Data Mining Applications in the Automotive Industry

Rudolf Kruse, Matthias Steinbrecher, Christian Moewes
Computational Intelligence Group, Department of Knowledge and Language Processing
Faculty of Computer Science, Otto-von-Guericke University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany

Abstract. Designing and assembling automobiles is a complex task which has to be accomplished in ever shorter cycles. However, customers have increasing desires w. r. t. reliability, durability and comfort. In order to cope with these conflicting constraints it is indispensable to employ tools that greatly simplify the analysis of data that is collected during all car lifecycle stages. We will present methods for pattern discovery tasks for the development stage, the manufacturing and planning stage as well as for maintenance and aftercare. The first approach will reinterpret a Bayesian network to induce association rules which are then visualized to find interesting patterns. The second part will use Markov networks to model the interdependencies related to the planning task when assembling a vehicle. The last part deals with finding recurring patterns in time series used for adjusting simulation parameters.

1. Introduction

The lifecycle of a modern car comprises a multitude of complex and interdependent tasks that start early during the development phase, guide and advice the production process and keep track of issues related to operating vehicles. We will present examples of data mining applications from all these three stages: development, production planning and fault analysis.[1] All contributions share the property that we use (or extract) rule patterns to explain the domain under analysis to the user. Rules (in form of association rules) are a well-understood means of representing knowledge and data dependencies. The inherent interdisciplinary character of the automobile development and manufacturing process requires models that are easily understood across application area boundaries. The understanding of patterns can be greatly enhanced by providing powerful visualization methods alongside with the analysis tools.

The next section will briefly sketch the underlying theoretical frameworks, after which we will present and discuss successfully applied fault analysis, planning and development methods, all of which have been rolled out to production sites of two large automobile manufacturers.

Section 2 provides some notations needed for the rest of the article. Section 3 discusses an approach based on graphical models to assess and reveal potential fault patterns inside vehicle data. Section 4 deals with the handling of production planning. Section 5 discovers rules from time series that are created by prototype simulations. Finally, section 6 summarizes the findings.

[1] For didactical reasons, we reverse the order.

4th International Workshop on Reliable Engineering Computing (REC 2010)
Edited by Michael Beer, Rafi L. Muhanna and Robert L. Mullen

ISBN: 978-981-08-5118-7. *Published by* Research Publishing Services.
doi:10.3850/978-981-08-5118-7_plenary-2

2. Background

We will now briefly discuss the notational underpinning that is needed to present the ideas and results from the industrial applications.

2.1. Graphical Models

As we have pointed out in the introduction, there are dependencies and independencies that have to be taken into account when reasoning in complex domains shall be successful. Graphical models are appealing since they provide a framework of modeling independencies between attributes and influence variables. The term "graphical model" is derived from an analogy between stochastic independence and node separation in graphs. Let $V = \{A_1, \ldots, A_n\}$ be a set of random variables. If the underlying probability distribution $P(V)$ satisfies some criteria (see e. g. (CGH97; Pea93)), then it is possible to capture some of the independence relations between the variables in V using a graph $G = (V, E)$, where E denotes the set of edges. The underlying idea is to decompose the joint distribution $P(V)$ into lower-dimensional marginal or conditional distributions from which the original distribution can be reconstructed with no or at least as few errors as possible (LS88; Pea88). The named independence relations allow for a simplification of these factor distributions. We claim, that every independence that can be read from a graph also holds in the corresponding joint distribution. The graph is then called an independence map (see e. g. (BSK09)).

2.1.1. *Bayesian Networks*

If we are dealing with an acyclic and directed graph structure G, the network is referred to as a Bayesian network. The decomposition described by the graph consists of a set of conditional distributions assigned to each node given its direct predecessors (parents). For each value of the attribute domains (dom), the original distribution can be reconstructed as follows:

$$\forall a_1 \in \mathrm{dom}(A_1) : \cdots \forall a_n \in \mathrm{dom}(A_n) :$$
$$P(A_1 = a_1, \ldots, A_n = a_n) = \prod_{A_i \in V} P\Big(A_i = a_i \mid \bigwedge_{(A_j, A_i) \in E} A_j = a_j\Big)$$

2.1.2. *Markov Networks*

Markov networks rely on undirected graphs where the lower-dimensional factor distributions are defined as marginal distributions on the cliques $C = \{C_1, \ldots, C_m\}$ of the graph G. The original joint distribution $P(V)$ can then be recombined as follows:

$$\forall a_1 \in \mathrm{dom}(A_1) : \cdots \forall a_n \in \mathrm{dom}(A_n) :$$
$$P(A_1 = a_1, \ldots, A_n = a_n) = \prod_{C_i \in C} \phi_{C_i}\Big(\bigwedge_{A_j \in C_i} A_j = a_j\Big)$$

For a detailed discussion on how to choose the functions ϕ_{C_i}, see e. g. (BSK09).

2.2. Association Rules

The introduction of frequent item set mining and subsequently association rule induction (AIS93; AMS+96) has created a prospering field of data mining. It is the simplicity of the underlying concept that allowed for a broad acceptance among all kinds of users no matter whether they possess a data analysis background or not. An association rule is basically an *if-then* rule. The *if*-part is called *antecedent* while the *then*-part is named the *consequent*. Both may consist of conjunctions of attribute-value pairs, however, the consequent often consists of only one pair. An example of an association rule could be

If a person is male and a smoker, his probability of having lung cancer is 10%.

This corresponds to the imagination that we pick a person at random from an underlying population (the database) and observe its properties, that is its attribute values. The above rule can then be represented in a more formal fashion as

$$\text{Gender} = \text{male} \ \wedge \ \text{Smoker} = \text{y} \ \rightarrow \ \text{Cancer} = \text{y}. \tag{1}$$

We refer to a database case as being covered by a rule if the antecedent and consequent attributes values match. For instance, a smoking man having lung cancer would be covered by the above rule. The general form of a rule has the following form:

$$A_1 = a_1 \wedge \cdots \wedge A_n = a_n \longrightarrow C = c \quad \stackrel{\text{abbr}}{=} \quad \mathbf{a} \rightarrow c$$

We will only discuss rules with one consequent attribute which will be a class variable. We thus use the notions class and consequent interchangeably.

