Learning the Causal Structure of Overlapping Variable Sets

David Danks

Institute for Human and Machine Cognition
University of West Florida
40 S. Alcaniz St.
Pensacola, FL 32501 U.S.A.
ddanks@ai.uwf.edu

Abstract. In many real-world applications of machine learning and data mining techniques, one finds that one must separate the variables under consideration into multiple subsets (perhaps to reduce computational complexity, or because of a shift in focus during data collection and analysis). In this paper, we use the framework of Bayesian networks to examine the problem of integrating the learning outputs for multiple overlapping datasets. In particular, we provide rules for extracting causal information about the true (unknown) Bayesian network from the previously learned (partial) Bayesian networks. We also provide the SLPR algorithm, which efficiently uses these previously learned Bayesian networks to guide learning of the full structure. A complexity analysis of the "worst-case" scenario for the SLPR algorithm reveals that the algorithm is always less complex than a comparable "reference" algorithm (though no absolute optimality proof is known). Although no "expected-case" analysis is given, the complexity analysis suggests that (given the currently available set of algorithms) one should always use the SLPR algorithm, regardless of the underlying generating structure. The results provided in this paper point to a wide range of open questions, which are briefly discussed.

1 The "Big Picture" Problem

Modern data collection has advanced to the point that the size and complexity of our datasets regularly exceed the computational limits of our algorithms (on modern machines). As a result, analysis is often rendered computationally tractable only when we consider proper subsets of the variables we have measured. In addition, the variables thought to be relevant often change over the course of an investigation, both in the data collection and analysis phases. For example, an unexpected correlation might suggest the need to find an unmeasured common cause. When these changes occur, we want to use as much information as possible from earlier analyses to minimize duplication of effort. Thus, in both of these situations, we must address the distinctive problems (such as integration of outputs and efficient use of prior learning in subsequent learning) that arise for learning on multiple overlapping sets of variables.

There is a further, more practical, motivation. Many social science datasets have overlapping variables but the datapoints are unlabelled (for privacy reasons), so that there is no possible way to create a "complete" dataset. For example, we might have a census dataset and an unemployment dataset, both of which have *Income* as a variable, but neither of which contains identifiers to be used for creation of a single, integrated dataset. Hence, in these domains where there are substantial practical barriers to creating a unified dataset, the problem of integrating the learning outputs becomes particularly salient.

At the same time as we face these difficulties, we want our analysis techniques to reveal causal relationships. Causal information allows for predictions about the (probabilistic) outcomes of interventions, and is more easily understood by human users of machine learning techniques. Hence, if possible, we want to use a representation that allows for causal inference and prediction.

To better understand these problems, we can try to express them more formally. Let $\boldsymbol{V}$ be the full set of variables under consideration. We assume that the variables are either all discrete or all continuous, though in the former case they need not have the same number of values. Let $\boldsymbol{S}_1, ..., \boldsymbol{S}_n$ be (nonempty) subsets of $\boldsymbol{V}$ such that $\boldsymbol{S}_1 \cap ... \cap \boldsymbol{S}_n \neq \emptyset$. We further assume that, throughout all stages of learning, there is some stationary generating process producing the data, and that we have sufficient data that the sample statistics are the same as the population statistics. In this paper, we will be concerned with the following two questions:

1. If we do not have joint data over $\boldsymbol{V}$, but we do have the outputs of some reliable, correct learning process (e.g., a machine learning algorithm) on $\boldsymbol{S}_1, ..., \boldsymbol{S}_n$, what can we learn about the relationships among $\boldsymbol{V}$ as a whole? That is, if we can only learn the causal structure of the subsets (because of lack of data), what can we learn (if anything) about the full structure underlying $\boldsymbol{V}$?
2. If we *do* have joint data over $\boldsymbol{V}$, as well as the learning outputs, how can we efficiently learn the full structure for $\boldsymbol{V}$? That is, how (if at all) can we use learning results over subsets to guide the learning for $\boldsymbol{V}$ as a whole?

For example, we might have three datasets (drawn from the same population) over the following variables:

1. $\boldsymbol{S}_1 = \{Education, ParentalEducation, Income\}$;
2. $\boldsymbol{S}_2 = \{Education, Housing, Income\}$; and
3. $\boldsymbol{S}_3 = \{Education, Age, NumberOfChildren, Income\}$.

In this example, the above two questions correspond to: (1) Given just these three datasets, what can be learned about the interrelationships among the six variables (including pairs, like *Housing* and *Age*, that do not appear in the same dataset)? and (2) How could we efficiently learn the full causal structure if we actually had a complete dataset?

On one level, it will be surprising if the answer to question 1 is anything other than "nothing." Any positive answer implies that we can determine something

about the relationship between two variables, despite the fact that we have no dataset that contains all of the (possibly) relevant variables, perhaps including the two target variables. Interestingly, we will find in Section 3 that, despite this restriction, we can still learn something in this situation.

