Causality-based cost-effective
action mining

Shamsinejadbabki, P, Saraee, MH and Blockeel, H

http://dx.doi.org/10.3233/IDA-130621

Title	Causality-based cost-effective action mining
Authors	Shamsinejadbabki, P, Saraee, MH and Blockeel, H
Type	Article
URL	This version is available at: http://usir.salford.ac.uk/id/eprint/48185/
Published Date	2013

USIR is a digital collection of the research output of the University of Salford. Where copyright permits, full text material held in the repository is made freely available online and can be read, downloaded and copied for non-commercial private study or research purposes. Please check the manuscript for any further copyright restrictions.

For more information, including our policy and submission procedure, please contact the Repository Team at: usir@salford.ac.uk.

Causality-based Cost-effective Action Mining

Pirooz Shamsinejadbabaki[1], Mohamad Saraee[2], and Hendrik Blockeel[3]

[1]Electrical and Computer Engineering Department, Isfahan University of Technology,Isfahan,Iran
[2]School of Computing, Science and Engineering, University of Salford-Manchester, Manchester, UK
[3]Department of Computer Science, KU Leuven, Leuven, Belgium

January 19, 2013

Abstract

In many business contexts, the ultimate goal of knowledge discovery is not the knowledge itself, but putting it to use. Models or patterns found by data mining methods often require further post-processing to bring this about. For instance, in churn prediction, data mining may give a model that predicts which customers are likely to end their contract, but companies are not just interested in knowing who is likely to do so, they want to know what they can do to avoid this. The models or patterns have to be transformed into actionable knowledge. Action mining explicitly addresses this.

Currently, many action mining methods rely on a predictive model, obtained through data mining, to estimate the effect of certain actions and finally suggest actions with desirable effects. A major problem with this approach is that predictive models do not necessarily reflect a causal relationship between their inputs and outputs. This makes the existing action mining methods less reliable. In this paper, we introduce ICE-CREAM, a novel approach to action mining that explicitly relies on an automatically obtained best estimate of the causal relationships in the data. Experiments confirm that ICE-CREAM performs much better than the current state of the art in action mining.

1 Introduction

Data Mining (DM) emerged as a response to the need of extracting understandable and ultimately useful patterns from large databases. The large majority of data mining methods focuses on extracting patterns or models that summarize the data. When the final goal of the data mining effort is to help business experts with decision making, such patterns or models are often not useful without

1

a considerable amount of extra work by domain experts. The more we can narrow the gap between the patterns mined and the knowledge required to make decisions, the more effectively data mining can be used in real-life problems. Action rule mining is motivated by this observation.

Action Rules are a type of knowledge that explicitly describes how changes in variables that can be influenced affect some target attribute that cannot be influenced directly [1].

They are much easier to apply than, for instance, association rules, because they specify exactly what should be done. For example [2], consider an intelligent loaning system for a bank that contains customers data. A traditional DM pattern in this system can be a classification rule like: *if sex = 'M' and service = 'L', then payback = 'False'.* However, an action rule can be like: *if sex = 'M' and service changes from 'L' to 'H', then payback will go from 'False' to 'True'.* In many business contexts, the second rule expresses more directly useful knowledge. Take, for instance, churn prediction: the company is not merely interested in predicting which customers it is going to lose, it wants to know what can be done to avoid this.

Action Mining (AM) is the process of learning action rules from data. Not much work has been done in this area up till now. Existing work includes Yang et al.'s method for learning actions from decision trees [3, 2], and several versions of Ras et al.'s DEAR system for discovering action rules [4, 5, 6]. In all of these methods, input data is in the form of a set of attribute-value pairs for each object. Furthermore, a certain profit is associated with specific values of one particular attribute, called the target attribute. These methods then try to uncover existing associations between the target attribute and other attributes, and use these associations for finding the most beneficial actions. The main difference between these methods is in the technique by which they find associations. For example, Yang's method uses decision trees, while DEAR 2 uses classification rules.

Despite all innovations presented in existing AM methods, they suffer from an important drawback: they implicitly rely on the assumption that the available models (decision trees, association rules) are causal. It is well-known from statistics that association or correlation does not imply causation. Even though the learned models do not merely express the existence of a correlation, but its nature (in the form of a predictive function), they suffer from the same problem. If a function $f : X \rightarrow Y$ learned from a data set is found to be accurate, this means that, when we *observe* $X = x$ in a new object, we can accurately predict that $Y = f(x)$; but if we manually *change* the object's X value to $X = x'$, there is no guarantee that Y changes into $f(x')$; in fact, Y may not be affected at all. Basically, the joint distribution over (X, Y) of objects with a manually changed X differs from that of the observed objects in the training set, therefore the training set is not representative for these and there is no guarantee that models learned from it will be accurate in this new distribution.

To make this more concrete, suppose ice cream consumption C and outside temperature T are correlated in some data set, and a data mining method constructs a model f that accurately predicts T from C. f will be accurate as

long as its input consists of a C value observed under normal conditions, but if C is artificially increased by a promotional campaign, the model will tend to overestimate T. In cases like this, if a high temperature is desired, existing action rule mining algorithms would suggest to increase ice cream consumption, which obviously will have no effect.

The main contribution of this paper is a method that proposes actions based on causal relationships between attributes. The method relies on causal networks (CNs), which are a powerful tool for representing causal relationships and performing inference on them. In particular, they correctly infer the effects of interventions (externally induced changes of values), which is exactly what we need here.

One might argue that such causal networks are rarely available in practice, when mining a database. However, one can try to construct them automatically from the data. There has been a lot of work on learning causal networks from data, starting with the foundational work of Pearl et. al. [7] and Spirtes et. al. [8]. They have shown that causal relationships can be inferred correctly under specific (but not under all) circumstances. This raises the following questions. Given a database, can we learn a causal network that is sufficiently accurate to be useful for action mining? And exactly how can such a causal network be used in this context? These are the questions we try to answer in this paper. They have not been studied before in action rule mining, and they pose a number of specific challenges that have not been addressed in the literature on causal networks, as will become clear later on.

In this text, we consider two scenarios. In the first scenario, which is mostly hypothetical but useful as a reference point, we assume that a correct causal network is available for the data. We present a method called Causal Relationship-based Economical Action Mining (CREAM) that, given an object and the causal network, predicts optimal (i.e., maximally cost-effective) actions. In the second, more realistic, scenario, we assume the causal network is not known. In that case, we can try to learn it from the data. This is a difficult task, and it is known that the correct network cannot be learned under most circumstances, but algorithms exist that return a partial network as a best possible estimate. A well known example is IC (Inductive Causation) [9]. ICE-CREAM (IC-enabled CREAM) refers to the complete approach of first running an inductive causation algorithm (IC or one of its variants), then CREAM. This approach requires extending CREAM so that it can handle partial, rather than complete, causal networks.

