From Temporal Rules to One Dimensional Rules

Kamran Karimi and Howard J. Hamilton

Department of Computer Science
University of Regina
Regina, Saskatchewan
CANADA S4S 0A2
{karimi, hamilton}@cs.uregina.ca

Abstract. In this paper we propose a new algorithm, called 1DIMERS (One Dimensional Investigation Method for Enregistered Record Sequences), to mine rules in any data of sequential nature, temporal or spatial. We assume that each record in the sequence is at the same temporal or spatial distance from others, and we do not constrain the rules to follow any monotonic direction, meaning that the rules can involve condition attributes in previous and next records relative to the decision attribute. Removing the conceptual temporal limitations makes 1DIMERS a generalised form of TIMERS (Temporal Investigation Method for Enregistered Record Sequences). TIMERS merges consequent records together, and then finds causal or acausal relationships among the variables in the merged records. The kind of rules discovered by TIMERS has also been called sequential rules. In general the passage of time is limited to one direction, which has been used in our previous work to distinguish between causality and acausality. Since in principle it is possible to move back and forth along a sequence, with general sequential data we can no longer intuitively speak of causality and acausality based on a direction. As a result, in 1DIMERS we substitute the terms "causality" and "acausality" with "forward-predictive" and "backward-predictive," respectively. 1DIMERS and TIMERS may each be applicable to a different problem depending on the user's choice, and we give examples of each program's applicability. In previous work we have been using C4.5 as the classifier for creating temporal rules. Here we employ CART as well, which has the ability to regress as well as classify, and show that the results are independent of the underlying rule discovery program.

1. Introduction

Given a sequence of records, the problem we are considering is finding rules for predicting the value of a decision attribute that appears in each record. The traditional approach is to look for a relationship among the decision attribute and other attributes within the same record. One example rule would be [If(a = 6) then (*decision* = *false*)]. This method may not produce good results if there is an *inter-record* relationship among the attributes. Bounded by temporal constraints, in such a case we usually expect only the previous records to affect the current decision attribute, but here we investigate the possibility that the decision attribute's value is determined by attributes not only in previous records, but in next records, or both previous and next records. One example rule in this context would be: [If{($a_{current-1}$ = 2) AND ($b_{current+1}$ = 2)} then ($decision_{current}$ = *false*)].

The attributes are now qualified with their position *relative* to the position of the decision attribute. In this example, "current-1" could be read as "previous," and "current+1" could be read as "next."

Relying on data records that appear one after the other in a sequence from a single source (so they are related), sets our approach apart from methods such as [16] than do not consider any ordering among the input records. In this paper we use the term "one-dimensional" instead of "sequential" to avoid confusion with common terminology. Though the data we deal with is sequential in nature, it need not obey any temporal order. Also, the attributes in each record in the sequence can represent any number of dimensions, temporal or spatial. "One-dimensional" here refers to the fact that there is a single temporal or spatial ordering among the records. Also, a sequence implies an unbreakable order, while in this paper we present rules that go back or forth (or both) in the sequence to predict the value of a decision attribute. This usage may not be suitable for real-time execution of temporal rules (we can't give a verdict until sometime in the future). We expect them to be of value in cases where predictive power is of more importance, or we are processing stored temporal data where at any given instant the future and past are available. Alternatively we may be processing spatial data, where future and past are simply substituted by notions of nearby locations (neighbourhood), and considered available. Lifting the restriction of following an inherent order in rules opens the door to new methods of analysing data. Other than that, there are hints that in spite of the intuitive appeal of finding patterns and rules in temporal sequences of data such as time series in a fixed temporal direction, in some cases the results may not be useful [6].

Sequential data and sequential rules have been studied before [1, 4, 20]. For example, in [4] the authors provide a genetic algorithm solution to the problem of detecting rules that manoeuvre a plane that is being chased by a missile in a two dimensional space. Discrete attributes such as speed, direction of the missile, turning rate of the plane, etc. are measured during 20 time steps. It is assumed that after 20 steps the missile will stop the chase. The rules discovered in that paper form part of a plan, and the genetic algorithm changes parts of the plan to make them better suitable to solving the problem. Since the aim of the plans is to prevent a hit, the system is developed to produce rules that come up with evasive actions. The rules are then used in a simulator to measure their effectiveness. Time is obviously the sequencing factor in this example. In this paper we provide one example of sequential data that resembles this application in the sense that it consists of 15 measurements made after a failure is detected in a robot. The observations are then used to classify failure types. However in general we are interested in predicting the value of an attribute that is included in each record. "Being hit," or "failure" does not appear in any of the records, while an attribute such as soil temperature can be measured at regular intervals along with other related variables. Other examples in this paper address the problem of predicting the value of such an attribute.

