS '02, pages 572–581.

Das, G., Lin, K.-I., Mannila, H., Renganathan, G., and Smyth, P. (1998). Rule Discovery from Time Series. In *Knowledge Discovery and Data Mining*, pages 16–22.

Daw, C. S., Finney, C. E. A., and Tracy, E. R. (2003). A Review of Symbolic Analysis of Experimental Data. *Review of Scientific Instruments*, 74:915–930.

Delgado, M., Marn, N., Sánchez, D., and amparo Vila, M. (2003). Fuzzy association rules: General model and applications. *IEEE Transactions on Fuzzy Systems*, 11:214–225.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 194–202.

Fu, T.-c. (2011). A review on time series data mining. *Eng. Appl. Artif. Intell.*, 24:164–181.

Fujimaki, R., Hirose, S., and Nakata, T. (2008). Theoretical analysis of subsequence time-series clustering from a frequency-analysis viewpoint.

Fukuda, T., Morimoto, Y., Morishita, S., and Tokuyama, T. (1996). Mining optimized association rules for numeric attributes. In *PODS '96: Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 182–191.

Garofalakis, M. N., Rastogi, R., and Shim, K. (1999). SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 223–234.

Han, J. (1999). Efficient mining of partial periodic patterns in time series database. In *Proc. Int. Conf. on Data Engineering*, pages 106–115.

Han, J., Gong, W., and Yin, Y. (1998). Mining segment-wise periodic patterns in time-related databases. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pages 214–218.

Han, J., Kamber, M., and Pei, J. (2006). *Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems)*. 2 edition.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 1–12.

Hand, D., Mannila, H., and Smyth, P. (2001). *Principles of Data Mining*. MIT Press, Cambridge, MA.

Harms, S. K., Deogun, J., Saquer, J., and Tadesse, T. (2001). Discovering representative episodal association rules from event sequences using frequent closed episode sets and event constraints. In *In Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 603–606.

Harms, S. K., Deogun, J., and Tadesse, T. (2002). Discovering sequential association rules with constraints and time lags in multiple sequences. In *In Proceedings of the 13th International Symposium on Foundations of Intelligent Systems*, pages 432–441.

Harms, S. K. and Deogun, J. S. (2004). Sequential association rule mining with time lags. *J. Intell. Inf. Syst.*, 22:7–22.

Hetland, M. and Sætrom, P. (2005). Evolutionary Rule Mining in Time Series Databases. *Machine Learning*, 58:107–125.

Hipp, J., Güntzer, U., and Nakhaeizadeh, G. (2000). Algorithms for Association Rule Mining – a General Survey and Comparison. *SIGKDD Explor. Newsl.*, 2:58–64.

Höppner, F. (2001). Discovery of Temporal Patterns. Learning Rules about the Qualitative Behaviour of Time Series. In *PKDD '01: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 192–203, London, UK.

Höppner, F. and Klawonn, F. (2009). Compensation of translational displacement in time series clustering using cross correlation. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, IDA '09, pages 71–82.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data Clustering: A Review. *ACM Comput. Surv.*, 31:264–323.

Keogh, E., Lin, J., and Fu, A. (2005). HOT SAX: efficiently finding the most unusual time series subsequence. In *Data Mining, Fifth IEEE International Conference on*.

Keogh, E., Lonardi, S., and Chiu, B. Y.-c. (2002). Finding surprising patterns in a time series database in linear time and space. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 550–556, New York, NY, USA.

Keogh, E. J., Chu, S., Hart, D., and Pazzani, M. J. (2001). An online algorithm for segmenting time series. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 289–296.

Last, M., Klein, Y., Kandel, A., and K, A. (2001). Knowledge discovery in time series databases. *IEEE Transactions on Systems, Man, and Cybernetics*, 31:160–169.

Laxman, S. and Sastry, P. (2006). A Survey of Temporal Data Mining. 31:173–198.

