TimeSleuth: A Tool for Discovering Causal and Temporal 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

Discovering causal and temporal relations in a system is essential to understanding how it works, and to learning to control the behaviour of the system. TimeSleuth is a causality miner that uses association relations as the basis for the discovery of causal and temporal relations. It does so by introducing time into the observed data. TimeSleuth uses C4.5 as its association discoverer, and by using a series of pre-processing and post-processing techniques to enable the user to try different scenarios for mining causality. The data to be mined should originate sequentially from a single system. TimeSleuth's use of a standard decision tree builder such as C4.5 puts it outside the current mainstream method of discovering causality, which is based on conditional independencies and causal Bayesian Networks. This paper introduces TimeSleuth as a tool, and describes its functionality. TimeSleuth expands the abilities of C4.5 in some important ways. It is an unsupervised tool that can handle and interpret temporal data. It also helps the user in analyzing the relationships among the attributes by enabling him/her to see the rules, and statistics about them, in tabular form. There is also a mechanism to distinguish between causality and acausal relations. The user is thus encouraged to perform experiments and discover the nature of relationships among the data.

1. Introduction

Knowing the causal relations between the input and output of a system allows us to predict the behaviour of that system, and if any of the inputs can be manipulated, then we may be able to control the system so as to have a desired output. Causality mining has been a source of extensive (sometimes philosophical) debate [3, 9, 16]. The question of whether we should use the term "temporal rule" or use "causal rule" is thus subject to discussion. We assume that what we observe in a system is a series of attributes taking on different values over time. Thus we only observe associations among the values of the attributes. Whether or not a temporal association portrays a causal or acausal relationship is not always obvious, hence the preference of some people to use the term "temporal rule" instead of "causal rule." Some researchers claim that under certain conditions causal relations can be discovered reliably [12]. The tool introduced in this paper uses the direction of time to distinguish between causal and acausal relations, with causal relations assumed to flow forward in time, and acausal relations assumed to flow backward. We define a temporal relation to be either causal or acausal, as long as the attributes involved in the relation appear at different times.

Many different methods have been employed to discover temporal rules and patterns. An *event sequence* is a series of temporally ordered events, with either an ordinal time variable (which gives the order but not a real-valued time) or no time variable. Each event specifies the values for a set of attributes. A pattern in an event sequence is called a *frequent episode* [10]. Algorithms such as Dynamic Time Warping and its variants measure the similarity of patterns that are stretched differently over time [8]. These methods have not been applied to searching for relations in causal data.

Interpreting association relations as causal relations requires justification. The current trend in causality mining involves using the statistical concept of conditional independence as a measure of the control one attribute may have over another. For example, given the three attributes x, y, and z, if x and y are independent given z, that is, $P(x, y \mid z) = P(x \mid z)$, then we can conclude that x is not a direct cause of y, and y is not a direct cause of x. In other words, z separates x and y from each other. This basic concept is used in Bayesian Networks to build graphs that show the conditional dependence of the attributes under observation. This graph is then *interpreted* as signifying causal relations. One well-known example of a system working on this basis is TETRAD [14].

The notion of conditional independence is void of time. The proponents of this mainstream method use temporal relationships, if they are available, to place constraints on the relationships among the attributes (if we know x always appears before y, then y cannot be a cause of x), but time is not essential to the working of their algorithms. These statistical methods have been used in many studies [2, 15], especially in social sciences, where the notion of causality is subject to debate. We tried TETRAD on a very simple domain, with straightforward causal rules that were not subject to interpretation and debate, and were disappointed with the results [4, 5]. Leaving out time when dealing with the notion of causality seemed counterintuitive to us. The result was an investigation into the use of a standard tool, C4.5 [13], to discover associations given temporally ordered input data. We chose this tool because it is available in source code, and has been used widely in the literature.

C4.5 is a decision tree generator that also produces classification rules. It is a supervised tool that takes as input a series of attributes, called condition attributes, and outputs the predicted value for a single decision attribute. The user has to specify which attribute is to be considered as the decision attribute. This requirement makes the algorithm fast compared to the unsupervised methods, because only one attribute is considered instead of all possible relations among all the attributes. C4.5 assumes that all the attributes are observed at the same. To go from normal associations to temporal and causal relations, we introduce time into the input as a preprocessing step, run C4.5, and then adjust the output to obey the original temporal relations as a postprocessing step. We thus use time, and not any statistical method, to justify the interpretation of association relations as causal ones. We have seen that this method gives results that agree better with common sense than statistical methods [4, 5].

