RFCT: An Association-Based Causality Miner

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 relations in a system is essential to understanding how it works and to learning how to control the behaviour of the system. RFCT is a causality miner that uses association relations as the basis for the discovery of causal relations. It does so by making explicit the temporal relationships among the data. RFCT uses C4.5 as its association discoverer, and by using a series of pre-processing and post-processing techniques enables the user to try different scenarios for mining causality. The raw data to be mined should originate from a single system over time. RFCT 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 variables by enabling him/her to see the rules, and statistics about them, in tabular form. The user is thus encouraged to perform experiments and discover any causal or temporal 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. If any of the inputs can be manipulated, then we may be able to control the system to obtain a desired output. Causality mining is the search for causal relations in data. It has been a source of extensive (sometimes philosophical) debate [1, 5, 9]. The question of whether we should use the term "temporal rule" or use "causal rule" is subject to discussion. What we observe in a system is a series of variables taking on different values. Thus we only observe associations among the values of the variables. Whether or not this temporal association portrays a causal relationship is not always obvious, hence the preference of some researchers to use the term "temporal rule." Other researchers claim that under certain conditions causal relations can be discovered reliably [7]. In this paper we avoid entering this debate and use the term "causal" in much the same sense as "temporal." Not distinguishing between causal and temporal is justified by the fact that the tool introduced in this paper is separate from the system under investigation, and it does not assume that the user can perform interventions in the system

to see how purposefully changing certain variables affects the others. We thus leave the determination of the nature of the relationship (causal or merely temporal) to the domain expert.

C4.5 [8] is a standard, widely used and available decision tree generator that also produces classification rules. It is a supervised tool that takes as input a series of variables, called condition variables, and outputs a tree or a set of rules to predict the value of a single decision variable. The user has to specify which variable is to be considered as the decision variable. This requirement makes the algorithm fast compared to the unsupervised methods. C4.5 does not distinguish the passage of time among the instances, so to go from associations to temporal and causal relations, we make the temporal relationship in the data more explicit as a preprocessing step, run C4.5, and then adjust the output to obey the original temporal relations. We thus use temporal order to justify the interpretation of association relations as causal ones. We have seen that this leads to results that agree with common sense [2, 3].

In the rest of the paper we present RFCT (**R**otated **F**lattening for **C**4.5 with enforced **T**emporal ordering), the tool we have developed for the discovery of temporal and causal rules. RFCT is written in Java and has a graphical user interface. It expands the abilities of C4.5 in many ways. Briefly, 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. Section 2 discusses the approach, and Section 3 concludes the paper.

2. The RFCT Approach

In this section, we describe the RFCT approach. First, we describe the preprocessing done to enable C4.5 to discover temporal relations. Input is a series of temporally ordered instances of m values. We then present the RFCT algorithm, and discuss the postprocessing that ensures that the user is presented with temporally valid output.

In the preprocessing stage, RFCT merges two or more consecutive records together, increasing the amount of information in each record. This operation is called flattening [2]. For example, suppose we have two records: $<x_{11}, x_{12}, x_{13}, x_{14}, x_{15}>$ observed at time 1, and $<x_{21}, x_{22}, x_{23}, x_{24}, x_{25}>$ observed at time 2. Flattening them with a window of size 2 merges them into the record $<x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}>$. In this flattened form it is easier to discover temporal and causal relations where the values at one time step have an influence on the values seen at the next time step. The number of records merged is determined by the time window.

The RFCT algorithm is shown in Figure 1. Since C4.5 expects the last value in each record to represent the decision attribute, a rotation of the fields is performed as necessary to ensure that the proper attribute is set as the decision variable. The $F_w(t,\boldsymbol{D})$ is the flattening operator, which given the unflattened input $\boldsymbol{D}$, the time window w, and the

current time t in the input, returns a flattened record made of w consecutive records, starting at index t, in $\boldsymbol{D}$. The resulting flattened records are then made available to C4.5.

```
Algorithm RFCT.
Input:  a set of m variables V; a data set D consisting of T records with m values; and
        a window size w.
Output: a set of decision rules R.
for j = 1 to m
    D' = rotate the values in each record d ∈ D such that value in v_j is the last
    member of d.
    Flat = the sequence of values yielded by F_w(t,D'),  for all t, w ≤ t ≤ T.
    R = R ∪ C4.5T(Flat)
end for
return R
```

Fig. 1. The RFCT Algorithm

Flattening causes a loss of temporal information, because every w consecutive records in the original input merge in the same flattened record, and are no longer distinguished by their temporal order. A non-temporal tool such as C4.5 treats all variables as if they were observed at the same time. Thus in the output a variable from the future may appear before a variable from the past. To make sure that any temporal order is respected in the output, RFCT uses a preprocessing step to re-order the variables as needed. This postprocessing step is meant to make the temporal order in the data explicit again.