Since not every database entry matching the antecedent also matches the consequent it is necessary to record this information. The probability that a database case matching the antecedent also matches the consequent, that is $P(c \mid \mathbf{a})$, is called the *confidence* of the rule. The above rule 1 has a confidence of 0.1. There is a multitude of other measures that quantify certain aspects of a rule (see, e. g., (YZ99)). We will briefly discuss those that are used in this paper.

The number of cases covered by the rule is referred to as the (absolute) *support* of the rule. The relative support equals $P(\mathbf{a}, c)$; it is the absolute support divided by the database size. The *recall* quantifies the fraction (or probability if you keep the above scenario of picking at random) of database cases matching the antecedent, given the consequent. In other words: What is the probability of a person being male and a smoker if this person has cancer? As a last measure (the only unbounded one) we introduce the *lift*. It represents the ratio between the confidence $P(c \mid \mathbf{a})$ and the marginal consequent probability $P(c)$: Let the marginal cancer rate be 0.01. Then, rule 1 has a lift of 10 since the confidence is ten times larger than the marginal cancer rate. We summarize the measures below:

- relative support: $\text{rel-supp}(\mathbf{a} \rightarrow c) = P(\mathbf{a}, c)$
- confidence: $\text{conf}(\mathbf{a} \rightarrow c) = P(c \mid \mathbf{a})$
- recall: $\text{recall}(\mathbf{a} \rightarrow c) = P(\mathbf{a} \mid c)$
- lift: $\text{lift}(\mathbf{a} \rightarrow c) = \dfrac{P(c \mid \mathbf{a})}{P(c)}$

3. Fault Analysis with Graphical Models

Data analysis is a vital component in strategic planning for companies that are aware of global competition, ever-shorter production cycles and increasing customer requirements. It is of paramount importance to identify meaningful patterns quickly within the collected data in order to respond to impending supply shortages or evolving problems with delivered products. We intend to enable users that not necessarily have a statistical background to assess and understand the identified patterns. This has been accomplished by devising appropriate visualization methods for patterns.

The modeling technique used for solving the outlined issues shall accommodate two main aspects: firstly, it must allow for a global view on the domain that is under analysis, i. e. the overall interconnections and interrelations between the attributes that describe a vehicle. As these are normally high-dimensional, a compact but still usable knowledge representation has to be found. Secondly, the user must be enabled to inspect any local dependency in greater details if he wishes to.

To illustrate these two claims in the realm of a vehicle manufacturer, assume that every vehicle configuration is stored in a database. Such a configuration often contains several tens to sometimes hundreds of attributes and hence dimensions. The stochastical dependencies—and more important: independencies—will be represented by a (directed) graph in which a node models an attribute amongst which the dependencies are reflected by edges. In our application this graph will be created from the database with optional preceding or subsequent expert-specified alterations. This will allow the user (e. g. an engineer or marketing analyst) to infer coarse-grained conclusions based on the potential effects between connected attributes. When it comes to a question that is narrowed down to a specific configuration fragment, the parameters attached to every node in the graph can reveal answers to quantitative questions such as "Whenever a repair report referenced transmission type X, there is a 40% chance of also having the engine type Y built into the car, which rises the faiure rate by 30%". In this case dependencies are contained in the vehicle database and are not known beforehand but are extracted to reveal possible hidden design flaws. This example calls for treatment methods that exploit the dependence structures embedded inside the application domains. We chose graphical models, more specific: Bayesian networks, to address these issues. The two abovementioned criteria map well onto the global and local components a graphical model consists of.

3.1. Data Description and Model Induction

For every car that is sold, a variety of data is collected and stored in corporate-wide databases. After every repair or check-up the respective records are updated to reflect the technical treatment. The analysis scenario discussed here is the interest of the automobile manufacturer to investigate car failures by identifying common properties that are exposed by specific subsets of cars that have a higher failure rate.

The decision was made to use Bayesian networks to model the dependence structure between these attributes to be able to reveal possible interactions of vehicle components that cause higher failure rates. The induction of a Bayesian network consists of identifying a good candidate graph that encodes the independencies in the database. The goodness of fit is estimated by an evalua-

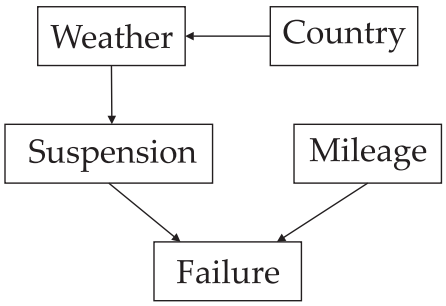

Figure 1. The qualitative component of an exemplary Bayesian network.

tion measure. Therefore, usual learning algorithms consist of two parts: a search method and the mentioned evaluation measure which may guide the search. Examples for both parts are studied in (CH92; HGC94; BK97).

Given a network structure, an expert user will gain first insights into the corresponding application domain. In Figure 1 one could identify the mileage to have a major (stochastical) impact on the failure rate and type. Of course, arriving at such a model is not always a straightforward task since the available database may lack some entries requiring the treatment of missing values. In this case possibilistic networks (BK98) may be used. However, with full information it might still be problematic to extract significant statistics since there may be value combinations that occur too scarcely. An expert can already benefit from the encoded stochastic direct and indirect dependencies in order to come up with hypotheses what attributes might be most predictive w. r. t. the failure attribute. However, the bare network structure does not reveal information about which *which* mileages have *what kind* of impact on *which* type of failure. Fortunately, this information can be retrieved easily in form of conditional probabilities from the underlying dataset, given the network structure. This becomes clear, if the sentence above is re-stated: Given a specific mileage, what is the failure probability of a randomly picked vehicle?

3.2. Model Visualization

Every attribute together with its direct parent attributes encodes a set of conditional probability distributions. For example, given a database D, the sub-network consisting of Failure, Suspension and Mileage in Figure 1 defines the following set of distributions:

$$P_D(\mathsf{Failure} \mid \mathsf{Suspension}, \mathsf{Mileage})$$

Given an attribute of interest (in most cases the class variable like Failure in the example setting) and its conditioning parents, every probability statement like

$$P(\mathsf{Failure} = \mathsf{Bearingsbroken} \mid \mathsf{Suspension} = \mathsf{Type\ X}, \mathsf{Mileage} = \mathsf{over100Kmi}) \quad = \quad p^*$$

can be considered an association rule:

If Suspension = Type 1 ∧ Mileage = over100Kmi, then there will be a bearings failure in $100 \cdot p^*\%$ of all cases.