In the rest of the paper we present TimeSleuth, the tool we have developed for the discovery of temporal and causal rules. TimeSleuth is an enhanced version of our RFCT software [7]. TimeSleuth is written in Java and has a graphical user interface. It expands the abilities of C4.5 in several important ways. It is an unsupervised tool that can handle and interpret temporal data. It also helps the user in analyzing the relationships by enabling the user to see the rules, and statistics about them, in tabular forms. For a theoretical treatment of TimeSleuth refer to [6].

The rest of the paper is organized as follows. Section 2 explains the input to the system, and shows how we embed temporal data in the input to C4.5 by performing a preprocessing step. Section 2 also talks about the postprocessing step necessary to make sure that the user is presented with temporally valid output. Section 3 describes how TimeSleuth works, and shows how the user-interface helps in conducting different experiments and making sense out of the results. Section 4 concludes the paper.

2. Preprocessing and Post-Processing

In this paper we consider a system as a black box that takes a set of inputs and produces a set of outputs. We observe the input and the output of the system over time, and save the observations in records at regular intervals. Assuming that we are observing 5 attributes, x_1 to x_5. We thus end up with a temporally ordered series of records as shown in Figure 1, where for any $t > 1$ and $5 > n > 1$, x_{tn} is the nth observed attribute at time t.

$<x_{11}, x_{12}, x_{13}, x_{14}, x_{15}>$
$<x_{21}, x_{22}, x_{23}, x_{24}, x_{25}>$
$<x_{31}, x_{32}, x_{33}, x_{34}, x_{35}>$
$<x_{41}, x_{42}, x_{43}, x_{44}, x_{45}>$

Figure 1. Data produced by a system

The trend in data mining is usually to ignore the temporal order among the records. Providing these data to C4.5 would allow it to find relationships among the attributes that are observed at the same time, which are non-temporal associations among the values of attributes. However, the simple operation of merging the consecutive records with each other, called *flattening*, will provide C4.5 with more information as the current observations and the previous ones are seen together. This brings the possible causes and effects together. The *flattening time window* determines how many consecutive records are to be merged together. The window size, determined by the user, shows for how long the past can affect the present. Guessing this number may not be easy, but as we will see later, TimeSleuth can explore a range of window sizes to choose a suitable one. As an example, Figure 2 shows the records of Figure 1 flattened with a time window of $w = 2$.

$<x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}>$ time 1:causes → time 2:effects
$<x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{31}, x_{32}, x_{33}, x_{34}, x_{35}>$ time 2:causes → time 3:effects
$<x_{31}, x_{32}, x_{33}, x_{34}, x_{35}, x_{41}, x_{42}, x_{43}, x_{44}, x_{45}>$ time 3:causes → time 4:effects

Figure 2. Records in Figure 1, flattened with $w = 2$.

With the exception of the first and the last records, each record appears twice, once as the "cause" part, and once as the "effect" part. The exceptions are because we cannot observe any causes for the first record or any effects for the last record. Generally, with a time window of w, each record appears in w different flattened records (with the exception of the first and last w records). The "effects" are assumed to exist only in the last time step. Thus with a time window of w, we are assuming that the previous w-1 records contains the causes for the values observed in the wth record.

The preprocessing phase looses the temporal information, because the merged records no longer contain the original order. A non-temporal tool such as C4.5 would thus treat all the attributes as if they were seen at the same time. Thus in the output an attribute from the past may appear ahead of an attribute from the future. To make sure that temporal order is respected in the output, TimeSleuth re-orders the attributes as needed. This postprocessing step is meant to recover the lost information. In particular, the rule generation part (c4.5rules) reorders the clauses of a rule to make sure that the attributes are encountered in temporal order. For example, if c4.5rules generates a rule such as: IF $\{(x_{11} = 1)$ AND $(x_{31} = 5)$ AND $(x_{24} = 1)\}$ THEN x_{35} = true, we transform it to the following: IF $\{(x_{11} = 1)$ AND $(x_{24} = 1)$ AND $(x_{31} = 5)\}$ THEN x_{35} = true.

