Using Feature Construction for dimensionality reduction in Big Data scenarios to allow real time classification of sequence data

Michael Schaidnagel[1], Fritz Laux[2] and Thomas Connolly[3]

Abstract: A sequence of transactions represents a complex and multi-dimensional type of data. Feature construction can be used to reduce the data´s dimensionality to find behavioural patterns within such sequences. The patterns can be expressed using the blue prints of the constructed relevant features. These blue prints can then be used for real time classification on other sequences.

Keywords: Feature construction, real time classification, big data

1 Introduction

A rapid advance in database technology nowadays allows storing massive amounts of data. The term 'Big Data' has become one of the biggest buzzwords in the last two years. This is also reflected by the massive interest of the research community in the topic. However, little of that data is actually analysed and used effectively. Reasons for that are the increasing complexity of the stored data (i.e. data sequences) and a more structural problem: In order to handle the data flood, companies tend to separate their operational systems from the analytical information systems. Operational systems are applications that operate the customer service, e.g., the booking system of an airline, the check-out application in an online store etc. Analytical information systems are databases that store large amounts of data such as the DWH or backup servers. This way, companies such as online retailers, airlines or computer game companies can ensure that their customers are served in a timely manner by rather lightweight application systems. These systems backup their data into the DWH on a regular basis. The data is aggregated according to the cubes that have been defined to fit the business needs. This structural necessity however, is contrary to new requirements and types of analysis that data miners are challenged with. Business needs include nowadays that data mining analysis is done on a customer (i.e., individual) level in real/near time, rather than on the aggregated form of cubes. The aggregation is necessary since not every detail can be stored for a longer time span. Unfortunately, a lot of valuable information contained in data sequences about individual customers is lost during this process. This work will show how feature

[1] University of the West of Scotland, School of Computing, PA1 2BE, Scotland, B00260359@studentmail.uws.ac.uk
[2] FH Reutlingen, Fakultät Informatik, Alteburgstr 150, 72762 Reutlingen, Fritz.Laux@reutlingen-universtiy.de
[3] University of the West of Scotland, School of Computing, PA1 2BE, Scotland, Thomas.Connolly@uws.ac.uk

construction can be used to simplify complex data sequences and therefore allow real time classification in big data scenarios. The rest of the paper is structured as follows: Section 2 will give a brief introduction into the related work. Section 3 will describe the structure of sequence data. Section 4 will give a short introduction into the field of feature construction. Section 5 will presents the underlying feature construction models and describes a framework that shows how feature construction can be used for real time classification. The concluding Section 6 will highlight the contributions of this paper.

2 Related Work

Recent contributions to the field of feature construction have been made by Shafti [SP09], who presents a technique named MFE3/GA. It searches through the initial space of attribute subsets to find subsets 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. A standard C4.5 decision tree learner is then applied for the data mining process. Only nominal attributes are being processed, so that class labels and continuous attributes need to be normalized. A feature is a bit-string of length N, where each bit shows the presence or absence of one the N original attributes. Another approach to feature construction is described by Morik and Köpcke [MK04]. They create features based on term frequency (TF) and inverse document frequency (IDF) features. The timestamped data is thereby transformed into frequency features as they are used in bag-of-words representations of text. A heuristic is used in order to estimate if a transformation of given raw data into frequency features will be beneficial or not. Since timestamped data often describes a status change of the same object, TF/IDF features use a Boolean representation to denote status changes of certain attributes. However, the current feature construction techniques have been designed for tupel based data, leaving the sequence dimension aside.

The dimensionality reduction described in this work has similarities with the Principal Component Analysis, which is a widely known statistical method to transform a set of observations into a smaller number of 'principal components'.

3 Sequence data

This research work is focusing on data sets that consist of transactions. These transactions represent complex vectors that can include both data types (categorical and ordinal) and another dimension of information: the time of the transaction. Prime example for transaction sequences are sessions in an online shop. Customers can view products and put them into their shopping basket. Every action can be represented in a database as a tuple that is associated with a timestamp. Following that, sequential data that is used in this work must include the following: a sequence identifier attribute $Sid \in \{sid_1, \dots, sid_s\}$, at least two attributes of arbitrary data type $A \in \{A_1, \dots, A_m\}$, a temporal attribute T indicating the time an event happened and a binary classification $L \in (0, 1)$. Following that the minimum structure for sequential data set D that can be classified with the suggested approach must at least satisfy the below depicted schema. The attributes A_m have a valid range of $W(A_m)$ and an attribute name of A_m = "A_m".