The value p^* is then the confidence of the corresponding association rule (cf. section 2). Of course, all known evaluation measures can be applied to assess the rules. With the help of such measures one can create an intuitive visual representation according to the following steps:

- For every probabilistic entry (i. e., for every rule) of the considered conditional distribution $P(C \mid A_1, \ldots, A_m)$ a circle is generated to be placed inside a two-dimensional chart.
- The gray level (or color in the real application) of the circle corresponds to the value of attribute C.
- The circle's area corresponds to the value of some rule evaluation measure selected before displaying. For the remainder of this chapter, we choose this measure to be the support, i. e., the relative number of vehicles (or whatever instances) specified by the values of C and $A_1, \ldots, A_m$. Therefore, the area of the circle corresponds to the number of vehicles.
- In the last step these circles are positioned. Again, the value of the x- and y-coordinate are determined by two evaluation measures selected in advance. We suggest these measures to be confidence and lift. Circles above the darker horizontal line in every chart mark subsets with a lift greater than 1 and thus indicate that the failure probability is larger given the instantiation of $A_1, \ldots, A_n$ in contrast to the marginal failure probability $P(C = c)$.

With these prerequisites we can issue the user the following heuristic in order to identify suspicious subsets:

Sets of instances in the upper right hand side of the chart may be good candidates for a closer inspection.

The greater the y-coordinate (i. e. the lift value) of a rule, the stronger is the impact of the conditioning attributes' values on the class variable. Larger x-coordinates correspond to higher confidence values.

3.3. Application

This section illustrates the proposed visualization method by means of three real-world datasets that were analyzed during a cooperate research project with a automobile manufacturer. We used the K2 algorithm (CH92) to induce the network structure and visualized the class variable according to the given procedure.

Figure 2 shows the analysis result of approximately 60000 vehicles. Attributes Precipitation and Transmission had most (stochastic) impact on the Failure variable. The subset marked by the arrow was re-identified by experts as a problem already known.

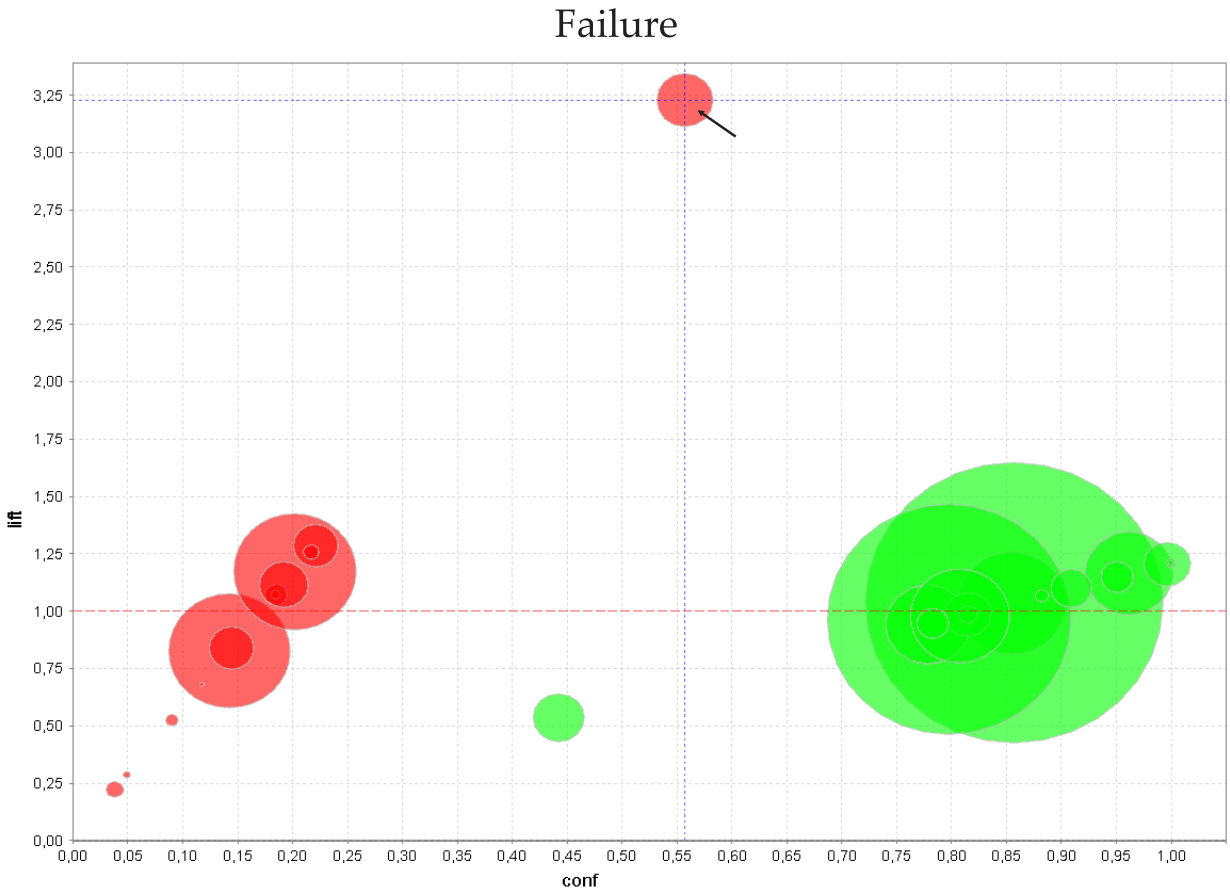

Figure 2. The subset marked by the arrow corresponds to 825 vehicles whose attributes values of Precipitation and Transmission yielded a causal relationship with the class variable.

4. Production Planning with Graphical Models

One goal of the project described here was to develop a system which plans parts demand for the production sites of the Volkswagen Group (GDM03). The market strategy is strongly customer-focused—based on adaptable designs and special emphasis on variety. Consequently, when ordering an automobile, the customer is offered several options of how each feature should be realized. The result is a very large number of possible car variants. Since the particular parts required for an automobile depend on the variant of the car, the overall parts demand can not be successfully estimated from total production numbers alone. The modeling of domains with such a large number of possible states is very complex. Therefore, decomposition techniques were applied and augmented by a set of operations on these subspaces that allow for a flexible parts demand planning and also provide a useful tool to simulate capacity usage in projected market development scenarios.

