The Representation Race – Preprocessing for Handling Time Phenomena

Katharina Morik

Univ. Dortmund, Computer Science VIII,
D-44221 Dortmund, Germany
morik@ls8.cs.uni-dortmund.de
http://www-ai.cs.uni-dortmund.de

Abstract. Designing the representation languages for the input,L_E, and output, L_H, of a learning algorithm is the hardest task within machine learning applications. This paper emphasizes the importance of constructing an appropriate representation L_E for knowledge discovery applications using the example of time related phenomena. Given the same raw data – most frequently a database with time-stamped data – rather different representations have to be produced for the learning methods that handle time. In this paper, a set of learning tasks dealing with time is given together with the input required by learning methods which solve the tasks. Transformations from raw data to the desired representation are illustrated by three case studies.

1 Introduction

Designing the representation languages for the input and output of a learning algorithm is the hardest task within machine learning applications. The "no free lunch theorem" actually implies that if a hard learning task becomes easy because of choosing appropriate representations, the choice of or the transformation into the appropriate representation must be hard [38]. The importance of L_H, the representation of the output of learning, is well acknowledged. Finding the hypothesis space with most easily learnable concepts, which contains the solution, has been supported by systems with declarative language bias [18], [11], [7] or representation adjustment capabilities [35],[39]. It is also the key idea of *structural* risk minimization, where the trade-off between complexity and accuracy of a hypothesis guides the learning process [37].

The importance of L_E, the representation of the input of learning, has received some attention only recently. Transforming the given representation of observations into a well-suited language L_E may ease learning such that a simple and efficient learning algorithm can solve the learning problem. For instance, first order logic examples and hypothesis space are transformed into propositional logic in order to apply attribute-value learning algorithms [23], [22]. Of course (and in accordance with the "no free lunch theorem"), the transformed set of examples might become exponentially larger than the given one. Only if some restrictions can be applied, the transformation plus the transformed learning problem are indeed easier than the original learning problem on the original representation. The central issue is to find appropriate restrictions and corresponding transformations for a given task [19].

The problem of designing L_E is not limited to the representation formalism but includes the selection or construction of appropriate features within a formalism [24]. The problem has become particularly urgent, since knowledge discovery confronts machine learning with databases that have been acquired and designed for processes different from learning. Given mature learning algorithms and the knowledge of their properties, the challenge is now to develop transformations from raw data L_E to suitable $L_{E'}$. The transformation can be a learning step itself so that L_{E_1} delivers $L_{H_1} = L_{E_2}$, or it can be another aggregation or inferential step. In general, we consider a series of transformations from the given raw data L_{E_1} to the input of the data mining step, L_{E_n}[1]. The technical term of "preprocessing" seems euphemistic when considering the effort spent on this transformation sequence in comparison to the effort spent on the data mining step. Rather we might view the exploration and design of tranformations a *representation race* where the winner leads to the most efficient and accurate learning of the interesting concept, rules, or subgroups. The new European project MiningMart aims at supporting end-users in winning the representation race.

This paper emphasizes the importance of transforming given data into a form appropriate for (further) learning. The MiningMart approach to supporting a user in this difficult task is illustrated by learning tasks which refer to time phenomena. First, the project is briefly described. Since it has just begun, only the main idea and the goals are reported. Second, time phenomena are discussed. Handling time is an excellent example of how data sets can be transformed in diverse ways according to diverse learning tasks and algorithms that solve them. Different views of time phenomena are elaborated and an overview of existing methods is given. Third, preprocessing operators for handling time phenomena are discussed on the basis of three case studies.

2 The MiningMart Approach

The MiningMart will be a system supporting knowledge discovery in databases. A set of transformation tools/operators will be developed in order to construct appropriate representations $L_{E'}$. Machine learning operators are not restricted to the data mining step within knowledge discovery. Instead, they are seen as preprocessing operators that summarize, discretize, and enhance given data. This view offers a variety of learning tasks that are not as well investigated as is learning classifiers. For instance, an important task is to acquire events and their duration (i.e. a time interval) on the basis of time series (i.e. measurements at time points). The tools improve the quality of data with respect to redundancy and noise, they assist the user in selecting appropriate samples, in discretizing

[1] This relates the issue of preprocessing closely to multistrategy learning [26].

numeric data and provide means for the reduction of the dimensionality of data for further processing. Making data transformations available includes the development of an SQL query generator for given data transformations and the execution of SQL queries for querying the database.

