# Automated Feature Construction for Classification of Time Ordered Data Sequences

Michael Schaidnagel, Thomas Connolly
School of Computing
University of the West of Scotland
Email: B00260359@studentmail.uws.ac.uk
Thomas.Connolly@uws.ac.uk

Fritz Laux
Faculty of Computer Science
Reutlingen University
Email: fritz.laux@reutlingen-university.de

*Abstract*—**The recent years and especially the Internet have changed the ways in which data is stored. It is now common to store data in the form of transactions, together with its creation time-stamp. These transactions can often be attributed to logical units, e.g., all transactions that belong to one customer. These groups, we refer to them as data sequences, have a more complex structure than tuple-based data. This makes it more difficult to find discriminatory patterns for classification purposes. However, the complex structure potentially enables us to track behaviour and its change over the course of time. This is quite interesting, especially in the e-commerce area, in which classification of a sequence of customer actions is still a challenging task for data miners. However, before standard algorithms such as Decision Trees, Neural Nets, Naive Bayes or Bayesian Belief Networks can be applied on sequential data, preparations are required in order to capture the information stored within the sequences. Therefore, this work presents a systematic approach on how to reveal sequence patterns among data and how to construct powerful features out of the primitive sequence attributes. This is achieved by sequence aggregation and the incorporation of time dimension into the feature construction step. The proposed algorithm is described in detail and applied on a real-life data set, which demonstrates the ability of the proposed algorithm to boost the classification performance of well-known data mining algorithms for binary classification tasks.**

*Index Terms*—**feature construction, sequential data, temporal data mining**

## I. Introduction

This work extends our previous work in the field of feature construction reported in [1]. It presents new feature construction techniques as well as new experimental results.

Significant amounts of data are being generated on a daily basis, in almost every industry and scientific research area. Advancements in computer science as well as computer hardware enable us to store these data. The rate of growth of data surpasses the capability of analysing all the stored data. It is believed that less than 10 % of all data stored is retrieved or analysed [2]. Particularly in the e-commerce area it is common to log all user activities in an online shop. Such data can be ordered by their timestamp and can be allocated to data sequences of particular users. However, the logged activities or actions are not stored in a form that enables immediate data mining. Therefore, it is important to pre-process the data before analyzing it (see also [3] [4]). When data is only represented by primitive attributes and there is no prior domain expert knowledge available, the pre-processing task becomes challenging and creates the need for automated techniques. At this point attribute selection and/or feature construction techniques need to be applied. Attribute selection can be defined as the task of selecting a subset of attributes, which are able to perform at least as good on a given data mining task as the original attributes set. The original values of the data set are called attributes, while the constructed data are called features. It is possible that primitive attributes are not able to adequately describe eventually existing relations among primitive attributes. Such interrelations (or also called interactions [6]) can occur in a data set if the relation between one attribute and the target concept depends on another attribute (see also [7]).

### A. Structure of the paper

The remainder of this section will provide a short introduction into the related fields of Feature Construction and Sequential Data Mining. It will also present the problem at hand. Section II will provide a short overview about the related research fields and briefly introduces well-known Feature Construction techniques. Subsection II-C will highlight our contribution to the particular research field of sequential data classification. The characteristics of such data are described in Section III. Our approach to sequential feature construction will be described in detail in Section IV. This is followed by an experimental analysis in Section V, in which we demonstrate the ability of our proposed algorithm to boost classification performance on a real-life data set. The paper then provides some conclusion (Section VI) and future work (Section VII).

### B. Feature Construction

Attribute selection alone can fail to find existing interaction among data. Therefore, one goal for feature construction is to find and highlight interactions. Feature construction can be defined as the process of creating new compound properties using functional expressions on the primitive attributes. There are also the terms Attribute Construction (Han and Kamber [3]) and Feature Extraction (Guyon et al. [9]) used in the literature to denote this research area. Guyon et al. are more focused on the Feature Selection task and uses the term Feature

Extraction as a compound term to denote both Feature Construction and Feature Selection tasks. This work will continue to use the term Feature Construction. Feature Construction is part of the data preparation step within the KDD process. There are two groups of data preparation techniques: one contains techniques that do not alter the space dimensionality of the given data, such as signal enhancement, normalization and standardization. The other group reduces or enlarges the feature space. Examples for this group are non-linear expansions, feature discretization [9]. Feature Construction can fit into both groups, depending on the goals of the data preparation step. Another goal of Feature Construction is to reduce the data dimension by removing redundant or irrelevant attributes [7]. This is done by constructing new features out of several of the given attributes to help the mining process [3]. In this case the constructed feature replaces the attributes it was constructed from [7]. However, it is important to not discard important information, which is necessary to describe the target hypothesis. If done correctly, Feature Construction is the key data preparation step to build classifiers that are able to describe complex patterns. The positive impact of feature construction was also shown in a comparative study focusing on predictive accuracy [7]. Liu and Motoda define Feature Construction as the process *'that discovers missing information about the relationships between features and augments the space of features by inferring or creating additional features'* [4]. This means that the original representation of data is altered and the feature space is extended by the new features. Usually, logical operators are used to combine features. A simple example for a Feature Construction technique on a two dimensional problem is the following: assume that $A_1$ is the width and $A_2$ is the length of a rectangle. This can be transformed into a one-dimensional problem by creating the feature $F_1$ as area $F_1 = A_1 * A_2$ [4]. However, the success of feature construction is dependent on the goal of the Data Mining problem at hand. There is no use in calculating the area as a feature; if the pattern (that can be used for discrimination between the two given labels) is connected to the aspect ratio of the rectangles. Shafti and Pérez distinguish between two types of features construction techniques in terms of their construction strategy:

- hypothesis-driven: create features based on a hypothesis (which is expressed as a set of rules). These features are then added to the original data set and are used for the next iteration, in which a new hypothesis will be tested. This process continues until a termination requirement is satisfied.
- data-driven methods: create features based on predetermined functional expressions, which are applied on combinations of primitive features of a data set. These strategies are normally non-iterative and the new features are evaluated by directly assessing the data.