Li, Y., Ning, P., Wang, X. S., and Jajodia, S. (2003). Discovering calendar-based temporal association rules. *Data Knowl. Eng.*, 44:193–218.

Lin, J. and Keogh, E. (2003). Clustering of streaming time series is meaningless. In *In Proceedings of the 8th ACM SIGMOD workshop on Research issues in*, pages 56–65.

Mannila, H. and Toivonen, H. (1996). Discovering Generalized Episodes Using Minimal Occurrences. In *Knowledge Discovery and Data Mining*, pages 146–151.

Mannila, H., Toivonen, H., and Verkamo, I. A. (1995). Discovering Frequent Episodes in Sequences (Extended Abstract). In *In 1st Conference on Knowledge Discovery and Data Mining*, pages 210–215.

Mannila, H., Toivonen, H., and Verkamo, I. A. (1997). Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1:259–289.

Méger, N. and Rigotti, C. (2004). Constraint-based mining of episode rules and optimal window sizes. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD04*, pages 313–324.

Mörchen, F. and Ultsch, A. (2005). Optimizing time series discretization for knowledge discovery. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD05*, pages 660–665.

Ozden, B., Ramaswamy, S., and Silberschatz, A. (1998). Cyclic Association Rules. pages 412–421.

Pei, J., Han, J., Mortazavi-asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16:1424–1440.

Petersohn, H. (2005). *Data Mining: Verfahren, Prozesse, Anwendungsarchitektur*. Oldenbourg.

Rasheed, F., Alshalalfa, M., and Alhajj, R. (2011). Efficient periodicity mining in time series databases using suffix trees. *IEEE Trans. on Knowl. and Data Eng.*, 23:79–94.

Rastogi, R. and Shim, K. (2001). Mining optimized support rules for numeric attributes. *Inf. Syst.*, 26:425–444.

Rastogi, R. and Shim, K. (2002). Mining optimized association rules with categorical and numeric attributes. In *TDKE*, volume 1, pages 129–50.

Roddick, J. F. and Spiliopoulou, M. (1999). A Bibliography of Temporal, Spatial and Spatio-temporal Data Mining Research. *SIGKDD Explor. Newsl.*, 1:34–38.

Saraee, M. H. and Theodoulidis, B. (1995). Knowledge discovery in temporal databases. In *In IEE Colloquium on Digest No. 1995/021(A*, pages 1–1.

Srikant, R. and Agrawal, R. (1996). Mining Sequential Patterns: Generalizations and Performance Improvements. In *EDBT '96: Proceedings of the 5th International Conference on Extending Database Technology*, pages 3–17.

Sun, X., Orlowska, M. E., and Li, X. (2003). Introducing uncertainty into pattern discovery in temporal event sequences. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 299–.

Tan, P.-N., Steinbach, M., and Kumar, V. (2006). *Introduction to Data Mining*. Pearson Education.

Udechukwu, A., Barker, K., and Alhajj, R. (2004). Discovering all frequent trends in time series. In *Proceedings of the Winter International Synposium on Information and Communication Technologies*, WISICT '04, pages 1–6.

Warren Liao, T. (2005). Clustering of time series data - a survey. *Pattern Recogn.*, 38:1857–1874.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*.

Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. In *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, PKDD '97, pages 78–87.

Wu, T., Chen, Y., and Han, J. (2010). Re-examination of Interestingness Measures in Pattern Mining: A Unified Framework. *Data Mining and Knowledge Discovery*.

Yang, J., Wang, W., and Yu, P. S. (2000). Mining Asynchronous Periodic Patterns in Time Series Data. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 275–279.

Yang, J., Wang, W., and Yu, P. S. (2001). Infominer: Mining Surprising Periodic Patterns. In *KDD '01: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 395–400.

Zaki, M. J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42(1/2):31–60.

Zhao, Q. and Bhowmick, S. S. (2003). Sequential pattern mining: A survey.