The remainder of the paper is organised as follows. Section 2 provides background for our work and also presents intuitive examples of the concepts used in the paper. Section 3 formally presents the 1DIMERS (One Dimensional Investigation Method for Enregistered Record Sequences) algorithm, which is a generalised version of TIMERS (Temporal Investigation Method for Enregistered Record Sequences). In Section 4 we present the

.

results of experiments with TIMERS, showing its effectiveness in solving temporal problems. The results of running TIMERS on a Robot learning problem, involving fixed number of relevant records ($w = n = 15$), are presented. It is a classification problem, with discrete values for the decision attribute. Other experiments in Section 4 show that TIMERS is effective for solving regression problems, where the decision attribute is continuous, and provide the results of running CART on two datasets. C4.5's results are provided for comparison purposes. 1DIMERS overlaps with TIMERS in its method, and for comparison's sake its results are provided in each case after trying TIMERS. Section 5 looks at another application domain that closely resembles the temporal domain, and that is spatial sequential data, and shows that the same techniques are effective there. TIMERS and 1DIMERS' results are presented and compared. Section 6 concludes the paper.

2. Background

Our previous work focuses on discovering temporal rules [7, 8] that allow us to predict what happens next, given previous observations. A temporal rule is a rule that involves time, i.e. the condition attributes appear at different times than the decision attribute. We divide temporal rules into two possible categories, causal, and acausal. A *causal* relation is one that involves attributes in the past affecting the decision attribute in the future [19]. The past affecting the future is the normal direction of time, and provides our definition of causality with an intuitive sense. We also consider the case where the future observations affect the past. We consider future affecting the past as a sign of *acausality* [15], or *temporal co-occurrence*. In an acausal relationship, the attributes just happen to be observed together over time, while none is causing the other. In this case there may be a hidden cause that has escaped our observation. Other than causality and acausality, the third possibility is that the relationship between the condition attributes and the decision attribute is *instantaneous*, meaning that value of the decision attribute is best determined by the condition attributes at the same time.

We proposed the TIMERS method to detect a causal or acausal relation among temporal sequences of data [11, 13, 14]. TIMERS provides a set of tests and guidelines, for judging the nature of a relationship. It is partly performed by software, and partly by the domain expert who is analysing the data. Following this algorithm, we generate classification rules from the data, using an operation called *flattening*. Flattening merges consecutive records in the normal, forward, direction of time (for the causality test) or the backward direction of time (for the acausality test). The number of records merged is determined by a time window *w*, and represents our guess as to how many records may be involved in an inter-record relationship. The quality of the rules, determined by their training or predictive accuracy, allows us to judge the data as containing a causal or acausal relation. TIMERS performs three tests: One without flattening, to test the instantaneous hypothesis, and two others to determine the temporal characteristics of the data. The order to consider goes from instantaneous, to acausal, to causal. So if the results of an instantaneous test is about the same or better than the other two, then we declare the relationship among the decision and condition attributes to be instantaneous. Otherwise if the results of the acausality test is about the same, or better than the causality test, then we

.

declare the relationship as acausal. Otherwise the relationship is causal. This order implies that when dealing with temporal relationships, the tendency is to declare it as acausal. More explanation is provided in [13], where an algorithm for flattening data in both forward and backward directions is provided.

In a sequential spatial dataset it is reasonable to assume that there may be connections between records at the neighbouring positions, before and after the current record. In this paper we introduce the *sliding position* flattening method which includes forward and backward flattening as special cases. The principle behind the sliding position method is that both previous and next records can be influential in determining the current value of the decision attribute. In a temporal domain this means considering both past and future observations. With any fixed window size *w*, the new flattening algorithm first places the current decision attribute at position one, and uses the next *w*-1 records to predict its value. This corresponds to a *backward flattening* in TIMERS, where future values are used to predict the past. Then the current attribute is set at position 2, and the previous record (position one) and the next *w*-2 records are used for prediction. This case has no correspondence in our previous algorithm in [13]. This movement of the current position continues and at the end it is set to *w*, and the previous *w*-1 records are used for prediction. This corresponds to *forward flattening* in TIMERS.