The transformation of the feature space is a standard procedure in Data Mining, since it may improve the recognition process of classifiers. In general the transformation function is denoted as $y = F(x)$. It is used to transform an $n$-dimensional original pattern $x$, that exists as a vector of the $n$-dimensional pattern space, into an $m$-dimensional pattern $y$ [10]. Finding a good transformation function is very domain specific and also depends on the available measurements [9]. After the transformation, data objects are represented as feature vectors in the expanded and augmented feature space. This effectively pulls apart examples of the same class, so that it is easier for the classifier to distinguish them [12]. However, Feature Construction must be used with some precautions: if a new classification problem is presented, it is not obvious, which of the various data representations should be used. It is also possible that none of the constructed features are able to express the target concept sufficiently. This can be the case in many real-life scenarios and it needs to be dealt with by using domain-specific knowledge. Feature Construction techniques are mostly based on a fixed set of basic operators. There is no easy way to alter the existing constructor set. This is also a disadvantage if the classification problem requires a combination of several construction functions to find a discriminatory form of data representation [11]. Some feature construction techniques only use Boolean representations of features, which cover only part of the potential relations between data attributes. Markovitch and Rosenstein [11] also points out that the basic Boolean operators such as AND and OR are already inherently represented in the structure of a decision tree.

### C. Sequential Data Mining

Feature Construction prepares the data before the actual mining is done. This can be difficult, if the data has a complex structure, such as sequences. Therefore, this section briefly introduces the rather young research field of Sequential Data Mining. Due to the increasing ability to store complex data sequences, it has become one of the most important and active subfields of data mining research. Dong and Pei [19] define it as special subfield of data mining for certain structured data. The term structured data thereby refers to data that is structured in an explicit way and comprises of a set of data items. In terms of sequences this structure can include (partial) orderings, temporal orderings, hierarchical structure as well as network structures. A more formal description of the sequence structure is also given in Section III. Other forms of structured data, which are not in the scope of this work, are for example tree data, graph data, time series data or also text data.

The complex data structure of sequences is what sets Sequential Data Mining apart from standard Data Mining. Although the structure makes it more difficult to mine sequential data, there is also the reward to access information that can be contained in the structure of a sequence. Bautista-Thompson and Brito-Guevara [20] stress that the collective behaviour and the hidden relations between such data, can contain decisive information. Furthermore, they point out that the structure of sequences can have a certain dynamics (such as stationary, random, complex).

*1) Tasks in Sequential Data Mining:* Sequential Data Mining was primarily applied in the field of bioinformatics on genomic data and also, in the field of business intelligence on transactional data (especially from the retailer industry). Therefore, the following three tasks of Sequential Data Mining have emerged:

- Clustering: This task is about the grouping of unlabelled sequences into clusters. In general, this task is solved by combining well-known clustering algorithms with an appropriate distance function that is applied to sequences. Therefore the special properties of sequences and their structure need to be taken into account.
- Classification: This is most common task that includes building a classifier that is able to distinguish between two existing classes (labels). This is normally achieved by combining standard classification methods in conjunction with suitable feature construction techniques. The goal for the classification can be to decide if a sequence belongs to a certain class or if a sequence contains a subsequence of interest and its position (especially interesting for comparison of genomic data). The presented work focuses on the classification task.
- Hybrid: As the name suggests, this task is concerned with both: the identification of sequences classes as well as the characterization of the occurring sequential patterns [19].

*2) Issues in Sequential Data Mining:* Research in Sequential Data Mining usually revolves around so-called sequential databases in order to find sequential patterns. Therefore, sequential data mining research should consider the following four technical issues:

- Concept formulation: creating new concepts that lead to advances in the research field
- Design: creating novel techniques that are able to handle large volumes of data with a large number of dimensions. The techniques need to be able to handle the complex data structure while being able to take advantage of the underlying structure of the given sequential data.
- Optimizing cluster/classifying quality: modifying/altering existing techniques to achieve a better accuracy. Quality measurements in terms of classification are accuracy, precision and recall. In terms of clustering inter-precision or inter-cluster similarity are used.
- Optimizing pattern interestingness: this task aims to improve techniques in terms of their usefulness for the user. Measures include support, confidence, lift, novelty and actionability. Xing et al. [21] state that in addition to accurate quality results, the interpretability of sequence classifiers is both important and difficult

This research work deals with all four issues and focuses on the concept formulation as well as the design of a new sequence classification algorithm.

### D. Problem description

As discussed in previous sections, the KDD process and Data Mining are about finding patterns in data. Initially these data comprised of static feature vectors that did not change over time [8]. The later years have brought more complex objects that need to be stored. The latest development in data collection and storage technologies allows companies to keep extremely large quantities of data relating to their daily activities [5]. This process introduced the temporal dimension into the field of Data Mining and allowed the storage of evolving (or dynamic) data over time. However, this dimension is neglected by most of the researchers: *'In Data Mining community, researchers pay little attention to time-stamps in temporal behavior [. . . ] during classification'* [14]. This is quite a sub-optimal situation since *'knowledge about the behavior of objects is an integral part of understanding complex relationships in real-world systems and applications'* [8]. Time is necessary to markup complex behaviour. Kriegel argues that due to historical reasons (i.e., given their static data during the 1980s) many researchers created their algorithms only for static descriptions of objects and are therefore not designed to input data with dynamic behavior. The inclusion of dynamic properties of modern complex data models would allow revealing the information hidden in their temporal aspect and in addition to that, describe the relationship between complex objects. The type of information that is visible in the temporal dimension of a series of events is called sequential pattern.

Most of the sequence analysis work (see also Section II) is focused on finding frequent item sets, associate them with a certain order and then predict what items are bought next in a sequence. The research work described in this article is about finding a technique that is able to take the time span between a series of events into account and unveil hidden information that can be used for classification purposes.

This work will present an algorithm that is able to find discriminatory patterns in temporal based data and use them for classification purposes. The algorithms suggested so far have not been able to use the information hidden in sequential and time ordered data. Such information can be captured by creating sequence based features out of the original attributes of the given data set. The time dimension is thereby used in the construction step.

