A Classifier Based on a Decision Tree with Temporal Cuts

Jan G. Bazan[1], Adam Szczur[1], Marian Rzepko[2], Paweł Król[2], and Andrzej Skowron[3,4]

[1] Interdisciplinary Centre for Computational Modelling
University of Rzeszów
Pigonia 1, 35 - 310 Rzeszów, Poland
`bazan@ur.edu.pl, adamszczur8@gmail.com`

[2] Physical Education Faculty, University of Rzeszów
Towarnickiego 3, 35-959 Rzeszów, Poland
`marianrzepko@poczta.onet.pl, pkrol@ur.edu.pl`

[3] Institute of Mathematics, The University of Warsaw
Banacha 2, 02-097 Warsaw, Poland

[4] Systems Research Institute Polish Academy of Sciences
Newelska 6, 01-447 Warsaw, Poland
`skowron@mimuw.edu.pl`

Abstract. A new method of decision tree construction from temporal data is proposed in the paper. This method uses the so-called temporal cuts for binary partition of data in tree nodes. The novelty of the proposed approach is that the quality of cuts is calculated not on the basis of the discernibility of objects (time points), but on the basis of the discernibility of time windows labeled with different decision classes. The paper includes results of experiments performed on our data sets and collections from machine learning repositories. In order to evaluate the presented method, we compared its performance with the classification results of a local discretization decision tree, and other methods well known from literature. Our new method outperforms these known methods.

Keywords: rough sets, discretization, classifiers, temporal cuts, temporal data

1 Introduction

One of the most important approaches to inducing classifiers is the decision tree approach. There are many methods for generation of decision trees. In this work, we take into account local discretization trees (see, e.g., [7, 1]). Not inadvertently in the name of this kind of trees appears the name of the method of discretization, because in practical applications related to data mining, the method of discretization plays a very important role. Most often the discretization is understood in such a way that it refers to methods of decreasing the number of values of certain attributes so that, on the one hand, it facilitates the generation of data mining models (e.g., classifiers) and, on the other, it increases their expression, e.g., in matching classified new objects.

The classic local discretization tree [7, 1] efficiently generates a high-quality decision tree for a given decision table. However, this method does not take into account the elapsed time, that can be recorded during data collection. In fact, such time can be taken into consideration, but as a simple conditional attribute only. Meanwhile, in many decision-making problems, time is more important. For instance, single objects (rows) are often treated as labeled time points and are grouped in so-called time windows. In such case of the studied decision problems it is often necessary to deal with time windows, not time points (decision attribute values are defined for time windows).

Therefore, in the paper we propose to modify the classical method of local discretization tree generation to build such trees taking into account the time that is represented in the data sets by an additional attribute. In general, this modification consists in the fact that when generating a decision tree for division of node, instead of using a classical cut (attribute and value), the so-called *a temporal cut* is used (attribute, value, and some alpha coefficient - see later for more details). The time efficiency of the proposed method is asymptotically the same as in case of the method of the classical local discretization tree generation.

To illustrate the method and to verify the effectiveness of presented classifiers, we have performed several experiments on the data sets obtained from Physical Education Faculty at the University of Rzeszów (Poland) (two data sets) and UC Irvine Machine Learning repository (five data sets) (see Section 2).

1.1 Temporal Information Systems

Data sets used for approximation of concepts may be represented using rectangular data tables. This representation is based on representing individual objects by rows of a given data table and attributes by columns of a given data table. In this paper, we consider *information systems* of the form $\mathbf{A} = (U, A)$ in Pawlak's sense (see, e.g., [8]) for representation of data tables, where A is a set of attributes or columns in the data table, while U is a set of objects (rows) in the data table.

In practical applications of data mining, we deal often with complex objects. Their complexity may be manifested, e.g., by the fact that each such object is represented by a list of objects (rows) from a given information system. If we represent information about complex objects in this way, then we need to find a mechanism making it to not lose information about complex objects that are represented by each row. This information may be represented by the distinguished information system attribute which we mark by a_{id}. Apart from that, one could take into account that the complex objects occurring in a real world change with time and their properties (object states) should be registered at different time instants (in other words time points). Hence, it is also necessary to store together with the information about a given object an identifier of time in which these properties are registered. This information may also be represented by the distinguished attribute of information system which we mark as a_t. Because we assume that the time points are linearly ordered, then attribute a_t must be enriched by a linear ordering on the set of values of this attribute. The possibility of sorting values is sometimes also useful for the a_{id} attribute. Therefore, in this paper we assume that both attributes a_{id} and a_t are numerical attributes.

Hence, in order to represent complex object states observed in complex dynamical systems, the standard concept of information system requires extension. Therefore, we define a temporal information system. *A temporal information system* (see, e.g., [3]) is a tuple $\mathbf{S} = (U, A, a_{id}, a_t)$, where:

1. (U, A) is an information system,
2. a_{id}, a_t are distinguished numerical attributes from the set A.