	Sid	A_1	...	A_m	L
t_1	sid_1	$a_{11}^{sid_1}$	...	$a_{1m}^{sid_1}$	*0*
t_2	sid_1	$a_{21}^{sid_1}$	...	$a_{2m}^{sid_1}$	*0*
t_3	sid_1	$a_{31}^{sid_1}$	...	$a_{3m}^{sid_1}$	*0*
t_4	sid_2	$a_{11}^{sid_2}$	...	$a_{1m}^{sid_2}$	*1*
t_5	sid_2	$a_{21}^{sid_2}$	...	$a_{2m}^{sid_2}$	*1*
...	...	...	...	...	...
t_n	sid_s	$a_{p1}^{sid_s}$	...	$a_{pm}^{sid_s}$	...

Tab. 1: Schema of sequential data

The schema depicted in **Fehler! Verweisquelle konnte nicht gefunden werden.** shows two sequences sid_1 and sid_2. If sequential data is ordered by the Sid column, it can be seen as a series of matrices. If the T, Sid and L columns are left out, only the attributes are left. By just focusing on the attributes $\mathrm{A} \in \{\mathrm{A}_1, \dots, \mathrm{A}_\mathrm{m}\}$ for the sequence sid_1, it is possible to represent the sequence as a $\mathrm{p} \times \mathrm{m}$ matrix M, as seen in (1) below:

$$M(Sid) = [a_{pm}] = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ a_{21} & \cdots & a_{2m} \\ \vdots & \ddots & \vdots \\ a_{p1} & \cdots & a_{pm} \end{bmatrix} \tag{1}$$

Note that the notation used in this chapter is largely based on Markovitch [Ma02]. The size of the whole data set D is $n \times m$, while the matrix of a sequence contains only $p \times m$ attribute values. The matrix can furthermore be split up into vectors $\vec{v}_1, \dots, \vec{v}_m$ with p dimensions as seen in (2) below:

$$\vec{v}_1 = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{p1} \end{pmatrix}, \dots, \vec{v}_m = \begin{pmatrix} a_{1m} \\ a_{1m} \\ \vdots \\ a_{pm} \end{pmatrix} \tag{2}$$

A vector is normally defined as a sequence of p elements $(a_1, a_2, \dots, a_p)$ where p is a positive integer. The elements normally consist of real numbers $a_\pi \in \mathbb{R}^n$. However, in this research work, vectors can also consist of categorical values, which are part of the definition domain $\mathbb{P}$. $\mathbb{P}$ can consists of all countable numbers and all characters strings with a length < 250.

4 Feature Construction

Feature Construction (FC) is about the construction of new information based on the given data (i.e. attributes). There are also other terms used in the literature to denote this research area. Han [HKP12] refers to it as 'attributes construction' and Guyon [Gu06] as 'feature extraction'. Guyon is more focused on the feature selection task and uses the term feature extraction as a sort of 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. One goal of feature construction is to reduce the data dimension by removing redundant or irrelevant attributes [SP09]. This is done by constructing new features out of the given attributes to help the mining process [HKP12]. In this case the constructed feature replaces the attributes it was constructed from [SP09]. However, it is important to not discard valuable 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 by Shafti [SP09] focusing on predictive accuracy.

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 [CSP07]. Finding a good transformation function is very domain specific and also depends on the available measurements [Gu06]. 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 [LZO00].

5 Dimensionality reduction model

This section will show how transactional sequence data can be reduced in dimensionality by using feature construction. The following subsection 5.1 and 5.2 will show how to use the generated features for classification. The concluding subsection 5.3 will describe a framework that shows how the models can be implemented in real life scenarios.