The main problem is, that nobody has yet been able to identify reliable rules predicting when one algorithm should be superior to others. Beginning with the MLT-CONSULTANT [34] there was the idea of having a knowledge-based system support the selection of a machine learning method for an application. The MLT-CONSULTANT succeeded in differentiating the nine MLT learning methods with respect to specific syntactic properties of the input and output languages of the methods. However, there was little success in describing and differentiating the methods on an application level that went beyond the well known classification of machine learning systems into classification learning, rule learning, and clustering. Also, the European STATLOG-Project [27], which systematically applied classification learning systems to various domains, did not succeed in establishing criteria for the selection of the best classification learning system. It was concluded that some systems have generally acceptable performance. In order to select the best system for a certain purpose, they must each be applied to the task and the best selected through a test-method such as cross-validation. Theusinger and Lindner [36] are in the process of re-applying this idea of searching for statistical dataset characteristics necessary for the successful applications of tools. An even more demanding approach was started by Engels [13]. This approach not only attempts to support the selection of data mining tools, but to build a knowledge-based process planning support for the entire knowledge discovery process. To date this work has not led to a usable system [14]. The European project METAL now aims at learning how to combine learning algorithms and datasets [8]. At least until today, there is not enough knowledge available in order to propose the correct combination of preprocessing operations for a given dataset and task.

The other extreme of the top-down knowledge-based approach to finding appropriate transformation sequences is the bottom-up exploration of the space of preprocessing chains. Ideally, the system would evaluate all possible transformations in parallel, and propose the most successful sequence of preprocessing steps to the user. This is, however, computationally infeasible. Therefore, the MININGMART follows a third way. It allows each user to store entire chains of preprocessing and analysis steps for later re-use in a case-base (for example, a case of preprocessing for mailing-actions, or a case of preprocessing for business reports). Cases are represented in terms of meta-data about operators and data, are presented to the users in business terms, and are made operational by SQL query generators and learning tools. The case-base of preprocessing and analysis tasks will not only assist the inexperienced user through the exploitation of experienced guidance from past successful applications, but will also allow any user to improve his or her skill for future discovery tasks by learning from the best-practice discovery cases.

3 Handling Time Phenomena

Most data contain time information in one way or another. Think, for instance, of a database storing warranty cases. Among data about the sold item including its production date, there would be data about the sale including the selling date, data about the warranty case together with the date of the claim, the expiration time of warranty, and the payment. Time stamps are natural attributes to all objects described in the database. Depending on the learning task, the same raw data are transformed into rather different example sets. Some of these simply ignore the time stamps, but others take particular care of time phenomena. In this section, first an overall view of time phenomena is presented. This is a necessary step towards a meta-level description of learning tasks related with time. Algorithms that solve one such task are briefly presented in the following subsections. This section concludes wih a list of $L_{E'}$ required by the learning methods.

3.1 Structuring time phenomena

For the overall view, we may structure time phenomena by two aspects, linear precedence and immediate dominance. These terms have been defined in natural language theory [15]. *Linear precedence* refers to the ordering of elements in a sequence. It is the relation between elements occuring along the time axis, horizontally depicted in Figure 1. Most statistical approaches are restricted to this aspect of time. *Immediate dominance* refers to categories of the time-dependent elements. Categories summarize observations to events of increasingly abstract levels[2]. The linear precedence relation between most abstract categories is propagated to the lowest level of interest, the actually observable actions or events. Sequencing rules often refer to categories (events) instead of their elements.

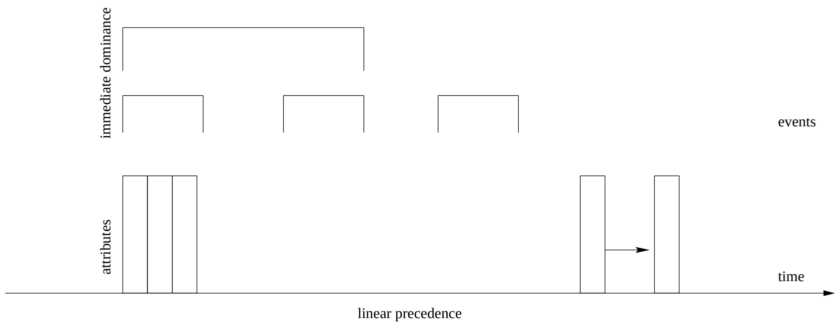

Fig. 1. The overall view of time phenomena

Learning tasks concerning linear precedence are:

[2] Mannila and Toivonen name the basic observations *events* and the higher level categories *episodes*.

Prediction: Given a sequence of elements until time point t_i, predict the element that will occur at time point t_{i+n}. We call n the horizon.

Characterization: Characterize a time ordered sequence of elements by its trend (i.e. the elements are increasingly or decreasingly ordered over time), a seasonal increasing or decreasing peak, or a cyclic ordering of elements. The cyclic ordering can be described by a function (e.g., sinus, cosinus, wavelet).

Time regions: Given time gaps between occurences of elements, predict a time interval in which an element is to be expected.

Level changes: Detect time points in a sequence of elements, where the elements are no longer homogenous according to some measure.