About an object $u \in U$ we say that it *represents* the current parameters of the complex object with identifier $a_{id}(u)$ at time point $a_t(u)$ in the temporal information system $\mathbf{S}$. Let us add that the a_{id} attribute is understood here as the identifier of a complex object that can be observed at multiple time points. Each such point is represented in the system $\mathbf{S}$ by one object (a line in the table). Then for each such object the value of the a_{id} attribute is the same. For example, consider a complex object that is a patient with a fixed identifier id_p. Information about this patient can be represented in $\mathbf{S}$ by list of objects, where each of such object represents the information about patient at one time point. Of course, in each of these objects, the value of the a_{id} attribute is the same, and is equal id_p.

About an object $u_1 \in U$ we say that it *precedes* an object $u_2 \in U$ in the temporal information system $\mathbf{S}$ if and only if $u_1 \neq u_2 \ \wedge \ a_{id}(u_1) = a_{id}(u_2) \ \wedge \ a_t(u_1) < a_t(u_2)$.

Objects from the temporal information system are grouped into time windows. *A time window* in the temporal information system $\mathbf{S}$ is an any sequence $(u_1, ..., u_k)$ of objects from U such that $k > 1$, for any $i, j \in \{1, ..., k\}$ holds $a_{id}(u_i) = a_{id}(u_j)$ and $a_t(u_i) < a_t(u_{i+1})$, for $1 \leq i < k$.

The family of all time windows from the temporal information system $\mathbf{S}$ is denoted by $\mathbf{TW}(\mathbf{S})$.

In the paper, we also assume that the maximal size of the time window for all complex objects is the same. For a given complex object, a window with maximal size (called *a maximal time window*) represents the whole history of changes in attribute values from set $A \backslash \{a_{id}\}$ for that complex object. The family of all such time windows from the temporal information system $\mathbf{S}$ is denoted by $MTW(\mathbf{S})$ (where $MTW(\mathbf{S}) \subseteq \mathbf{TW}(\mathbf{S})$).

It is easy to see that for a given $u \in U$ exists only one time window $w = (u_1, ..., u_k) \in MTW(\mathbf{S})$ such that $u = u_i$ for exactly one $i \in 1, ..., k$. For a given $u \in U$ such time window we denote $w(u)$.

In practice, there is a need to consider a number of decision problems that require approximation of concepts. In the case of the temporal information system, these concepts are usually related to time windows, and therefore the membership of time widows to concepts must be somehow represented. As with classic decision tables, we can use here a special attribute called a decision attribute.

In the paper we assume that the decision attribute will have the same values for all time points that represent information about a given complex object. In other words, the specific value of the decision attribute is associated with all time points representing the given complex object.

Hence, the concept of temporal information system requires one more extension. Therefore, we define a temporal decision table. *A temporal decision table* is a six-element tuple $\mathbf{T} = (U, A, W, a_{id}, a_t, d)$, where:

1. (U, A, a_{id}, a_t) is a temporal information system that we denote by $\mathbf{S}$,
2. $W \subseteq \mathbf{TW}(\mathbf{S})$ is a set of time windows,
3. d is a distinguished attribute called a *decision attribute* from the set A, such that $d \neq a_{id}$, $d \neq a_t$, and for any $u_1, u_2 \in U$, if $a_{id}(u_1) = a_{id}(u_2)$ then $d(u_1) = d(u_2)$.

It is worth noting that with the above representation of a time window membership to concepts, the decision attribute value is always the same for all objects belonging to a given time window. Besides, it is easy to see that the temporal decision table for complex objects is a special case of the temporal information system.

If $\mathbf{S} = (U, A, a_{id}, a_t)$ is a temporal information system, $\mathbf{T} = (U, A, W, a_{id}, a_t, d)$ is a temporal decision table, and $W = MTW(\mathbf{S})$ then temporal decision table $\mathbf{T}$ we call *a temporal decision table with maximal time windows* and to simplify the description, we denote such table as $\mathbf{MT} = (U, A, a_{id}, a_t, d)$ instead of $\mathbf{MT} = (U, A, MTW(\mathbf{S}), a_{id}, a_t, d)$.

1.2 Classical Approach to Generation Classifiers for Temporal Decision Tables

A typical approach to inducing classifiers for a temporal decision table is to create a new classical decision table, whose rows correspond to the time windows, the decision attribute is defined for time windows, and the conditional attributes are defined over properties of time windows (usually suggested by a human expert). This consists in the fact that using some conditional attributes or a subset of conditional attributes from the input temporal decision table, a collection of attributes describing properties of a given time window is defined. For example, for any numeric attribute from the original temporal decision table, the following attributes can be defined: mean, median, standard deviation, maximum, and minimum value of this attribute. Sometimes a new attribute corresponds to the value of the first or last time point in a time window. Such attributes are often called time patterns, and a new decision table constructed using temporal patterns is called a *temporal pattern table* (see, e.g., [3, 2, 1]). Next, using such temporal pattern table, classifiers can be constructed using various methods known from the literature (see, e.g., [4, 5]).

Although often, such methods applied to the temporal decision table lead to the high quality classifiers, they have some disadvantages. Namely, it is not always possible to manually create all the time patterns that are relevant for the satisfactory approximation of the decision attribute. So, if there are not available relevant patterns using values of attributes for time points, then some of the important information from a given temporal decision table may be lost and thus the quality of the resulting classifier may not be satisfactory.