## II. Related Work

Earlier work in the field of feature construction was done by Setiono and Liu [13]. They used a neuronal network to construct features in an automatic way for continuous and discrete data. Pagallo [15] proposed FRINGE, which builds a decision tree based on the primitive attributes to find suitable boolean combinations of attributes near the fringe of the tree. The newly constructed features are then added to the initial attributes set and the process is repeated until no new features are created. Zupan and Bohanec [16] used a neuronal net for attribute selection and applied the resulting feature set on the well known C4.5 [17] induction algorithm. Feature construction can also be used in conjunction with linguistic fuzzy rule models. García [18] et al. use previously defined functions over the input variables in order to test if the resulting combination returns more information about the classification than the single variables. This process can lead to fuzzy rules of the following schema, which can include functions in the antecedent:

IF $x_1$ IS $A_1$AND $SUM(x_1, x_2)$ IS $A_3$ THEN $Y$ IS $B$

$A_3$ and $B$ represent fuzzy subset values that belong to the function‘s domain. The used function $SUM(x_1, x_2)$ is thereby treated as a new variable. However, in order to deal with increasing complexity of their genetic algorithm in the empirical part, García only used three functions ($SUM(x_i, x_j)$, $PRODUCT(x_i, x_j)$, $SUBSTRACT\_ABS(x_i, x_j)$) to enlarge the feature space. Another approach to feature construction, which utilizes a genetic algorithm, is described by Alfred [23]. Although, his approach is not using different functions to create new combinations of features, it can create a big variety of features since it is not limited to binary combination. That means that it is able to combine more than two attributes at a time. The genetic algorithm selects thereby the crossover points for the feature sequences. Sia [24] proposes a 'Fixed-Length Feature Construction with Substitution' method called FLFCWS. It constructs a set that consist of randomly combined feature subsets. This allows initial features to be used more than once for feature construction.

The next two subsections present the two most famous Feature Construction techniques for sequential data in greater detail.

## A. MFE3/GA

Shafti [7] presents MFE3/GA (Multi-Feature Extraction using GA), a method that uses a global search strategy (i.e., finding the optimal solution) to reduce the original data dimensions and find new non-algebraic representations of features. Her primary focus is to find interactions between the original attributes (such as the interaction of several cards in a poker game that form a certain hand). MFE3/GA basically searches through the initial space of attribute subsets to find subset of interaction attributes as well as a function over each of the found subsets. The suitable functions are then added as new features to the original data set. The C4.5 learner is then applied for the data mining process. So far only nominal attributes are being processed, so that class labels and binary/continuous attributes need to be normalized. A feature is thereby a bit-string of length $N$, where each bit shows the presence or absence of one of the $N$ original attributes. Subsets of these features are associated with a function defined over the attributes in the subset. This allows a non-algebraic (operator-free) representation of the original attributes. The output of the associated functions $f_i$ for each subset $S_i = (x_{i_1}, \ldots, x_{i_m})$ are basically the binary class labels. The labels are retrieved from the training samples that match the subset. It can be possible that both labels for one subset are occuring in the training data. In this case a so called mixed-tuple label can be associated with the subset (other labels are types pure and unknown).

## B. FeatureMine

Lesh, Zaki and Ogihara present FeatureMine [12] - another well known feature construction technique for sequential data. It combines two data mining paradigms: sequence mining and classification algorithms. They understand sequences as a series of events, e.g., $AB \rightarrow B \rightarrow CD$. There is also a timestamp associated with each event. FeatureMine starts by mining frequent and strong patterns within the sequences. Frequency is defined by a threshold that is specified by the user. Strong is defined as a confidence level that needs to be over a user specific threshold. The found frequent sequence patterns are pruned and selected using some heuristics. The prevailing sequences lattices are stored in a matrix $n * m$ database layout, whereby the rows $n$ represents the sequences and the columns $m$ represent the prevailing sequence lattices. The cells of the matrix contain and boolean indicator if a sequence contains the corresponding sequence lattice. The constructed features are associated with a class label and then feed into the Naive Bayes classification algorithms.

## C. Contribution

We propose an automated algorithm that is able to systematically construct and assess suitable new features based on data sequences for binary classification tasks. It thereby is also able to utilize the time dimension in a sequence of events in order to access information, which can have a significant impact on the discriminatory power of features. Thereby, the algorithm transforms sequential data into tuple-based data in a way, that allows standard algorithm such as Neuronal Networks, Bayesian Belief Network, Decision Trees or Naive Bayes to be applied on sequential data.

So far, feature construction techniques build new features by combining columns of a data set (i.e., 'horizontally'). We also apply these techniques with a larger variety of mathematical operators. In addition to that, we are able to utilize the time elapsed between data points. Our approach is novel, since we include the vertical dimension of data, i.e., the rows of a sequence, in order to create new features. This is achieved by combining numeric values (or its probabilities in terms of string attributes) of the corresponding occurrences. The original values are aggregated during the feature construction process. This allows to store sequence based information on tuple level. As a result of that, the above mentioned standard algorithms can be applied (not all are able to handle sequenced data sets right away).

The proposed techniques are extending the given problem-space and search for a combination of dimensions that allow to separate the binary classes that need to be classified. It thereby utilizes abstracted patterns that can occur in the data and is able to validate the created combinations.

## III. General characteristics of sequential data

This work often refers to the term sequential data. Thereby, we understand data, that can be ordered by time and can be grouped to logical units (i.e., the sequence). A simple example for that are sessions in an online shop. Customers

TABLE I: Schema of sequence data

| $r$ | $t$ | $s_{id}$ | $a_1$ | $a_2$ | $\dots$ | $a_i$ | $s_{label}$ |
|---|---|---|---|---|---|---|---|
| $r_1$ | $t_1$ | $s_{id_1}$ | $a_{1_1}$ | $a_{2_1}$ | $\dots$ | $a_{i_1}$ | 0 |
| $r_2$ | $t_2$ | $s_{id_1}$ | $a_{1_2}$ | $a_{2_2}$ | $\dots$ | $a_{i_2}$ | 0 |
| $r_3$ | $t_3$ | $s_{id_1}$ | $a_{1_3}$ | $a_{2_3}$ | $\dots$ | $a_{i_3}$ | 0 |
| $r_4$ | $t_4$ | $s_{id_2}$ | $a_{1_4}$ | $a_{2_4}$ | $\dots$ | $a_{i_4}$ | 1 |
| $r_5$ | $t_5$ | $s_{id_2}$ | $a_{1_5}$ | $a_{2_5}$ | $\dots$ | $a_{i_5}$ | 1 |
| $r_6$ | $t_6$ | $s_{id_2}$ | $a_{1_6}$ | $a_{2_6}$ | $\dots$ | $a_{i_6}$ | 1 |
| $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ |
| $r_m$ | $t_m$ | $s_{id_n}$ | $a_{1_m}$ | $a_{2_m}$ | $\dots$ | $a_{i_m}$ | $\dots$ |