Clustering: Given subsequences in a sequence of events find clusters of similar sequences.

Note, that methods about linear precedence can be used to solve the problem of forming (basic) categories. It is evident, that finding trends, seasons, cycles, level changes, and clusters can be used to discretize time series. Hence, these learning tasks can be considered as preprocessing for the learning tasks concerning immediate dominance. They are valuable tasks in their own right, though.

Learning tasks concerning immediate dominance are:

Frequent Sequences: Given sequences of events, learn the precedence relation between sets of events. The sets of events in precedence relation have also been called *episodes*.

Non-determinate sequence prediction: Given a sequence of observations and the background knowledge about characterizations and categories of the basic observations, learn a set of rules that is capable of producing legal sequences.

Relations: Given events and their duration (i.e. a time interval), learn sequences of events in terms of relations between time intervals. Time relations are the ones defined in Allen's time calculus [4, 5]: overlap, inclusion, (direct) precedence, ...

Higher-level categories: Given events and their duration together with a classification in terms of a category c of the next higher level, learn the definition of c.

Non-determinate sequence prediction has been solved by [25] and has currently received attention in the context of biochemical analyses [31]. It is also the task that has to be solved for language learning. Since the datasets for sequence prediction do not include any explicit time stamp, we do exclude this very interesting issue here.

A more detailed structure of time phenomena distinguishes between handling abstract and actual time. Consider, for instance, the action of sweetening tea. This category summarizes the actions of putting sugar into the tea and stiring. These categories, in turn, can be instantiated by various alternative observable actions (e.g., using a spoon or pouring the sugar into the cup). It is not at all important, how long after putting the sugar in, one has to start stiring. Nor is the actual time in seconds interesting for the duration of stiring – stiring is performed as long as the sugar is not yet dissolved. This illustrates abstract time. Consider, in contrast, the duration of drawing the tea. Here, the actual time of 3 minutes is important. In principle, all the tasks listed above could be solved with respect to abstract or to actual time. However, learning tasks concerning linear precedence are typically solved using actual time.

An important choice when describing actual time is the scale. We may use seconds, minutes, days, months or even milleniums. Moreover, even for the same granularity, we may choose different scales of referene. For instance, measurements of vital signs in intensive care units are recorded on a minute to minute basis. Counting the minutes does not start at midnight (day time), but when the patient is connected to the monitoring machines (duration of stay). Transforming the data from one scale to the other allows to discover different regularities. The morning visit, for instance, explains why the therapy is adjusted in a time interval where the patient's state it not worse than, say, at 2 o'clock in the night. Other therapeutical interventions can better be explained using the scale referring to the duration of stay. Hence, the description of L_E should indicate the scale.

3.2 Statistical Approaches

Statistical approaches view time series as observing a process where a measurement depends on previous measurements[3]. In principle, the time axis is structured into three areas: the relevant past, the current observations and the observation to be predicted. Diverse functions are chosen to compute a value for the measurements of the relevant past: the average (in simple moving average procedures), the weighted average, where more recent measurements are multiplied by a higher weight than the ones that occured longer ago (weighted moving average), or the weights are such that weights for the relevant past and the weight for the current observation sum up to 1 (exponential moving average), smoothing algorithms use the median for values of the relevant past. Another algorithm uses the gradient [30]. Autocorrelation procedures (ARMA) consider whether past values and current value show the same ($r = 1$) or opposite direction ($r = -1$) and possibly use r^2. Choices regarding moving average models refer to noise models, the number of observations in the window (lag) of the relevant past, and whether more than one current observation is considered.

Filtering approaches consider the function over time and filter out the peaks (high pass) or the slow move (low pass). Possibly, Fourier analysis is applied decomposing the original curve into a set of sinus and cosinus curves. This is, of course, only possible, if measurements are not received on-line, but the curve is given in total.

Multivariate time series analysis is capable of considerung the dynamics over time of up to about 5 attributes. Frequently, a multivariate time series is decom-

[3] Since this overview of statistical approaches corresponds to textbooks, I do not give references, if the particular method will not be used in succeeding sections. Focusing on data mining, [32] explains statistical approaches comprehensively.

posed into a set of univariate time series, thus disregarding the dependencies of different attributes.

In addition to the learning task of predicting the next measurement, the detection of trends, cycles, and seasons is investigated. For abstracting the time point view of linear precedence into time intervals of actual time, the detection of level changes can be used. An interesting recent approach is to transform the time series into a phase space [6]. The visualisation clearly shows regularities that cannot be recognized in the original form. The summary of time intervals according to a level can be seen as a first step towards immediate dominance.