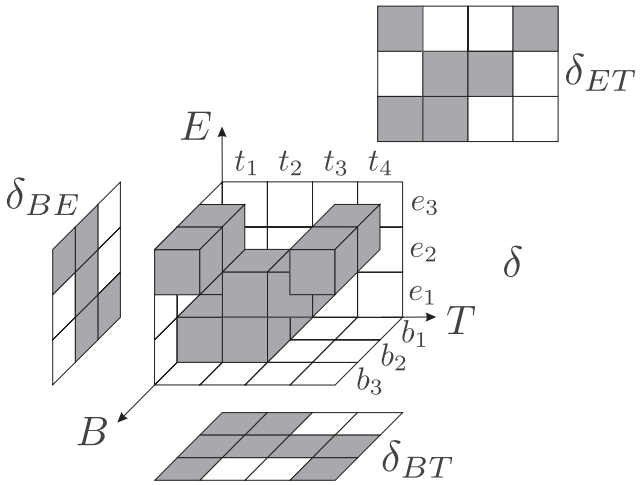

Figure 3. The 3-dimensional space $\mathrm{dom}(E) \times \mathrm{dom}(T) \times \mathrm{dom}(B)$ is thinned out by a rule set, sparing only the depicted value combinations. Further, one can reconstruct the 3-dimensional relation δ from the two projections δ_{ET} and δ_{BT}.

4.0.1. *Data Description and Model Induction*

The first step towards a feasible planning system consists of the identification of valid vehicle variants. If cars contain components that only work when combined with specific versions of other parts, changes in the predicted rates for one component may have an influence on the demand for other components. Such relations should be reflected in the design of the planning system.

A typical model of car is described by approximately 200 attributes, each consisting of at least 2, but up to 50 values. This scaffolds a space of possible car variants with a cardinality of over 10^{60}. Of course, not every combination corresponds to a valid specification. To ensure only valid combinations, restrictions are introduced in form of a rule system. Let us assume we are dealing with three variables E, T and B representing engine type, transmission type and brake type with the following respective domains:

$$\mathrm{dom}(E) = \{e_1, e_2, e_3\}, \qquad \mathrm{dom}(T) = \{t_1, t_2, t_3, t_4\}, \qquad \mathrm{dom}(B) = \{b_1, b_2, b_3\}$$

A set of rules could for example contain statements like

$$\text{If } T = t_3 \text{ then } B = b_2$$
$$\text{or}$$
$$\text{If } E = e_2 \text{ then } T \in \{t_2, t_3\}$$

A comprehensive set of rules cancels out invalid combinations and may result in our example in a relation as depicted in figure 3.

It was decided to employ a probabilistic Markov network to represent the distribution of the value combinations. Probabilities are thus interpreted in terms of estimated relative frequencies. Therefore, an appropriate decomposition has to be found. Starting from a given rule base R and a production history to estimate relative frequencies from, the graphical component is generated as follows: We start out with an undirected graph $G = (V, E)$ where two variables F_i and F_j

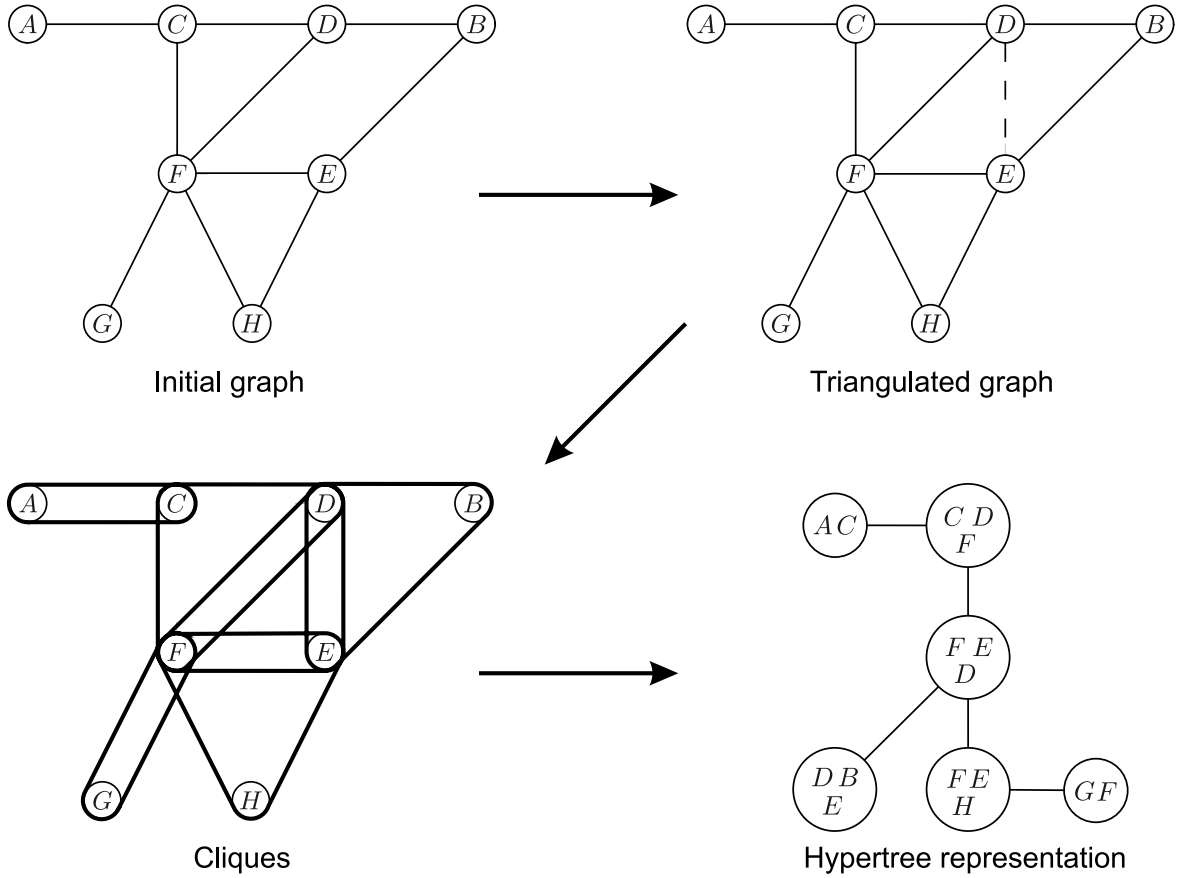

Figure 4. Transformation of the model into hypertree structure. The initial graph is derived from the rule base. For reasoning, the hypertree cliques have to have the running intersection property which basically allows for a composition of the original distribution from the clique distributions. This property can be asserted by requiring the intial graph to be triangulated.