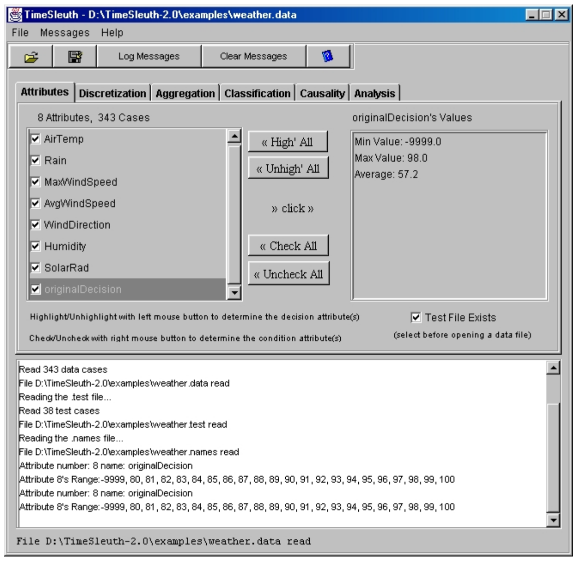
This modification to produce temporal rules does not require any changes in C4.5's algorithms in the decision tree generator or in the rule generator. Another way to ensure that temporal order is respected is to change the tree generation algorithm in such a way that the decision tree is temporally sorted, as explained in [7].

3. TimeSleuth: How it Works

In this section we introduce TimeSleuth as a tool. [4, 5] provide reports on the experiments performed with this tool, and compare it with other causality miners, notably TETRAD. TimeSleuth is an application program written in Java, and thus usable anywhere a Java interpreter is available. However, it needs two of the C4.5 package's executable files, c4.5 and c4.5rules. TimeSleuth is a causal rule discoverer, and mainly ignores the decision trees created by C4.5 in the process of generating rules. Its functionality is divided between the TimeSleuth application and C4.5. This requires certain modification to be done in both c4.5 and c4.5rules sources, and then they should be compiled. After this c4.5 and c4.5rules understand new command line options, which are used by TimeSleuth to communicate with them. C4.5 is available publicly in source form, and standard patch files to make it compatible to TimeSleuth are included with the TimeSleuth package. However, TimeSleuth can be used, with reduced abilities, even if C4.5 has not been patched and recompiled. An option in TimeSleuth allows the user to inform it of this situation, so it will not provide the non-patched C4.5 programs with options they cannot understand.

The TimeSleuth application provides the user with a Graphical User Interface (GUI). Its input is the same as that of standard C4.5, which is a set of two files: A file with a .names extension contains the names of the condition attributes, while the decision attribute remains nameless. TimeSleuth calls the decision attribute simply "originalDecision." The second file, with a .data extension, contains the observations of the condition and decision attributes. These are called "cases," and take the form of rows of data that correspond in order to the contents of the .names file. The value of the decision attribute always comes in the last column. To measure the predictive accuracy of the resulting decision tree or rules, the user can provide a .test files, containing unseen cases. A .test file has the same format as a .data file. After C4.5 is done, it evaluates the tree or the rules, based on the contents of the .test file, and reports the results. All three files (names, data, test) should have the same name, and their extension determines their type.

Figure 3 shows a snapshot of TimeSleuth's input handling panel. After reading the input files, a list of the discovered attributes is presented to the user. The user selects a decision attribute by highlighting it. The user should inform TimeSleuth about the presence of a .test file by clicking on the appropriate checkbox.


TimeSleuth - D:\TimeSleuth-2.0\examples\weather.data

8 Attributes, 343 Cases

- [✓] AirTemp
- [✓] Rain
- [✓] MaxWindSpeed
- [✓] AvgWindSpeed
- [✓] WindDirection
- [✓] Humidity
- [✓] SolarRad
- [✓] originalDecision

originalDecision's Values

Min Value: -9999.0
Max Value: 98.0
Average: 57.2

Highlight/Unhighlight with left mouse button to determine the decision attribute(s)

Check/Uncheck with right mouse button to determine the condition attribute(s)

