Using Dependence Diagrams to Summarize Decision Rule Sets

Kamran Karimi

Department of Software Engineering
Lakehead University
Thunder Bay, Ontario
Canada, P7B 5E1
kkarimi@lakeheadu.ca

Howard J. Hamilton

Department of Computer Science
University of Regina
Regina, Saskatchewan
Canada, S4S 0A2
hamilton@cs.uregina.ca

Abstract. Generating decision rule sets from observational data is an established branch of machine learning. Although such rules may be well-suited to machine execution, a human being may have problems interpreting them. Making inferences about the dependencies of a number of attributes on each other by looking at the rules is hard, hence the need to summarize and visualize a rule set. In this paper we propose using dependence diagrams as a means of illustrating the amount of influence each attribute has on others. Such information is useful in both causal and non-causal contexts. We provide examples of dependence diagrams using rules extracted from two datasets.

1 Introduction

In a system exposed as a number of variables that change value over time, exploring the possible influence of some variables on others is important. Dependence of one variable on a second variable may denote an association or a causal relation. In the case of an association, we can expect a change in the value of the first variable when we observe a change in the second one, though we cannot control this process. In the causal case, we can possibly control the value of the first variable by changing the value of the second variable.

Much work has been done in extracting sets of rules that apply to the exposed variables of a system. The rules in such rule sets may be numerous and complicated, making it hard for a human to understand the dependencies represented by these rules. In this paper we introduce dependence diagrams as a way to help a user visually summarize a rule set by displaying the variables' influence on each other.

In a rule, such as {if x_1 = 5 and a_1 = Move_Right, then x_2 = 6} (Rule 1), the variables x_1 and a_1 are called the *condition attributes*, while the variable x_2, whose value is being determined, is called the *decision attribute*. In this rule, x_1 and a_1 are used to predict the value of x_2. The *accuracy* of such a rule denotes how often the prediction is correct when applied to test data. Such a rule specifies a classification, and this format is widely used to represent decision rules in software such as C4.5 [9].

Existing representations and tools used for analyzing rule sets are inadequate to give a user a quick sense of the dependencies among variables. Decision tables [6] represent a decision space in textual form, but they do not provide visualization or support for summarizing multiple rules or rule sets. Decision trees [9] can be displayed in a visual form helpful to understanding the influences of the condition attributes on a single decision attribute, but they do not show the complex interdependencies among variables when several can be treated as decision attributes. Decision trees can also be very big, making it hard to assess the importance of a given attribute in the decision making process.

Software tools such as MineSet [2] and CART [1] enable the user to see decision rules in a variety of formats, but the summarization property is lacking. The General Logic Diagram (GLD) [7] uses a two-dimensional grid to represent a multi-dimensional decision space. It is similar to a Karnaugh map, but it can represent multi-valued discrete attributes. Although this representation is effective for visualizing all the rules, it is not well suited to representing the amount of influence of the attributes. A new representation is required that allows us to see the inter-relation and influence among the different attributes, and at the same time is independent of the number of generated rules. Focusing on summarization, such a representation should not depend on individual rules. Its complexity and size should depend on the number of variables rather than the number of the rules.

Informally, in the context of a given number of attributes and a set of decision rules generated based on those attributes, a *dependence diagram* is a graphic representation that shows the amount of influence of the condition attributes on the decision attribute. When we change the decision attribute and generate a new set of decision rules, the same dependence diagram can be used to show the results. So a dependence diagram can describe multiple rule sets, each with a different decision attribute, as long the same attributes are involved. To provide examples of dependence diagrams, in this paper we use a synthesized database and a real weather database. For both these datasets, we show how dependence diagrams can be used to summarize the role of the condition attributes in determining the decision attributes' value.

Normally, in a given rule set, the decision attribute and the condition attributes are assumed to have been observed at the same time, so there is no notion of the passage of time between observing the attributes. Dependence diagrams can show such *atemporal* (or *instantaneous*) relationships. However, they can also be of particular interest in understanding potentially causal rules, a concept that is explained next.

Temporal and atemporal rules can be discovered from sequential data using the TimeSleuth software [3]. Assuming that time may have passed between the observations of variables allows interesting possibilities in analysing the data, and can lead to the generation of *temporal* decision rules Temporal decision rules that allow prediction of the future events using only past events are potential candidates for causal rules and are called *p-causal rules*. Such rules can be further evaluated by a domain expert to determine whether they represent actual causal rules. Crucially, dependence diagrams can aid in visualizing dependencies among attributes to aid the domain expert in evaluating the potential causality of these dependencies.