are connected by an edge $(F_i, F_j) \in E$ if there is a rule in R that contains both variables. To make reasoning efficient, it is desirable that the graph has hypertree structure. This includes the triangulation of G, as well as the identification of its cliques. This process is depicted in figure 4. To complete the model, for every clique a joint distribution for the variables of that clique has to be estimated from the production history.

4.0.2. *Operations on the Model*

A planning model that was generated using the above method, usually does not reflect the whole potential of available knowledge. For instance, experts are often aware of differences between the production history and the particular planning interval the model is meant to be used with. Thus, a mechanism to modify the represented distribution is required. Planning operators have been developed (GK05) to efficiently handle this kind of problem, so modification of the distribution and restoration of a consistent state can be supported.

4.0.2.1. *Updating* Consider a situation where previously forbidden item combinations become valid. This can result for example from changes in the rule base. The relation in figure 3 does not allow engine type 2 to be combined with transmission type 1 because $(e_2, t_1) \notin E \times T$. If this option becomes valid probability mass has to be transferred to the respective distribution. Another scenario would be the advent of a new engine type, i. e. a change in the domain itself. Then, a multitude of new probabilities have to be assessed. A further related problem arises when subsets of cliques are altered while the information of the remaining network is retained. Both scenarios are addressed with the updating operation.

This operation marks these combinations as valid by assigning a positive near-zero probability to their respective marginals. Due to this small value, the quality of the estimation is not affected by this alteration. Now instead of using the same initialization for all new combinations, the proportion of the values is chosen in accordance to an existing combination, i. e. the probabilistic interaction structure is copied from reference item combinations.

Since updating only provides the qualitative aspect of the dependence structure, it is usually followed by the subsequent application of the revision operation, which is used to reassign probability mass to the new item combinations.

4.0.2.2. *Revision* The revision operation, while preserving the network structure, serves to modify quantitative knowledge in such a way that the revised distribution becomes consistent with the new specialized information. There is usually no unique solution to this task. However, it is desirable to retain as much of the original distribution as possible so that the principle of minimal change (Gär88) should be applied. Given that, a successful revision holds a unique result (GBKD04). As an example for a specification, experts might predict a rise of the popularity of a recently introduced navigation system and set the relative frequency of this respective item from 20% to 30%.

4.0.2.3. *Focusing* While revision and updating are essential operations for building and maintaining a distribution model, it is much more common activity to apply the model for the exploration of the represented knowledge and its implications with respect to user decisions. Typically users would want to concentrate on those aspects of the represented knowledge that fall into their domain of expertise. Moreover, when predicting parts demand from the model, one is only interested in estimated rates for particular item combinations. Such activities require a focusing operation. It is implemented by performing evidence-driven conditioning on a subset of variables and distributing the information through the network. Apart from predicting parts demand, focusing is often employed for market analyses and simulation. By analyzing which items are frequently combined by customers, experts can tailor special offers for different customer groups. To support planning of buffer capacities, it is necessary to deal with the eventuality of temporal logistic restrictions. Such events would entail changes in short-term production planning so that consumption of the concerned parts is reduced.

4.0.3. *Application*

The development of the planning system explained was initiated in 2001 by the Volkswagen Group. System design and most of the implementation is currently done by Corporate IT. The mathematical modeling, theoretical problem solving, and the development of efficient algorithms have been

entirely provided by ISC Gebhardt.[2] Since 2004 the system is being rolled out to all trademarks of the Volkswagen Group. With this software, the increasing planning quality, based on the many innovative features and the appropriateness of the chosen model of knowledge representation, as well as a considerable reduction of calculation time turned out to be essential prerequisites for advanced item planning and calculation of parts demand in the presence of structured products with an extreme number of possible variants.

5. Adjusting Monitored Experiments to Real-World Cases by Matching Labeled Time Series Motifs

Conducting field tests of complex systems to evaluate their behavior is usually expensive and time consuming. One requirement is that the designed tests should be as similar as the behavior of their pendants which are produced in series and used in the real world. Based on these experiments which quantitatively describe criteria (e. g., lifetime, errors, loadings), the quality of a system might be improved.

In order to evaluate these criteria, sensor data are recorded over long periods of time from the test and the real objects, respectively. The timely behavior of real-world systems might contain many different processes that have not been considered in the tests so far. We describe a very effective algorithm to find interesting recurrent patterns, so-called *motifs*.

One task then would be to match discovered motifs to test criteria of the time series. Every time series containing a motif thus can be labeled by at least one test criterion if the relevant test has been designed thoroughly. Hence a motif and its label can be regarded as a discovered rule's antecedent and consequent, respectively.

Having identified a set of rules in all experiments, we try to retrieve a subset of it in real-world data. As a consequence unseen time series can be assigned and compared to the given experimental criteria.

5.1. Memory-Efficient Representations

Owing to many and especially slow accesses to the original data on the hard disc, one should use an approximation of every time series that fits into the main memory of a computer and contains all essential and interesting features. There are dozens of different kinds of time series approximations, e. g., discrete Fourier transform (DFT), discrete wavelet transform (DWT), piecewise linear models (PAA), piecewise constant models (APCA), singular value decomposition (SVD), symbolic representations. The latter ones benefit by being applicable to algorithms that originate from text processing and bioinformatics, e. g., hashing, Markov models, suffix trees, etc.

In current research the symbolic representation of Lin and Keogh (LKLC03) wins out even over well-known approximations. Their symbolic aggregate approximation (SAX) transforms a univariate time series sequence into a word of defined length n over a chosen alphabet A with $|A| = a$. The SAX algorithm is rather simple but intuitive.