[✓] Test File Exists
(select before opening a data file)

```
Read 343 data cases
File D:\TimeSleuth-2.0\examples\weather.data read
Reading the .test file...
Read 38 test cases
File D:\TimeSleuth-2.0\examples\weather.test read
Reading the .names file...
File D:\TimeSleuth-2.0\examples\weather.names read
Attribute number: 8 name: originalDecision
Attribute 8's Range:-9999, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100
Attribute number: 8 name: originalDecision
Attribute 8's Range:-9999, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100
```

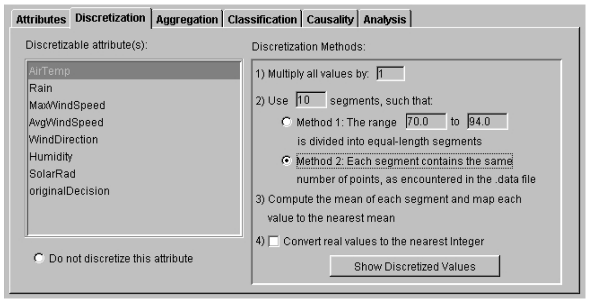
File D:\TimeSleuth-2.0\examples\weather.data read

Figure 3. TimeSleuth 's input handling panel.

The data used in Figure 3 and the following figures contains weather observations gathered hourly. The decision attribute, "originalDecision" in TimeSleuth, is the Soil Temperature.

Unlike with standard C4.5, in TimeSleuth the user can choose more than one decision attribute. In such cases, C4.5 is invoked multiple times, each time with a different attributed as the decision attribute. TimeSleuth automatically generates the appropriate .data, .test, and .names files.

The user can choose to discretize the attributes using the "Discretization" panel in Figure 4. Two methods are available. In Method one, the attribute's range of values, as present in the input data file, is divided into segments of equal length. Method 2 takes into consideration the distribution of the values, and divides the range into segments that contain the same number of values. This ensures the values in a denser region of the attribute's range are preserved. The steps taken in the discretization process are shown in Figure 4.



Discretizable attribute(s):

AirTemp
Rain
MaxWindSpeed
AvgWindSpeed
WindDirection
Humidity
SolarRad
originalDecision

Do not discretize this attribute

Discretization Methods:

1) Multiply all values by: 1

2) Use 10 segments, such that:

Method 1: The range 70.0 to 94.0 is divided into equal-length segments

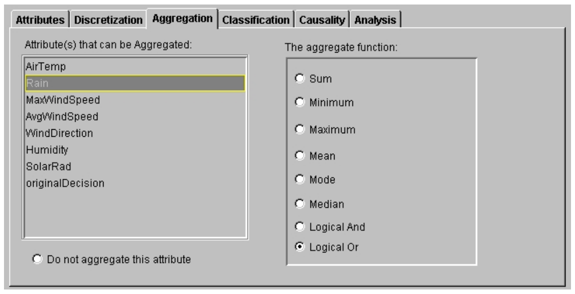
[✓ Method 2: Each segment contains the same number of points, as encountered in the .data file]

3) Compute the mean of each segment and map each value to the nearest mean

4) ☐ Convert real values to the nearest Integer

Figure 4. The Discretization panel

The main assumption in TimeSleuth is that the program is provided with the values that are snapshots: They are observed at different time steps. However, the user can instruct TimeSleuth to use the aggregate value of an attribute in forming its rules. For example, the user may decide to investigate the effect of the minimum value of the Air Temperature during a certain time window, as shown in Figure 5. In such a case, TimeSleuth computes the minimum, and uses its value instead of individual temperature values. Going back to the example of Figure 2, choosing the aggregate function min() on x_1 would result in the record: $< x_{12}, x_{13}, x_{14}, x_{15}, x_{22}, x_{23}, x_{24}, x_{25}, \min(x_{11}, x_{11})>$. In the actual output file, the last attribute will be the decision attribute because that is the format expected by C4.5. The effect of having aggregate attributes is explained below, when we introduce the classification panel.