C4.5 is a decision tree builder, and the classification rules it outputs are derived from the decision tree. So there are two distinct programs that the user should run in the C4.5 package. The first one, c4.5, is the decision tree builder, while the c4.5rules derives rules from the decision tree created by c4.5. Standard C4.5 uses a greedy algorithm with one step look-ahead to expand the tree, and all condition variables are supposed to be available for choosing. With flattened data this creates problems both in the tree and in the derived rules because C4.5 may employ the variables out of their temporal order. This can create problems especially when the tree is being traversed in real time, where the records keep coming and decisions have to be made immediately before seeing the next record. To make sure that the decision tree respects the temporal constraints, we have modified C4.5's tree building algorithm. Whereas previously C4.5's only consideration was to build a more compact tree with the least amount of error, now an additional constraint forces the algorithm to choose variables in the correct temporal order. As an example, if the algorithm has chosen a variable from time step n, then from that branch in the tree it has to choose variables that appear in time step m, $m \geq n$. This is done even if there are variables that would result in a tree with less error, but appeared in a time step previous to n. As a side effect, decision trees may be of lower quality [4]. Figure 2 shows two example trees. We are assuming that each of T_i $1 \leq i \leq 3$ contains variables from time step i. At left we see a tree created normally (that violates temporal order), while at right the temporal order is respected.

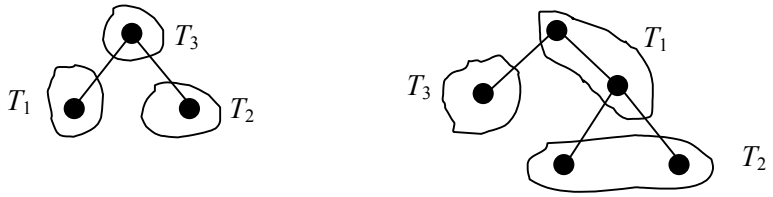

Fig. 2. Left: A tree that violates temporal order, and Right: a tree that respects temporal order.

Temporal order in different branches is considered independent. This makes the root the most important. If it is chosen from a later time step, nothing from the previous time steps may be used, restricting the algorithm's ability to construct a tree. The current version of RFCT stops at this stage, but to improve the process of building such temporal trees, one can turn the problem into a Constraint Satisfaction Problem [6] where the choice of one variable limits the future choices in that branch of the tree. As is normal in constraint satisfaction problems, we may need to backtrack up the tree branches and choose other condition variables [4].

The other way to make sure temporal orders are respected is to leave C4.5 to create the decision trees as usual, but to rely on the rule generation part (c4.5rules) to "sort" the variables in the output. For example, if c4.5rules generate a rule such as: If $\{(x_{11} = 1)$ AND $(x_{31} = 5)$ AND $(x_{24} = 1)\}$ then x_{35} = true, We could modify it to look like this: 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, whether in the decision tree generator or in the rule generator.

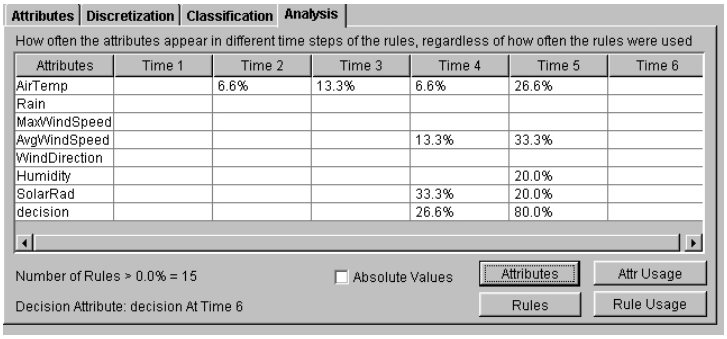

Fig. 3. How often each attribute appears in the rules

Unlike with standard C4.5, in RFCT the user can choose more than one decision attribute. In such cases, C4.5 is invoked multiple times, each time with one selected attribute as the decision attribute. RFCT's graphical user interface helps in interpreting the results by using tables that show information temporally. The rules generated by c45rules have a confidence level. To filter the rules, the user can specify a minimum confidence level, and as a result only rules will higher values will be presented. The user can see how the selected time window has affected the results by having the rules laid out according to the time step in which each variable appears. The user can also see how important each variable has been in forming the rules, as in Figure 3. RFCT also 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 variables will be. RFCT's built-in discretization ability allows it to better handle non-continuous attributes.

3. Concluding Remarks

RFCT is an unsupervised tool based on C4.5 that is targeted at discovering causal and temporal rules. It relies on associations, but instead of statistical methods such as conditional independence, it assumes temporally ordered observations, and finds temporally ordered rules based on that assumption. RFCT allows the user to experiment with different scenarios and see what kinds of rules are discovered when different window values are used. The tabular output of information about the rules and the variables helps the user assimilate the discovered relations.

In theory, the RFCT approach can be used with any association discoverer, not just C4.5, so it is possible to replace C4.5 with another tool if its source code is available. The RFCT package includes sources, executable files, on-line help, and C4.5 patch files. It is available for download from http://www.cs.uregina.ca/~karimi/downloads.html.

References

1. Freedman, D. and Humphreys, P., *Are There Algorithms that Discover Causal Structure?*, Technical Report 514, Department of Statistics, University of California at Berkeley, 1998.
2. 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.
3. 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.
4. Karimi, L. 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.

5. 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.
6. Nadel, B.A., Constraint Satisfaction Algorithms, *Computational Intelligence*, No. 5, 1989.
7. Pearl, J., *Causality: Models, Reasoning, and Inference*, Cambridge University Press. 2000.
8. Quinlan, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.
9. 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.