5.1 Construction Models

This subsection will show the five contractions models U_1, U_2, U_3, U_4, U_5 that combine the original data attributes and aggregate the sequences in a certain way, which leads to a reduction in the horizontal as well as vertical dimension of the data. For the first construction technique U_1, each element of the vector $\vec{v} \in (\vec{v}_1, \dots, \vec{v}_m)$ of an sequence sid_s is transformed into a set $\{a_{pm}\}$. The set only contains distinct values of the corresponding attribute vector. The cardinality of the set contains information about the cardinality of occurrences in the corresponding attribute vector, which can be useful and distinctive in certain situations.

$$\begin{aligned} &U_1 \colon \mathbb{P}^{\mathrm{p}} \to \mathbb{R}^+ \\ &u_1(v_\mu) \mapsto y_\mu := |\{a_{\mu i} | a_{\mu i} \in \vec{v}_\mu\}| \leq p \end{aligned} \tag{3}$$

It is also possible that the cardinality isn´t directly visible if just one attribute is examined, due to feature interaction. In order to access this information it is possible to concatenate two categorical vectors and form a set with distinctive occurrences of the constructed pairs, as shown in (4) and (5).

$$\begin{aligned} &U_2 \colon \mathbb{P}^p \times \mathbb{P}^p \to \mathbb{R}^+ \\ &u_2(\vec{v}_\mu \circ \vec{v}_\nu) \mapsto y \end{aligned} \tag{4}$$

$$y := u_2(\vec{v}_\mu, \vec{v}_\nu) = \left|\{v_{\pi\mu} \; concat \; v_{\pi\nu} | v_{\pi\mu} \in \vec{v}_\mu, v_{\pi\nu} \in \vec{v}_\nu\}\right| \forall\, \pi = 1, \dots, p \tag{5}$$

Feature interaction between two continuous attributes can be highlighted by combining the two using arithmetic operators. In contrast to categorical attributes, a greater variety of concatenations can be produced, yielding a higher chance to find distinctive information. The corresponding constructor function calculates in essence the scalar products of each attribute vectors $\vec{v}_\mu$ and $\vec{v}_\nu$ as in formula (7).

$$\begin{aligned} &U_3 \colon \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} \\ &(\vec{v}_\mu, \vec{v}_\nu) \mapsto y := U_3(\vec{v}_\mu, \vec{v}_\nu) \end{aligned} \tag{6}$$

$$U_3(\vec{v}_\mu, \vec{v}_\nu) = |\{v_{\pi\mu} \circ v_{\pi\nu} | v_{\pi\mu} \in \vec{v}_\mu, v_{\pi\nu} \in \vec{v}_\nu\}| \forall\, \pi = 1, \dots, p \tag{7}$$

Note that the features are calculated using a ring operator '$\circ$'. The ring operator is of the basic arithmetic operators +,-,*,/. The vectors are collapsed (aggregated) by the ring operator in order to create the new feature value for the particular sequence. So the dimensionality of the original matrix is transformed from $p \times m$ to $1 \times (0.5m^2$-$0{,}5m)$, so that one hand the dimensionality is reduced vertically from p to 1, while it is increased horizontally from m to $(0.5m^2\text{-}0{,}5m)$ due to the combination of attributes. The combination isn´t strictly $(m \times m)$, since some combinations can be left out. E.g. multiplying a number x by a number y will yield the same results as multiplying y by x. The dimensional transformation shown in (6) can also be called mapping. The inner product therefore is a specific kind of mapping that maps vectors from $\mathbb{R}^p \times \mathbb{R}^p$ into $\mathbb{R}$.

The next model shows how to discover sequence-based behaviour patterns by using a weights derived from temporal axis $\vec{v}_t$. As a first step, the temporal axis vector $\vec{v}_t$ needs to be determined for every sid. For a given sid, the temporal data sequence is given by $t_1, t_2, \dots, t_z$ with $t_i \leq t_z, i < j$. Thereby, t_1 is the minimum value $t_{min} \coloneqq t_1$ and t_z is the maximum value $t_{max} \coloneqq t_z$. The emphasis can either be (a) on the past values that are considered more important $\vec{v}_{t_{max}}$ or (b) on the more recent values that are considered more important $\vec{v}_{t_{min}}$. For type (a), the maximum temporal value t_{max} of a sequence is substracted from every other t value of the sequence, see also (8).

$$\vec{v}_{t_{max}} \coloneqq \begin{pmatrix} t_{max}\text{-}t_1 \\ t_{max}\text{-}t_2 \\ \vdots \\ t_{max}\text{-}t_z \end{pmatrix} = t_{max}\text{-}\begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_z \end{pmatrix} \tag{8}$$

For type (b), the minimum temporal value of a sequence is deducted from every other t value of the sequence, as it can be seen in (9).