The task of clustering subsequences has been solved in order to obtain categories as input to finding frequent sequences [9]. All subsequences of window length w are formed and similar subsequences are clusterd together. The clusters are labeled. The original sequence is transformed into the sequence of labels. The categories apply to overlapping sections of the original curve. This has to be taken into account, when using clustering as preprocessing for rules discovery.

The notion of *examples* becomes difficult when investigating time series. For instance, all minutely measurements of n attributes of a process are just one example for an n-variate time series. The prediction task is solved for one example, although by the technique of moving windows, many subseries are obtained and exploited for learning. The learning result is then applied to the very same process. If the process is something like the stock market or the wheather, there is, in fact, no other similar process available. We do not want to generalise the American, the Japanese, and the German stock market, if they are not (yet) observations of the same global economical process. Nor do we want to generalise the "wheather" of different planets. Hence, time series analysis really differs from the well established paradigm of empirical risk minimization, which assumes many independent observations of different individuals (processes). Let us look at other time stamped data. If warranty claims are analysed, the recall of mailing actions, or christmas sales, the aim is to generalize over sets of customers. This is in accordance with the principle of risk minimization. In order to apply time series methods, we have to perform the generalization step in advance. This can easily be done, for instance, by summing up the sales data of all shops. The result is one time series. It looks exactly like the time series of one process. However, it makes a difference in that less observations from the past are needed, because the present "observation" is already empirically based.

3.3 Frequent groups

The discovery of subgroups is one of the most common tasks of knowledge discovery. Originated in the database field, there is no assumption about the process producing the data. The typical database stores a huge amount of independent elements (e.g., contracts, sales, warranty cases). Frequent patterns in the data generalize over masses of time series. Association rules describe that some elements frequently occur together (frequent item sets). According to the confidence measure, the set is divided into an indicator set and an expected set. Although presented for basket analysis, the well-known APRIORI algorithm exploits an ordering of the items [1, 2]. Hence, it can easily be applied to sequences [3]. We only need to interpret the ordering as the time attribute. The confidence measure requires some adjustment. Rules of the form $A \rightarrow_T B$ state that if A occurs, then B occurs within time T. The frequency $F(A, B, T)$ is determined by counting, how often A precedes B, given a window of size T. The confidence for the rule can be defined as $\frac{F(A,B,T)}{F(A)}$, where $F(A)$ denotes the frequency of A [9]. A variety of algorithms concerning the discovery of frequent sequences exist, ranging from just testing a user-specified sequence pattern [16] to relational approaches [10]. Frequent subsequences can be detected using atual or abstract time. The algorithms can be applied directly to the time stamped data, or a categorisation step is performed in advance.

3.4 Relational learning

Often, it is interesting to find relations between durations of diverse events or categories. We might be interested in dependencies between events that are produced by different processes or abstract relations such as "as long as" or "directly after" or "in parallel". A natural way to represent time relations are rules with time points as chaining arguments, chain rules [12], [33]. Although chain rules are not restricted to time as the chaining arguments, they are particularly well suited for modeling time phenomena in both aspects, linear precedence and immediate dominance.

General chain rule: Let S be a literal or a set of literals. Let $args(S)$ be a function that returns the Datalog arguments of S. A normal clause is a general chain rule, iff its body literals can be arranged in a sequence $B_0 \leftarrow B_1, B_2,, ..., B_k, B_{k+1}$ such that there exist Datalog terms $Begin, End \in args(B_0)$, $Begin, T_1 \in args(B_1)$, $T_1, T_2 \in args(B_2)$, ...,$T_{k-1}, T_k \in args(B_k)$, and $T_k, End \in args(B_{k+1})$.

Chain rules can express relations between time intervals, form higher-level categories, and dependencies between different multivariate time series. They require facts as input that include two arguments referring to time points that mark the begin and the end of the time interval. Most algorithms of inductive logic programming are capable of learning chain rules. Hence, they solve the learning tasks related with immediate dominance. Either actual or abstract time may be used. However, they are weak in numerical processing. Therefore, time series with numerical attributes should be discretized beforehand.

3.5 Required L_E

The input formats for the selection of methods presented, are now listed. We write attributes A_j and their values a_j, abstract time points T_i and actual time points t_i, the class that is described by the attributes I_l and an instance i_l. This notation is meant to be close to the one of database theory. Note, however, that the semantic notion of the class being described can be mapped to the database relation or its key or even to one of the attributes of the database relation.

L_{E_1} **multivariate time series:** From a vector with measurements of attributes $A_1, \ldots A_k$, i.e. $i_l : t_1 a_{1_1} \ldots a_{1_k}, \ldots, t_i a_{i_1} \ldots a_{i_k}$ methods solving the prediction task output $i_l : t_{i+n} a_{i+n_1} \ldots a_{i+n_k}$, methods characterizing the time series output a label for a trend (e.g., increasing), a time interval for a season, or a function for the cycle. Note, that a_{i_j} are numerical values.