The novel approach is computationally more intensive than some existing approaches, so the question is how much can be gained by using it. We experimentally evaluate this on a number of datasets, comparing ICE-CREAM to the state of the art (which uses models that are not necessarily causal). The comparison shows that taking induced causality into account results in much better action recommendations.

The rest of this paper is structured as follows. Related work is discussed in Section 2. In Section 3 we describe background and definitions for action mining. Causal networks, as the backbone of our methods, are briefly explained

3

in Section 4. We describe the core of our approach, the CREAM method, in Section 5, and the full method, ICE-CREAM, in Section 6. In Section 7, we experimentally compare our methods with Yang et al.'s method, testing them on different data sets. Finally, in Section 8 we conclude and discuss perspectives of causality-based action mining.

2 Related Work

Action mining is part of a subdomain of data mining called Actionable Knowledge Discovery (AKD), which is concerned with finding patterns that are not only interesting but also actionable, preferably with minimal further effort required of domain experts. Piatetsky-Shapiro and Matheus[10] define a pattern as actionable if the user can act upon it to gain some advantage in the application domain. Although actionability could be considered a standard requirement in data mining (it is roughly the same as "usefulness", and no one wants to find useless patterns), most traditional systems do not explicitly maximize some well-defined actionability criterion, and the patterns they return do not explicitly express desirable actions: further inference is needed to identify those.

We can categorize the existing work in AKD in different ways. A first dichotomy relates to whether a standard or novel type of patterns is returned. Some methods (e.g. [11]) define a measure of interest called actionability, and filter patterns resulting from traditional methods based on this measure. Other methods try to learn a new type of knowledge, often called *action rules* or simply *actions*, from data. We here focus on the second type.

Assume objects are described by a set of attributes. An action is an externally induced change of value of one attribute. Such an action can be taken with the goal of changing the state of the object into a desirable state. An action rule is an if-then rule that predicts the effect of applying certain actions under particular conditions. We define action mining as the process of finding desirable actions (for a given object), or finding rules that can predict desirable actions, by analyzing data. Following terminology from predictive modeling, we call methods that predict desirable actions for a given object *transductive*, and methods that learn action rules (which can afterwards be used to predict actions), *inductive*.

One of the first efforts towards action mining is the work by Ras et al.[1]. Their goal was to know "what actions should be taken to improve the profitability of customers". They define the concept of action rules, distinguishing between flexible attributes, of which the value can be changed directly, and stable attributes, which cannot be changed directly. Valid actions consist of changing the value of a flexible attribute. This work led to the definition of a system called DEAR, which in successive work [4, 5, 6, 12] was developed into a full-fledged system that includes handling of missing values, outlier detection, and more.

Wang et al.[13] propose a method that takes as input a set of predefined actions and a so-called influence matrix that shows the effect of each action on

the flexible attributes.

Tzacheva et al.[14] also use the influence matrix for finding action paths which are a sequence of actions. Characteristic for these approaches is that they require background knowledge (the influence matrix) that is not learned from the data, but assumed to be known. This reduces their applicability. We call these methods *informed*, as opposed to *uninformed* methods such as the DEAR series.

He et al.[15] present an Apriori-like method for learning action rules that takes into account the cost of action rules in addition to their support and confidence. Yang et al. [3, 2] present a method that first learns a decision tree from previous data and then uses the tree for finding the most profitable action for upcoming cases. Characteristic for these approaches is that a cost is associated with each action, which will be subtracted from the eventual profit obtained. The goal is then to find the set of actions that maximizes the net profit. One could call this "economical action mining". Obviously, the setting with costs generalizes over that without costs (one could simply set all action costs to zero).

The method we propose in this paper is transductive (it predicts actions but does not yield generally applicable action rules), uninformed, and uses costs. From this point of view, the most closely related work is that of Yang et al. We therefore discuss it in more detail.

Yang et al.'s method first learns a decision tree, then analyzes the tree to find the optimal actions for each object. As the first step is pretty standard, we here focus on the second step. We illustrate it on the example decision tree in Fig. 1 [2], which represents a simplified decision tree built from customer profiles in a bank. The number below each leaf node is the probability that a random object in that node is in some desired state. This probability is estimated from past data. In Yang's method, an action on an object is defined as a set of attribute value changes that moves an object from one leaf node to another one in the decision tree. For example, consider an instance ($Service = L$, $Sex = M$, $Rate = L$). It is in leaf node B, but if Service were H instead of L, it would be in node D, in which the probability of being in the desired state is higher. It is therefore assumed that the action ($service, L \to H$) will increase the probability that the customer is in the desired state.

Yang et al. associate a fixed profit p with the desired state, define the expected profit in a leaf as the proportion of cases in that leaf in the desired state, multiplied by p, and define the net profit of an action as the difference in expected profit between the from-leaf and the to-leaf, minus the cost of the action. A leaf-node search algorithm then selects for each object the action with maximum net profit.

Compared to Yang et al., the crucial new element in our method is that it tries to extract causal relationships from the data, and uses a model based on these causal relationships, instead of a non-causal model, to reason about the effects of actions. In this respect, the most closely related work in association rule discovery is probably that of Silverstein et al. [16], who proposed a method for finding association rules that encode causal relationships. These causal as-

Service

L   H

M

Sex   Rate

F   M   C   L   H
           0.1

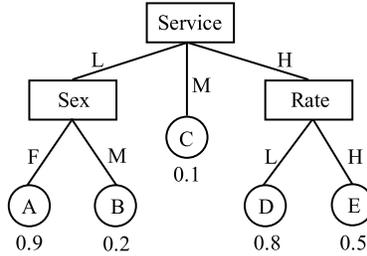A   B   D   E

0.9   0.2   0.8   0.5

Figure 1: A sample decision tree of customer profiles

sociation rules are not action rules by themselves, but could be post-processed using methods such as DEAR. To our knowledge, such an approach has never been tried in the field of action mining. The approach we propose here is an alternative to it that relies on more recent results in mining causal relationships.