One can ask if methods based on automatic searching of features of time windows directly from the original temporal decision table can lead to better results? Searching methods for such features can provide access to the full information encoded in the time window. The method described in this paper can be considered as an exemplary method in this direction.

1.3 Temporal Cuts and Templates

The selection method of an attribute and its value (for numeric attributes often called the cut), that can be uses in the partition of a given data table, is a crucial in the tree construction method based on local discretization [7]. Note that during applying of this method the analysis of values of the decision attribute for training objects is performed. Thus, one of the most important concepts presented in the strategy of generating the local discretization tree is a cut that defines binary partition of the object set based on the attribute and its value.

Similarly to the classic approach, in this paper we use the notion of cut but in this case the cut is used not for defining a binary partition of the object set but for defining a binary partition of the time windows set. Therefore, in this case we call our cut in a special way, that is *a temporal cut*. Formally, a *temporal cut* is a triple (a, v, α) that is defined for a given *temporal decision table with maximal time windows* $\mathbf{MT} = (U, A, a_{id}, a_t, d)$, where $a \in A$, v is the value of the attribute a, and $\alpha \in [0.0, 1.0]$.

Moreover, any temporal cut (a, v, α) defines four *temporal templates*, where by a temporal template we understand a description of a set of objects grouped into time windows, that are defined in two different ways for numerical and symbolical attributes.

In case of numerical attributes, we define the following four templates for a temporal cut $c = (a, v, \alpha)$ and the table $\mathbf{MT}$:

– the first temporal template, called *a left-up template*, is described by a formula:
$TLU(c) = \{x \in U : \frac{card(\{u \in w(x):\ a(u) \geq v\})}{card(w(x))} \geq \alpha\}$,
– the second temporal template, called *a right-up template*, is described by a formula:
$TRU(c) = \{x \in U : \frac{card(\{u \in w(x):\ a(u) \geq v\})}{card(w(x))} < \alpha\}$,
– the third temporal template, called *a left-down template*, is described by a formula:
$TLD(c) = \{x \in U : \frac{card(\{u \in w(x):\ a(u) < v\})}{card(w(x))} \geq \alpha\}$,

– the fourth temporal template, called *a right-down template*, is described by a formula:
$TRD(c) = \{x \in U : \frac{card(\{u \in w(x):\ a(u) < v\})}{card(w(x))} < \alpha\}$.

Notice that the above templates are defined as descriptions of object sets, but they are actually descriptions of the collection of time windows in which the objects described by the template are grouped together.

An object $x \in U$ matches the template $TLU(c)$, if $a(u) \geq v$ holds for not less then $\alpha \cdot 100\%$ time points from the time window $w(x)$, otherwise the object x does not match the template $TLU(c)$. Whereas, an object $x \in U$ matches the template $TRU(c)$, if $a(u) \geq v$ holds for less then $\alpha \cdot 100\%$ time points from the time window $w(x)$, otherwise the object x does not match the template $TRU(c)$.

Similarly, an object $x \in U$ matches the template $TLD(c)$, if $a(u) < v$ holds for not less then $\alpha \cdot 100\%$ time points from the time window $w(x)$, otherwise the object x does not match the template $TLD(c)$. Whereas, an object $x \in U$ matches the template $TRD(c)$, if $a(u) < v$ holds for less then $\alpha \cdot 100\%$ time points from the time window $w(x)$, otherwise the object x does not match the template $TRD(c)$.

While, in case of symbolical attributes, we define four following templates for a temporal cut $c = (a, v, \alpha)$ and the table $\mathbf{MT}$:

– the first temporal template, called *a left-up template*, is described by a formula:
$TLU(c) = \{x \in U : \frac{card(\{u \in w(x):\ a(u) = v\})}{card(w(x))} \geq \alpha\}$,
– the second temporal template, called *a right-up template*, is described by a formula:
$TRU(c) = \{x \in U : \frac{card(\{u \in w(x):\ a(u) = v\})}{card(w(x))} < \alpha\}$,
– the third temporal template, called *a left-down template*, is described by a formula:
$TLD(c) = \{x \in U : \frac{card(\{u \in w(x):\ a(u) \neq v\})}{card(w(x))} \geq \alpha\}$,
– the fourth temporal template, called *a right-down template*, is described by a formula:
$TRD(c) = \{x \in U : \frac{card(\{u \in w(x):\ a(u) \neq v\})}{card(w(x))} < \alpha\}$.

Notice that the above templates are defined as descriptions of object sets, but they are actually descriptions of the collection of time windows in which the objects described by the template are grouped together.

An object $x \in U$ matches the template $TLU(c)$, if $a(u) = v$ holds for not less then $\alpha \cdot 100\%$ time points from the time window $w(x)$, otherwise the object x does not match the template $TLU(c)$. Whereas, an object $x \in U$ matches the template $TRU(c)$, if $a(u) = v$ holds for less then $\alpha \cdot 100\%$ time points from the time window $w(x)$, otherwise the object x does not match the template $TRU(c)$.