$L_{E_{1'}}$ **univariate time series:** The vector of measurements here only contains one numerical attribute: $i_l : t_1 a_1, \ldots, t_i a_i$. The output for prediction, trend, season, or cycle is similar to the one of multivariate series. Methods for detecting level changes deliver $i_l : t_m, t_n a_{mn}$, where a_{mn} is a computed value, e.g., the average. Clustering delivers a sequence of $Label_j[t_i, t_{i+w}]$, where w is the window size and the label is some computed summary of the attribute values $a_i \ldots a_{i+w}$.

L_{E_2} **nominal valued time series:** From a vector of nominal attribute values that can already be considered events, i.e. $i_l : t_1 a_{1_1} \ldots a_{1_k}, \ldots, t_i a_{i_1} \ldots a_{i_k}.$, the time region approach [40] outputs a set of rules of the form
$I : a_u, \ldots, a_v \rightarrow_{[t_b, t_e]} a_z$, where $u, v, z \in [1, k]$ and $b, e \in [1, i]$. The rule states that within the time interval $[t_b, t_e]$ the event a_z is to be expected if $a_u, \ldots, a_v$ have been observed.

L_{E_3} **sequence vectors:** A large set of vectors with nominal or numerical attribute values is the input to finding frequent sequences. The scheme of the vectors is similar to univariate time series, but the example set always consists of a large number of individuals that are described by the attribute. The time span is fixed to the given number of fields in the vector. The scheme $I : T_1 A_1, \ldots, T_i A_i$. is instantiated by all individuals about which data are stored in the database. The time points can vary from instance to instance, but the ordering is fixed. Rules learned are of the form $I : a_u, \ldots, a_v \rightarrow_{[t_e]} a_z$ with the meaning introduced for nominal valued time series.

L_{E_4} **facts:** A set of facts possibly concerning individuals of different classes, indicating a time interval of abstract or actual time ($[T_b, T_e]$) for an event given by several attributes are the input to relational learning. The number and type of attributes may vary for different predicates p that instantiate P. The facts have the form $P(I_1, T_b, T_e, A_r, \ldots, A_s)$, where some attribute A can denote another class.
If the learning task is to define higher-level categories, a classifying fact must be given for each example that is represented by a set of facts of the above form. The classifying fact has at least a time interval as arguments and the predicate denotes the higher-level category.

We have now developed a set of frequently used representations for the input of (time related) learning. In addition, many methods require the parameters window size w, the number of current observations *head*, and the prediction horizon n. The time scale has to be indicated by the granularity and the starting point of reference. Whereas L_{E_1} and L_{E_2} internally produce a large set of data for one example by moving windows, L_{E_3} and L_{E_4} are representations for sets of examples (independent observations). L_{E_4} in addition possibly combines different classes of individuals within one example. The notation for the examples already covers some semantical aspects. This is important in order to preprocess data appropriately, namely, to distinguish between attributes that refer to time, to a class, to features of an individual, or to relations between individuals of different classes.

4 Preprocessing for Time Phenomena

In order to discuss the transformations into the desired formats, let us now look at typical cases of raw data. We illustrate the representations by three cases that each stands for a large range of applications. The first case is a typical database with time-stamped database tuples, the second is a set of robot traces, where the measurements of 24 sensors are recorded over time, the third is a database of intensive care patients with their vital signs and infusions measured every minute.

The most frequent representation of raw data is

$L_{E_{DB1}}$ **database table:** a set of individuals and a set of time points is given according to the scheme $I: T_1A_1 \dots A_k, \dots, T_iA_1 \dots A_k$. It is a large set of multivariate time series with nominal or numerical values for the attributes.

Let us now look at the options for transforming the data into appropriate representations for learning.

4.1 The shop application – Representing time implicitly

In our first example, I is instantiated by shops, $i = 108$ denotes the weeks of two years, A_j is an item, and a_j its sale. In our application, the task was to predict the sales for an item in a time horizon n, that varies from $n = 4$ to $n = 13$. The prediction is necessary for optimizing the storage of goods. Of course, seasonal effects are present. They are already stored as binary flags within the database. The learning method was the regression mode of the support vector machine (SVM) [37]. The SVM requires input vectors of fixed length with numerical values. The method does not handle time explicitly. It is a rather common approach to compile time phenomena into attributes that are then handled by the learning method as any other attribute. The most frequently used choices are:

Multivariate to univariate transformation: For each attribute A_1 to A_k, store a vector for the corresponding univariate time series: $I: t_1a_1, \dots, t_ia_i$. The result are k vectors for all $i \in I$. In our example, where the sales of 50 items were analyzed, 50 vectors were stored for each of the 20 shops.