TimeSleuth can combine a number of sequential input records into one merged record. To uncover potentially causal rules, TimeSleuth examines the effects of a condition attribute, from previous time steps, on the decision attribute. Similarly, to identify *acausal* (i.e., not causal, but still temporal) rules, it examines the effects of condition attributes on a decision attribute in a previous time step. If previous values can be *retrodicted* (predicted in reverse), the relationship is assumed to be acausal. The number of consecutive records merged together is determined by a *window size*. Suppose at each time step we register the current position of an object along the x-axis, and also the direction of the movement that will be attempted during the next time step (left or right). So the effect of the attempted movement can be discovered in the next time step. Rule 1, for example, has been produced by merging two consecutive records in such an environment. x_1 and a_1 can be read as the previous position and movement directions, while x_2 can be read as the current location.

We can compare a dependence diagram to a causal Bayesian network [8] for the task of displaying causal rule sets. Both display causal influences, but they employ different methods to derive their corresponding graphs, and their interpretations are different. In a Bayesian network, one can traverse the graph transitively from a grandparent to a parent and then a child node, and so on, while one cannot traverse a dependence diagram.The rest of the paper is organized as follows. Section 2 introduces dependence diagrams first via an example and then formally. Section 3 discusses pruning dependence diagrams. In Section 4 we present examples of dependence diagrams from two data sets. Section 5 concludes the paper.

2 Dependence Diagrams

As previously mentioned, a dependence diagram summarizes multiple sets of decision rules in a compact manner and shows the amount of influence that each condition attribute has on the decision attributes' values. The diagram is not equivalent to a set of decision rules, as the original rule sets cannot be extracted from a dependence diagram. Each attribute in a dependence diagram can be a decision attribute as well as a condition attribute.

In a dependence diagram, attributes are denoted as nodes in a graph and connected together. The strength of the connection (the weight of the edge) depends on how often they are used in predicting each other's values. By definition, a decision attribute *d depends on* another (condition) attribute c_i if attribute c_i appears in rules that are used to predict attribute *d*. A dependence diagram can summarize instantaneous, acausal, or possibly causal rule sets, as defined in [4, 5].

Before presenting a formal definition, we first provide an example of a dependence diagram. In Figure 1, a dependence diagram for temporal data from a simulated robot is shown. The data comes from a simulated robot doing a random walk in a two-dimensional space, where *x* and *y* denote the position, *a* is the random action taken (the direction of movement), and *f* shows the presence or absence of food at the robot's location. The data represent repeated observations of the robot's state, consecutively ordered by time.

Figure 1 shows that we can determine the value of *x* reliably (accuracy of 100% as indicated in the node corresponding to *x*) from the values of *x* and *a*. In particular, the previous values of *x* and *a*, namely x_{t-1} (read: *x* at time *t*-1, or the previous time step) and a_{t-1}, are used in all the rules that predict the current value of *x*, namely x_t, (read: *x* at time step *t*, or the current time). This is shown by links from nodes *x* and *a* to node *x* having a weight of 100%. The times of occurrence for the attributes are not shown in a dependence diagram. Figure 1 also shows that the value of *a* can be determined with an accuracy of 47% by using the values of *a*, *x*, *y*, and *f* attributes. The value of *a* depends mostly on *x* and *y* (*x* appears in 80% of the rules, while *y* appears in 76% of the rules), somewhat on *a*, (which appears in 64% of the rules), and to a lesser extend on *f* (40% of the rules). A low accuracy value or link weight can suggest that a relationship does not exist or that insufficient evidence is available. For example, several links have 0% weight. Such relationships can be pruned from the diagram.