Similarly, an object $x \in U$ matches the template $TLD(c)$, if $a(u) \neq v$ holds for not less then $\alpha \cdot 100\%$ time points from the time window $w(x)$, otherwise the object x does not match the template $TLD(c)$. Whereas, an object $x \in U$ matches the template $TRD(c)$, if $a(u) \neq v$ holds for less then $\alpha \cdot 100\%$ time points from the time window $w(x)$, otherwise the object x does not match the template $TRD(c)$.

If c is a temporal cut then we denote by $T(c)$ the template defined by the c, keeping in mind that it might be one of two following templates $TLU(c)$, $TRU(c)$, $TLD(c)$, or $TRD(c)$.

In addition, to simplify the description, instead of $T(c)$, we sometimes write T, when the temporal cut c is fixed. Besides, if T is a template defined for some temporal cut c, by $\neg T$ we understand the template $TRU(c)$ when $T = TLU(c)$, the template $TLU(c)$ when $T = TRU(c)$, the template $TRD(c)$ when $T = TLD(c)$, or the template $TLD(c)$ when $T = TRD(c)$.

Finally, if T is a template defined for the temporal decision table with maximal time windows $\mathbf{MT} = (U, A, a_{id}, a_t, d)$, by $\mathbf{MT}(T)$ we denote a subtable of $\mathbf{MT}$ containing all objects from U matching the template T.

By $TDis(T)$ we denote the number of pairs of time windows from the different decision classes discerned by the template T. We explain the calculation of $TDis(T)$ using an example from Figure 1. There are 10 objects represented by four time points from two decision classes: X and O. The first class contains objects: x_1, x_2, x_3, x_4, x_5, while the second class contains objects: o_1, o_2, o_3, o_4, o_5. The attribute values for each object are changing with time.

A simple (not temporal cut (a, v)) divides the set of objects into two subsets L and R. In our example, for the first time point t_1, the subset $L = \{x_1, x_2, x_3, o_1, o_2\}$ and $R = \{x_4, x_5, o_3, o_4, o_5\}$. In the next time point, the value of attribute a of object x_3 has changed significantly, leading to changes of L and R: $L = \{x_1, x_2, o_1, o_2\}$ and $R = \{x_4, x_5, o_3, o_4, o_5, x_3\}$. In t_3 the contents of these sets are as follows: $L = \{x_1, x_2, o_1, o_2, o_5\}$, $R = \{x_4, x_5, o_3, o_4, x_3\}$, and in t_4 : $L = \{x_1, x_2, o_1, o_2, o_4\}$ and $R = \{x_4, x_5, o_3, x_3, o_5\}$.

Now, let us consider a temporal cut $c = (a, v, \alpha)$ with a numerical attribute a, template $T = TLU(c)$, where $\alpha = 0.7$. We calculate a measure $TDis(T)$ for the temporal cut T, which is the number of time windows pairs discerned by this template.

The template T divides time windows into two groups that match it, and those that do not match. For a time window that matches the template T, at least 70% of the time points of this window have the attribute value a greater than or equal to v. For time windows with objects x_1, x_2, o_1 and o_2 such a condition is not satisfied for any time point. Therefore, these objects do not match the T template. In the time window of the object x_3, this condition is satisfied at the point $t2$, $t3$, and $t4$. This means that $\frac{3}{4} = 0.75 > 0.7$ and therefore x_3 matches the T template. For time windows of objects x_4, x_5, and o_3 such condition is satisfied at every time point. Therefore, these objects also match the T template. On the other hand, for the object o_4, this condition is satisfied at $t1$, $t2$, and $t3$. This means that $\frac{3}{4} = 0.75 > 0.7$ and therefore the o_4 matches the T template. Lastly, for the object o_5 the condition satisfies at $t1$, $t2$, and $t4$. This means $\frac{3}{4} = 0.75 > 0.7$ and therefore the o_5 object matches the T template.

Summarizing, time windows of complex objects x_3, x_4, x_5, o_3, o_4, o_5 match the T template, while time windows of complex objects x_1, x_2, o_1 and o_2 do not match the T template. Hence, the number of pairs of complex objects (time windows) from the different decision classes discerned by the template T is given by $TDis(T) = 2 \cdot 3 + 3 \cdot 2 = 12$. After calculation of the value of this measure for all possible cuts, one can greedily choose one of the cuts and divide the entire set of objects into two parts on its basis. Of course, this approach can be easily generalized to the case of more than two decision classes. In our approach, the number $TDis(T)$ is treated as the quality of the cut that was used for definition of the template T.