In the causal network literature, the focus has largely been on the challenging task of learning causal networks from data. Pearl's seminal work [7] has given rise to a large number of algorithms that try to learn such networks. It is well-known how probabilistic inference in such networks can be done. In the context of action rule mining, however, new types of problems arise, which are of a more practical nature and which have not been considered in the causal networks literature:

1. Given a desirable state for one variable, and costs associated to actions that change the value of a variable, what is the most economical way of causing the target variable to take that value? This corresponds to optimizing the expected value of a function of some variables in the causal network.

2. The causal network derived form the data is often only partial; not all causal directions can typically be determined. The answer to question 1 can therefore only be computed by making additional assumptions or approximations. How well can the question be answered, when this is taken into account?

3. How can action rules be derived from the given data? An action rule is a generally applicable rule that states under what circumstances which variables should be changed. These action rules can in principle be obtained by computing the answer to question 1 under many different circumstances, then learning rules from all these answers.

In this text, we are mostly concerned with questions 1 and 2.

3 The learning setting

As terminology and definitions in the literature vary somewhat, we here formally define concepts as they are used in this paper.

The input data for action learning is observational and in tabular form. Each column represents an attribute and each row an object. We assume there are n objects and m attributes. We write an object's state as $(A_1 = a_1, A_2 = a_2, \cdots, A_m = a_m)$, with each A_i an attribute and a_i a value from its domain $Dom(A_i)$. Given an object o, its value for attribute A is written as $A(o)$. We assume there are no missing values.

We assume that all attributes are discrete; this is not a strong limitation, as real-valued attributes can always be discretized before applying the method. $\mathbf{A}$ denotes the set of all attributes in the system. One particular binary attribute is called the target attribute, and denoted by T. One of T's values is called the goal, denoted by t_g; with this goal a profit $p_g > 0$ is associated. With the other value of T a profit of 0 is associated.

An *action* is a structure of the form $(A, a \to a')$, with A a non-target attribute and $a, a' \in Dom(A)$. It corresponds to changing the value of attribute A from a to a' by means of an external intervention. With each action α a cost $C(\alpha)$ is associated. An action $(A, a \to a')$ is *compatible* with an object if the object's value for A equals a. When an action α is applied to an object o, we write the resulting state as $o\alpha$. Note that, while an action $(A, a \to a')$ directly changes only A, this change may in turn cause changes to other attributes besides A, according to the causal relationships that exist in the data. So, given an object o and an action $\alpha = (A_i, a \to a')$, there is no guarantee that $A_j(o\alpha) = A_j(o)$ for each $j \neq i$. Note that no additional cost is associated with these indirect changes, only the action itself has a cost.

An *action set* Γ is a set of actions in which each attribute occurs at most once. An action set is compatible with an object if all its actions are compatible with it. Given an object o and a compatible action set $\Gamma = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, we define $o\Gamma = o\alpha_1\alpha_2 \cdots \alpha_k$. It denotes the state of object o after applying all actions to o. We define the cost of an action set as $C(\Gamma) = \sum_{\alpha \in \Gamma} C(\alpha)$.

The *net profit np* of an action set and an object o is defined as $np(\Gamma, o) = p_g - C(\Gamma)$ if $T(o) \neq t_g$ and $T(o\Gamma) = t_g$, and $np(\Gamma, o) = -C(\Gamma)$ otherwise.

We define the task of *economical action mining* as follows: Given an object space $\mathcal{O} = Dom(A_1) \times \cdots \times Dom(A_m)$, a data set $D \subseteq \mathcal{O}$, a set of possible actions and their associated costs, a target attribute T with associated goal t_g and profit p_g, and a set of objects O (which may or may not overlap with D), find for each $o \in O$ an action set Γ_o such that $\sum_{o \in O} np(\Gamma_o, o)$ is maximal.

It is useful to look back at Yang et al.'s work at this point. It addresses the task just described. In doing so, it implicitly assumes two things. First, changing an attribute's value does not affect any other attributes except the target. That is, given an object o and an action $\alpha = (A_i, a \to a')$, $A_j(o\alpha) = A_j(o)$ whenever $j \neq i$ and $A_j \neq T$. This is only true if there is never a causal relationship between two non-target attributes. Second, each non-target attribute that occurs in the decision tree causally influences the target attribute in exactly the way that

the tree suggests. Both are strong assumptions, which can easily be violated in practice. In this work, we do not make such assumptions.

4   Causal Networks

It was long deemed impossible to learn causal relationships from merely observational data, until Spirtes et al. [8] and Pearl [7] showed that under certain conditions one can infer some causal relationships from non-experimental data. These causal relationships are typically shown using a Directed Acyclic Graph (DAG) called a *causal network* (CN). In a CN, each node represents a variable and each edge represents a direct causal effect from the parent to the child. Causal networks are essentially Bayesian networks, with the additional guarantee that edges coincide with causal influences.

Causal networks can be used like Bayesian networks for probabilistic inference. Queries for the (conditional) probabilities of events, prior and posterior marginals, most probable explanations and maximum a posteriori hypotheses can be answered (see [17] section 5.2 for more details). However, causal networks have the additional advantage that they can be used for reasoning about the effects of *interventions*: what can we infer if we *set* $A = a$, rather than observe $A = a$? Such interventions are basically the same as what we call actions in action mining.

One of the most subtle tasks in causal inference is learning the structure of causal networks from merely observational data. Much work has been done in this area. The Inductive Causation (IC) algorithm proposed by Verma and Pearl [9] is the backbone of many proposed methods. This algorithm uses conditional independency relationships between attributes for finding causal links. In this text, we use inductive causation or IC to refer to the whole class of algorithms that induce causation using the same basic principles.

It is known that not all causal relationships can be discovered from observational data; causal relationships can be identified only under specific circumstances. As a result, IC returns only a partially directed acyclic graph (PDAG). In a PDAG, some edges are directed, and some undirected. Directed edges indicate causal relationships that have been identified with certainty. Undirected edges indicate that there is a relationship, but its type and direction cannot be determined with certainty.

In the following, we first explain the idea of causality-based action mining, and propose the CREAM method for extracting actions from a given causal network. Next, we will describe challenges in learning actions from only observational data, and present the ICE-CREAM method as a solution for them.

5   Mining Actions from Causal Networks (CREAM)

In this section, we consider the following task: given a causal network in the form of a DAG and an object o, find actions that effectively and economically
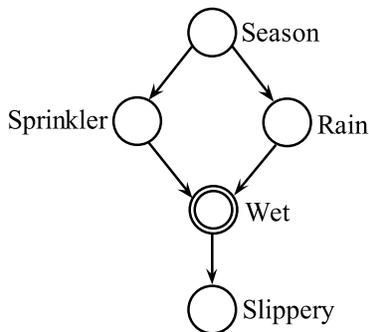
Figure 2: A sample causal network including five boolean attributes. The target node is shown with a double circle

put o into the desired state (i.e., $T(o) = t_g$).