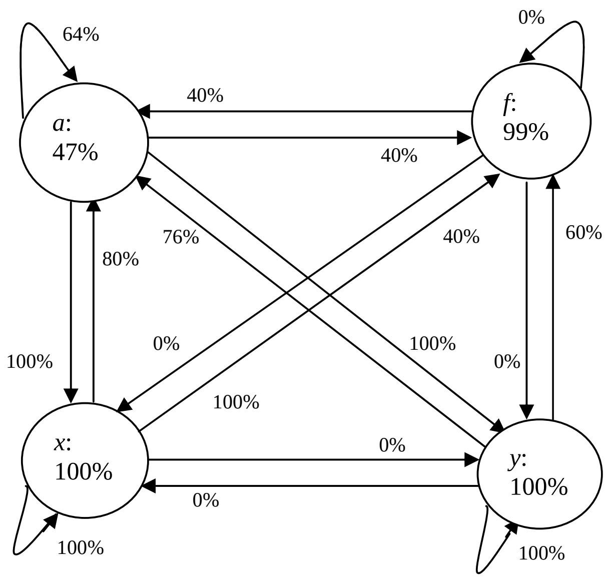

Fig. 1. A dependence diagram for the robot data

We now formally define a dependence diagram. For any rule set *R* that predicts the value of attribute *d*, if condition attribute c_i appears in at least one rule in *R*, then *d* depends on c_i. This type of dependence is called *static dependence*, because it is determined by the static form of the rule set, as opposed to the run-time or dynamic behaviour of the rule set.

Definition 1: If $r \in R$ and $c_i \in$ CONDITIONS(*r*), then *d* = DECISION(*r*) *depends on* c_i.

Ignoring the time indexes, the percentage of rules in a rule set *R* in which attribute c_i appears determines the strength of dependence of *d* on c_i.

Definition 2: Static Dependence Strength: *d* statically depends on c_i with *strength* *s*% with respect to a rule set *R*, written as $D_d(c_i) = s\%$, if c_i appears in *s*% of the rules in *R*.

In rule r, d statically depends on c_i with strength s% at time step t if c_i appears in s% of the rules at time step t, written as $D_{d,t}(c_i)$.

For a temporal rule computed using a window size of $w > 0$, we calculate $D_d(c_i)$ as follows:

$$D_d(c_i) = D_{d,1}(c_i), \quad \text{if } w = 1 \text{ or } w = 2$$
$$D_d(c_i) = \max(D_{d,1}(c_i), \ldots D_{d,w-1}(c_i)), \quad \text{otherwise.}$$

Example: Let the set of condition attributes be {a, b, c} and the decision attribute be d. Suppose the window size is 3 and the rule set R contains 2 rules: {[If at Time 1: (a = 1) And at Time 2: (a = 1) and (b = 2), Then at Time 3: (d = true)], [If at Time1: (a = 3) Then at Time 3: (d = false>)]}. Here, $D_d(a)$ = 100% (a appears in both rules) and $D_d(b)$ = 50% (b appears in half of the rules).

From the previous example, we have $D_{d,1}(a)$ = 100%, $D_{d,2}(a)$ = 50%, $D_{d,1}(b)$ = 0%, $D_{d,2}(b)$ = 50%.

Dynamic dependence is similar to static dependence, except the strength of the dependence is determined according to the rules that actually get used for determining the value of the decision attribute. In other words, only the rules that get fired by the test dataset are considered for determining the strength of the edges in the dependence diagram.

Definiton 3: Dynamic Dependence Strength: d dynamically depends on c_i with *strength* s% with respect to rule set R and data set T, written as $DD_d(c_i) = s$%, if c_i appears in s% of the rules that are fired when rule set R is applied to data set T.

In the previous example, suppose only the first rule is fired. In this case we have: $DD_d(a)$ = 100%, $DD_d(b)$ = 100%, $DD_{d,1}(a)$ = 100%, $DD_{d,2}(a)$ = 0%, $DD_{d,1}(b)$ = 0%, and $DD_{d,2}(b)$ = 100%.

We use a threshold value to prune weak and accidental dependencies.

Definition 4: Threshold Dependence and Independence: d is *dependent* on c_i if $D_d(c_i) > \varepsilon$, where ε is a user-specified threshold. Otherwise, d is *independent* of c_i.

Definition 5: Dependence Diagram: A *dependence diagram* is a possibly cyclic, directed, weighted graph <N, L>, where N is a set of nodes, each representing an attribute, and L is a set of directed links, each representing the dependence strength of a decision attribute on a condition attribute. The direction of the link is from the condition attribute to the decision attribute. A node represents a decision attribute with respect to the links pointing to it, and a condition attribute with respect to the links pointing away from it.

In a static dependence diagram, weights are assigned to the links but not the nodes. No weights are assigned to nodes because the rules are not run on any data set and so no accuracy values are available. In a dynamic dependence diagram, weights are assigned to both the nodes and the links. The weight of a node is the training or testing accuracy of the rules created for predicting the decision attribute that is represented by the node.

To create a dependence diagram, the rule sets for predicting the values of one or more decision attributes are first generated from a training data set. A static dependence diagram can be drawn directly from the rule sets, but for a dynamic dependence diagram, each rule set must be evaluated on a selected dataset. For a given rule set, only one static dependence diagram can be derived, but multiple dynamic dependence diagrams can be derived by changing the test data set.