Sliding windows: Choose a window size of w consecutive time points, store the vector $i : t_1 a_{1_1} ... a_{1_k}, ..., t_w a_{w_1} ... a_{w_k}$, move the starting point by m steps and repeat, until $t_w = t_i$ (in our example $t_i = 108$). The result is a set of $i - w$ vectors for one time series (in our case, window sizes 3, 4, 5 were tried yielding 103 to 105 vectors).

Summarizing: Attribute values within a window of past observations are summarized by some function $f(a_{i_j}, ..., a_{i+w_j})$ (e.g., average, gradient, variance). The original time series is replaced by the discretized one:
$i : [t_1, t_w] f(a_{1_j}, ..., a_{w_j}), [t_m, t_{m+w}] f(a_{m_j}, ..., a_{m+w_j}), \ldots$.
Some approaches do not fix the window size, but find it in a data-driven fashion [9], [30]. Whereas most approaches deliver overlapping time intervals, [30] deliver a discretized time series with consecutive time intervals. Summarizing time windows is also viewed as a method of feature construction. In our shop example, we did not summarize the time series.

Multiple learning: Instead of handling diverse individuals in one learning run, a learning run can be started for each individual. The learning result of this run is used for the prediction concerning this individual only. In our shop example, for each shop (20) and each item (50), a separate learning of support vectors was started. The results are then used to predict the sales of this item in this particular shop.

Aggregation: Aggregating the shops by summing up the sales made in all shops did not perform well in our application. In principle, however, this aggregation is a common transformation. It constructs $i' \in I$ and a'_j and hence produces one time series for all individuals.

The resulting representation for the SVM that proved successful by cross validation was: $i : t_{i+w} a_{i+w_j} season, \ldots, t_i a_{i_j}$. This is a quite common combination of transformations if we follow a statistic-oriented approach: multivariate to univariate, sliding windows, and multiple learning.

4.2 The applications in intensive care

The second example, records of patients in an intensive care unit, offers raw data of the $L_{E_{DB}}$ form. The difference is that the length of the time series is not determined once for all patients but denotes the length of the patent's stay in the intensive care unit. Hence, the database table is organised as consecutive parts of the time series.

$L_{E_{DB2}}$ **tuples for time points:** The database no longer stores all measurements of one individual in one row, but only the measurements at one point in time: $I : TA_1 \ldots A_k$. There are several rows for one individual. The number of measurements needs not be equal for different individuals.
$i_1 : t_1 a_{1_1} \ldots a_{1_k} \ldots i_1 : t_i a_{i_1} \ldots a_{i_k}$
$\ldots$
$i_z : t_l a_{l_1} \ldots a_{l_k}$

We explored a variety of learning tasks within this application. The learning when and how to change the dosage of drugs is – with respect to prepocessing and learning – similar to the shop example, but we had to combine different rows of the table first.

Chaining database rows: Select all rows concerning the same individual and group its attributes by the time points. Output a vector of the length of the time series of this individual. The result is a multivariate time series.

Again, we used the SVM, now in the classification mode [17]. We experimented with many different features that were formed by sliding windows and different summarization methods. However, the past did not contribute to learning the decision rule. Hence, we learned from patients' state at t_i whether and how to intervene at t_{i+1}[29]. Therefore, in the end we could use the original data. The real difference to the shop application is that the decision rule is learned from a large training set of different patients and then applied to previously unseen patients and their states.

For a different learning task in the intensive care application, a method for time series analysis was applied. The learning task was to find outliers and detect level changes. A new statistical method was used [6]. It transforms measurements of one vital sign of the patient such that the length of the window is interpreted as dimensions of Euclidian space. Choosing $w = 2$, the measurements of two consecutive time points are depicted as one point in a two-dimensional coordinate system. It turns out, that outliers leave the ellipse of homogeneous measurements, and a level change can be seen as a new ellipse in another region of the space. The method is a special case of sliding windows.

In the intensive care application, current work now uses the detected level changes as input to a relational learning algorithm. This allows to combine various time series and detect dependencies among parameters, deviations from a stable (healthy) state, and therapeutical interventions. The learning task is to find time relations that express therapy protocols, in other words effective sequences of interventions.

4.3 The application in robot navigation

The third application to be presented here, is about sensor measurements of a mobile robot. The raw data are of the $L_{E_{DB2}}$ type, where I denotes mission or path, from which the measurements are taken, k is the number of sensors (in our case 24), and the only attribute is the measured distance to some (unknown) object. The learning tasks are higher-level concepts that can be used for navigation planning and execution [21]. Using chain rules with abstract time arguments allows to apply the learned knowledge to different environments [4]. However, relational learners that are capable of learning them, require facts as input, where the predicate indicates the summary of measurements and two arguments indicate the time interval in which it is valid. The requirements were

[4] The post-processing of learned rules into real-time control is summarized in [28].

further that the transformation can be applied on-line, i.e. purely incrementally, and the time intervals do not overlap. This excludes the standard methods from statistics as well as the approach of [9]. Hence, we developed our own method that closes a time interval if the gradient of the current summary and the current measurement varies more than a given threshhold [30]. Predicate symbols denote classes of gradients, e.g. `increase, decrease, peak`. The first step of preprocessig was to chain database rows in order to acquire a 24-variate time series for each mission. The second was to transform each one into 24 univariate time series. To these our variant of summarizing was applied.