The nodes in the DAG correspond to attributes. We call the node that corresponds to the target attribute the target node. We say a node x is *upstream* from a node y (and y is *downstream* from x) if and only if there is a directed path from x to y. A node is a *cause node* if it is upstream from the target node.

For example, in Fig. 2, if the target node is "Wet", then "Season", "Rain" and "Sprinkler" are cause nodes.

Clearly, only changes in cause nodes can influence the target node, therefore these are the only candidates for actions. In the following, we explain how to compute the effect of an action.

5.1   Inferring the impact of an action on an object

Inference in causal networks is a bit more complicated than that in Bayesian networks, because interventions (attributes set to a particular value) must be treated differently from observations. Assume we have an edge $A \rightarrow B$, i.e., A causally influences B. When we observe B, this gives us information about A, so we must update our belief about A, but when we set the value of B, this has no effect on the value of A, so our belief about A should remain unchanged. This is achieved in practice as follows: whenever a variable is given a value by intervention, its incoming edges are removed from the DAG. The standard inference procedure for Bayesian networks is then used on the new DAG, without further distinguishing observed nodes from action nodes. This gives correct inference under all circumstances. For further information, we refer to Pearl [7].

In the context of action mining, we use this technique as follows. Given a causal network CN and an action $\alpha = (A, a \rightarrow a')$, we define CN_α as the network constructed from CN by removing all incoming edges to A, and setting $A = a'$. Given an action set Γ, CN_Γ denotes the network obtained by applying this procedure consecutively for all actions in Γ.

9
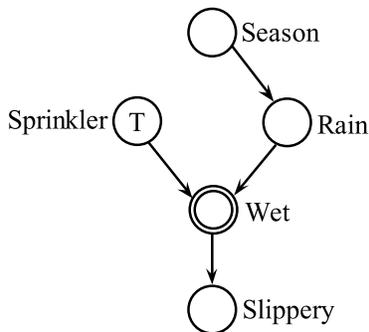
Figure 3: Submodel of the causal network in Fig. 2 obtained after applying action $\alpha(Sprinkler, False \rightarrow True)$.

For example, Fig. 3 illustrates the CN_α resulting from applying action $\alpha = (Sprinkler, False \rightarrow True)$ to the network depicted in Fig. 2.

After creating CN_Γ it can be used for reasoning about the effect of Γ on the value of other attributes. In the next subsection we use CN_Γ for computing the effect of Γ on the value of the target attribute.

5.2 Net profit

We define the *effectiveness* of an action α with respect to an object o as the gain in probability that o is in the goal state. Formally:

$$eff_\alpha(o) = Pr(T = t_g | C' = C'(o\alpha)) - Pr(T = t_g | C = C(o)) \tag{1}$$

where C is the set of cause nodes, C' is the set of cause nodes that are not downstream from the action attribute, $C(o)$ are the values for the cause nodes in o and $C'(o\alpha)$ are the values for the nodes in C' in the modified object. The reason for using C' instead of C for $o\alpha$ is that cause nodes downstream from a changed attribute can no longer be considered evidence. (For instance, to answer the question "will it be slippery if I turn the sprinkler on", in a context where the grass is dry, we need to add evidence that the sprinkler is on but also drop the evidence that the grass is dry, as turning the sprinkler on causally affects the value of Wet, and thus renders this evidence invalid.) As explained before, $Pr(T = t_g | C' = C'(o\alpha))$ is computed using the modified network CN_α, whereas $Pr(T = t_g | C = C(o))$ is computed using the unmodified network. Both are computed using standard inference procedures for Bayesian networks.

The effectiveness of an action set Γ is computed similarly. We define

$$eff_\Gamma(o) = Pr(T = t_g | C' = C'(o\Gamma)) - Pr(T = t_g | C = C(o)). \tag{2}$$

$Pr(T = t_g | C' = C'(o\Gamma))$ is computed similarly as for a single action, with the following differences: CN_Γ is used for the inference, and C' contains all nodes that are not downstream from any action attribute in CN_Γ.

10

The net profit, finally, is computed as follows:

$$np(\Gamma, o) = \mathit{eff}_\Gamma(o) \cdot p_g - C(\Gamma) \qquad (3)$$

Our task is to find, for each object o that is not in the desired state, the action set Γ that maximizes $np(\Gamma, o)$. In the next subsection, we discuss two strategies for searching the space of all possible action sets.

5.3   Exhaustive vs Greedy Search

Any action of the form $(A, a \to a')$, with A a cause node and a the observed value for it, is a candidate action for trying to change the target attribute.

With $\mathbf{C}$ the set of attributes that correspond to cause nodes, this gives $\sum_{A \in \mathbf{C}}(|Dom(A)| - 1)$ candidate actions, which is $O(md)$ with m the number of attributes and d the largest domain size. The number of action sets is $\prod_{A \in \mathbf{C}} |Dom(A)|$ (each attribute may or may not occur in the action set, and if it occurs, there are $|Dom(A)| - 1$ possible values for a'). This is $O(d^{m-1})$.

Clearly, searching the space of action sets exhaustively may be intractable for medium-sized or larger datasets. We therefore compare two different strategies. The first one is exhaustive search (ES): for each object, we evaluate all action sets and select the best one.

The second and faster strategy is greedy search (GS). Given an object o, we start with an empty set Γ. In each iteration, we extend Γ with one action, namely the one that, when added to Γ, results in the highest np for Γ. This is continued until no action is found that improves $np(\Gamma)$.

In the first iteration, GS checks at most $d(m-1)$ actions, and afterwards, in the i'th iteration, at most $d(m-i)$, since an already added attribute need not be reconsidered for addition in later iterations. In the worst case, GS iterates $m-1$ times, so the number of action sets that is evaluated for each object is at most $d \sum_{i=1}^{m-1}(m-i)$, or $O(dm^2)$.

Considering all objects in the set O, the number of evaluations of action sets is $O(|O|dm^2)$ for greedy search, and $O(|O|d^{m-1})$ for exhaustive search. Note that each evaluation includes probabilistic inference in the causal network, which is itself a relatively expensive operation.