If all rules in a rule set get executed, then the static and dynamic link strengths will be the same. This case holds in Figure 1, where the weights on the links show the static (and dynamic) strengths, while the weights of the nodes show the results from a particular run.

Transitivity does not apply to dependence diagram graphs, because by definition only the immediate links to and from a node are meaningful. Thus, dependence diagrams cannot be traversed. This characteristic sets the dependence diagram apart from many other types of graph.

3 Pruning Dependence Diagrams

If the user of a dependence diagram is not interested in nodes or links that have low weight, the diagram can be pruned. Pruning can be performed at two levels, using a link pruning threshold and a node pruning threshold. At the link level, a link is pruned if it does not have enough strength, or in other words, enough importance, in determining the value of a decision attribute. At the node level, a node is pruned if its weight is too low, or in other words if it cannot be predicted accurately enough. Node-level pruning is only possible with dynamic dependence diagrams.

Each link in the dependence diagram has an associated weight that represents the static or dynamic dependence strength of a decision attribute on a condition attribute, as determined by the number of times the attribute has appeared in the rules predicting the decision attribute. In *link oriented pruning***,** links that have a weight below the link pruning threshold are removed.

In *node oriented pruning***,** the nodes are examined one by one. Each node represents an attribute. Associated with each node is a weight representing the training or testing accuracy for that attribute when it was used as the decision attribute. Any node with a weight below the node pruning threshold is pruned unless there are links that point away from that node to other nodes. In other words, a node that is used is classifying another attribute is not removed. All links that point to a pruned node are also pruned, regardless of their strength.

It is important to note that link-level pruning must be performed before node-level pruning. A node with a low accuracy denotes a decision attribute with low predictability, but the same attribute may be important in predicting the value of another decision attribute. The examples provided later in the text will make this point clear.

Link- and node-level pruning allow the user to choose the amount of detail in a dependence diagram. The user can thus concentrate on more influential attributes. This ability is important because the attributes in a data set may not all have equal importance, and when selecting the attributes to record in the data set, one may have chosen irrelevant attributes because of a lack of a priori knowledge.

4 Examples of Dependence Diagrams

In this section, we show dependence diagrams for two temporal datasets. The first dataset is from a discrete event simulator called URAL [11], where known atemporal and temporal rules govern an artificial environment. The second dataset is from a weather station in Louisiana AgriClimatic Information System [10]. We used TimeSleuth to generate rules from the datasets and then constructed a variety of dependence diagrams.

4.1 Data domains

The URAL dataset is a synthetic dataset derived from consecutive observations of a artificial robot moving in a simulated world. The world is a rectangular, 8×8 board. The robot performs a random walk in the domain: at each time-step, it randomly decides on one of the following actions *a*: left (L), right (R), up (U), or down (D). Left and right correspond to moving along the *x*-axis and up and down to moving along the *y*-axis. We used 2500 records for training, and 500 for testing the rules (predictive accuracy). When predicting the attribute *x*, we expect that a rule set derived using a window size of 1 will not contain highly accurate rules. The reason is that, based on our understanding of the domain, the current value of *x* depends on the previous value of *x*, and the previous direction of movement. The same holds for *y*. So we expect that a rule set derived from a window size of 2, called the p-causal rule set, will contain highly accurate rules.

The second example, the Louisiana weather dataset, is a real-world dataset from weather observations in Louisiana. It contains observations of 8 environmental attributes gathered hourly from 22/7/2001 to 6/8/2001. There are 343 training records, each with the air temperature, the soil temperature, humidity, wind speed and direction and solar radiation, gathered hourly. 38 other records were used for testing the rules and generating predictive accuracy values. Since this dataset describes real phenomena, interpreting the dependencies and relationships in it is harder than for the robot dataset.

It should be emphasized that although both these data sets are from temporal domains, a dependence diagram can be generated from any given data set. For a non-temporal dataset, we refrain from mentioning causality or acausality, but we can interpret the links in a dependence diagram as associations.

4.2 The dependence diagrams

Suppose a user wants a dynamic dependence diagram for the domain of the artificial robot. To generate such a dependence diagram, the user first creates a data set containing observations of the values of the four attributes. The user then generates four p-causal rule sets with a window size of 2, each with one of the available attributes set as the decision attribute. The results can be used to generate the dependence diagram shown in Figure 1.