Input to relational learning was first the set of all summarized univariate time series of all missions, together with the classification of the higher-level category (e.g., `sensor_along_wall`). The learned rules describe sequences of summarized sensor measurements that define the higher category. A classification corresponding to the placement of the sonar sensor at the robot was then used to combine sequences of several sensors. This led to the learned definition of sensor group features. In a bootstrap manner a hierarchical logic program was learned, that integrates all 24 sensors. Moreover, irrelevant relations between summarized measurements and their time intervals are filtered out by relational learning. The low accuracy of 27.1% (for `sensor_along_wall`) and 74.7% (for `sensor_through_door`) at the lowest level increased to 54.3% (`along_wall`) and 93.8% (`through_door`) at the highest level where all perceptions and actions are integrated [20]. This is surprising, because the learned rules of the lower level produced the examples for learning at the next higher level. It clearly shows the importance of taking into account the aspect of immediate dominance when handling time. Handling time with respect to linear precedence alone is unable to discover dependencies between 24 time series in several missions. It also illustrates the power of preprocessing: we could well consider all learning steps at lower levels as a chain of transformations that allow the highest-level data mining step.

5 Conclusion

In this paper, nine time-related learning tasks were presented, together with classes of algorithms that solve them. Five input languages for the methods were distinguished. Given two standard representations of time-stamped data in databases, it was shown, how they can be transformed into the desired languages for learning. All the transformations proved their value in many applications – not only the ones named in the paper. However, a uniform description of data, learning tasks, methods and L_E transformations was missing. The description shown in this paper can now be made operational as meta-data and transformation tools. I am certain, that the lists of tasks and transformations is not complete and that new publications will contribute more tasks and methods. This is not a counter argument, though. In contrast, it emphasizes the need for a preprocessing library.

Since we do not know which representation will turn out to be the best for a learning task, we have to try out several representations in order to determine the winner. This is a tedious and time consuming process. It is the goal of the MiningMart project to supply users with a workbench offering preprocessing tools in a unified manner. Moreover, a case base will present for several applications the winners of the representation race.

Acknowledgements

This work has been partially funded by the European Commission, IST-11993 (MiningMart). I particularly thank those partners with whom I enjoyed fruitful discussions along proposal writing.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, Washington, D. C., may 1993.
2. Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI Press/The MIT Press, Cambridge Massachusetts, London England, 1996.
3. Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *International Conference on Data Engineering*, Taipei, Taiwan, mar 1995.
4. J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.
5. James F. Allen. Maintaining knowledge about temporal intervals. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, chapter VII, pages 509–523. Morgan Kaufman, Los Altos, CA, 1985.
6. M. Bauer, U. Gather, and M. Imhoff. Analysis of high dimensional data from intensive care medicine. Technical Report 13/1998, Sonderforschungsbereich 475, Universit"at Dortmund, 1998.
7. Francesco Bergadano and Daniele Gunetti. *Inductive logic programming:from machine learning to software engineering*. The MIT Press, Cambridge, Mass., 1996.
8. Pvael Brazdil. Data transformation and model selection by experimentation and meta-learning. In C.Giraud-Carrier and M. Hilario, editors, *Workshop Notes – Upgrading Learning to the Meta-Level: Model Selection and Data Transformation*, number CSR-98-02 in Technical Report, pages 11–17. Technical University Chemnitz, April 1998.
9. Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule Discovery from Time Series. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 16 – 22, Ney York City, 1998. AAAI Press.
10. Luc Dehaspe and Hannu Toivonen. Discovery of Frequent DATALOG Patterns. *Data Mining and Knowledge Discovery*, 3(1):7 – 36, 1999.