Attribute(s) that can be Aggregated:

AirTemp
Rain
MaxWindSpeed
AvgWindSpeed
WindDirection
Humidity
SolarRad
originalDecision

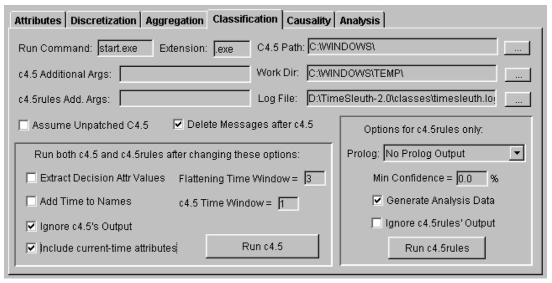
Do not aggregate this attribute

The aggregate function:

Sum
Minimum
Maximum
Mean
Mode
Median
Logical And
[✓ Logical Or]

Figure 5. The Aggregation panel

The actual running of C4.5 happens in the Classification panel, as shown in Figure 6. It allows the user to specify the location of C4.5's executable files and other related directories. The user can also provide any optional run-time arguments to C4.5, even though the arguments needed for TimeSleuth's functioning will be automatically provided. Here the user can inform TimeSleuth that C4.5 has not been patched. The default values provided in this panel make sure that C4.5 will function in a backward compatible way.



Run Command: start.exe
Extension: .exe
C4.5 Path: C:\WINDOWS\
c4.5 Additional Args:
Work Dir: C:\WINDOWS\TEMP\
c4.5rules Add. Args:
Log File: D:\TimeSleuth-2.0\classes\timesleuth.lo

☐ Assume Unpatched C4.5
☑ Delete Messages after c4.5

Run both c4.5 and c4.5rules after changing these options:

☐ Extract Decision Attr Values
Flattening Time Window = 3
☐ Add Time to Names
c4.5 Time Window = 1
☑ Ignore c4.5's Output
☑ Include current-time attributes

Options for c4.5rules only:

Prolog: No Prolog Output
Min Confidence = 0.0 %
☑ Generate Analysis Data
☐ Ignore c4.5rules' Output

Figure 6. The Classification panel.

If C4.5 is not patched, TimeSleuth can still be used to discover causal relations, because it can perform the flattening operation with no need for C4.5 to be patched. However, there is a potential problem here: C4.5 does not rely on the names of the attributes to identify them. Rather, it uses their locations in the .data file. If TimeSleuth simply flattens the data and copies the names in the .names file multiple times, then in the output the attribute's names from different time steps would be confused. The x_{11}, x_{21}, x_{31}, etc. from Section 3 are not actually differentiated in a .names file, and all are called x_1 by C4.5. If c4.5rules has been patched, it actually outputs temporal information in the rules. So the rule from section 3 looks like this:

IF {At Time 1: ($x_1 = 1$) AND At Time 2: ($x_4 = 1$) AND AT Time 3: ($x_1 = 5$)} THEN At Time 3: x_5 = true

This outlines the way the patched c4.5rules outputs its results. As seen later, TimeSleuth uses a tabular form to display the same information. If c4.5rules is unpatched, then the user can instruct TimeSleuth to add a time index (_t<time>) to the names it generated. So c4.5rules' output would be temporally out of order, but still understandable because the attributes from different time steps are distinguishable. An example is the following rule: IF {(x_1_t1 = 1) AND (x_1_t3 = 5) AND (x_4_t2 = 1)} THEN x_5_t3 = true. TimeSleuth can output these rules as Prolog statements, making them machine executable.

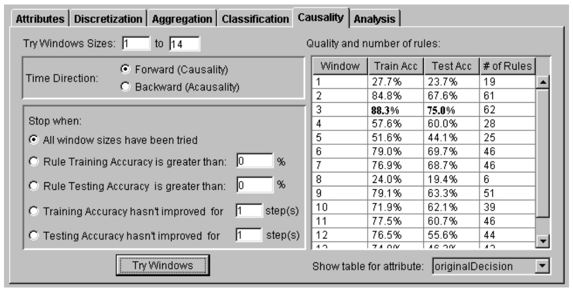
If aggregate attributes are present, then the output rules will include the keyword "During Time Window" to make it clear that the aggregate value of the specified attribute is seen during the whole window. A rule would look like: IF {During Time Window: $x_3 >= 0$ AND At Time 1: ($x_1 = 1$) AND At Time 2: ($x_4 = 1$) AND AT Time 3: ($x_1 = 5$)} THEN At Time 3: x_5 = true

C4.5 is supervised, because the user has to indicate the decision attribute, and also has to tell C4.5 exactly which values will be taken on by the decision attribute. TimeSleuth allows the user to specify more than one decision attribute, and can optionally extract the values taken on by the decision attribute from the .data file. It outputs a warning message if the .test file contains a value that is not seen in the .data file. These abilities turn

TimeSleuth into an unsupervised tool, as the user simply has to run the program with minimal instructions as to the target attribute and the values it can have.