The user may decide to prune any link with strength below 80% and any unneeded node with accuracy below 50%. In this case, 12 out of 16 links are removed, but none of the four nodes are removed. The results are shown in Figure 2. Notice that the node corresponding to *a* is not pruned (even though its accuracy is below the threshold), because it is an important participant in determining the values of *x* and *y*.

From the diagram, we can see that the value of *x* can be predicted with 100% accuracy using previous values of *a* and *x*, and the value of *y* can be predicted with 100% accuracy from the previous values of *a* and *y*. The values of *a* and *f* cannot be predicted with sufficient accuracy to meet the constraints.

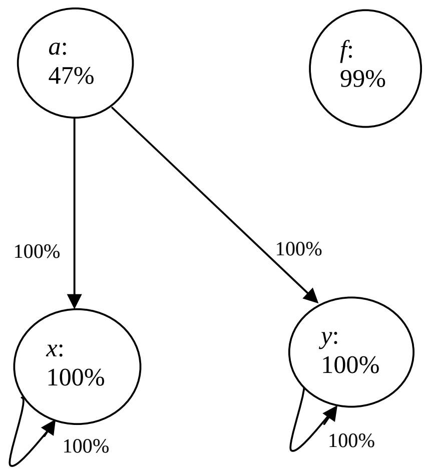

Fig. 2. The pruned dependence diagram for the robot data

The choice of appropriate threshold values depends on the domain and is left to the domain expert. A domain expert can observe the relative strengths and choose to allow only certain links to remain. Various threshold values can be tried to obtain different pruned dependence diagrams.

We can observe the influence of the attributes on each other in the dependence diagram shown in Figure 2. If a link exists from a node to itself, then the value of the corresponding attribute at a different time (preceding or succeeding) is used for predicting the present value. A node can only point to itself if the window size is bigger than 1.

If a rule set is derived from a p-causal investigation with TimeSleuth, the links in either static or dynamic dependence diagrams for the rule set may be examined by a domain expert to see if the dependencies they represent could be causal. These links are based on potentially causal relationships. However, if the rule set is derived from an instantaneous or acausal investigation, the presence of a link in a dependence diagram is unrelated to the existence of causality. In this case, none of the dependencies are suggested as p-causal.

Considering the second dataset, suppose the user is interested in a dynamic dependence diagram for the weather dataset, where the strength of every link is at least 40% and the strength of every node (unless otherwise needed) is at least 50% for a p-causal investigation with window size 2. As with the robot data, the user first constructs the diagram with all nodes and links. Figure 3 shows the diagram after pruning the links by removing any link with strength less than 40%.