As an example consider four temporally consecutive records, each with four fields: <1, 2, 4, true>, <2, 3, 5, false>, <6, 7, 8, true>, <5, 2, 3, true>. Suppose we are interested in predicting the value of the last (Boolean) variable. Using a window of size 3, we can merge them as in Table 1. The decision attribute is indicated in **bold** characters. When it comes to the record involving the decision attribute, we do not consider any condition attributes in the same record as the decision [13]. The *Record.value* notation in Table 1 means that we are only including the decision attribute. For example, <R_1, R_2, R_3.**false**> would contain <1, 2, 4, true, 2, 3, 5, true, **false**>, where *false* is the decision attribute in R_3. This is to make sure that minimum amount of data is shared between the original record and the flattened record.

Instantaneous. w = 1 (original data)	Forward (Causality). w = 3	Backward (Acausality). w = 3	Sliding position. w = 3
R_1 = <1, 2, 4, **true**>	<R_1, R_2, R_3.**false**>	<R_3, R_2, R_1.**true**>	<R_2, R_3, R_1.**true**>
R_2 = <2, 3, 5, **true**>	<R_2, R_3, R_4.**true**>	<R_4, R_3, R_2.**true**>	<R_1, R_3, R_2.**true**>
R_3 = <6, 7, 8, **false**>			<R_1, R_2, R_3.**false**>
R_4 = <5, 2, 3, **true**>			<R_3, R_4, R_2.**true**>
			<R_2, R_4, R_3.**false**>
			<R_2, R_3, R_4.**true**>

Table 1. Results of flattening using the forward, backward, and sliding position methods

Normal flattening (vs. sliding position flattening) with a window size of *w* reduces the number of records by *w*-1. It is possible that not all the data in a dataset follow each other temporally, but only every *n* records. For example, every two records were generated one after the other, but there is no relationship between the first record and the third record, or any other record. In this case we consider the window size *w* to be the same as *n*, and

.

perform flattening so that every consecutive *n* records are merged into one, and thus the number of flattened records is divided by *n*. Sliding position flattening increases the number of records.

TIMERS and 1DIMERS perform a series of pre-processing operations, notably flattening, then provide the processed data to another software to generate rules or trees and evaluate them. A post-processing phase can then follow, in which the data are presented to the user in a sequentially meaningful way. We have tried C4.5 [17] before, and because of its availability of source code, have been able to integrate it into our TimeSleuth software [9, 12]. TimeSleuth performs all processing before and after running C4.5 and hence partially implements both the TIMERS and 1DIMERS algorithms. Results of comparing TimeSleuth with other causality miners appear in [14]. Being a classifier, to apply data with continuous variables to C4.5, one has to perform discretisation on the decision attribute, which is not always reasonable with continuous data. In this paper we use another package called CART [2] which can classify as well as perform regression. We evaluate our method with CART as the underlying rule-discoverer, and we show that it performs with little basic variation with different rule-discovery programs.

To use CART we had to perform many of the pre- and post-processing operations "by hand," i.e., using tools that were not integrated into CART. We performed flattening with TimeSleuth, and then deleted the current-time attributes using Mcrosoft Excel. The results for CART were not presented in a temporally valid way since CART, like most other data mining and machine learning algorithms, does not consider any order among its input records. So in the output a variable from the future could precede a variable from the past, for example.

3. The 1DIMERS Algorithm

Modern physics has established time and space as a unity, where one is inconceivable without the other. However, time remains an anomaly because unlike the spatial dimensions, it seems that one cannot move back in time, although experiments have shown that at the particle level, this is in fact possible [5]. Discovering temporal associations that predict the future, based on past observations, is possible, and one can conceptually use the same idea for one-dimensional space as well. Our previous work has used the distinction between moving back and forth in time as the basis of distinguishing causality on one side, and acausality (or temporal co-occurrence) on the other. Since we do not consider an explicit representation of time as necessary (time is implicitly present in the order of the records), a measure such as length can be substituted for time.

Consider the problem of drilling a well. The well can be regarded as a one-dimensional entity. As the drill is making its way through the ground, new points are explored and registered. When we stop, we have a series of records that follow each other along the line. While it seems that the data was produced in a certain temporal order, one could argue that if the drilling were started from the opposite side, then we would be encountering the points from the reverse direction of time. It makes perfect sense to analyse the drilling data in any direction of time, with the results being valid in both cases.

.

Of course now it is not possible to talk about cause and effect because what happens to precede something in one direction, will be following it in the opposite direction.