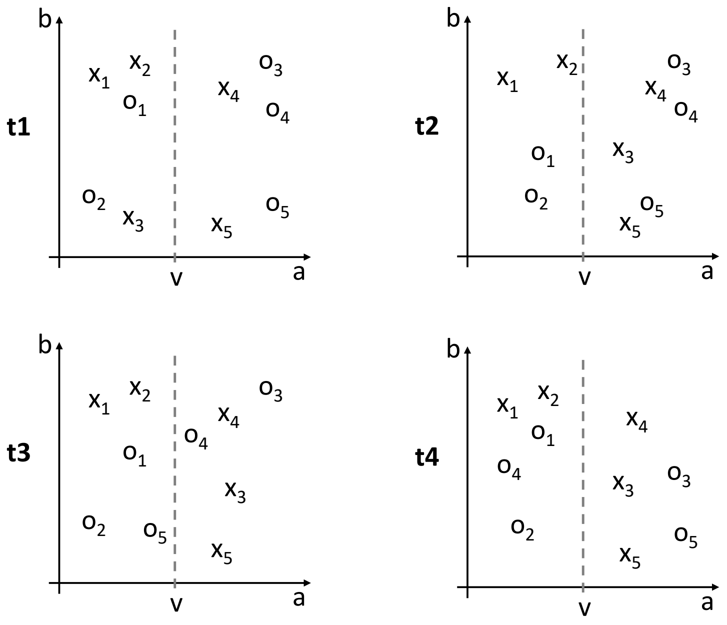

Fig. 1. Visualization of temporal cut in two-dimensional space

It should be noted that above quality measure $TDis(T)$ can be calculated for the given cut in time $O(n)$, where n is the number of objects in the temporal decision table. But searching for the optimal temporal cut requires the calculation of quality measures for all the potential cuts. For this purpose it is necessary to check all potential cuts, including all conditional attributes in a specific order. This can be done using various methods. One of such methods for numerical attributes firstly sorts the objects relative to the given attribute for which searching the optimal partition is performed. This allows us to determine the optimal cut in $O(n \cdot log\ n)$ time, where n is the number of all objects.

1.4 Calculation of Temporal Cuts

In this section, we present the algorithm for selection of temporal cuts. In order to find globally the best temporal cut, this algorithm can be executed for all numerical attributes different from a_{id}, a_t and d. For ease of discussion, we assume that there are only two decision classes C_0 and C_1 in the data set. This approach can be easily generalized to the case of more than two decision classes.

Algorithm 1: Computation of temporal cuts on the basis of selected attribute

```
Input: Let $\mathbf{MT} = (U, A, a_{id}, a_t, d)$ be a temporal decision table with maximal
       time windows and decision classes $C_0$ and $C_1$, $a \in A \setminus \{a_{id}, a_t\}$, and
       $\alpha \in [0.0, 1.0]$.
Output: The collection $RESULT$ of all temporal cuts on the basis of the
        attribute $a$ along with their quality.
1 begin
2     Sort the values of the numerical attribute $a$ and attribute $a_{id}$.
3     For each appearing cut $c$ on $a$, by browsing the values of the attribute $a$
        from the smallest to the largest, determine the following parameters:
        – $LD(a, c, C_0, \alpha)$, $LD(a, c, C_1, \alpha)$, $RD(a, c, C_0, \alpha)$, $RD(a, c, C_1, \alpha)$,
          i.e., numbers of time windows corresponding to objects that are matching
          the template $TLD(c)$ from class $C_0$, $TLD(c)$ from class $C_1$, $TRD(c)$ from
          class $C_0$, and $TRD(c)$ from class $C_1$, respectively,
        – the quality of cut $c$ on $a$ defined by the following formula
          $Q(c) = LD(a, c, C_0, \alpha) \cdot RD(a, c, C_1, \alpha) + LD(a, c, C_1, \alpha) \cdot RD(a, c, C_0, \alpha)$,
        – add temporal cut $c$ and its quality $Q(c)$ to the collection $RESULT$.

4     For each appearing cut $c$ on $a$, by browsing the values of the attribute $a$
        from the highest to the lowest, determine the following parameters:

        – $LU(a, c, C_0, , \alpha)$, $LU(a, c, C_1, \alpha)$, $RU(a, c, C_0, \alpha)$, $RU(a, c, C_1, \alpha)$,
          i.e., numbers of time windows corresponding to objects that are matching
          the template $TLU(c)$ from class $C_0$, $TLU(c)$ from class $C_1$, $TRU(c)$ from
          class $C_0$, and $TRU(c)$ from class $C_1$, respectively,
        – the quality of cut $c$ on $a$ by the following formula
          $Q(c) = LU(a, c, C_0, \alpha) \cdot RU(a, c, C_1, \alpha) + LU(a, c, C_1, \alpha) \cdot RU(a, c, C_0, \alpha)$,
        – add temporal cut $c$ and its quality $Q(c)$ to the collection $RESULT$.

5 end
```

Due to the sorting operation, the Algorithm 1 runs in time $O(n \cdot log\ n)$, where n is the number of objects from the table $\mathbf{MT}$.

Note, that the algorithm of calculation temporal cuts for symbolical attribute is simpler. We need only one viewing of all objects (time points) from temporal decision table and storing in memory some statistics about the time windows and values of the given attribute to determine the quality of cuts. For this purpose, we use advanced structures of data with acces in constant time. Thus, the complexity of calculation optimal cut for symbolical attributes is $O(n \cdot k)$, where n is the number of objects from the table **MT** and k is the number of diffrent values of given symbolical attribute.

1.5 Construction of the Decision Tree with Temporal Cuts

In this section, we present the algorithm for construction of a decision tree with temporal cuts, based on the above considerations (see Algorithm 2). Due to the fact that the algorithm uses temporal cuts, the decision tree produced by this algorithm is called the TC-tree or TC-decision tree.