We are now ready to present the full algorithm for extracting optimally economical actions for an object o from a causal network. The approach is called CREAM (Causal Relationships based Economical Action Mining). Algorithm 1 presents pseudo-code for the greedy approach. The algorithm using exhaustive search is similar, simply replacing the stepwise construction of an optimal action set with an exhaustive enumeration and evaluation of all action sets. The procedure $\mathit{findCandidateActions}$ returns all actions that are compatible with o and for which the action attribute is a cause node. The set $\mathbf{O}^-$ excludes objects that are already known to be in the desired state; no actions are sought for those.

The CREAM algorithm can be used to find the most profitable action set for any object, assuming a complete causal network is known. This may not

Algorithm 1 The CREAM algorithm for learning cost-effective action sets from causal networks (greedy version).

$1:$ **procedure** CREAM $(\mathrm{T}, O, \mathcal{C}, p_g, \mathcal{CN})$

Input:target attribute T,

      object data O,

      cost data $\mathcal{C}$,

      profit p_g,

      underlying causal network $\mathcal{CN}$,

Output: one action set for each object $o \in O$

$2:$ $\mathbf{O}^- \leftarrow \{o \in O | Pr(T(o) \neq t_g) > 0\}$

$3:$ **for each** o **in** $\mathbf{O}^-$**do**

$4:$ $\mathbf{I} \leftarrow \textit{findCandidateActions(o, } \mathcal{CN})$

$5:$ $\Gamma \leftarrow$ empty action set

$6:$ finished$\leftarrow$ false

$7:$ **repeat**

$8:$ $\alpha_{max} \leftarrow \arg\max_{\alpha \in \mathbf{I}} np(\Gamma \cup \{\alpha\}, o)$

$9:$ **if** $np(\Gamma \cup \{\alpha_{max}\}, o) > np(\Gamma, o)$ **then**

$10:$ $\Gamma \leftarrow \Gamma \cup \{\alpha_{max}\}$

$11:$ $\mathbf{I} \leftarrow \mathbf{I} - \{\alpha_{max}\}$

$12:$ **else**

$13:$ finished $\leftarrow$ true

$14:$ **until** $\mathbf{I} = \emptyset$ **or** finished

$15:$ assign Γ to o

---

be the case in practice. In the next section, we describe how a partial causal network can be learned from data, and how CREAM can be adapted to handle such partial networks.

6   Mining Actions from Observational Data (ICE-CREAM)

As mentioned before, it is possible to learn causal relationships between attributes from merely non-experimental data to a limited extent. The IC algorithm [9] does exactly this. It returns a partial DAG, in which some edges have no direction, reflecting the fact that the direction of causality could not be established in these cases.

Given a dataset without a causal network, one can run an algorithm such as IC to learn a causal network, then run CREAM to extract optimal actions from this network. This method, as a whole, is an uninformed action mining method that we call ICE-CREAM (IC-Enabled CREAM). While this may seem straightforward, a number of challenges need to be addressed. All of them are related to the fact that CREAM expects a DAG, but IC returns only a PDAG. In the following subsection, we discuss these challenges and explain how we extend CREAM in order to address them. Next, we will discuss ICE-CREAM as a whole.

6.1   Handling uncertainty

IC returns a partial DAG (PDAG): a graph in which some edges are directed and others are not, without directed cycles. This PDAG represents a set of DAGs, namely, the set of all DAGs that can be found by directing the undirected edges one way or the other, in such a way that the resulting DAG is "compatible" with the PDAG. Full details are beyond the scope of this paper, but essentially, a DAG is compatible with a PDAG if its edges have the same direction as the corresponding ones in the PDAG, and it contains no causal substructures that, if they are indeed present, should have been identified by IC, but were not. Given a PDAG G, we write the set of DAGs compatible with it as $dagset(G)$.

Allowing for undirected edges is a challenge for CREAM, as it causes uncertainty about which nodes are downstream from which other nodes. To meet this challenge, we integrate uncertainty into our method. We introduce a new concept, *action quality*, that is related to how likely it is that the action attribute causally influences the target attribute. In the following, we define action quality and explain how we compute it for an action.

Given a PDAG G and a probability distribution p over $dagset(G)$ (with $p(g)$ indicating the probability that g is the real causal network), we define the quality Q_α of an action $\alpha = (A, a \rightarrow a')$ as the probability that there exists a directed path from A to T in the real causal network (in other words, the probability that T is downstream of A). In a DAG, this value is one for cause

13

attributes and zero for all others. However, in a PDAG, this value is not easily computable.

We say a path ρ is partially directed if its edges are directed or undirected, and all its directed edges go in the same direction. Let $\mathbf{P}$ be the set of all partially directed paths between the action attribute A and the target attribute T for some action α, and let D_ρ represents the event "ρ is directed from A to T in the real network". Then we have

$$Q_\alpha = Pr(\bigcup_{\rho \in \mathbf{P}} D_\rho). \tag{4}$$

For a partially directed path to be directed if we traverse the path from the action attribute toward target attribute, all undirected edges of the path should take the direction of motion. If we assume D_ε represents event "undirected edge ε has a direction compatible with the path", then

$$Pr(D_\rho) = \prod_{\varepsilon \in \mathbf{E}^u} Pr(D_\varepsilon) \tag{5}$$

where $\mathbf{E}^u$ denotes the set of all undirected edges in ρ. It is worth noting that here we assume that the direction of each undirected edge is independent of the direction of other edges, which it is not always true (due to the constraints of acyclicity and unidentifiability of causal substructures). This is a first simplifying assumption.

A second assumption that we need to make is about the prior probabilities of edges having a certain direction. These are necessarily subjective. Looking at one undirected edge $A - B$ in isolation, without additional background knowledge such as chronological data, there is no reason to believe that the edge is directed either way, so we can assign a probability of 0.5 to each direction.

This is a subjective probability: its value represents our "degree of belief" about the direction, 0.5 meaning we have no reason to think one direction is more likely than the other.

Using 0.5 as the probability that an undirected edge "has the right direction", we have

$$D_\rho = (0.5)^{|\mathbf{E}^u|}. \tag{6}$$

The quality of an action set is now defined as follows:

$$Q_\Gamma = \prod_{\alpha \in \Gamma} Q_\alpha \tag{7}$$

It estimates the probability that all actions in the set causally influence the target attribute. ("Estimates", because it treats all the D_ρ with $\rho \in \mathbf{P}$ as independent, this on top of the fact that the Q_α are estimates themselves.)