can view products and put them into their shopping basket. Every action can be represented in a data set $E$ as a row $r$ with several attributes $a_i \in E$. Each row is provided with a timestamp $t$. A row can be associated to a logical unit $s_{id}$ (in our case the session id). There are $n$ sequences $s_{id_n}$ in a data set $E$. Each sequence $s_{id_n}$ consist of at least one row $r$. The number of rows in a sequence equals to the length of a sequence $ls$, so that $1 \leq ls \leq m$. Table I depicts the general schema of sequential data: It is important to differentiate between the number of rows (or tuples) $m$ of a data set and the number of sequences $n$. Sequence $s_{id_1}$, for example, has a length $ls$ of three and contains a matrix such as $s_{id_1} = \begin{pmatrix} a_{1_1} & a_{2_1} & \dots & a_{i_1} \\ a_{1_2} & a_{2_2} & \dots & a_{i_2} \\ a_{1_3} & a_{2_3} & \dots & a_{i_3} \end{pmatrix}$

In order to use our proposed method, which is described in detail in the following section, the user has also to annotate the following columns on a data set:

- $t$: timestamp column that is used for temporal based features. It is used to calculate the time elapsed between the collected data points of a sequence.
- $s_{id}$: sequence identifier column that is used for sequence aggregation. It identifies events/objects that can be logically associated to one entity
- $s_{label}$: the proposed algorithm requires a binary column as target value. This is needed in order to automatically calculate the information gain of newly constructed features. Every sequence must only have one label, i.e., a customer in an online shop is either a returning customer or not (it can not be both at the same time).

During the feature construction process, we will create a feature table, which includes the $s_{id}$, $s_{label}$ and the created features $f_p \in S$. $S$ is thereby defined as a set of constructed features. Please refer to Table II, for a schema of such a table.

TABLE II: Schema of feature table

| $s_{id}$ | $f_1$ | $f_2$ | $\dots$ | $f_p$ | $s_{label}$ |
|---|---|---|---|---|---|
| $s_{id_1}$ | $f_{1_1}$ | $f_{2_1}$ | $\dots$ | $f_{p_1}$ | 0 |
| $s_{id_2}$ | $f_{1_2}$ | $f_{2_2}$ | $\dots$ | $f_{p_2}$ | 1 |
| $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ |
| $s_{id_n}$ | $f_{1_n}$ | $f_{2_n}$ | $\dots$ | $f_{p_n}$ | $\dots$ |

The data sequences are aggregated on a tuple-based level. This enables the application of many standard classification algorithms.

## IV. Feature Construction for Data Sequences

Our goal is to extend and search the initial problem space as much as possible. Problem space is thereby defined through the primitive (original) attributes $E$, which are used to solve a binary classification task. The accessible feature space expands, if more features are constructed. Albeit, this leads to an increase in search time, it brings a higher chance to find discriminatory features. In order to keep things as simple as possible, we describe the algorithm in five different subsections, each describing a certain sort of features construction technique. Please note that the initial attributes are, in a first step, categorized in string and numeric attributes. Reason for this is, that not all described functions are applicable on string values. Note, that after each feature construction technique, we normalize the newly generated features with min-max normalization, depicted in (1). This provides an easy way to compare values that are on different numerical scales or different units of measure.

$$Normalized(e_i) = \frac{e_i - E_{min}}{E_{max} - E_{min}}, for\ E_{max} > E_{min} \quad (1)$$

The first Subsection IV-A will show construction techniques for both string and numeric attributes. The second Subsection IV-B describes construction techniques for string-only attributes. After that we will focus in the third Subsection IV-C on numeric-only construction techniques. Subsection IV-D describes temporal based feature construction techniques. This section is concluded by Subsection IV-E, which describes feature construction based on sequence distribution.

### A. Distinct occurrences based features

The general idea for this feature construction technique is to analyze if different occurrences per sequence allows to discriminate between the given labels. Basically, we aggregate all sequences $s_{id_n}$ and count the distinct occurrences (so no duplicates are counted) for each given string as well as for each numeric attribute $a_{i_m}$. The constructed features $f_p$ are then collected in $S$, together with its corresponding sequence identifier $s_{id}$ and the corresponding session label $s_{label}$. Please note that the sequence identifier $s_{id}$ is unique in $S$ (as opposed to $E$). The corresponding pseudo-code is depicted in Fig. 1.

In order to assess the quality of the new constructed feature $f_i$, we calculate two measurements in order to assess the quality. The first one is the average of all aggregated values per label $s_{label} \in \{0, 1\}$. The normalized difference between both averages is called split and is calculated as depicted in (4).

$$avg_0 = avg(\{f_p \in S | s_{label} = 0\}) \quad (2)$$
$$avg_1 = avg(\{f_p \in S | s_{label} = 1\}) \quad (3)$$
$$split_{f_i} = \frac{|avg_0 - avg_1|}{avg_0 + avg_1} \quad (4)$$

The second measurement to assess the quality of the constructed features is the number of zero and NULL values for each target label. This is a support measurement that denotes if the achieved split value is based on many sequences or not.

```
Input: $E$ // set of nominal and continuous attributes
    $s_{label} \in \{0,1\}$ // binary label indication
Def: $a_i \in E$ // single attribute or a column in a data set
    $s_{id} = (r_1, r_2, \ldots, r_m)$ // sequences of rows $r_i$
    $S = \emptyset$ // set of constructed features
for each $a_i \in E$ {
    for each $s_{id} \in E$ {
        $f_p := (|\{a_{i_n}\}|, s_{id}, s_{label})$
        $S := S \cup f_p$
        }
    }
return $S$
```

Fig. 1: Pseudo-code feature construction based on distinct occurences per label

So there could be the situation that a constructed feature has a high split value, but might be useless since it cannot be used very often due to large number of 0 values for the particular features.