1DIMERS is an evolution of TIMERS. It changes the terminology from a temporal domain to a spatial domain and provides a more general flattening method, as intuitively described in Section 2. In 1DIMERS an instantaneous rule becomes a *punctual* rule (happens on a point instead of at an instant). A causal rule becomes a *forward-predictive* rule. An acausal rule becomes a *backward-predicitve* rule. The intuitive distinction of causal vs. acausal rules does not exist here. "Forward" and "backward" in 1DIMERS simply refer to the original direction of the data. The lack of distinction between the two possible directions of movement on a line side steps some conceptual problems and debates about causality [3]. In 1DIMERS two new categories are added to one-dimensional rules. The first one is called "linearly extended." A relation is called linearly extended when it is not punctual, and there is no strong evidence that either direction (forward or backward) result in a better predictive ability. The second new category is called *bidirectional predictive* and applies to cases where both directions result in similar results. In TIMERS such cases would be labelled acausal in a temporal domain because. The bias towards acausality in TIMERS is because causality is a strong assumption about any relation. In the general one-dimensional domain, where the restrictions and implications of time do not hold, we can employ a more varied terminology. The sliding position flattening operator is presented in Algorithm 1.

```
For (i = 0; i ≤ |D| - w; i++)
{
        flattenedRecord = <>
        for(j = 1; j < pos, j++)              // previous records
            flattenedRecord += D_{i+j}
        for(j = pos + 1; j ≤ w, j++)          // next records
             flattenedRecord += D_{i+j}
         flattenedRecord += Field(d, D_{i+pos})     // the decision attribute
        output(flattendRecord)
}
```

Algorithm 1. The Sliding position flattening method

Formally, the flattening operator $F(\boldsymbol{w}, \boldsymbol{pos}, \boldsymbol{D}, \boldsymbol{d})$ takes as input a window size $\boldsymbol{w}$, The position of the decision attribute within the window $\boldsymbol{pos}$, the input records $\boldsymbol{D}$, and the decision attribute $\boldsymbol{d}$, and outputs flattened records according to Figure 1. D_i returns the ith record in the input **D**. *Field*() returns a single field in a record, as specified by its first variable. The += operator stands for concatenating the left hand side with the right hand side, with the results going to the right hand side variable. <> denotes an empty record. This flattening algorithm is simpler than the one presented in [13]. The 1DIMERS algorithm is presented in Figure 2 below. *F*() is the flattening operator as defined in Algorithm 1.

.

Input: A sequence of sequentially ordered data records D, minimum and maximum flattening window sizes α and β, where $\alpha \leq \beta$, a minimum accuracy threshold Ac_{th}, tolerance values ε_i, and a decision attribute d. The attribute d can be set to any of the observable attributes in the system, or the algorithm can be tried on all attributes in turn.
Output: A verdict as to whether the relation among the decision attribute and the condition attributes is punctual, forward-predictive, backward-predictive, bidirectional predictive, or linearly extended.
RuleGenerator() is a function that receives input records, generates decision trees, rules, or any other representation for predicting the decision attribute, and returns the training or predictive accuracy of the results.

1DIMERS(D, α, β, Ac_{th}, ε, d)
ac_p = RuleGenerator(D, d); // punctual accuracy. window size = 1
for ($w = \alpha$ to β)
    for(pos = 1 to w)
        $ac_{w,pos}$ = RuleGenerator(F(w, pos, D, d), d)
    end for
end for

ac_b = max($ac_{\alpha,1}$, …, $ac_{\beta,1}$) // The best value with the decision attribute in the past (backward)
ac_f = max($ac_{\alpha,\alpha}$, …, $ac_{\beta,\beta}$) // The best value with the decision attribute in the future (forward)
ac_l = max($ac_{\alpha,\theta}$, …, $ac_{\beta,\theta}$), $\forall\ w, \alpha \leq w \leq \beta$, then $1 < \theta < w$ // Best value in-between

// Maybe there is not enough related information, or the variables are random
if ($Ac_{th} >_{\varepsilon_1}$ max(ac_p, ac_l, ac_f, ac_f) then discard results and stop.

verdict = "for attribute " + d + ", "

if ($ac_p \geq_{\varepsilon_2}$ max(ac_l, ac_f, ac_b) then verdict += "the relation is punctual"
else if($ac_l >_{\varepsilon_3}$ max(ac_f, ac_b)) then verdict += "the relation is linearly extended"
else if ($ac_f \approx_{\varepsilon_4} ac_b$) then verdict += "the relation in bidirectional-predictive"
else if ($ac_b >_{\varepsilon_5} ac_f$) then verdict += "the relation is backward-predictive"
else verdict += "the relation is forward-predictive"

return verdict.