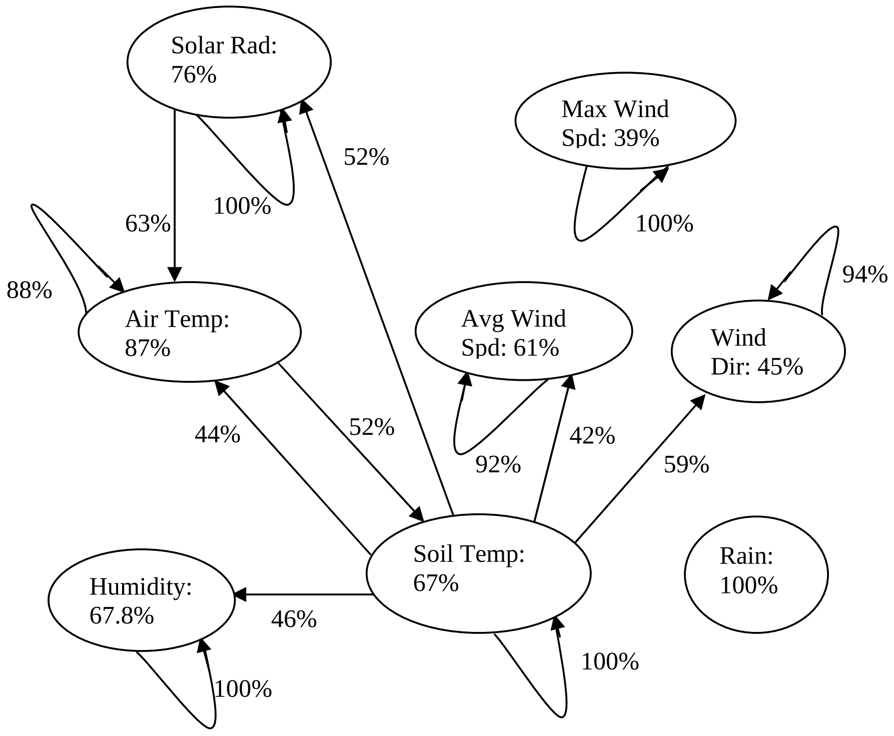

Fig. 3. Dynamic dependence diagram for the weather data

Finally, we prune unneeded nodes with accuracies less than 50%. The results are given in Figure 4. The dynamic dependence diagram in Figure 4 displays how the values of the attribute are related to each other. For example, the value of Soil Temperature is highly dependent on its own previous value and somewhat dependent on Air Temperature.

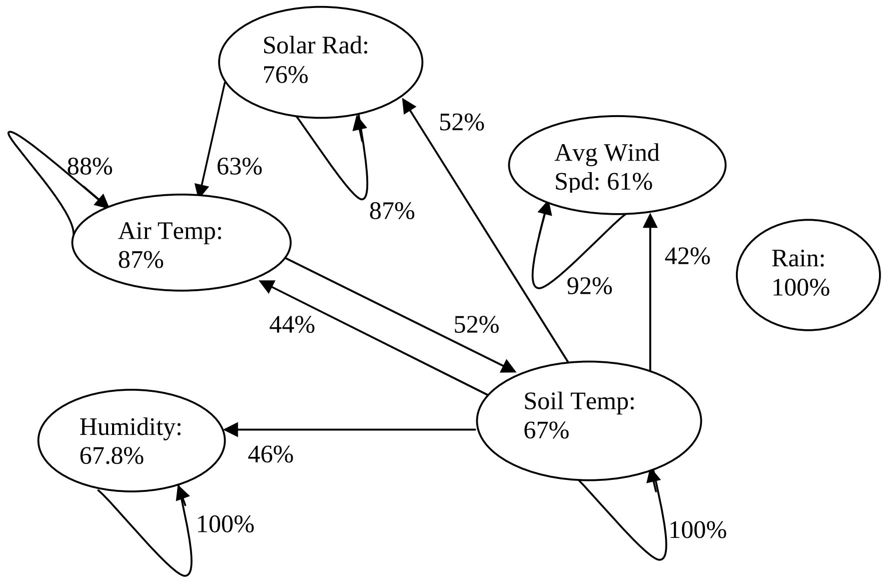

Fig. 4. The pruned dynamic dependence diagram for the weather data.

5 Concluding Remarks and Future Work

Making sense of the relations implied in a rule set is not easy because of the textual representation of such rules. We introduced dependence diagrams as a way of summarizing the relationships among a number of variables. They provide a visual aid for understanding the amount of influence of the attributes on each other. We provided examples of dependence diagrams that describe relations in synthetic and real datasets. A dependence diagram can summarize many rule sets generated from the same dataset, each with a different decision attribute. The complexity of a dependence diagram depends on the number of attributes rather than the number of generated rules.

Dependence diagrams have not yet been integrated into TimeSleuth, so they have to be derived manually. Automating their generation would allow the user to prune the diagram using different threshold values in real-time and notice important relationships more easily.

References

[1] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J., *Classification and Regression Trees*, Wadsworth Inc., 1984.

[2] Brunk, C., Kelly, J., and Kohavi, R., MineSet: An Integrated System for Data Access, Visual Data Mining, and Analytical Data Mining, *The Third Conference on Knowledge Discovery and Data Mining (KDD-97)*, Newport Beach, USA, August 1997.

[3] Karimi, K. and Hamilton, H.J., TimeSleuth: A Tool for Discovering Causal and Temporal Rules, *14th IEEE International Conference On Tools with Artificial Intelligence (ICTAI'2002)*, Washington DC, USA, November 2002. pp. 375-380.

[4] 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, pp. 234-240.

[5] 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, pp. 175-189.

[6] Kohavi, R. The Power of Decision Tables, *Proceedings of the European Conference on Machine Learning*, 1995, pp. 174-189.

[7] Michalski, R. S., A Planar Geometric Model for Representing Multidimensional Discrete Spaces and Multiple-valued Logic Functions, *Technical Report UIUCDCS-R-78-897*, University of Illinois at Urbana-Champaign, USA, 1978.

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

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

[10] http://typhoon.bae.lsu.edu/datatabl/current/sugcurrh.html. Contents change with time.

[11] http://www.cs.uregina.ca/~karimi/downloads.html/URAL.java