Algorithm 2: Construction of the TC-decision tree

```
Input: Let $\mathbf{MT} = (U, A, a_{id}, a_t, d)$ be a temporal decision table with maximal
       time windows and $\alpha \in [0.0, 1.0]$.
Output: The TC-decision tree computed for the table $\mathbf{MT}$.
1 begin
2     Find the optimal temporal cut $c$ with the parameter $\alpha$ in the table $\mathbf{MT}$
        (using Algorithm 1).
3     Split the table $\mathbf{MT}$ into two subtables $\mathbf{MT}(T)$ and $\mathbf{MT}(\neg T)$ such that
        $\mathbf{MT}(T)$ contains the objects matching a template $T$, and $\mathbf{MT}(\neg T)$
        contains the objects matching a template $\neg T$.
4     Assign $\mathbf{MT}_l = \mathbf{MT}(T)$ and $\mathbf{MT}_r = \mathbf{MT}(\neg T)$
5     If the tables $\mathbf{MT}_l$ and $\mathbf{MT}_r$ satisfy the stop condition, then finish the tree
        construction else repeat steps 1-3 for all the subtables which do not satisfy
        the stop condition.
6 end
```

Stop condition of partition is satisfied when the given subtable contains only objects from one decision class or the considered cut does not have any effect, i.e., there are no new pairs of time windows from different decision classes separated by the cut.

1.6 Using the Decision Tree with Temporal Cuts to Classify Objects

The decision tree with temporal cuts can be treated as a classifier for the concept represented by decision attribute from a given temporal decision table **T**.

The algorithm for classification of new objects is the same as for classic method (see [7]). Note, however, that the template defined for temporal cuts are different from those in the classic decision tree. Therefore, for a given test object, its time points must be analyzed. We classify a tested object starting from the root of the tree using Algorithm 3.

Algorithm 3: Classification by the TC-decision tree

```
Input: Let $\mathbf{MT} = (U, A, a_{id}, a_t, d)$ be a temporal decision table with maximal
       time windows, $T$ be a template of the table $\mathbf{MT}$ and $u$ be a new
       (tested) object.
Output: The value of decision attribute for the object $u$.
1  begin
2      if u matches template T found for T then
3          go to subtree related to $\mathbf{MT}(T)$
4      else
5          go to subtree related to $\mathbf{MT}(\neg T)$
6      end
7      if u is at the leaf of the tree then
8          go to line 12
9      else
10         repeat lines 2-11 substituting $\mathbf{MT}(T)$ (or $\mathbf{MT}(\neg T)$) for $\mathbf{T}$
11     end
12     Classify $u$ to the decision class attached to the leaf;
13 end
```

It is not hard to see that the time complexity of Algorithm 3 depends on the length of the largest path in the decision tree with temporal cuts. The time complexity of Algorithm 3 is of order $O(l \cdot m)$, where m is the number of conditional attributes in the table $\mathbf{T}$ and l is the largest length of path in the decision tree with temporal cuts.

2 Experiments and Results

The experiments have been performed on the data sets obtained from UC Irvine (UCI) Machine Learning Repository [6] and Physical Education Faculty at the University of Rzeszów (Poland).

From the UCI there come one collection that is related to topic: Robot Execution Failures. This collection contains five data sets named *LP1*, *LP2*, *LP3*, *LP4* and *LP5*, which deals with the following decision values: failures in approach to grasp position, failures in transfer of a part, position of part after a transfer failure, failures in approach to ungrasp position and failures in motion with part. As values of conditional attributes at each time point, the instantaneous values of the various forces and torques measurements on a robot are recorded.

The collection from Physical Education Faculty contains two data sets: Basketball and Volleyball. Data was collected in 2014. The experiments have been carried out with a group of 61 persons - 31 for Basketball and 30 for Volleyball. The first dataset contains results for 15 female basketball players (Polish Junior Champions) and 16 women who did not play basketball professionally (young students of Physical Education Faculty). And the second - 13 male volleyball players (Resovia players - many times Polish Champions, one of the best team in the world) and 17 men who did not play volleyball professionally (students).

Each person stood on a balance board which was comprised of a dynamometric platform and a computer with software for recording and processing of diagnostic tests. The subjects stood for 30 seconds in a relaxed position on a platform, with upper extremities positioned along the body. The persons behaviour has been recorded by the AccuGait apparatus. Each record is represented by 6 attributes: triaxial forces of body pressure in the X, Y and Z directions (Fx, Fy,Fz) and 3-axial torque (MFx, MFy, MFz). Table 1 shows the summary of the characteristics of the data sets.

Table 1. Experimental data set details

Name of collection	Objects	Time points	Attributes	Classes
Robots LP1	88	15	6	normal (n), collision (c), front collision (f_c), obstruction (o)
Robots LP2	47	15	6	normal(n), front collision (f_c), back collision (bk_c), collision to the right, (r_c), collision to the left (l_c)
Robots LP3	47	15	6	ok, slightly moved (s_m), moved (m), lost (l)
Robots LP4	117	15	6	normal (n), collision (c), obstruction (o)
Robots LP5	164	15	6	normal (n), bottom collision (bm_c), bottom obstruction (bm_o), collision in part (p_c), collision in tool (t_c)
Basketball	31	1350	6	Yes (Y), No (N)
Volleyball	30	1350	6	Yes (Y), No (N)

The aim of conducted experiments was to compare the quality of our classification algorithm described in this paper with other classification methods developed in WEKA (see [5]) and RSES (see [4]) systems, well known from literature. For our classifier were used the original data sets, while the aggregated data sets were used for other classifying methods. In the aggregated data we have defined 5 new features for each of original attributes. The new features was based on statistical measures like minimum, maximum, average, median and standard deviation. The values of this attributes was calculated for each object based on values of their time points.