Question 2 is more straightforward, since we would expect that some speed-up would be possible (since we have already done some learning). An algorithm that uses the initial learning is provided in Section 4.1. The complexity analysis in Section 4.2 then shows that the algorithm is always more efficient than a comparable reference algorithm on the worst-case scenario. That is, for all parameterizations of the worst-case, we always gain an advantage by using prior learning.

A question similar in spirit was previously considered by Fienberg and Kim in the context of log-linear models (a type of undirected graph) [5]. They considered the problem of finding the full structure for $\boldsymbol{V} \cup X$ when we are given each marginal structure for $\boldsymbol{V}$, conditional on some value of X. Since there are conditions under which log-linear models can model a Bayesian network (the formalism used here), their work shares a similar flavor to the ideas in this paper. It differs, however, by considering variable addition (rather than overlap), focusing on a particular parameterization, and assuming that the full data is (implicitly) available.

Before proceeding on to the heart of the paper, we note that, for the purposes of this paper, we will restrict ourselves to the case of exactly two overlapping sets, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, but we will not assume that one set is a proper subset of the other.

2 Bayesian Networks

We cannot address the two principal questions of this paper without examining them from within a particular formalism. In this paper, we will use Bayesian networks (or simply Bayes nets). In addition, we will assume that all of the variables in $\boldsymbol{V}$ are discrete. Nothing in this paper hinges on the latter assumption; there are corresponding representations for continuous variables, and the SLPR algorithm (presented in Section 4.1) refers only to (conditional) independence, which is defined for both discrete and continuous variables. The remainder of this section is meant simply as a quick overview of Bayes net terminology and concepts. Several excellent introductions to Bayes nets are [7,11,12,14]; more detailed snapshots of the current state of the field can be found in [7,9].

2.1 The Bayes Net Formalism

Suppose that $\boldsymbol{V} = \{V_1, ..., V_n\}$ is a set of random variables, and let $\boldsymbol{v} = \{v_1, ..., v_n\}$ be the values of the variables for some datapoint. A Bayes net for $\boldsymbol{V}$ is composed of two inter-connected elements:

- A directed acyclic graph over $\boldsymbol{V}$; and

– A joint probability distribution (j.p.d.) over the possible joint variable values.

There is a node for each variable in $\boldsymbol{V}$ in the graph, and nodes are (sometimes) connected by an edge with exactly one arrowhead. The edge points from the *parent* variable to the *child* variable. We use similar relation terminology (e.g., ancestor, descendant) to describe other variables. So, for example, in the simple graph $A \rightarrow B \leftarrow C$, A, C are B's parents, and B is A, C's child. Every variable is its own ancestor and descendant, so $\{A, B, C\}$ are B's ancestors. B is also called a *collider* in the above graph, since the two edges "collide" at it. A *directed path* from V_i to V_j is a sequence of the form $V_i \rightarrow ... \rightarrow V_j$, with all edges oriented in the same direction. A *trek* is a pair of directed paths (possibly one with zero length) that are both from the same variable (called the *source*). A graph is *acyclic* if there is no directed path (of non-zero length) from a variable to itself.

The graph and j.p.d. are related[1] through the following two assumptions:

Markov Assumption A variable A is (probabilistically) independent of all (graphical) non-parental non-descendants, conditional on A's (graphical) parents.

Faithfulness (Stability) Assumption If variables A and B are (probabilistically) independent conditional on set $\boldsymbol{S}$, then there is no (graphical) edge between them.

Notice that these two assumptions are each other's converse: the Markov assumption says "No edge implies conditional independence" and the Faithfulness assumption says "Conditional independence implies no edge."

The Markov assumption enables us to decompose the j.p.d. into the product of n simpler probabilities based on the graph. Specifically, if $pa\left(V_i\right)$ is the set of parents of V_i, then the j.p.d. can be expressed as:

$$P\left(V_1, ..., V_n\right) = \prod_{i=1}^{n} P\left(V_i | pa\left(V_i\right)\right) \tag{1}$$

For example, any Markov j.p.d. for the graph $X \rightarrow Y \rightarrow Z$ must factor as $P\left(X, Y, Z\right) = P\left(X\right) P\left(Y|X\right) P\left(Z|Y\right)$.

The Faithfulness assumption rules out cases in which two pathways exactly cancel each other out. That is, if there is an edge between two variables, they must be associated regardless of conditioning set. For example, suppose that running increases your metabolism, which would normally lead you to lose weight. But suppose it also makes you hungrier, so you eat more and (normally) gain weight. Faithfulness (or rather, a causal version of it) states that these two processes do not exactly cancel out (so that your weight gain or loss is independent

[1] I deliberately use the neutral term "related." Contextual factors determine which component is primary. In most machine learning contexts, we have a j.p.d. and we want to learn the graph; in most expert knowledge contexts, we have a graph and we want to make predictions about the j.p.d. In this paper, we will operate in both contexts: from graph to j.p.d. (and back to graph) in Section 3, and from j.p.d. to graph in Section 4.

of whether you go running). This assumption is necessary to learn the most parsimonious graph(s) for a particular j.p.d., and is assumed by all current Bayes net learning methods (discussed below).