[2] Intelligent Systems Consulting Gebhardt

Firstly, the sequence is separated into n equal parts. Then the mean of every interval is computed as representative of all values in that interval. This method is also called piecewise aggregate approximation (PAA) (KCPM01).

After that step the essentially shorter sequence of mean values is discretized as follows. Every mean value of the PAA sequence is assigned to one of the a letters such that the occurrence of every letter in the sequence is equally probable. This is achieved by assuming that the PAA sequence's range of values is normally distributed. Furthermore, this distribution is split up into parts such that all parts share the area under the Gaussian curve. This assumption can be made due to the following fact. Long time series may not be normally distributed, but their short sequences certainly are to a high degree (LKLC03).

While other symbolic representations generate a word from time series data as well, SAX is yet one of a kind compared with them. It does not only compress the sequence. SAX also enables us to measure a distance $d^*(Q, C)$ between two SAX words which is a lower bound of the Euclidean distance between the original sequences Q and C, formally

$$d^*(Q, C) \leq d(Q, C).$$

For the rest of the paper we assume that the similarity is determined by the Euclidean distance. So, a lower bound means that if two SAX words are dissimilar, then their original sequences are dissimilar as well. Consequently, algorithms that are based on SAX produce identical results compared to algorithms that work with the original data. Merely similar SAX words should be compared in the Euclidean space again. Fortunately, those accesses to the original data are only very rare since most of the comparisons are based on dissimilar sequences.

Having a memory-efficient representation we can concentrate ourselves on finding similar sequences efficiently. In the following we proceed from the assumption that every time series is approximated by SAX since the next algorithms are based on hashing.

5.2. Motif Discovery in Time Series

If we are able to find recurrent sequences that are similar to each other, then problems such as clustering or classification of time series are much easier to solve. These similar sequences are called *motifs* due to the vocabulary that is used in bioinformatics. This originates from the fact that in this domain, motifs correspond to recurrent strings (usually from a DNA).

In the article from Chiu et al. (CKL03) SAX is associated with motif discovery in univariate time series for the first time. In order to find all motifs of a time series of length l, it is separated by a sliding window with certain width w into $(l - w + 1)$ sequences. Every sequence is transformed into a SAX word and saved into a $(l - w + 1) \times n$ matrix which we call SAX matrix.

The positions of possible motifs are then guessed using the random projection algorithm proposed by Buhler and Tompa (BT02). Actually, the positions are found by pairwise comparisons of the SAX words. So, for each of those $(l - w + 1)^2$ comparisons, we firstly reserve one entry in a collision matrix $\mathbf{M}$ which can be implemented efficiently by a hash table. In the beginning, let every entry $\mathbf{M}(i, j)$ be zero for $1 \leq i, j \leq l - w + 1$.

Although usually $n << w$, it is not preferable to compare every single character of the saved SAX words in the matrix with each other. Buhler and Tompa rather had the idea that there exist

so-called *don't care symbols* of which we do not know where they might be in the words. These symbols would correspond to, e. g., a noisy motif, a dilation/contraction of a temporal sequence.

Accordingly the SAX matrix is projected down to $1 \leq k < n$ randomly chosen columns. Afterwards all rows of the projected matrix are compared with each other. If two projected SAX words in the rows i and j are equal, then the value in $\mathbf{M}(i, j)$ is incremented by one.

The projection is repeated t times since one can assume that some of the hidden motifs will share one entry in $\mathbf{M}$ after t iterations. Additionally, it is improbable that many random sequences will collide together with an already found motif. Therefore they would have to be identical to this motif in all k positions.

Since the algorithm cannot know if a collision entry in $\mathbf{M}$ is a motif or not, the user must specify a threshold $1 \leq s \leq k$. All $\mathbf{M}(i, j) \geq s$ thus would be motif candidates. Remembering that we deal with temporal and not DNA sequences, the problem of motif discovery becomes harder as we find similar occurrences of the i-th sequence in its direct neighborhood. Those sequences which are named *trivial matches* (CKL03) are heuristically removed from the set of potential motifs at the end of the discovery.

Although comparatively, many parameters have to be determined, i. e., n, a, w, k, t, s, random projection is robust against slight changes of the SAX parameters n and a as well as the projection size k (CKL03). Also the number of projections t can be set large enough in order to create some collisions. However, there are two questions left: How many and in particularly what kind of motifs do we have to find?

If we set w and s too large on the one hand, then we may not find lots of "short" motifs. On the other hand, we will get completely different results if we set w and s too small. Then we will probably find many random consensuses that do not correspond to any real motif. Therefore, the choice of these two parameters should be made carefully. Expert's knowledge may help in such a situation.

As a side remark, we would like to mention that Yankov et al. (YKM$^+$07) extended time series random projection to a non-Euclidean distance measure, i. e., *uniform scaling*. With this method one can find motifs that are not exactly w time stamps long. This approach is indeed limited such that w must be chosen based on the respective application.

5.2.1. *Subdimensional Motifs*

The random projection to find time series motifs (CKL03) was originally only designed for one-dimensional datasets. If we deal with multivariate time series, then there exist several ways how to tackle this problem.

The simplest idea is to map the p dimensions down onto one and then use random projection. For instance, Tanaka et al. (TIU05) have transformed the input dimensions by means of principal component analysis (PCA) into solely the first principal component. Finally, the approach from Chiu et al. (CKL03) could be applied to the new univariate time series.

A first approach of Minnen et al. (MSEI07) is founded on the idea that p dimensions also generate p SAX words. These SAX words are then concatenated and treated like a SAX representation of a long univariate time series. As a consequence, the method from Chiu et al. (CKL03) can be applied in this case as well.

Though notice that both approaches can only discover motifs that span all dimensions. This will be problematic in particular if we *a priori* do not know in which of the dimensions we can observe any motif. In practice it can also happen that a time series motif's attributes can differ quite from another one's attributes. Such multivariate time series motifs that do not span all dimensions are called *subdimensional*.