Algorithm 2. The 1DIMERS method

We use the ε subscripts in the comparison operators to allow the domain expert to ignore small differences. We define $a >_\varepsilon b$ as $a > b + \varepsilon$ and $a \approx_\varepsilon b$ as $|a - b| \leq \varepsilon$. The value of ε, a non-negative number, is determined by a domain expert. If the results for different window values are about the same, we suggest using the smallest window size.

4. Experimental Results

This section provides the results of experiments with the TIMERS and 1DIMERS methods. There is a clear temporal element in the datasets used in the experiments. The

.

results confirm that regression and classification both give consistent results. In all experiments we use training accuracy values.

Even though 1DIMERS is more general and includes TIMERS, in some cases its additional analysis methods may not be applicable or needed. As shown in the experiments below, this includes cases where the semantics of the data do not allow a sliding position flattening (robot failure data in section 4.1), and hence 1DIMERS is not applicable. Another example is in strictly temporal datasets where there is a strong causal relationship (artificial robot data 4.2.1), where 1DIMERS's new flattening method would not produce any better results than TIMERS.

4.1 Classifying Robot Failures

In this section we attend to the problem of failure detection in a robot that grabs, moves and puts down objects. Upon encountering a failure, the force and torque values in the *x*, *y*, and *z* axis (a total of 6 values) are recorded 15 times at regular intervals. The whole process takes 315 ms. The results are then used to classify the type of error that occurred. In [18], five strategies have were to create decision rules for solving the problem. The first one uses the 6 sensor values as they are, while in others these values are processed first, and then used in the decision making process. The 5th strategy combines all the data available to other strategies. The observations have been divided into 5 learning problem datasets. LP1 (failure in approach to grasp), LP2 (failure in transfer), LP3 (failure in positioning a part after a transfer), LP4 (failures in approach to ungrasp), and LP5 (failure in motion with part). Here we observed the force and torque values after an error had already occurred. This kind of data must be processed by the standard TIMERS method, as it does not make sense to place the decision attribute (occurrence of failure) within the observed records. Thus a sliding position flattening is impossible.

To obtain the results in Table 2, we used TIMERS to merge every 15 consecutive records into a single one, and used C4.5 to create decision trees. We tried both the forward and the backward directions of time. C4.5 was invoked with default parameters, with the exception of the values marked with a *, where the -g (use gain) option was used to generate the decision tree. In these cases the values obtained by default arguments appear in parenthesis. The window size was fixed at 15. The five strategies covered in [18] are presented as S1 to S5. The best value for each learning problem among the 5 strategies is presented in bold.

Problem	S1	S2	S3	S4	S5	TIMERS (forward)	TIMERS (backward)
LP1	78%	80%	**96%**	85%	89%	97.7%	97.7%
LP2	45%	57%	51%	**68%**	64%	95.7%	95.7%
LP3	49%	75%	**87%**	85%	83%	85.1%* (48.0)	97.9%
LP4	65%	60%	**95%**	77%	83%	100%* (94.9)	100%* (99.1)
LP5	69%	63%	72%	49%	**77%**	90.9%* (89.0)	90.9%* (82.3)

Table 2. Accuracy values for the robot learning problem

We see that TIMERS gives either better or nearly the same accuracy values as the best of the 5 strategies in [18]. It is also more consistent compared to the other 5 strategies in

.

terms of the quality of results. While TIMERS and S1 both use the original values of force and torque, TIMERS performs considerably better without requiring the user to come up with ways to process data. This is a desirable quality because it frees the user from having to guess which processing method should be used in any particular case.

4.2. Evaluation of Regression on Temporal data

In this subsection we compare the effectiveness of our TIMERS and 1DIMERS methods using two different rule discovery approaches, that of C4.5 and CART. We see that the results are consistent in both cases. The data under investigation is temporal, hence using TIMERS with its temporal terminology is more appropriate. We also provide the results of 1DIMERS' sliding position flattening, and compare the results with those of TIMERS.

We will use two temporal datasets. The first one is from an artificial life program called URAL [21], and involves an artificial robot moving through a two-dimensional board. It can move to left, right, up and down. The goal is for us to discover the effects of moving, on the robot's position, expressed by a *x* and *y* pair. The board is 8 × 8 and there are 1000 observed records. This data comes from a controlled environment with no exceptions, and hence the rules are easy to learn. We consider the results of this test as a form of "sanity check" and have been using them as such in our papers. The second dataset is from a weather station in Louisiana. It includes 342 records of air temperature, the soil temperature, humidity, wind speed and direction and solar radiation, gathered hourly.