Undirected edges pose some difficulties for computing the action set effectiveness eff_Γ, too. When creating CN_Γ from CN, edges from parents of the action attribute should be deleted, but with undirected edges it is not certain which nodes are the parents. In ICE-CREAM, we have chosen to delete all

14

directed edges and leave undirected edges unaffected. Other alternatives would be to delete also the directed edges, or delete each of them with some probability. None of these are correct under all circumstances. While some of these alternatives may work better than others on average, we have not attempted to determine an optimal alternative.

Finally, cause nodes cannot be determined with certainty either. We simply take a node as cause node if it can be a cause node according to the PDAG (that is, if undirected edges can be directed such that it becomes a cause node).

We now define the net profit of an action set in ICE-CREAM as follows:

$$np^\star(\Gamma, o) = Q_\Gamma \cdot \mathit{eff}^\star_\Gamma(o) \cdot p_g - C(\Gamma) \tag{8}$$

where $\mathit{eff}^\star_\Gamma(o)$ estimates $\mathit{eff}_\Gamma(o)$ as just explained.

We use CREAM* to refer to the version of CREAM that handles partial DAGs by using $np^\star$ instead of np.

6.2   ICE-CREAM

At the highest level of abstraction, the ICE-CREAM method can be described as follows:

1. Learn a partial causal network structure from data.

2. Estimate the value of the CN parameters from the data.

3. Call CREAM* on the learned CN.

We now analyze the complexity of each step separately.

Step 1 requires an algorithm that induces causal relationships. Many improved versions of the original IC algorithm have been proposed; we use the Grow Shrink Markov Blanket Algorithm by Margaritis [18]. Assuming a constant upper bound on the size of Markov blankets, the complexity of this algorithm is $O(e^2 + nm^3)$, with e the number of edges in the network. For more details about the complexity of the algorithm and some improvements on it, we refer to [18].

Since all attributes are observable and have no missing values, the parameters of the CN (conditional probabilities) are easily computed as relative frequencies. This takes one pass over the data, which has n rows and m columns. Thus, step 2, learning the parameters, takes $O(nm)$ time.

Just like CREAM, CREAM* can be implemented using a greedy (GS) or an exhaustive search (ES). The number of $np^\star$ calculations is $O(|O|dm^2)$ for the GS version, and $O(|O|d^{m-1})$ for the ES version. The computation time for $np^\star$ is dominated by that of Q_Γ. Indeed, $C(\Gamma)$ is clearly $O(m)$, and the main operation in $\mathit{eff}^\star_\Gamma(o)$ is computing $Pr(T = t_g | C' = C'(o\Gamma))$ and $Pr(T = t_g | C = C(o))$, which can be done in $O(m \cdot \exp(w))$ of time, with w the treewidth of the network (see, e.g., chapter 7 of [17]).

Q_Γ is more complicated, however.

To find Q_Γ, we compute $Pr(\bigcup_{\rho \in \mathbf{P}} D_\rho)$ for each action in Γ, which can be expanded as follows (inclusion-exclusion principle):

$$
\begin{aligned}
Pr(\textstyle\bigcup_{\rho \in \mathbf{P}} D_\rho) \quad = \quad & \textstyle\sum_{\rho \in \mathbf{P}} Pr(D_\rho) \\
& - \textstyle\sum_{\rho_1, \rho_2 \in \mathbf{P}} Pr(D_{\rho_1} \cap D_{\rho_2}) \\
& + \textstyle\sum_{\rho_1, \rho_2, \rho_3 \in \mathbf{P}} Pr(D_{\rho_1} \cap D_{\rho_2} \cap D_{\rho_3}) \\
& - \cdots \\
& + (-1)^{|\mathbf{P}|+1} Pr(\textstyle\bigcap_{\rho \in \mathbf{P}} D_\rho)
\end{aligned}
\tag{9}
$$

For computing the probability that k paths exist, the main operation is counting the number of distinct undirected edges in all k paths. In the worst case where $k = |\mathbf{P}|$ and each path contains all network edges this operation will take $O(e|\mathbf{P}|)$ time where e is the number of edges in CN and can be bounded by $m(m-1)/2$ for a complete graph. So that $O(e|\mathbf{P}|)$ can be rewritten as $O(m^2|\mathbf{P}|)$. We pessimistically consider this bound for computing each term in Equation 9.

The number of terms in the equation is $2^{|\mathbf{P}|} - 1$. Therefore, computing Q_α is $O(m^2|\mathbf{P}|2^{|\mathbf{P}|})$ and consequently computing Q_Γ is $O(m^3|\mathbf{P}|2^{|\mathbf{P}|})$. It is clear that, as $|\mathbf{P}|$ grows, the computation time of Q_Γ becomes prohibitive. In practice, an upper bound on the number of paths considered for each action may be needed. In our experiments, we set this upper bound to 10.

7 Experiments

7.1 Experimental questions

We have described two systems. The first, CREAM, computes optimal actions from a given causal network in a principled way. The ES exhaustive search version (ES) provably gives the optimal action set but may be infeasible in practice, while the greedy version(GS) is not guaranteed to give the best possible result. The second system, ICE-CREAM, combines an inductive causation algorithm with an extension of CREAM. Here, additional uncertainty is introduced by the fact that we have only an approximation of the real causal network.

In this context, CREAM(ES) is the gold standard by which the other algorithms should be compared. The difference in performance between CREAM(ES) and CREAM(GS) relates to the use of a greedy search algorithm. The difference between ICE-CREAM (ES or GS) and the corresponding CREAM relates to uncertainty about the causal network, as well as to the fact that in order to deal with this uncertainty, many approximate methods are introduced.

We will compare these methods also with Yang et al.'s approach [3], which, for brevity and for lack of a better name, we simply refer to as Yang. To our knowledge, this method is the state of the art in economical action mining. Any difference between Yang and (ICE-)CREAM can be related to many things, since the methods are quite different, but an essential difference is that Yang does not use any causal reasoning at all.

The experimental questions we want to address are:

16

1. What is the loss incurred by making CREAM search greedily instead of exhaustively?

2. What is the loss incurred by using the uninformed ICE-CREAM instead of the informed CREAM?

3. How much can one gain by using the computationally complex ICE-CREAM, which tries to use causal reasoning, instead of Yang's method, which is much more efficient but does not use causal reasoning at all?

7.2 Evaluation procedure

Experimental evaluation of action mining methods is difficult. To find out the effect of an action on an object, it needs to be actually performed, so that its impact can be assessed. But existing datasets are by definition observational: we can see what is in the dataset, but we cannot experimentally find out what the target would have been if we had manually changed the value of some input attribute. Thus, the question is how we can mimic such an experimental setting.