Table 2 contains all methods used in experiments and their short names used in the resulting table.

For testing quality of classifiers we applied leave-one-out (LOO) technique. The LOO technique involves a single object from the original data set as the validation data, and the remaining observations as the training data. This is repeated such that each observation in the sample is used once as the validation data.

The results of experiments are given in Table 3. As we see, for all data sets, overall results of our method was $7-23\%$ better than all other methods (omitting

Table 2. Classification methods used in experiments

Method	Designation
C4.5 (WEKA)	C4.5
NaiveBayes (WEKA)	NB
SVM (WEKA)	SVM
k-NN (WEKA)	kNN
RandomForest (WEKA)	RF
Multilayer Perceptron (WEKA)	MP
Global discretization + all rules (RSES)	GDRul
Local discretization + all rules (RSES)	LDRul
Classic Decision Tree (measure of cuts quality: MAX_DISC_PAIR)	CDT
Decision Tree with Temporal Cuts	TCDT

results for Basketball data set and SVM and MP methods). Also for almost all decision classes our method was better.

Table 3: Results of experiments

		Dataset						
Method		B-ball	V-ball	R_lp1	R_lp2	R_lp3	R_lp4	R_lp5
C4.5	overall	0.806	0.9	0.659	0.511	0.553	0.709	0.524
	class	Y: 0.8	Y: 0.923	n: 0.85	n: 1.0	ok: 1.0	n: 0.913	n: 0.907
		N: 0.813	N: 0.822	c: 0.824	bk_c: 0.5	m: 0.313	c: 0.712	t_c: 0.148
				o: 0.647	f_c: 0.167	s_m: 0.222	o: 0.476	p_c: 0.596
				fr_c: 0.294	r_c: 0.0	l: 0.0		bm_o: 0.048
					l_c: 0.0			bm_c: 0.538
NB	overall	0.806	0.9	0.614	0.617	0.638	0.752	0.451
	class	Y: 0.733	Y: 0.846	n: 0.85	n: 0.895	ok: 0.895	n: 1.0	n: 0.93
		N: 0.875	N: 0.941	c: 0.647	bk_c: 0.625	m: 0.75	c: 0.795	t_c: 0.296
				o: 0.529	f_c: 0.0	s_m: 0.111	o: 0.333	p_c: 0.34
				fr_c: 0.471	r_c: 0.0	l: 0.0		bm_o: 0.286
					l_c: 0.778			bm_c: 0.154
SVM	overall	0.968	0.833	0.545	0.468	0.553	0.615	0.439
	class	Y: 1.0	Y: 0.846	n: 0.85	n: 1.0	ok: 1.0	n: 0.0	n: 0.977
		N: 0.938	N: 0.824	c: 0.412	bk_c: 0.125	m: 0.438	c: 0.945	t_c: 0.0
				o: 0.618	f_c: 0.0	s_m: 0.0	o: 0.143	p_c: 0.468
				fr_c: 0.176	r_c: 0.0	l: 0.0		bm_o: 0.143
					l_c: 0.122			bm_c: 0.192
kNN	overall	0.903	0.767	0.659	0.489	0.574	0.786	0.622
	class	Y: 0.933	Y: 0.846	n: 0.85	n: 0.895	ok: 0.895	n: 0.913	n: 0.884
		N: 0.875	N: 0.706	c: 0.941	bk_c: 0.375	m: 0.438	c: 0.849	t_c: 0.444
				o: 0.529	f_c: 0.0	s_m: 0.333	o: 0.429	p_c: 0.617
				fr_c: 0.412	r_c: 0.2	l: 0.0		bm_o: 0.429
					l_c: 0.222			bm_c: 0.538
RF	overall	0.871	0.9	0.693	0.511	0.574	0.778	0.598
	class	Y: 0.933	Y: 0.923	n: 0.85	n: 0.947	ok: 0.947	n: 1.0	n: 0.93
		N: 0.813	N: 0.882	c: 0.882	bk_c: 0.375	m: 0.563	c: 0.877	t_c: 0.185
				o: 0.735	f_c: 0.0	s_m: 0.0	o: 0.19	p_c: 0.681
				fr_c: 0.235	r_c: 0.0	l: 0.0		bm_o: 0.333
					l_c: 0.333			bm_c: 0.538