### B. Concatenation based features

The purpose of this type of feature construction is to highlight simpler interactions among data. We systematically concatenate every string attribute in pairs of two and then again, count the distinct value-pairs per sequence identifier. Thereby, interactions such as, if $a_1$ AND $a_2$ have low value-pair variety for label 0, but a high value-pair variety for label 1, are highlighted. Even for data sets with a high number of different occurrences, this kind of feature construction will highlight distinct occurrences between both labels. This procedure is only applicable on string attributes. This approach is similar to most common column combinations that is described widely in the literature (e.g., [7], [16], [23]). However, we once again use this technique on a different abstraction layer since we aggregate via the sequence identifier $s_{id}$. The corresponding pseudo-code is depicted in Fig. 2.

The algorithm copies the input attribute list $E$ for looping purposes into a second variable $E_2$. Right after the second loop, it deletes the current attribute from the copied list ($E_2 - a_{2i}$). Reason for this is to avoid the same features to occur twice, due to symmetric properties. If, for example, we combine column $a_i = X$ and $a_j = Y$ of a data set, we will yield feature $XY$. This feature will have the same variability per sequence as the vice versa feature $YX$. The construction of such features can be avoided by deleting the current feature from the copied feature list $E_2$.

### C. Numeric operator based features

The basic idea of this feature construction technique is to combine two numeric attributes with basic arithmetic operators such as '+', '-', '*' or '/'. Garcia [18] and Pagallo [15] for instance are using similar techniques with fewer operators. In addition to the repeated use of arithmetic operators we, once again, use the sequence identifier attribute to aggregate the constructed features for each sequence. Lets put this into an

```
Input: $E$ // set of nominal attributes
    $s_{label} \in \{0,1\}$ // binary label indication
Def: $a_i \in E$ // single attribute or a column in a data set
    $s_{id} = (r_1, r_2, \ldots, r_m)$ // sequences of rows $r$
    $S = \emptyset$ // set of constructed features
    $E_2 = E$ // copy of $E$, used for looping
    $con()$ // concatenates two values
for each $a_i \in E$ {
    //remove $a_i$ to avoid vice versa features
    $E_2 := E_2 - \{a_i\}$
    for each $a_j \in E_2$ {
        for each $s_{id} \in E$ {
            $f_p = (|(con(a_i, a_j))|, s_{id}, s_{label})$
            $S = S \cup f_p$
            }
        }
    }
return $S$
```

Fig. 2: Pseudo-code feature construction based on concatenated string attributes

example: attributes $a_i$ and $a_j$ are combined with the multiplication operator '*' for a sequence $s_{id_1}$. The resulting feature $f = a_i * a_j$ is derived from the sequence $s_{id_1} = \begin{pmatrix} a_{i_1} & a_{j_1} \\ a_{i_2} & a_{j_2} \\ a_{i_3} & a_{j_3} \end{pmatrix}$

The sequence consists of three data points. In the aggregation phase, we sum up the multiplied attributes for all sequences $\sum_{j=1}^{3} f_{ij}$. This process is repeated for all possible combinations of numeric attributes for all of the above mentioned mathematical operators. The full pseudo-code is depicted in Fig. 3. For this technique, we also avoid vice versa features as described in previous Subsection IV-B.

### D. Temporal axis based features

The general idea for this feature construction technique is to use the time axis, which is given in each sequence by the time indicator column $t$. This is applicable for both, numeric as well as string attributes. However, for string attributes, there needs to be some preparations done, which are explained further down in this subsection. We continue here to describe the process for numeric attributes. What the algorithm basically does, is to multiply the time interval (e.g., days, hours, minutes), between earliest data point and the current data point with the numeric value of corresponding attribute, which results in a weighting.

Table III shows this for two example sequences. We have two attributes $a_i$ and $a_j$ for two sequences as well as the $t$ column. In order to calculate the temporal based feature for attribute sequence $s_{id} = 1$ in terms of attribute $a_i$, we first have to calculate the time between the earliest data point $min(t)$ with $t \in sequence(s_{id})$ and each of the 'current' data points $t$. In Table III, this is depicted by the $\Delta time\_in\_days$ column. The next step is to multiply the value of each $t_i$ in $s_{id} = 1$ with

```
Input: $E$ // set of primitive numeric attributes
    $s_{label} \in \{0,1\}$ // single value label indication
Def: $a_i \in E$ // single attribute or a column in a data set
    $s_{id} = (r_1, r_2, \ldots, r_m)$ // sequences of rows $r$
    $S = \emptyset$ // set of constructed features
    $E_2 = E$ // copy of $E$, used for looping
    $O$ // set of arithmetic operators
    $ls$ // length of a sequence $s_{id}$
for each $a_i \in E$ {
    //remove $a_i$ to avoid vice versa features
    $E_2 := E_2 - \{a_i\}$
    for each $a_j \in E_2$ {
        for each $o \in O$ {
            for each $s_{id} \in E$ {
                $f_p = (\sum_{i=1}^{ls}(a_i \ o \ a_j), s_{id}, s_{label})$
                $S = S \cup f_p$
            }
        }
    }
}
return $S$
```

Fig. 3: Pseudo-code feature construction based on numeric attributes

its corresponding delta time value: $(a_{i_1} * 1, a_{i_2} * 11, \ldots, a_{i_4} * 24)$. The sum of this value is the new time based constructed feature $f_p$. This process is repeated for all sequences $s$ and for all numerical attributes $E$.

TABLE III: Example for creating temporal based features

| $s_{id}$ | $t$ | $min(t)$ per $s_{id}$ | $\Delta time-in\ days$ | $a_i$ | $a_j$ | $s_{label}$ |
|---|---|---|---|---|---|---|
| 1 | 01.01.2013 | 01.01.2013 | 1 | $a_{i_1}$ | $a_{j_1}$ | 0 |
| 1 | 10.01.2013 | 01.01.2013 | 11 | $a_{i_2}$ | $a_{j_2}$ | 0 |
| 1 | 15.01.2013 | 01.01.2013 | 16 | $a_{i_3}$ | $a_{j_3}$ | 0 |
| 1 | 23.01.2013 | 01.01.2013 | 24 | $a_{i_4}$ | $a_{j_4}$ | 0 |
| 2 | 24.01.2013 | 24.01.2013 | 1 | $a_{i_5}$ | $a_{j_5}$ | 1 |
| 2 | 28.01.2013 | 24.01.2013 | 5 | $a_{i_6}$ | $a_{j_6}$ | 1 |
| 2 | 30.01.2013 | 24.01.2013 | 7 | $a_{i_7}$ | $a_{j_7}$ | 1 |

However, there are two directions of including the time for this feature construction technique. What we described above puts a stronger emphasis on the recent history. It is also possible to increase the weight of the past by using the (max_date - current_date) operator to calculate the $\Delta time\_in\_days$ column. An example of this is depicted in Table IV. The complete pseudo code is depicted in Fig. 4.