In the following tables, the values under "classification" represent the percentage of correct classifications done on training data (training accuracy) while the values for "regression" represent the error (mean square error). So higher values for classification are better, while lower values for regression are desired. We did not change the presentation to stay closer to the actual output of the programs we use.

4.2.1 The Artificial Robot

Each record in this dataset contains a *x* and *y* position value, the direction of movement at the time, and also a binary variable indicating the presence or absence of food. We set the decision attribute to be the current value of *x*, and the other three attributes are set as the condition attributes. There is no relationship between the current value of *x*, and the current values of *y*, direction of the movement, or the presence of food, so we predict that an instantaneous test (no flattening, or setting the window size to 1) will give poor results. Intuitively we know that the current value of *x* depends on the previous value of *x*, and the previous direction of movement. This temporal relationship makes us consider the relationship as a causal one. The acausal hypothesis says that you can tell where you were before if you know where you are now. This hypothesis is clearly wrong, as we could have ended at the current position from a different number of previous positions. Hence we do not expect to get good results with our acausality test. Results of using normal flattening are shown in Table 3, where the "Classification" column indicates the

.

percentage of correct classifications, while the "Regression" column (applicable only to CART) shows the mean square error.

Window	Normal Flattening	CART		C4.5
		Classification	Regression	Classification
1	N/A	24.6%	1.687	46.0%
2	Forward	100%	0	**100%**
	Backward	71.7%	0.469	70.6
3	Forward	100%	0	100%
	Backward	74.1%	0.435	71.7%
4	Forward	100%	0	100%
	Backward	76.2%	0.408	72.8%
5	Forward	100%	0	100%
	Backward	79%	0.378	74.5%

Table 3. CART and C4.5's results with the robot data

As shown in Table 3, CART and C4.5 behave similarly when provided with the same data. After the data have been flattened, the difference in results between the two programs diminishes significantly. The conclusion is the same in both cases: value of *x* is in a causal relation with other attributes, because a causality test provides better results than either the instantaneous or acausal tests. More specifically, the previous *x* and direction of movement causally determine the current value of *x*. This trend (100% accuracy for the causal test) is continued with window sizes higher than 5.

Using the sliding position flattening gives the results shown in Table 4, which are consistent with our expectations. With any position bigger than 1, the previous record, containing the relevant information for accurate prediction of current *x* value, is included in the flattened data. C4.5 discovers the correct temporal relation between the current value of *x* and the previous *x* and movement direction, and results are 100% accuracy with sliding positions of 2 or more.

In Tables 3 and 4 we see that there are slight differences between the results obtained with normal flattening in the acausal mode on one hand, and with sliding position flattening when the position is one, on the other hand. As shown in Table 1, the same information is provided to C4.5 in both cases, so one may expect the same results. However, the order of the attributes in the flattened records is different. This different ordering is evident in Table 1, and causes the results to vary.

.

Window	Position	Accuracy
2	1	70.6%
2	2	**100%**
3	1	71.5%
3	2	100%
3	3	100%
4	1	72.7%
4	2	100%
4	3	100%
4	4	100%
5	1	75.1%
5	2	100%
5	3	100%
5	4	100%
5	5	100%

Table 4. The results of using the sliding position flattening window to predict the value of *x*

4.2.2 The weather data

The subject of experiments in this subsection is a real-world dataset from weather observations in Louisiana [23], and hence interpreting the dependencies and relationships is harder. The main aim, however, is to compare CART and C4.5's results so as to evaluate TIMERS' consistency in giving a verdict based on the quality of the rules. We have set the soil temperature to be the decision attribute. The results obtained with normal flattening are shown in Table 5.

Window	Normal Flattening	CART		C4.5
		Classification	Regression	Classification
1	N/A	47.8%	1.375	27.7%
2	Forward	58.5%	441.48	82.78%
	Backward	60.5%	441.47	75.1%
3	Forward	78.0%	0.41	86.8%
	Backward	79.5%	0.47	**87.1%**
4	Forward	80.6%	0.37	84.4%
	Backward	80.3%	0.45	84.7%
5	Forward	64.0%	3.05	86.7%
	Backward	63.4%	470.65	82.9%

Table 5. CART and C4.5's results with Louisiana weather data.