Formally, we denote a multivariate sequence as $w \times p$ matrix which stores w real values for each of the p attributes. We define the distance d_{mult} of two multivariate sequences $\mathbf{Q} = [Q_1, \ldots, Q_p]$ and $\mathbf{C} = [C_1, \ldots, C_p]$ by the Euclidean norm

$$d_{\text{mult}}(\mathbf{Q}, \mathbf{C}) = \|\mathbf{d}\|_2 = \sqrt{\sum_{j=1}^{p} |d_j|}$$

whereas $\mathbf{d} = (d_1, \ldots, d_p)$ and $d_j \equiv d(Q_j, C_j)$ corresponds to the Euclidean distance between Q_j and C_j for $1 \leq j \leq p$.

According to our literature research, there is so far only one approach that tries to discover subdimensional motifs. Minnen et al. (MIES07) improve their original idea to concatenate the SAX words of every dimension. They increment the collision matrix $\mathbf{M}$ *per attribute* at the appropriate entry for every projected SAX word that matches another one.

Afterwards all elements of $\mathbf{M}$ that are greater than s are picked out and must be examined further. Note that although we have two positions for each pair of sequences, nonetheless we do not know its relevant dimensions. Furthermore, there is not any assignment of the pairs of sequences to the potential motifs yet.

Before we can perform this assignment, we have to extract the subdimensions of the sequences by means of the following naïve idea. For every pair of sequences we sort all distances $d_1, \ldots, d_p$ in an ascending order. Then the distance is accumulated in that order for every single dimension until a certain threshold $r_{\max}$ is exceeded. The attributes of the smallest distances thus correspond to the pair of sequences' relevant subdimensions.

These heuristics can be also improved by not regarding attributes having smallest distances, but using only probably relevant attributes to compute the distance (MIES07). Therefore one estimates the empirical frequency distribution $P(d_j)$ over the distances between some non-trivial matches for every dimension $1 \leq j \leq p$ by random sampling. Later on the distances $d_1^*, \ldots, d_p^*$ are computed for every entry $\mathbf{M}(i, j) \geq s$. If the value of the cumulative distribution function $P(d_j \leq d_j^*)$ is smaller than the dimension relevance r_{rel} which is specified by the user, then the j-th dimension be relevant.

Determined all pairs of subdimensional sequences, the trivial matches have to be eliminated as it was done in the univariate case of motif discovery. With this idea (MIES07), motifs do not need to span all dimensions. This would an asset compared to (TIU05; MSEI07) when the set of attributes does contain, e. g., very noisy signals, uninformative dimensions.

Disadvantages of this method for subdimensional motif discovery are the threshold parameter $r_{\max}$ and r_{rel}, respectively. Both extremely depend on the sequence length w. So, if domain knowledge is present, then it is suitable to use $r_{\max}$ as threshold. Otherwise one must estimate the distribution $P(d)$ and handle with r_{rel}.

5.3. Labeling Discovered Motifs

Having identified a set of subdimensional motifs, we merely found multivariate time series sequences of certain length w that recur at least twice. Note that we can find random motifs accidentally as well. Thus it is probable that a motif which recurs only twice might not be what we are looking for.

Yet, motifs that recur more often should be labeled meaningfully from mainly experts who designed the experiments. They usually possess the necessary knowledge to interpret both simple and complex curve progressions. This labeling can be done, e. g., by means of the test criteria.

If there is no expert's knowledge available, then one can fall back on methods from fuzzy set theory (FST) (DP00). In FST one tries to model imprecise, vague or even uncertain concepts, e. g., sensor measurements, such that the human being obtains a better understanding of these concepts.

For instances, every attribute can be regarded as linguistic variable (Zad75). In doing so, the attribute's range of values is separated into a so-called fuzzy partition. Every partition is described by a fuzzy set A. Thus every value x can be assigned to a membership degree $\mu_A(x) \in [0, 1]$ of the fuzzy set A.

We consider the measured velocity v as an illustrating example. The velocity can be described by some linguistic terms, e. g., *fast*, *medium*, *slow*. Every expression corresponds to a fuzzy partition which then again is described by a fuzzy set, i. e., A_{fast}, A_{medium}, A_{slow}.

If we want to assign a discovered motif to a linguistic term, for example we can compute the mean $\bar{v}$ of all velocity values in the respective sequence. The linguistic term with the highest of the three membership degrees $\mu_{A_{\text{fast}}}(\bar{v})$, $\mu_{A_{\text{medium}}}(\bar{v})$ and $\mu_{A_{\text{slow}}}(\bar{v})$ is then labeled to the motif.

If the experiments are designed thoroughly (i. e., they do not contain contradictory linguistic terms), then it is assumed that a time series which contains the labeled motif can be labeled in the same way. If this is not the case, we can firstly compute the relative frequencies of labeled motifs in a time series, and secondly assign several labels to this time series to a certain degree.

Every labeled motif and its linguistic term can thus represent the antecedent and the consequent of a rule, respectively. We can further hope that such a consequent corresponds to a test criterion. From the monitored experiments we finally obtain a set of rules which can be interpreted in terms of natural language by more or less great efforts.

5.4. Matching Labeled Motifs

So far we solely considered the data coming from the field tests. The system that needs to be tested may behave completely different in a real-world environment, e. g., when it is utilized by an end user. In this situation we face the problem that systems under real-world loads might not follow any designed schedule model.

Usually the only thing what remains to evaluate these systems is monitored sensor data that hopefully contains motifs similar to the ones from the experiments. These real-world data is foremost approximated memory-efficiently (see Section 5.1) before we try to find motifs in the data (cf. Section 5.2). Now we can try to label the newly discovered motifs with similar linguistic terms by means of the already labeled motifs from the field tests. In machine learning, this would correspond to classification that is based on unsupervised learning.

Remember that it is very important to choose an adequate distance measure in order to compare two motifs. For example Lin und Keogh (LKLC03) have developed not only SAX but the so-called MINDIST function which computes the distance between two SAX words. It is preferable to use this function since the sequences are stored as SAX words anyway. Of course, other distance measures could be used as well.

No matter which measures we choose, eventually every real-world time series can be matched with previously unknown criteria. Taking everything into account, we can state that a classification into different criteria is thus a trivial consequence. Nevertheless, we have to consider that this classification should be carried out rather fuzzy than crisp. Accordingly, the usage of fuzzy clustering methods (HKKR99) seems to be desirable.

5.5. Adjusting the Experiments

Having finally discovered all motifs of the real-world data and labeled them to the already existing ones, experts should have a closer look at the results of the matching. The goal should be to adjust the original experiments such that they will resemble the time series more than before.