The above mentioned techniques are applicable on numeric attributes. For string attributes, it is possible to replace the string by the posterior probability $p(\theta|x)$ (see also Hand [26], pp. 117-118 and pp. 354-356). Thereby, $\theta$ represents the probability of the parameters for a given evidence $x$. In our example case, we have the distribution of our two labels as parameters $\theta$ and occurrences of $a_i$ as evidence $x$.

Based on this the posterior probability can be calculated as

TABLE IV: Example for creating temporal based attributes with a stronger emphasis on the distant past

| $s_{id}$ | $t$ | $max(t)$ per $s_{id}$ | $\Delta time-in\ days$ | $a_i$ | $a_j$ | $s_{label}$ |
|---|---|---|---|---|---|---|
| 1 | 01.01.2013 | 23.01.2013 | 24 | $a_{i_1}$ | $a_{j_1}$ | 0 |
| 1 | 10.01.2013 | 23.01.2013 | 14 | $a_{i_2}$ | $a_{j_2}$ | 0 |
| 1 | 15.01.2013 | 23.01.2013 | 9 | $a_{i_3}$ | $a_{j_3}$ | 0 |
| 1 | 23.01.2013 | 23.01.2013 | 1 | $a_{i_4}$ | $a_{j_4}$ | 0 |
| 2 | 24.01.2013 | 30.01.2013 | 7 | $a_{i_5}$ | $a_{j_5}$ | 1 |
| 2 | 28.01.2013 | 30.01.2013 | 3 | $a_{i_6}$ | $a_{j_6}$ | 1 |
| 2 | 30.01.2013 | 30.01.2013 | 1 | $a_{i_7}$ | $a_{j_7}$ | 1 |

```
Input: $E$ // set of continuous/numeric attributes
    $t$ // time indicator column
    $s_{label} \in \{0,1\}$ //binary label indication
Def: $a_i \in E$ // single attribute or a column in a data set
    $s_{id} = (r_1, r_2, \ldots, r_m)$ // sequences of rows $r$
    $S = \emptyset$ // set of constructed features
    $E_2 = E$ // copy of $E$, used for looping
    $ls$ // length of a sequence $s_{id}$
    $max()$ // returns max value of a set
for each $a_i \in E$ {
    for each $s_{id}$ {
        $f_p = (\sum_{i=1}^{ls}((\max_{k=1,\ldots,ls}(t_k) - t_i) * a_i), s_{id}, s_{label})$
        $S = \{S \cup f_p\}$
    }
}
return $S$
```

Fig. 4: Pseudo-code feature construction of temporal based attributes

depicted in (5)

$$p(s_{label} = 1|a_i) = \frac{p(a_i|s_{label}=1) * p(s_{label}=1))}{p(a_i)} \quad (5)$$

In order to apply this for string based attributes, we can construct new features $f$ for string attributes as depicted in (6)

$$f_p = \sum_{i=1}^{ls} (\max_{k=1,\ldots,m}(t_k) - t_i) * (p(s_{label} = 1|a_i)) \quad (6)$$

If there are occurrences in the data that have a great tendency towards a particular label (i.e., having a high probability for one label), we can make this pattern visible by multiplying the posterior possibility with the temporal axis of the given sequence.

However, if there are too many different occurrences, lets say more than 1.000 different values per attribute, this technique could have problems dealing with very small probabilities. So, it is recommended to take the logarithm of the posterior probability for cases with high cardinality.

### *E. Sequence distribution based features*

It is also possible that a discriminatory pattern evolves around distributions of numeric values in the given sequences.

```
**Input:** $E$ // set of continuous/numeric attributes
    $s_{label} \in \{0,1\}$ // binary label indication
**Def:** $a_i \in E$ // single numeric attribute in a data set
    $s_{id} = (r_1, r_2, \ldots, r_m)$ // sequences of rows $r$
    $S = \emptyset$ // set of constructed features
    $E_2 = E$ // copy of $E$, used for looping
    $O$ // set of arithmetic operators
**for each** $a_i \in E$ {
        **for each** $s_{id} \in E$ {
          $f_p = (STD\_DEV(s_{id}), s_{id}, s_{label})$
          $f_p = f_p \cup (VAR(s_{id}), s_{id}, s_{label})$
          $f_p = f_p \cup (AVG(s_{id}), s_{id}, s_{label})$
          $S = S \cup f_p$
        }
    }
**return** $S$
```

Fig. 5: Pseudo-code feature construction based on sequence distribution

Therefore, this feature construction technique is focusing on patterns that are based on variability, standard deviation and average. This construction techniques highlights patterns as for example:

- one numeric value of a class is oscillating while the value is stable for the other class
- the values for one class are more spread out than for the other class
- the average value of an attribute per sequence of a certain class is in general higher or lower, then of the other class

In principle, we calculate the above mentioned values for each sequence of each numeric attribute in a data set. The full pseudo-code is depicted in Fig. 5.

## V. Experimental setup and Results

This section is divided into three subsection in which we will first look at the technical framework we used during our experiments. This is followed by a brief look at the data profile and the corresponding classification task. The third subsection will then compare and discuss the results of our experiments.