$$\vec{v}_{t_{min}} \coloneqq \begin{pmatrix} t_1\text{-}t_{min} \\ t_2\text{-}t_{min} \\ \vdots \\ t_z\text{-}t_{min} \end{pmatrix} = t_{min}\text{-}\begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_z \end{pmatrix} \tag{9}$$

The time axis vector is multiplied with each attribute vector of a matrix $[a_{ij}]$ of a sequence sid_s. This allows weighting the attribute´s behaviour over the course of time. So far the weights have been multiplied with the corresponding attributes. However, other arithmetic operations are possible to yield more features and potentially catch a behavioural pattern. The weight is influenced by the values of corresponding attributes as well as the density of the time vector used. Note here that the two axis aren´t forming the exact mirror opposite of each other, which will yield a higher chance to find distinctive information. The sum of these products then creates the temporal-based feature value for an attribute. The following formula (10) shows the calculations in detail:

$$\begin{aligned} &U_4\colon \mathbb{R}^p \times \mathbb{R}^p_+ \to \mathbb{R} \\ &u_4(\vec{v}_\mu, \vec{v}_t) \mapsto U_3(\vec{v}_\mu, \vec{v}_t) \forall \mu = (1,2,\dots,m), \vec{v}_t = (\vec{v}_{t_{min}}, \vec{v}_{t_{max}}) \end{aligned} \tag{10}$$

The last feature construction model is calculated as follows: the sum, standard deviation, and the variability are calculated for each numeric attribute of a sequence. Note that the dimensionality for each numeric attribute $A \in (A_1, \dots, A_m)$ is increased by the factor of three (each of the construction techniques). Following that, for each attribute vector $V \in (\vec{v}_1, \dots, \vec{v}_m)$ of a matrix $M = \left[a_{pm}\right]$ of a sequence the average (12), the variance (13) and the standard deviation (14) are calculated:

$$U_5: \mathbb{R}^p \rightarrow \mathbb{R} \tag{11}$$

$$u_5 = \frac{1}{p}(a_{1\mu} + a_{2\mu} + \cdots + a_{p\mu}) \tag{12}$$

The variance measures the dispersion of an independent variable A over its mean [CSP07]. The average value is also referred to as the mean value $E[A]$ of a discrete probability, i.e., the distribution of continuous occurrences in a vector $[\vec{v}_m]$. The variance of a vector for an attribute A can then be described as shown in (13):

$$\mathrm{var}\left(\vec{\mathrm{v}}_{\mu \mathrm{j}}\right) = \frac{1}{\mathrm{p}} \sum_{\pi=1}^{\mathrm{p}} \left(\mathrm{a}_{\pi\mu} \text{-} \mathrm{U}_5(\vec{\mathrm{v}}_\mu)\right)^2 \; \forall \, \mu = 1, \dots, m \tag{13}$$

The standard deviation is then the square root of the variance of a vector as it can be seen in (14):

$$\sigma = \sqrt{\mathrm{VAR}} \tag{14}$$

5.2 Classification model

The feature construction techniques described in previous section generate a large amount of features, which need to be assessed if they are useful for classification. Therefore, the next step is to use feature selection, which in general is another step in the KDD process. The performance of classification algorithms can deteriorate, if the wrong input is given and also the computational costs can increase tremendously. Reason for the deterioration in performance is the tendency of classifiers to overfit, if provided with misleading information. In order to avoid this, data miners created methods such as feature selection to decrease dimensionality of the data and as a result of that, increase classification performance and also the execution times. A supervised filter model is adopted in this work to find the most suitable features created (for more details refer to [Sc14]). Since the features can only be generated if training data is available, the classification model that is presented here is a supervised learning technique. The training data TD typically has the form as shown in (15):

$$TD = \{(\vec{v}_1, l_1), (\vec{v}_2, l_2), \dots, (\vec{v}_s, l_s)\} \tag{15}$$

So one label is associated with the same label value l, which is drawn from a discrete set of binary classes $L \in \{0, 1\}$. The label is based on a sequence level and not on a tuple level. This might seem to be counter intuitive, but it must be considered that the goal is to find a pattern in the behaviour within the whole sequence and not on the tuple level. So based on the training data, the objective is to find a set of relevant features rf_i that are able to capture the underlying behaviour model. These features are then used on the test data set to associate new vectors from the test data set with one of the label classes l. In terms of classification, such a model is called a classifier Φ. It is able to assign a label to the new vectors based on any pattern x that was learned from the features that were derived from training data [CSP07]. The pattern x in the proposed technique consist of a set of relevant features rf.