There are two time windows in the classification panel. The first one, called "Flattening Time Window" is as the name implies. Its value is used to flatten the input data for both c4.5 and c4.5rules. This value is then provided to c4.5rules so that it can sort the output temporally. However, this value is not provided to the tree generation program, c4.5. So the decision tree is generated with no regard to any time window. The value "c4.5Time Window" is meant to affect the decision tree, as explained in [7]. This value, if different than 1, should be the same as the Flattening time window. The rules generated by c45rules have a confidence level. In order to filter the rules, a user can specify a minimum confidence level, and only rules will higher confidence values will be presented.

Finding a suitable window size for the data under investigation can be a challenge. For this reason TimeSleuth allows the user to run it in a batch mode, wherein it tries consecutive values for the time window. Training and testing accuracy can be employed to guide the search. As shown in Figure 7, the user can decide to explore all values, or stop after the accuracy has reached a threshold, or after the accuracy has stopped improving. The running time is determined by C4.5's speed, which in our experiments has been good, even for fairly large values of the window size.



Try Windows Sizes: 1 to 14

Time Direction: [✓ Forward (Causality)] Backward (Acausality)

Stop when:
[✓ All window sizes have been tried]
Rule Training Accuracy is greater than: 0 %
Rule Testing Accuracy is greater than: 0 %
Training Accuracy hasn't improved for 1 step(s)
Testing Accuracy hasn't improved for 1 step(s)

Quality and number of rules:

Window	Train Acc	Test Acc	# of Rules
1	27.7%	23.7%	19
2	84.8%	67.6%	61
3	**88.3%**	**75.0%**	62
4	57.6%	60.0%	28
5	51.6%	44.1%	25
6	79.0%	69.7%	46
7	76.9%	68.7%	46
8	24.0%	19.4%	6
9	79.1%	63.3%	51
10	71.9%	62.1%	39
11	77.5%	60.7%	46
12	76.5%	55.6%	44

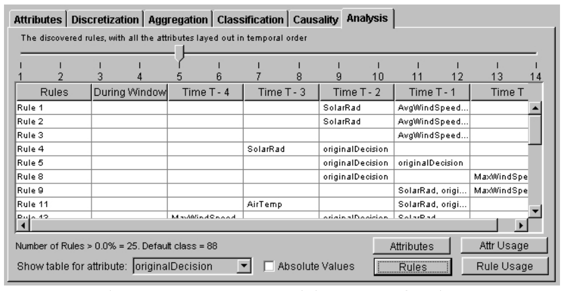
Show table for attribute: originalDecision

Figure 7. Exploring different window sizes.

The user can perform the acausality test by choosing the backward flow of time. If the results are better than in the forward flow, then the relationship is acausal.

TimeSleuth's user interface provides help in making sense of the rules and relations among different attributes in the rules. Using the Analysis panel the user can instruct c4.5rules to output data necessary for tabular display of rules and the relevant statistics. This changes C4.5's normal output from text-based to tabular.

In the analysis panel, the user can see how the selected time window has affected the resulting rules. As shown in Figure 8, the user can opt to see the rules laid out according to the time steps in which the attributes appear. This display shows how important each attribute has been in forming the rules.



The discovered rules, with all the attributes layed out in temporal order

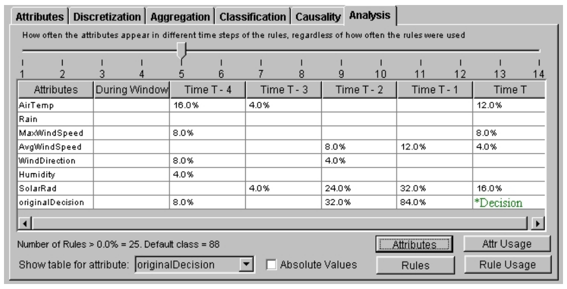
Rules	During Window	Time T - 4	Time T - 3	Time T - 2	Time T - 1	Time T
Rule 1				SolarRad	AvgWindSpeed...	
Rule 2				SolarRad	AvgWindSpeed...	
Rule 3					AvgWindSpeed...	
Rule 4			SolarRad	originalDecision		
Rule 5				originalDecision	originalDecision	
Rule 8				originalDecision		MaxWindSpe
Rule 9					SolarRad, origi...	MaxWindSpe
Rule 11			AirTemp		SolarRad, origi...	