### A. Technical Framework and Infrastructure

All implementations and experiments were carried out on a Microsoft Windows Server 2008 R2 Enterprise Edition (6.1.7601 Service Pack 1 Build 7601) with four Intel Xeon CPUs E5320 (1.86 GHz, 1862 MHz). The available RAM comprised of 20 GB installed physical memory and 62 GB virtual memory (size of page file 42 GB). The widespread freeware data mining software RapidMiner (version 5.2.008) was used for the standard methods under comparison: Decision Tree, Naive Bayes, Neuronal Network and Random Forrest (for a closer description please also see Witten [25] pp. 191-294, Han [3] pp. 291-337). The method Bayesian Belief Network required the installation of the free RapidMiner extension WEKA. We used the default parameters for all of the above mentioned classification algorithms.

### B. Data Profile

The data we used for our experiments was retrieved from the DataMiningCup 2013. The training as well as the test data set can be downloaded on the following site: http://www.data-mining-cup.de/en/review/dmc-2013/. The given historical data from an online shop consisting of session activities from customers. The goal of the task is to classify sessions into a buyer and a non-buyer class. The parameters of the train data was predefined by the task of the DataMiningCup 2013 and are as follows:

- total number of rows: 429,013
- number of sessions: 50,000
- number of numeric attributes: 21
- number of string attributes: 2

The test data was also given by the DataMiningCup requirements, which had the following parameters:

- total number of rows: 45,068
- number of sessions: 5,111
- number of numeric attributes: 21
- number of string attributes: 2

Most of the given attributes are numeric. Please note that there is no exact time column given. Therefore, we used a artificial $id$ column to map the temporal order of the various sessions. We also used this column to calculate the temporal based features described in Subsection IV-D.

### C. Comparison of original attributes vs constructed features sets

As a first step, we used the given primitive attributes to solve the task. We used the accuracy measurement (7) due to a similar label distribution (45 % to 55 %) and both labels are associated with the same 'costs' for misclassification.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (7)$$

As it can be seen in Fig. 6, the Naive Bayes classification algorithm was able to achieve better result than the base line (the other algorithms defaulted and predicted label = 0 for all sessions). The Bayesian Belief Networks are not applicable for situations in which the same $s_{id}$ can occur several time (therefore a accuracy rate of 0 %). In a next step, we used our suggested feature construction algorithm in order to aggregate the sessions and find useful features. During this process, a grand total of 860 features were created:

- # of distinct occurrences based features: 19
- # of string concatenation based features: 2
- # of arithmetic based features: 760
- # of temporal axis based features: 20
- # of sequence distribution based features: 59

All constructed features were normalized with the min-max normalization. They were, in a first series of experiments, assessed by calculating the split value for each feature. The

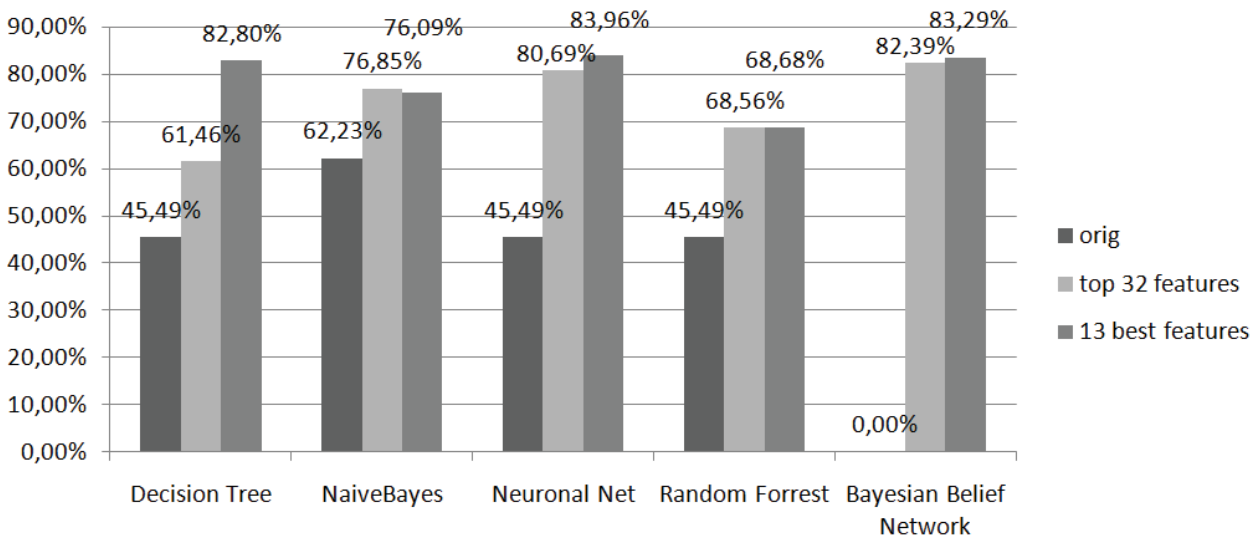

Fig. 6: Accuracy rate comparison original data set with primitive attributes variations of constructed features.

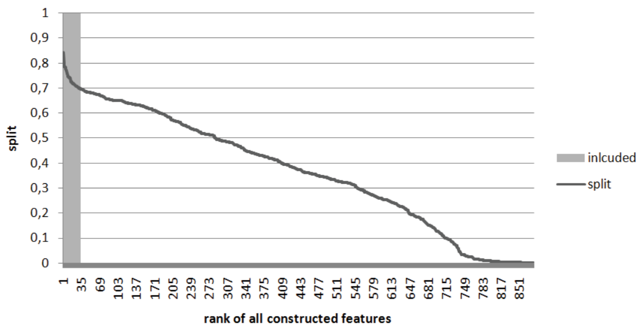

Fig. 7: All constructed features ranked by their split value.

features were ranked by their split value, as it can be seen in Fig. 7. The best feature achieve a split value of 0.843, the lowest of 0.0003. In order to keep execution times low, we chose only the top 32 constructed features from the ranked list for our second run. Fig. 6 shows the impressive improvement for the compared standard methods. Since the $s_{id}$ is unique for the constructed features set, the Bayesian Belief Networks are applicable.