The relationship between the soil temperature and other variables is not instantaneous, as observed by relatively poor results with a window of 1 (instantaneous test). The accuracy goes up after flattening, implying that there is a temporal relationship at work (the current value of the soil temperature has a close relationship with the previous values of the soil temperature, among others). TIMERS allows the user to use his domain knowledge when labelling a relationship, especially when the results are similar. In this case we decide to declare the relationship as acausal, because the accuracy values in the

.

two directions of time are not much different. With different time window values, CART displayed more variation than C4.5, but the user is still able to make a decision as to the acausal nature of the relationship.

The results of trying the same data with 1DIMERS' sliding position flattening, as implemented with TimeSleuth are shown in Table 6.

Window	Position	Accuracy
2	1	75.1%
2	2	82.7%
3	1	85.3%
3	2	82.4%
3	3	86.8%
4	1	85.3%
4	2	85.9%
4	3	83.2%
4	4	84.4%
5	1	85.0%
5	2	**87.0%**
5	3	85.0%
5	4	83.8%
5	5	86.7%

Table 6. Results of Sliding position flattening on the weather data.

All values in Table 6 are similar. 1DIMERS gives the verdict of "bidirectional predictive" for these results, which is in contrast to TIMERS' verdict of "acausal." TIMERS would lump both bidirectional and backward predictive verdicts under the verdict of acausal. TIMERS has less resolution in its verdicts, but given the temporal nature of the data, we would choose TIMERS' temporal verdict.

5. Spatial Data

For the experiments described in this section, we used data generated while drilling an oil well [22]. It includes observations about the characteristics of the rock being pierced, including the porosity of the rock (its capacity to hold oil) and different resistance values. The records were registered every 0.5 metres, between the depths of 7400 and 8907.5 metres. The decision attribute was set to be the porosity. TIMERS and 1DIMERS are both tried on this dataset.

To produce results with C4.5, we discretised the value of the porosity to 20 different values. In this case, CART was more effective with regression than classification. Classification took a much longer time to finish, and was done mainly for comparison with C4.5. Table 7 shows the results of the normal flattening methods.

.

Window	Normal Flattening	CART		C4.5
		Classification	Regression	Classification
1	N/A	35.9%	0.010	42.0%
2	Forward	40.4%	0.010	**45.3%**
	Backward	40.1%	0.009	44.0%
3	Forward	38.0%	0.009	41.0%
	Backward	38.4%	0.009	42.0%
4	Forward	37.5	0.009	42.8%
	Backward	31.5%	0.009	41.4%
5	Forward	36.9%	0.009	39.9%
	Backward	29.5%	0.009	39.3%

Table 7. CART and C4.5's results on drilling-sample data. TIMERS method

We get very similar results with the previous and next samples (backward and forward). TIMERS declares the relationship between the porosity and the condition variables to be instantaneous because, the instantaneous test gives about the same results as the temporal tests, and TIMERS gives precedence to being instantaneous. Hence the porosity at each point depends on the current values of the other variables at the same point. This result matched our expectations, because many of the fields in the data, including the resistance values, are related to porosity.
We also applied 1DIMERS, as implemented in TimeSleuth, to this data. The results are given in Table 8.

Window	Position	Accuracy
2	1	44.0%
2	2	45.3%
3	1	42.2%
3	2	**49.0%**
3	3	41.0%
4	1	43.1%
4	2	47.6%
4	3	48.2%
4	4	42.8%
5	1	38.9%
5	2	46.9%
5	3	46.4%
5	4	46.8%
5	5	39.9%

Table 8. C4.5's results with the 1DIMERS method.

According to the results in Table 8, and assuming that a 49.0% result is sufficiently better than 44.0%, the relationship among the porosity and the other variables is best described as linearly-extended, with a window size of 3 and a sliding position of 2. In other words, to predict the porosity at a given depth, the two neighbouring values, half a metre above and below, should be used.

.

Throughout Table 8 we get better results with a window position bigger than 1 and smaller than the window size, implying a definite neighbourhood relationship between porosity and the other attributes.

6. Concluding Remarks

We introduced 1DIMERS as a conceptual evolution of TIMERS for application on any one dimensional data, and gave the example of a dataset containing samples taken at regular intervals from an oil well. The similarities between a spatial line and temporal line made this generalisation intuitive. We also demonstrated that this method can be used with different underlying rule discoverers, and provide consistent results. CART was employed as an alternative to C4.5, and its ability to generate regression trees allowed us to work with datasets that C4.5 could not handle efficiently. TIMERS/1DIMERS can be implemented both at the rule level and at the tree level [10]. TimeSleuth is an example of a software package that implements the TIMERS and 1DIMERS methods at the rule level.