We have proceeded as follows. Given a causal model M, one can generate a dataset through random sampling from the distribution defined by the causal model. This dataset is used as input for the system, which next predicts action sets for a number of objects. For each object and action set, since we know the true model M, we can use it to determine the effect of the action set on the object. This simulates the "online experimenting" that one would need in order to evaluate on real datasets.

Our different datasets will have different ranges of costs and profits. To make them more comparable, we express the effectiveness of an action set Γ on an object o using the Normalized Net Profit (nnp), which is defined as

$$nnp(\Gamma, o) = \max(\frac{np(\Gamma, o)}{p_g}, 0), \tag{10}$$

where np is computed according to the real causal network. It is obvious that nnp falls between 0 and 1. It is one when the action set has cost zero and with certainty changes T into t_g. $np(\Gamma, o)/p_g$ can be negative; this happens when the cost of the action set exceeds its expected profit. In such cases, in order to limit the effect of excessive costs, we set nnp to zero.

Concretely, for generating test data, we first select a CN and then simulate observational data based on it. These artificial data then are passed to CREAM, ICE-CREAM, and Yang's method. Finally, the CN is used for measuring the normalized net profit of the action sets recommended by both methods.

We have used the following packages in our implementations:

- "bnlearn", an R package for learning CN structure that implements Margaritis' method. [19, 20]

- "JSmile", a Java library for learning CN parameters and performing inference with the CN [21]

17

Table 1: Artificial networks used for evaluation.

Network Name	# Nodes	# Arcs	# Parameters
sample7	7	6	30
sample15	15	27	372
sample30	30	64	788
sample45	45	103	1010

Table 2: Real networks used for evaluation.

Network Name	# Nodes	# Arcs	# Parameters
Fire	6	5	24
Chest Clinic	8	8	36
usa2000	8	11	255
Headache	12	11	180
Alarm	37	46	752
Hailfinder	56	66	3741

- the "Weka" Java library for decision tree learning, in our implementation of Yang's method [22]

7.3 Data

7.3.1 Causal Networks

We have selected two types of CNs for simulating data: real and artificial. The real networks are gathered from the Bayesian network literature and tools samples. The Alarm and Hailfinder networks are from [23] and [24], respectively. Fire, Headache, Chest Clinic and usa2000 have been chosen from samples of Hugin GUI [25]. The artificial networks were created manually, with their structure and parameters chosen mostly randomly, but in such a way that a meaningful causal network is obtained. Networks were created with a varying number of variables, and all variables are discrete, as for the real networks.

Tables 1 and 2 show the nodes, edges, and parameters in each of the networks. For each network, a single target attribute was chosen randomly and manually among the attributes near the end of the network.

7.3.2 Observational Data

We have generated observational data from each network according to its distribution. The number of objects created for each network was chosen depending on the network size and on memory limitations of the tools used in our experiments. It is shown in Table 3.

Table 3: Number of records generated for each network

Network Name	Number of Records
sample7	12800
sample15	20000
sample30	20000
sample45	20000
Fire	6400
Chest Clinic	2560
usa2000	11644
Headache	262144
Alarm	173328
Hailfinder	50000

7.3.3 Cost Data

We have randomly generated cost data for each data set. A randomly chosen subset of the attributes (20% of the total) was considered stable; these attributes are given a infinite cost.

Among the remaining 80%, some attributes were chosen to be semi-stable, in the sense that changing their value from a to a' has a finite cost but the opposite change has infinite cost (to simulate, for instance, attributes such as "experience level", which can be increased through training but cannot be decreased). 5% of all cost values were set to infinity to model such semi-stable attributes. Other costs were chosen randomly from a uniform distribution between 0 and 10. p_g was set to 100 in all cases.

7.4 Results

We have experimented with both exhaustive and greedy search (ES, GS).

Because ES is computationally prohibitive in large networks, we added a time threshold to this method, which varied from one to a few seconds per object, based on the size of network. When the threshold was exceeded, the best result until then was returned. The exhaustive search ran to completion in four datasets: Fire, ChestClinic, usa2000 and sample7.

For each network, we have repeated the following procedure 10 times (except for Hailfinder, where it was repeated only 5 times due to its computational cost): Generate random cost data for the network (according to the procedure explained above); randomly select 100 objects where $T \neq t_g$; use each method to find the most profitable action for each object in the set; report for each method the average nnp obtained. These results were again averaged over the 10 (or 5) runs with different random costs. The final result is shown in Table 4.

Inspection of the table shows the following.

1. As expected, the results for ICE-CREAM, which is uninformed, are less

Table 4: Average normalized net profit obtained by CREAM, ICE-CREAM and Yang's method on different networks. Asterisks indicate where ES was interrupted.

Network	CREAM(ES)	CREAM(GS)	ICE-CREAM(ES)	ICE-CREAM(GS)	Yang
ChestClinic	0.58	0.58	0.49	0.49	0.41
Fire	0.81	0.81	0.81	0.81	0.80
usa2000	0.75	0.71	0.66	0.59	0.56
Headache *	0.73	0.72	0.71	0.71	0.22
Alarm *	0.56	0.56	0.54	0.54	0.11
Hailfinder *	0.89	0.90	0.80	0.79	0.63
sample7	0.35	0.35	0.34	0.34	0.25
sample15 *	0.39	0.39	0.36	0.36	0.23
sample30 *	0.35	0.37	0.28	0.30	0.14
sample45 *	0.40	0.39	0.35	0.34	0.17

good than for the informed CREAM method. However, the difference is often small. This suggests that ICE-CREAM can be used effectively in real-world problems.

2. ICE-CREAM outperforms Yang in each network. Again, this is not surprising, given that Yang implicitly relies on strong assumptions about causality, which seem unrealistic in practice, and are definitely invalid for the datasets used here.

3. Greedy search works well, in comparison to exhaustive search. On the four datasets where ES could be completed, GS obtained the same result in three datasets, and only slightly worse in the fourth (usa2000). This holds for CREAM as well as for ICE-CREAM.

As said, there are many differences between ICE-CREAM and Yang, so one may wonder to what extent the improved performance is related to the use of causal information, and not simply to the fact that the dataset has a structure that lends itself better to modeling with Bayesian networks (of which causal networks are an instance) than with decision trees. To check this, we have learned a Bayesian network and a decision tree on each dataset, and evaluated it using cross-validation. This type of evaluation tells us how well the used learning method can learn a model that fits the observational data well. Table 5 shows the results. Clearly, decision trees model the data at least as well as Bayesian networks. The difference in performance between ICE-CREAM and Yang is therefore not related to the target function being inherently more suitable for learning Bayesian networks. It is related to the fact that ICE-CREAM explicitly performs causal inference.