However, focusing only on the split measurement for feature selection is not enough. In a second range of experiments, we only included those features, which achieved a minimum split value of 0,70 and a had a minimum support value of 0,50. A total number of 13 features met these criteria (10 operator numeric and 3 sequence distribution based features). The results for the best feature are shown in Fig. 6. It can be seen that the smaller constructed feature space is able to perform better or at least as good as the top 29 features only ranked by split. This shows that complex problems with sequential data can be simplified and solved by features construction. We can also see that for this data set, operator numeric features turned out to be the most benefiting ones. Reason for this is that there were only two string attributes in the original data set as well as the lack of a proper timeline (see also Section V-B). This means that in this data set, the pattern to distinguish between the two given labels is not that dependent on the temporal dimension than in other data sets (e.g., [1]). This also highlights the importance of the presented features selection techniques. Without them, arbitrary and useless features would have mislead the used classifiers.

## VI. Conclusion

Data pre-processing and selection are important steps in the data mining process. This can be challenging, if there is no domain expert knowledge available. The algorithm proposed in this work helps, not only to understand the patterns within the data, but also, to simplify more complex data structures (such as sequential data). This is achieved by various aggregation and combination techniques that allow to increase the feature space of a given data set and eventually, to highlight present feature interactions. The feature construction algorithm can be applied in conjunction with well known standard algorithms and boosts classification performance in a big variety of fields with similar specifications (such as the detection of credit card fraud, network intrusions, bots in computer games). Its systematic approach can also help domain experts to find previously unknown interactions among data and therefore, to get a better understanding of their domain.

## VII. Future Work

Further ways for extending the features space could be to implement more numerical features generated by logarithm, exponential function or combinations of more than two attributes. The algorithm itself could be optimized to assess the quality of a candidate feature before actually calculating it. Another development direction could be to align the constructed features in a way, that would allow to classify data without the help of one of the standard algorithms.

## REFERENCES

[1] M. Schaidnagel and F. Laux, "Feature construction for time ordered data sequences," in *Proceedings of the Sixth International Conference on Advances in Databases, Knowledge, and Data Applications*, Chamonix, April 20-24, 2014, pp. 1-6.

[2] W. Lee, "A Data mining framework for constructing features and models for intrusion detection systems," PhD thesis, Columbia University, Graduate School of Arts and Sciences, 1999.

[3] J. Han and M. Kamber, *Data mining: concepts and techniques* 2. edition pp. 48-97 second edition, San Francisco, Morgan Kaufmann, 2006.

[4] H. Liu and H. Motoda, *Feature extraction, construction and selection: a data mining perspective*, Boston, Kluwe Academic Publisher, 1998.

[5] W. Lin, M. Orgun, and W.J. Graham, "An overview of temporal data mining," in *Proceedings of the 1st Australian data mining workshop*, Canberra, Australia, 2002, pp. 83-90.

[6] L. S. Shafti and E. Pérez, "Constructive induction and genetic algorithms for learning concepts with complex interaction," in *Proceedings of The Genetic and Evolutionary Computation Conference*, Washington, June 2005, pp. 1811-1818.

[7] L. S. Shafti and E. Pérez, "Data reduction by genetic algorithms and non-algebraic feature construction: a case study," in *Proceedings of: Eighth International Conference on Hybrid Intelligent Systems*, Barcelona, September 2008, pp. 573-578.

[8] H.P. Kriegel, K. M. Borgwardt, P. Krger, A. Pryakhin, M. Schubert, and A. Zimek, "Future trends in data mining," in *Data Mining and Knowledge Discovery*, vol. 15, no. 1, Springer, 2007, pp. 87-97.

[9] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*, Berlin, Springer, 2006.

[10] K. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan,*Data mining: a knowledge discovery approach*, New York, Springer US, 2007.

[11] S. Markovitch and D. Rosenstein, "Feature generation using general constructor functions," in *Machine Learning*, vol. 49, no. 1, Kluwer Academic Publishers, 2002, pp. 59-98.

[12] N. Lesh, M. J. Zaki, and M. Ogihara, "Scalable feature mining for sequential data," in *IEEE Intell. Syst.* No. 2, 2000, pp. 48-56.

[13] R. Setiono and H. Liu, "Fragmentation problem and automated feature construction," in *Proceedings of: 4th Conference on Data Mining and Optimization (DMO)*, Langkawi, September 2012, pp. 53-58.

[14] Y. Yang, L. Cao, and L. Liu, "Time-sensitive feature mining for temporal sequence classification," in *Proceedings 11th Pacific Rim International Conference, Wellington*, December 2013, pp. 315326.

[15] G. Pagallo, "Learning DNF by decision trees," *Machine Learning*, pp. 71-99 Kluwer Academic Publishers, 1990.

[16] B. Zupan and M. Bohanec, "Feature transformation by function decomposition," in *Journal IEEE Intelligent Systems archive*. Volume 13 Issue 2, March 1998, pp. 38-43.

[17] J.R. Quinlan, "C4.5: programs for machine learning". Morgan Kaufmann, 1993.

[18] D. García, A. González, and R. Pérez, "A two-step approach of feature construction for a genetic learning algorithm," in *Proceedings of: IEEE International Conference on Fuzzy Systems*, Taipei, June 2011, pp. 1255-1262.

[19] G. Dong and J. Pei, *Sequence data mining*, New York, Springer US, 2007.

[20] E. Bautista-Thompson and R. Brito-Guevara, "Classification of data sequences by similarity analysis of recurrence plot patterns," in *Proceedings of Seventh Mexican International Conference on Artificial Intelligence*, Tuxtla Gutirrez, Mexico, 2008.

[21] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," in *ACM SIGKDD Explorations Newsletter*, No 1, 2010, pp. 40-48.

[22] D. García, Antonio González, and R. Pérez, "An iterative strategy for feature construction on a fuzzy rule-based learning algorithm," in *Proceedings of: 11th International Conference on Intelligent Systems Design and Applications*, Cordoba, November 2011, pp. 1235-1240.

[23] R. Alfred, "DARA: data summarisation with feature construction," in *Proceedings of: Second Asia International Conference on Modelling & Simulation*, Kuala Lumpur, May 2008, pp. 830-835.

[24] F. Sia and R. Alfred, "Evolutionary-based feature construction with substitution for data summarization using DARA," in *Proceedings of: fourth Conference on Data Mining and Optimization (DMO)*, Langkawi, September 2012, pp. 53-58.

[25] I. Witten and F. Eibe, *Data mining : practical machine learning tools and techniques* 2. edition, San Francisco, Morgan Kaufmann, 2005, pp. 48-97.

[26] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press, 2001.