In total, three different possibilities have to be distinguished. If an unseen motif (coming from any real-world case) could be matched easily with a motif from a field test, then we can assume that we found some important feature of the system behavior. At any rate, such characteristics should be kept in all experiments in the next generation of system tests.

Experts would probably react differently in the case that a motif is exclusively discovered in field tests and not in real-world case. Such a feature should most likely be removed from the experiments after expert opinion. It is clear that this type of motif does not matter at all.

If there are in turn motifs in unseen time series that do manifest themselves in any trial, then experts have to adjust at least one trial. After all, this motif seems to be a recurrent feature of the system which occurred either never or not often enough in the field tests.

When all motifs are examined and the test design is improved, the next generation of experiments can be performed. The gained knowledge about, e. g., loading, service live, which results from the tests should consequently be more consistent with the serial product used in the reality. Finally, these experiences can possibly provoke enhancements of the systems.

6. Summary

Data mining tools provide a valuable tool to manage and process overwhelmingly large information flows. In this article we gave an overview on three such approaches that stemmed from the automobile manufacturing setting. Firstly, we presented an approach to find fault patterns that may lead to an earlier identification of production problems. Secondly, we gave an overview on the design of a production planning and prediction system and discussed necessary operations for model alteration. Finally, we dealt with the question how field tests of systems that are produced in series might be adjusted to real circumstances.

References

Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining Association Rules between Sets of Items in Large Databases. In Peter Buneman and Sushil Jajodia, editors, *Proc. ACM SIGMOD Int. Conf. on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

Christian Borgelt and Rudolf Kruse. Some experimental results on learning probabilistic and possibilistic networks with different evaluation measures. In *1st International Joint Conference on Qualitative and Quantitative Practical Reasoning (ECSQARU/FAPR'97)*, pages 71–85, Bad Honnef, Germany, 1997.

Christian Borgelt and Rudolf Kruse. Probabilistic and possibilistic networks and how to learn them from data. In O. Kaynak, L. Zadeh, B. Turksen, and I. Rudas, editors, *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*, NATO ASI Series F, pages 403–426. Springer-Verlag, New York, NY, USA, 1998.

Christian Borgelt, Matthias Steinbrecher, and Rudolf Kruse. *Graphical Models — Representations for Learning, Reasoning and Data Mining*. John Wiley & Sons, United Kingdom, 2nd edition, 2009.

Jeremy Buhler and Martin Tompa. Finding motifs using random projection. *Journal of Computational Biology*, 9(2):225–242, 2002.

Enrique Castillo, José Manuel Gutiérrez, and Ali S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer Verlag, 1997.

Gregory F. Cooper and Edward Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9:309–347, 1992.

Bill Yuan-chi Chiu, Eamonn J. Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs. In Lise Getoor, Ted E. Senator, Pedro Domingos, and Christos Faloutsos, editors, *KDD*, pages 493–498. ACM, 2003.

Didier Dubois and Henry Prade, editors. *Fundamentals of Fuzzy Sets*. Kluwer Academic Publishers, Boston, MA, USA, 2000.

P. Gärdenfors. *Knowledge in the Flux—Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge, 1988.

Jrg Gebhardt, Christian Borgelt, Rudolf Kruse, and Heinz Detmer. Knowledge Revision in Markov Networks. *Journal on Mathware and Soft Computing, Special Issue "From Modelling to Knowledge Extraction"*, XI(2–3):93–107, 2004.

J. Gebhardt, H. Detmer, and A. L. Madsen. Predicting Parts Demand in the Automotive Industry — An Application of Probabilistic Graphical Models. In *Proc. International Joint Conference on Uncertainty in Artificial Intelligence (UAI 2003), Bayesian Modelling Applications Workshop, Acapulco, Mexico, 4.-7. August 2003*, 2003.

Jrg Gebhardt and Rudolf Kruse. Knowledge-Based Operations for Graphical Models in Planning. In Lluis Godo, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, LNAI 3571*, pages 3–14. Springer, Berlin, 2005.

David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. Technical Report MSR-TR-94-09, Microsoft Research, Advanced Technology Division, Redmond, WA, 1994. Revised February 1995.

Frank Höppner, Frank Klawonn, Rudolf Kruse, and Thomas Runkler. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. John Wiley & Sons Ltd, New York, NY, USA, 1999.

Eamonn J. Keogh, Kaushik Chakrabarti, Michael J. Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, and Bill Yuan-chi Chiu. A symbolic representation of time series, with implications for streaming algorithms. In Mohammed Javeed Zaki and Charu C. Aggarwal, editors, *DMKD*, pages 2–11. ACM, 2003.

S. L. Lauritzen and D. J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society, Series B*, 2(50):157–224, 1988.

David Minnen, Charles Lee Isbell, Jr, Irfan A. Essa, and Thad Starner. Detecting subdimensional motifs: An efficient algorithm for generalized multivariate pattern discovery. In *ICDM*, pages 601–606. IEEE Computer Society, 2007.

David Minnen, Thad Starner, Irfan A. Essa, and Charles Lee Isbell, Jr. Improving activity discovery with automatic neighborhood estimation. In Manuela M. Veloso, editor, *IJCAI*, pages 2814–2819, 2007.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

J. Pearl. Aspects of Graphical Models Connected with Causality. In *49th Session of the International Statistics Institute*, 1993.

Yoshiki Tanaka, Kazuhisa Iwamoto, and Kuniaki Uehara. Discovery of time-series motif from multi-dimensional data based on mdl principle. *Machine Learning*, 58(2-3):269–300, 2005.

Dragomir Yankov, Eamonn J. Keogh, Jose Medina, Bill Yuan-chi Chiu, and Victor B. Zordan. Detecting time series motifs under uniform scaling. In Pavel Berkhin, Rich Caruana, and Xindong Wu, editors, *KDD*, pages 844–853. ACM, 2007.

Y. Y. Yao and Ning Zhong. An Analysis of Quantitative Measures Associated with Rules. In *Methodologies for Knowledge Discovery and Data Mining*. Springer-Verlag Berlin, 1999.

Lotfi A. Zadeh. The concept of a linguistic variable and its applications to approximate reasoning–I. *Information Sciences*, 8(3):199–249, 1975.