Number of Rules > 0.0% = 25. Default class = 88

Show table for attribute: originalDecision Absolute Values

Figure 8. Temporal layout of rules.

In Figure 9, many of the condition attributes used to determine the value of the soil temperature (originalDecision) come from previous time steps. In other words, the current temperature of the soil depends on attributes measured previously. Using standard C4.5 with such data obviously would not be as revealing. As seen in Figure 9, the previous value of soil temperature appears in 84.3% of the rules, which supports the common sense guess that the current temperature is determined mostly by the corresponding observation an hour ago.



How often the attributes appear in different time steps of the rules, regardless of how often the rules were used

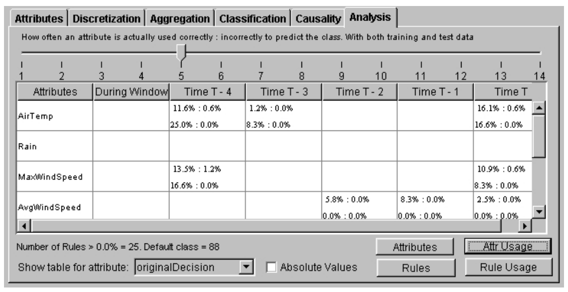
Attributes	During Window	Time T - 4	Time T - 3	Time T - 2	Time T - 1	Time T
AirTemp		16.0%	4.0%			12.0%
Rain						
MaxWindSpeed		8.0%				8.0%
AvgWindSpeed				8.0%	12.0%	4.0%
WindDirection		8.0%		4.0%		
Humidity		4.0%				
SolarRad			4.0%	24.0%	32.0%	16.0%
originalDecision		8.0%		32.0%	84.0%	*Decision

Number of Rules > 0.0% = 25. Default class = 88

Show table for attribute: originalDecision Absolute Values

Figure 9. Statistics about the attributes

In Figure 10, TimeSleuth shows the frequency of attribute usage in rules that were actually fired. In other words, the more a rule has been used (on test data or on training data), the more important those attributes will be. Both training and testing results are displayed. The two values are separated by ":".



How often an attribute is actually used correctly : incorrectly to predict the class. With both training and test data

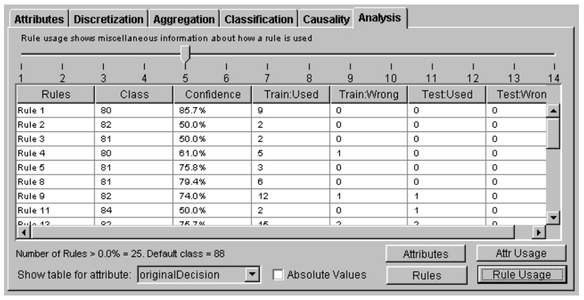
Attributes	During Window	Time T - 4	Time T - 3	Time T - 2	Time T - 1	Time T
AirTemp		11.6% : 0.6% 25.0% : 0.0%	1.2% : 0.0% 8.3% : 0.0%			16.1% : 0.6% 16.6% : 0.0%
Rain						
MaxWindSpeed		13.5% : 1.2% 16.6% : 0.0%				10.9% : 0.6% 8.3% : 0.0%
AvgWindSpeed				5.8% : 0.0% 0.0% : 0.0%	8.3% : 0.0% 0.0% : 0.0%	2.5% : 0.0% 0.0% : 0.0%

Number of Rules > 0.0% = 25. Default class = 88

Show table for attribute: originalDecision Absolute Values

Figure 10. Frequency of attribute usage in rules

Figure 11 shows some addition information about the rules and how they were used. The column headers are self-explanatory.