11. Luc DeRaedt. *Interactive Theory Revision: an Inductive Logic Programming Approach.* Acad. Press, London [u.a.], 1992.
12. G. Dong and S. Ginsburg. On the decomposition of chain datalog programs into p (left-)linear l-rule components. *Logic Programming*, 23:203 – 236, 1995.
13. Robert Engels. Planning tasks for knowledge discovery in databases; performing task–oriented user–guidance. In *Proc. of th 2nd Int. Conf. on Knowledge Discovery in Databases*, aug 1996.
14. Robert Engels, Guido Lindner, and Rudi Studer. A guided tour through the data mining jungle. In *Proceedings of the 3nd International Conference on Knowledge Discovery in Databases (KDD-97)*, August 14–17 1997.
15. Gerald Gazdar and Chris Mellish. *Natural Language Processing in PROLOG.* Addison Wesley, Workingham u.a., 1989.
16. Valery Guralnik and Jaideep Srivastava. Event detection from time series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 33 – 42, San Diego, USA, 1999.
17. T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. MIT-Press, 1999.
18. J.-U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task–oriented models. In Stephen Muggleton, editor, *Inductive Logic Programming.*, number 38 in The A.P.I.C. Series, chapter 16, pages 335–360. Academic Press, London [u.a.], 1992.
19. Jörg-Uwe Kietz and Marcus Lübbe. An efficient subsumption algorithm for inductive logic programming. In W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning IML–94*, San Francisco, CA, 1994. Morgan Kaufmann.
20. Volker Klingspor. *Reaktives Planen mit gelernten Begriffen.* PhD thesis, Univ. Dortmund, 1998.
21. Volker Klingspor and Katharina Morik. Learning understandable concepts for robot navigation. In K. Morik, V. Klingspor, and M. Kaiser, editors, *Making Robots Smarter – Combining Sensing and Action through Robot Learning.* Kluwer, 1999.
22. S. Kramer, B. Pfahringer, and C. Helma. Stochastic propositionalization of non-determinate background knowledge. In D. Page, editor, *Procs. 8th International Workshop on Inductive Logic Programming*, pages 80 – 94. Springer, 1998.
23. Nada Lavrač and Sašo Džeroski. *Inductive Logic Programming — Techniques and Applications.* Number 148 in Artificial Intelligence. Ellis Horwood, Hertfortshire, 1994.
24. H. Liu and H. Motoda. *Feature Extraction, Construction, and Selection: A Data Mining Perspective.* Kluwer, 1998.
25. R.S. Michalski and T.G. Dietterich. Learning to predict sequences. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning - An Artificial Intelligence Approach Vol II*, pages 63–106. Tioga Publishing Company, Los Altos, 1986.
26. Ryszard Michalski. Inferential learning theory as a basis for multistrategy task-adaptive learning. In Michalski and Tecuci, editors, *Multistrategy Learning.* George Mason University, USA, 1991.
27. D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, New York u.a., 1994.

28. Katharina Morik. Tailoring representations to different requirements. In Osamu Watanabe and Takashi Yokomori, editors, *Algorithmic Learning Theory – Procs. 10th Int. Conf. ALT99*, Lecture Notes in Artificial Intelligence, pages 1 – 12. Springer, 1999.
29. Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach – A case study in intensive care monitoring. In *Proc. 16th Int'l Conf. on Machine Learning (ICML-99)*, Bled, Slowenien, 1999.
30. Katharina Morik and Stephanie Wessel. Incremental signal to symbol processing. In K.Morik, M. Kaiser, and V. Klingspor, editors, *Making Robots Smarter – Combining Sensing and Action through Robot Learning*, chapter 11, pages 185 –198. Kluwer Academic Publ., 1999.
31. S. Muggleton, A. Srinivasan, R. King, and M. Sternberg. Biochemical knowledge discovery using inductive logic programming. In Hiroshi Motoda, editor, *Procs. First International Conference on Discovery Science*. Springer, 1998.
32. Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, 1999.
33. Anke D. Rieger. *Program Optimization for Temporal Reasoning within a Logic Programming Framework*. PhD thesis, Universität Dortmund, Germany, Dortmund, FRG, 1998.
34. D. Sleeman, R. Oehlman, and R. Davidge. Specification of Consultant–0 and a Comparision of Several Learning Algorithms. Deliverable D5.1, Esprit Project P2154, 1989.
35. Devika Subramanian. A theory of justified reformulations. In D. Benjamin, Paul, editor, *Change of Representation and Inductive Bias*, pages 147–167. Kluwer, 1990.
36. C. Theusinger and G. Lindner. Benutzerunterstützung eines KDD-Prozesses anhand von Datencharackteristiken. In F. Wysotzki, P. Geibel, and K. Schädler, editors, *Beiträge zum Treffen der GI-Fachgruppe 1.1.3 Machinelles Lernen (FGML-98)*, volume 98/11 of *Technical Report*. Technical University Berlin, 1998.
37. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.
38. D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fé Institute, Santa Fé, CA., 1995.
39. Stefan Wrobel. *Concept Formation and Knowledge Revision*. Kluwer Academic Publishers, Dordrecht, 1994.
40. Wei Zhang. A region-based approach to discovering temporal structures in data. In Ivan Bratko and Saso Dzeroski, editors, *Proc. of 16th Int. Conf. on Machine Learning*, pages 484 – 492. Morgan Kaufmann, 1999.