Each relevant feature is only able to show a part of the pattern. So in order to complete the pattern, the relevant features need to be brought together to form the complete pattern. The features are normalized in order to make them comparable. Once again, each relevant feature vector rf_i can be expressed as a vector of feature values and its corresponding label vector:

$$rf_i = \begin{bmatrix} rf_i^{sid_1} & l_1 \\ rf_i^{sid_2} & l_2 \\ rf_i^{sid_s} & l_s \end{bmatrix} \forall\, i = 1, \dots, \#rf \tag{16}$$

The relevant features are then grouped in two groups according to their tendency. The first type of features $\mathrm{rf}_{i\,nom}$ tends to 1 if normalized and will be summed up in the nominator of a fraction. The denominator, in contrary, is composed of the second type of features $\mathrm{rf}_{i\,denom}$, which tend to 0. If the quotient of the normalization expression insn′t defined, it will be discarded. The fraction formula depicted in (18), is used for calculating a signal value that can then be held against a threshold z for binary classification. The assembling of the interactive features will result in a high signal value if the sequence in question is similar to the average of all sequences of the target label.

$$\Phi: rf \rightarrow [0,1] \rightarrow \{0,1\} \tag{17}$$

$$\Phi(rf_1, \dots, rf_i) := \frac{\sum rf_{i\,nom}}{\sum rf_{i\,denom}} \begin{cases} \geq z & \mapsto 1 \\ < z & \mapsto 0 \end{cases} \tag{18}$$

5.3 Framework

The introduced feature construction techniques can be automated. The next logical step is to use these techniques to form an adaptive framework for sequence classification that is able to adapt itself to changes in the underlying patterns. It is thereby especially designed for dynamic data situations in which the status of a transaction can change in the course of time.

The framework consists of two systems: the first system is referred to as the live system. It can, for example, be the system that processed credit card transactions or activities in an online store. Status changes are processed as updates of transactions in real time. So in order to achieve real time classification, this system needs to be able to classify sequences with as little effort as possible. The decision (e.g., fraudulent or genuine, malign or benign) is based on a signal value that is calculated using the transaction history of the corresponding sequence. The calculation of the signal value is carried out by the feature assembler, which uses the formula of features as described in Subsection 5.2. The execution time of the feature assembler is very low, since it only has to fetch the corresponding transactions of the to-be-classified sequence from the database and run through it one time. During that run, all features can be constructed. It is hereby important to note that not the absolute value of features is used by the feature assembler, but the templates (or blue-prints) on how the features have been generated from a sequence. This information is provided by the feature pool, which is kept up to date by the second system.

The second system hosts the feature construction algorithm, which was introduced in Section 5.1. The features are constructed using the training data storage. After a certain period of time, e.g., a week or a month, the second system uses a sliding time window to query for new training data. So the idea is to use the abstracted knowledge that has been found in the training data, for the classification of all transactions for a certain period of time. Following that, if the underlying pattern changes in the stored training data, other features will be selected by the feature selection process and updated in the dynamic feature pool. This allows the framework to adapt to behavioural changes without any human interaction.