TimeSleuth, the program that partially implements TIMERS/1DIMERS, can be freely downloaded from http://www.cs.uregina.ca/~karimi/downloads.html. Its user interface employs a temporal/causal terminology.

References

1. Antunes, C. and Oliveira, A., Using Context-Free Grammars to Constrain Apriori-based Algorithms for Mining Temporal Association Rules, *Workshop on Temporal Data Mining (KDD2002)*, Edmonton, Canada. July, 2002.
2. Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth Inc., 1984.
3. Freedman, D. and Humphreys, P., *Are There Algorithms that Discover Causal Structure?*, Technical Report 514, Department of Statistics, University of California at Berkeley, 1998.
4. Grefenstette, J.J., Ramsey, C.L., Schultz, A.C, Learning Sequential Decision Rules Using Simulation Models and Competition, *Machine Learning 5(4)*, 1990, pp. 355-381.
5. Hawking, S.W., *The Universe in a Nutshell*, Bantam Books, 2001.
6. Lin, J, Keogh, E. and Truppel, W., Clustering of streaming time series is meaningless, The eighth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, San Diego, California, USA, 2003, pp. 56-65.
7. Karimi, K. and Hamilton, H.J., Finding Temporal Relations: Causal Bayesian Networks vs. C4.5, *The Twelfth International Symposium on Methodologies for Intelligent Systems (ISMIS'2000)*, Charlotte, NC, USA, October 2000, pp. 266-273.
8. Karimi, K. and Hamilton, H.J., Learning With C4.5 in a Situation Calculus Domain, *The Twentieth SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence (ES2000)*, Cambridge, UK, December 2000, pp. 73-85.
9. Karimi, K. and Hamilton, H.J., RFCT: An Association-Based Causality Miner, *The Fifteenth Canadian Conference on Artificial Intelligence (AI'2002)*, Calgary, Alberta, Canada, May 2002, pp. 334-338.
10. Karimi, K. and Hamilton, H.J., Temporal Rules and Temporal Decision Trees: A C4.5 Approach, *Technical Report CS-2001-02*, Department of Computer Science, University of Regina, Regina, Saskatchewan, Canada, December 2001.

.

11. Karimi, K. and Hamilton, H.J., Discovering Temporal Rules from Temporally Ordered Data, *The Third International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2002)*, Manchester, UK, August 2002, pp. 334-338.
12. Karimi, K., and Hamilton, H.J. TimeSleuth: A Tool for Discovering Causal and Temporal Rules, *The 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2002)*, Washington DC, November, 2002, pp. 375-380.
13. Karimi, K., and Hamilton, H.J., Distinguishing Causal and Acausal Temporal Relations, *The Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'2003)*, Seoul, South Korea, April/May 2003.
14. Karimi, K. and Hamilton H.J., Using TimeSleuth for Discovering Temporal/Causal Rules: A Comparison, *The Sixteenth Canadian Artificial Intelligence Conference (AI'2003)*, Halifax, Nova Scotia, Canada, June 2003.
15. Krener, A. J. Acausal Realization Theory, Part I; Linear Deterministic Systems. *SIAM Journal on Control and Optimization*. 1987. Vol 25, No 3, pp. 499-525.
16. Pearl, J., *Causality: Models, Reasoning, and Inference*, Cambridge University Press. 2000.
17. Quinlan, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.
18. Seabra Lopes, L. and Camarinha-Matos, L.M. Feature Transformation Strategies for a Robot Learning Problem, *Feature Extraction, Construction and Selection. A Data Mining Perspective*, H. Liu and H. Motoda (edrs.), Kluwer Academic Publishers, 1998.
19. Schwarz, R. J. and Friedland B., *Linear Systems*. McGraw-Hill, New York. 1965.
20 Sætrom, P. and Hetland, M.L., Unsupervised Temporal Rule Mining with Genetic Programming and Specialized Hardware, *International Conference on Machine Learning and Applications* (ICMLA'2003), 2003.
21. http://www.cs.uregina.ca/~karimi/downloads.html/URAL.java
22. http://explorer.ndic.state.nd.us/. Our data came from a sample CD.
23. http://typhoon.bae.lsu.edu/datatabl/current/sugcurrh.html. Contents change with time.

.