Continued on next page

Table 3 – continued from previous page

Method		B-ball	V-ball	R_lp1	R_lp2	R_lp3	R_lp4	R_lp5
MP	overall	0.935	0.8	0.648	0.489	0.574	0.718	0.543
	class	Y: 1.0 N: 0.875	Y: 0.846 N: 0.765	n: 0.8 c: 0.882 o: 0.588 fr_c: 0.353	n: 0.895 bk_c: 0.5 f_c: 0.0 r_c: 0.0 l_c: 0.222	ok: 0.842 m: 0.625 s_m: 0.111 l: 0.0	n: 0.87 c: 0.781 o: 0.333	n: 0.884 t_c: 0.259 p_c: 0.532 bm_o: 0.238 bm_c: 0.538
GDRul	overall	0.839	0.9	0.591	0.511	0.596	0.778	0.598
	class	Y: 0.873 N: 0.806	Y: 0.93 N: 0.877	n: 0.839 c: 0.779 o: 0.526 fr_c: 0.237	n: 0.958 bk_c: 0.376 f_c: 0.331 r_c: 0.191 l_c: 0.0	ok: 1.0 m: 0.376 s_m: 0.226 l: 0.35	n: 0.94 c: 0.84 o: 0.378	n: 0.962 t_c: 0.344 p_c: 0.569 bm_o: 0.331 bm_c: 0.531
LDRul	overall	0.839	0.9	0.625	0.574	0.553	0.684	0.585
	class	Y: 0.873 N: 0.806	Y: 0.93 N: 0.877	n: 0.739 c: 0.895 o: 0.582 fr_c: 0.3	n: 0.958 bk_c: 0.624 f_c: 0.0 r_c: 0.191 l_c: 0.337	ok: 1.0 m: 0.376 s_m: 0. 111 l: 0.0	n: 0.985 c: 0.716 o: 0.239	n: 0.915 t_c: 0.456 p_c: 0.441 bm_o: 0.469 bm_c: 0.531
CDT	overall	0.839	0.9	0.648	0.489	0.596	0.701	0.476
	class	Y: 0.867 N: 0.813	Y: 0.923 N: 0.882	n: 0.75 c: 0.824 o: 0.647 fr_c: 0.353	n: 0.947 bk_c: 0.375 f_c: 0.0 r_c: 0.0 l_c: 0.222	ok: 0.947 m: 0.5 s_m: 0.222 l: 0.0	n: 0.957 c: 0.74 o: 0.286	n: 0.907 t_c: 0.185 p_c: 0.34 bm_o: 0.286 bm_c: 0.462
TCDT	overall	0.935	0.967	0.92	0.745	0.787	0.957	0.799
	class	Y: 0.933 N: 0.938	Y: 1.0 N: 0.941	n:1.0 c: 0.882 o: 0.971 fr_c: 0.75	n: 1.0 bk_c: 0.714 f_c: 0.5 r_c: 0.4 l_c: 0.556	ok: 1.0 m: 0.8 s_m: 0.556 l: 0.0	n: 1.0 c: 0.986 o: 0.81	n: 0.977 t_c: 0.538 p_c: 0.681 bm_o: 0.905 bm_c: 0.885
	alpha	0.988	0.963	0.869	0.851	0.858	0.91	0.904

3 Conclusions

A new method for a decision tree construction from temporal data was presented in the paper. This method introduces the so-called temporal cuts, used for binary partition of data in tree nodes, based on analysis values of the attributes of all the time points of training objects from a time window. Classifier based on the decision tree with temporal cuts outperforms other classifiers used in the experiments. The approach can be used for exploration of data sets from many domains such as sport, robotics, medicine, finance, industry, transport, telecommunication, and others.

Acknowledgement

This work was partially supported by three following grants of the Polish National Science Centre: DEC-2013/09/B/ST6/01568, DEC-2013/09/B/NZ5/00758, and also by the Centre for Innovation and Transfer of Natural Sciences and Engineering Knowledge of University of Rzeszów, Poland.

References

1. Bazan, J.G., Bazan-Socha, S. and Buregwa-Czuma, S., Dydo, L., Rzasa, W., Skowron, A.: A classifier based on a decision tree with verifying cuts. Fundamenta Informaticae, 143 (1-2), 1–18 (2016)
2. Bazan, J.G., Bazan-Socha, S., Buregwa-Czuma, S., Pardel, P.W., Sokolowska, B.: Predicting the presence of serious coronary artery disease based on 24 hour Holter ECG monitoring. In: M. Ganzha, L. Maciaszek, M. Paprzycki (eds.), *Proceedings of the Federated Conference on Computer Science and Information Systems*, 2012, pp. 279-286, IEEE Xplore - digital library.
3. Bazan, J.G.: Hierarchical classifiers for complex spatio-temporal concepts. In: Transactions on Rough Sets IX, pp. 474–750. Springer (2008)
4. Bazan, J.G., Szczuka, M.: The rough set exploration system. In: Transactions on Rough Sets III, pp. 37–56. Springer (2005)
5. Frank, E., Hall, M., Witten, I.: The weka workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", 4th edn. Morgan Kaufman, Burlington (2016)
6. Lichman, M.: UCI Machine Learning Repository (2013), http://archive.ics.uci.edu/ml
7. Nguyen, H.S.: Approximate boolean reasoning: Foundations and applications in data mining. LNCS Transactions on Rough Sets V 4100, 334–506 (2006)
8. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177, 3–27 (2007)