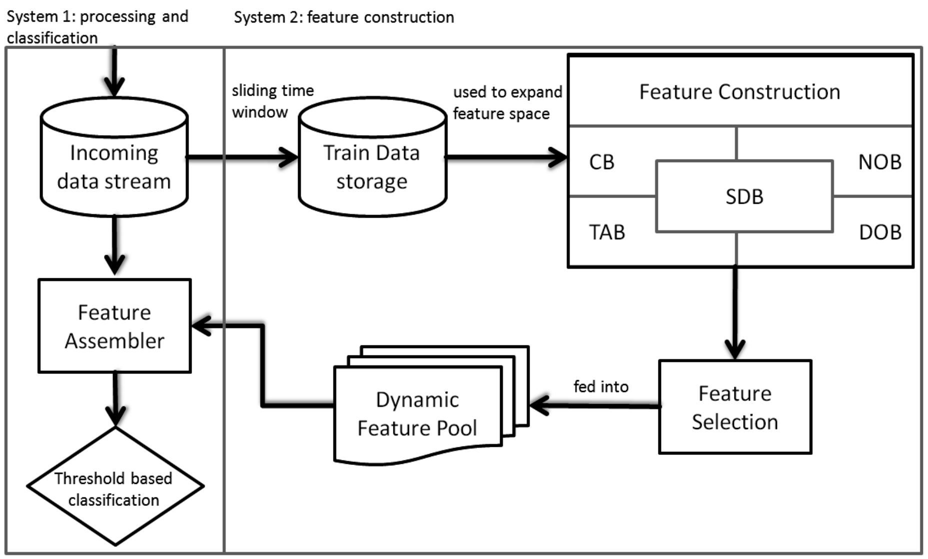

Figure. 1: Overview framework for feature construction and classification

6 Conclusion

The proposed framework and the underlying concepts enrich the field of data mining in several ways. First of all, it extends the theory of classification by extending feature construction to the sequential dimension as a preparation step to classification. The research work produced a synthesis of previous work as well as the creation of novel insights to expand the theory of classification algorithms on complex and sequential data.

The research work at hand furthermore provides new insights in the field of data preparation. The proposed automated feature construction techniques enable a systematic way to find and assess features in complex data structures and store them in a simpler, yet meaningful way. Thereby the time dimension of a sequence of actions is utilized in order to access information, which can have a significant impact on the discriminatory power of features. So far, feature construction techniques build new features 'horizontally' by combining attributes of a data set. The proposed research work is novel, since it extends the combination 'vertically' by aggregating the time axis of a sequence and create features by combining numeric values of the corresponding occurrences. The original values are aggregated during the feature construction process and this allows storing sequence based information on tuple level.

So far, the approach can be applied on scenarios with binary labels. In order to apply the approach to multi-label situations, a one-vs.-rest classification approach needs to be implemented. An example for a one-vs.-rest classification is the following: assume that a data set consists of samples that are associated to either class *a*, *b*, *c*, or *d*. In order to apply a binary classification algorithm on the problem, an iterative approach needs to be taken. First, all samples of class *a* are relabeled as e.g. 1, while all other classes are relabelled as 0. Then the algorithm is trained on the data set, yielding features that are able to distinguish class *a* from all other classes. In the next iteration, all samples of class *b* are relabelled as 1 and class *a*, *c*, and *d* are relabelled as 0. Then the training starts again to yield features that are able to distinguish *b* from all other classes and so forth. Another approach for multiclass scenarios could be to take a closer look at the signal values of the various classes. It is possible that classes have certain signal value intervals which could then be used for interval based classification. An example for that could be that a signal value between 0 and 0.3 is associated with label *a*, while the interval 0.3 to 0.6 is associated with the label *b* etc.

References

[CSP07] Cios, K.; J. Swiniarski, R. W; Pedrycz, W.; Kurgan, L. A.: The Knowledge Discovery Process: A Knowledge Discovery Approach. US: Springer US, 2007

[Gu06] Guyon, I.: Feature extraction: Foundations and applications. Berlin [u.a]: Springer, 2006 (Studies in Fuzziness and Soft Computing 207)

[HKP12] Han, J.; Kamber, M.; Pei, J.: Data mining: Concepts and techniques, third edition. 3. Aufl. Waltham, Mass: Morgan Kaufmann Publishers, 2012 (The Morgan Kaufmann series in data management systems)

[LZO00] Lesh, N.; Zaki, M.; Oglhara, M.: Scalable feature mining for sequential data. In: IEEE Intelligent Systems 15 (2000), Nr. 2, S. 48–56. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=850827 – Last accessed 2013-12-13

[Ma02] Markovitch, S.; Rosenstein, D.: Feature Generation Using General Constructor Functions. In: Machine Learning 49 (2002), Nr. 1, S. 59–98. URL http://dx.doi.org/10.1023/A:1014046307775 – Last accessed 2014-03-18

[MK04] Morik, K.; Köpcke, H.: Local Pattern Detection: Features for Learning Local Patterns in Time-Stamped Data. Heidelberg: Springer Berlin, 2004 (Lecture Notes in Artificial Intelligence 3539)

[Sc14] Schaidnagel, M.; Laux, F.: Feature Construction for Time Ordered Data Sequences. In: Proceedings of the Sixth International Conference on Advances in Databases, Knowledge, and Data Applications.

[SP09] Shafti, L. S.; Pérez, E.: Feature Construction and Feature Selection in Presence of Attribute Interactions, Bd. 5572. In: 4th International Conference, HAIS 2009, Salamanca, Spain, June 10-12, 2009. Proceedings, S. 589–596