Rule usage shows miscellaneous information about how a rule is used

Rules	Class	Confidence	Train:Used	Train:Wrong	Test:Used	Test:Wron
Rule 1	80	85.7%	9	0	0	0
Rule 2	82	50.0%	2	0	0	0
Rule 3	81	50.0%	2	0	0	0
Rule 4	80	61.0%	5	1	0	0
Rule 5	81	75.8%	3	0	0	0
Rule 8	81	79.4%	6	0	0	0
Rule 9	82	74.0%	12	1	1	0
Rule 11	84	50.0%	2	0	1	0

Number of Rules > 0.0% = 25. Default class = 88

Show table for attribute: originalDecision

☐ Absolute Values

Figure 11. Rule usage and other related data

In TimeSleuth, to determine the quality of a set of rules and the strength of the relations among the attributes that is implied by the rules, we use the training or testing accuracy of the set of rule.

4. Concluding Remarks

We introduced TimeSleuth, an unsupervised, learning tool based on C4.5 that is targeted for discovering causal and temporal relations. We saw that discovering causal relationships using associations is the norm in the literature. TimeSleuth does the same, but instead of using statistical methods it employs the common sense notion of temporal order to go from association relationships to temporal ones. The option to set the flow of time backward or forward allows the user to determine the causality or acausality of the rules. Often the exact nature of a temporal relationship is not known. TimeSleuth helps the user to experiment with different scenarios and see what kind of rules are discovered when different time window values are used. The tabular output of information about the rules and the attributes helps the user assimilate the discovered relations.

TimeSleuth helps the user to analyze the rules produced by C4.5 by showing how important each attribute is at different times in determining the value of the decision attribute. This ability makes TimeSleuth a data mining tool (with output suitable for a domain expert) as well as a machine learning tool (with output consisting of ready-to-execute rules).

TimeSleuth is freely available for download from http://www.cs.uregina.ca/~karimi/downloads.html or by contacting the authors. The package includes on-line help, example files (including the weather data used in figures in this paper), and the patch files for C4.5. Also available on the same web address are c4.5.exe and c4.5rules.exe, patched and compiled for Microsoft Windows 95 and up.

References

[1] Berndt, D. J. and Clifford, J., Finding Patterns in Time Series: A Dynamic Programming Approach, *Advances in Knowledge Discovery and Data Mining*. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, et al. (eds.), AAAI Press/ MIT Press, pp. 229-248, 1996.

[2] Bowes, J., Neufeld, E., Greer, J. E. and Cooke, J., A Comparison of Association Rule Discovery and Bayesian Network Causal Inference Algorithms to Discover Relationships in Discrete Data, *Proceedings of the Thirteenth Canadian Artificial Intelligence Conference (AI'2000)*, Montreal, Canada, 2000, pp. 326-336.

[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] 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, October 2000.

[5] 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.

[6] 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. 25-30.

[7] 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.

[8] Keogh, E. J. and Pazzani, M. J., Scaling up Dynamic Time Warping for Data Mining Applications, *The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000)*, August 2000, pp. 285-289.

[9] Korb, K. B. and Wallace, C. S., In Search of Philosopher's Stone: Remarks on Humphreys and Freedman's Critique of Causal Discovery, *British Journal of the Philosophy of Science 48*, pp. 543-553, 1997.

[10] Mannila, H., Toivonen, H. and Verkamo, A. I., Discovering Frequent Episodes in Sequences, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp. 210-215, 1995.

[11] Nadel, B.A., Constraint Satisfaction Algorithms, *Computational Intelligence*, 5, pp. 188-224, 1989.

[12] Pearl, J., *Causality: Models, Reasoning, and Inference*, Cambridge University Press. 2000.

[13] Quinlan, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[14] Scheines, R., Spirtes, P., Glymour, C. and Meek, C., *Tetrad II: Tools for Causal Modeling*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.

[15] Silverstein, C., Brin, S., Motwani, R. and Ullman, J., Scalable Techniques for Mining Causal Structures, *Proceedings of the 24th VLDB Conference*, pp. 594-605, New York, 1998.

[16] Spirtes, P. and Scheines, R., Reply to Freedman, In McKim, V. and Turner, S. (editors), *Causality in Crisis*, University of Notre Dame Press, pp. 163-176, 1997.