Table 5: Predictive accuracy of Bayesian networks and decision trees for the considered datasets.

Network Name	BN	DT
sample7	88.36	88.97
sample15	75.72	83.85
sample30	76.38	78.47
sample45	75.96	77.03
Fire	99.64	99.67
Chest Clinic	84.98	84.98
usa2000	83.20	85.32
Headache	81.32	91.74
Alarm	97.74	99.81
Hailfinder	93.42	99.82

8   Conclusions and Future Work

In this paper, we have introduced the idea of using causal relationships in action mining, and argued why this should lead to better results. We have proposed a novel method called ICE-CREAM that first builds a causal network that indicates hypothesized causal relationships among the variables, then uses this causal network to recommend actions that can be taken to change the state of given objects into some goal state. Experiments show that ICE-CREAM leads to much higher gains than the best known current method for economical action mining, and confirm the expectation that this is due to the explicit use of causal relationships in ICE-CREAM.

While the proposed method is already a clear improvement upon the state of the art in action mining, it is still subject to improvement. Handling the uncertainty in the automatically induced causal network is currently the main bottleneck; the current method is computationally expensive and makes many assumptions that are known to be wrong. Recent results on efficiently computing path probabilities in probabilistic graphs [26] could be exploited to improve this. Further, the current version of ICE-CREAM assumes there are no missing data, handles only discrete attributes, and assumes a binary target attribute.

Our current method is transductive: it does not return generally applicable action rules, but action set recommendations for concrete objects. A simple way to derive an action rule learner from it would be the following: predict action sets for a large number of objects, then use a standard rule learner to learn which actions are predicted under which circumstances.

Another topic for further work is the evaluation of action miners. This evaluation requires experimental control over the data, which means it cannot be performed using existing datasets. Ideally, one should have access to real-life situations that allow for active experimentation. This may be difficult to obtain in practice. Advanced and well-studied methods for simulating such situations would be a welcome alternative.

In principle, the proposed method is not limited to economical applications, but could also be useful in, for instance, life sciences. It is to be expected, however, that as the complexity of the real causal relationships increases, the difference between approximate causal inference using an incomplete network and using no causal reasoning at all diminishes. It remains to be explored for what range of applications the method is practically useful.

Acknowledgements

References

[1] Z. Ras and A. Wieczorkowska, "Action-rules: How to increase profit of a company," *Principles of Data Mining and Knowledge Discovery*, pp. 75–116, 2000.

[2] Q. Yang, J. Yin, C. Ling, and R. Pan, "Extracting actionable knowledge from decision trees," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 1, pp. 43–56, 2007.

[3] Q. Yang, J. Yin, C. Ling, and T. Chen, "Postprocessing decision trees to extract actionable knowledge," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 685–688.

[4] Z. Ras and L. Tsay, "Discovering extended action-rules (system dear)," *Intelligent Information Systems*, pp. 293–300, 2003.

[5] L. Tsay and Z. Ras, "Action rules discovery: System dear2, method and experiments," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 17, no. 1-2, pp. 119–128, 2005.

[6] ——, "Action rules discovery system dear_3," *Foundations of Intelligent Systems*, pp. 483–492, 2006.

[7] J. Pearl, *Causality: models, reasoning, and inference*. Cambridge Univ Press, 2000, vol. 47.

[8] P. Spirtes, C. Glymour, and R. Scheines, *Causation, prediction, and search*. MIT press, 2001, vol. 81.

[9] T. Verma and J. Pearl, "Equivalence and synthesis of causal models," *Uncertainty in artificial intelligence 6*, vol. 6, p. 255, 1991.

[10] G. Piatetsky-Shapiro and C. Matheus, "The interestingness of deviations," in *Proceedings of the AAAI-94 workshop on Knowledge Discovery in Databases*, vol. 1, 1994, pp. 25–36.

[11] A. Silberschatz and A. Tuzhilin, "On subjective measures of interestingness in knowledge discovery," in *Proceedings of KDD-95: First International Conference on Knowledge Discovery and Data Mining*, 1995, pp. 275–281.

[12] Z. Ras, E. Wyrzykowska, and H. Wasyluk, "Aras: Action rules discovery based on agglomerative strategy," *Mining Complex Data*, pp. 196–208, 2008.

[13] K. Wang, Y. Jiang, and A. Tuzhilin, "Mining actionable patterns by role models," in *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*. IEEE, 2006, pp. 16–16.

[14] A. Tzacheva and Z. Ras, "Association action rules and action paths triggered by meta-actions," in *Granular Computing (GrC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 772–776.

[15] Z. He, X. Xu, S. Deng, and R. Ma, "Mining action rules from scratch," *Expert Systems with Applications*, vol. 29, no. 3, pp. 691–699, 2005.

[16] C. Silverstein, S. Brin, R. Motwani, and J. Ullman, "Scalable techniques for mining causal structures," *Data Mining and Knowledge Discovery*, vol. 4, no. 2, pp. 163–192, 2000.

[17] A. Darwiche, *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009, vol. 54, no. 2.

[18] D. Margaritis, "Learning bayesian network model structure from data," DTIC Document, Tech. Rep., 2003.

[19] M. Scutari, "Learning bayesian networks with the bnlearn r package," *Arxiv preprint arXiv:0908.3817*, 2009.

[20] ——, "bnlearn, bayesian network structure learning," http://www.bnlearn.com/.

[21] decision system laboratory of Pittsburgh university, "Jsmile, java interface for smile," http://genie.sis.pitt.edu/about.html.

[22] the university of Waikato, "Weka, data mining software in java," http://www.cs.waikato.ac.nz/ml/weka/.

[23] I. Suermondt and R. Chavez, "The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks."

[24] B. Abramson, J. Brown, W. Edwards, A. Murphy, and R. Winkler, "Hailfinder: A bayesian system for forecasting severe weather," *International Journal of Forecasting*, vol. 12, no. 1, pp. 57–71, 1996.

[25] "Hugin lite 7.5 samples," http://www.hugin.com.

[26] B. Gutmann, I. Thon, A. Kimmig, M. Bruynooghe, and L. De Raedt, "The magic of logical inference in probabilistic programming," *TPLP*, vol. 11, no. 4-5, pp. 663–680, 2011.