Most importantly for our present purposes, Bayes nets have proven to be excellent models of causation. In particular, the "asymmetry of intervention" (i.e., an intervention on a variable influences its effects, but not its causes) is quite easily represented in a Bayes net, and we can thus predict the (probabilistic) result of a particular intervention [12,14]. Furthermore, the learning algorithms discussed in the next section have been shown to accurately recover causal structure (in appropriate situations). There are debates about the exact nature of the assumptions needed to extract causal information from data (e.g., [10,15]); we will not deal with those issues in this paper.

2.2 Learning Bayes Nets

Two different strategy types have been used for learning Bayes nets from data: constraint-based search and Bayesian updating. These strategy types differ in process, not in asymptotic behavior, though they do have different performance profiles. Constraint-based procedures (e.g., [3,14]) use patterns of conditional and unconditional independencies and associations in the data to determine the equivalence class of graphs that could possibly have produced that pattern. An equivalence class (in this context) is a set of graphs that all imply (by the Markov and faithfulness conditions) the same conditional and unconditional associations and independencies. The equivalence class output is usually represented as a partially directed graph with associated rules for transforming the output into the full equivalence class.

For Bayesian updating (e.g., [2,6,8]), we assign a probability distribution to the space of possible graphs. That distribution encodes our prior beliefs about the likelihood of each graph. As we receive data, we use standard Bayesian updating to revise both the probability distribution over the search space, and also the parameter probability distributions for each possible graph. The output produced by a Bayesian procedure is thus an updated (and asymptotically correct) probability distribution for the space of possible graphs.

Bayesian search procedures are more flexible than constraint-based procedures, and often give more useful information. However, if we have n different variables in our system, then the number of possible graphs is at least exponential in n. Therefore, Bayesian search procedures are only practical if we use a series of heuristics, even though we can prove that the use of these heuristics is sometimes incorrect.

There are also barriers peculiar to this problem to using Bayesian updating. There is no principled way (at present) to "redistribute" a probability distribution for graphs over $\boldsymbol{S}_1$ or $\boldsymbol{S}_2$ into a probability distribution for graphs over $\boldsymbol{V}$ as a whole [3]. In addition, even if we had such a method, we would potentially need to resolve conflicts in the marginal distributions over the shared variables, and there are theorems suggesting that there may not be principled Bayesian solutions for

such conflicts [13]. Hence, we will focus on constraint-based procedures for the remainder of this paper.

We can now recast the questions from Section 1 in Bayes net terms. Suppose we are attempting to learn some causal structure for a large set of variables $\boldsymbol{V}$, and suppose $\boldsymbol{V}$ is divided into overlapping subsets $\boldsymbol{A}$ and $\boldsymbol{B}$. We will primarily be interested in the three distinct (non-empty) subsets: the shared variables $\boldsymbol{M} = \boldsymbol{A} \cap \boldsymbol{B}$, and the unshared variables $\boldsymbol{X} = \boldsymbol{A} \setminus \boldsymbol{M}$ and $\boldsymbol{Y} = \boldsymbol{B} \setminus \boldsymbol{M}$. Further suppose that we have learned the patterns (i.e., the equivalence classes of graphs) that are Markov and faithful to $\boldsymbol{A}$ and $\boldsymbol{B}$: $Patt_{\boldsymbol{A}}$ and $Patt_{\boldsymbol{B}}$, respectively. Let $Patt_{\boldsymbol{V}}$ be the (unknown) pattern that is Markov and faithful to (data over) $\boldsymbol{V}$. The questions from Section 1 can now be re-expressed as:

1. Given only $Patt_{\boldsymbol{A}}$ and $Patt_{\boldsymbol{B}}$ as inputs, what can we determine about possible edges in $Patt_{\boldsymbol{V}}$? (discussed in Section 3)
2. Given $Patt_{\boldsymbol{A}}$, $Patt_{\boldsymbol{B}}$, and a sufficiently large dataset of complete data over $\boldsymbol{V}$, is there an algorithm for learning $Patt_{\boldsymbol{V}}$ that is less computationally complex than a standard learning algorithm that uses only the data over $\boldsymbol{V}$? (discussed in Section 4)

We also introduce one further assumption here:

Causal Sufficiency Assumption $\boldsymbol{V}$ contains all common causes of variables in $\boldsymbol{V}$.

This assumption is essentially a closure assumption that says that there are no unobserved common causes (though there can be other unobserved causes of variables in $\boldsymbol{V}$). The appropriateness of this assumption is obviously quite context- and knowledge-dependent. It is, however, assumed by all Bayesian learning algorithms, and is regularly assumed in practice by users of constraint-based methods (at least at first). We will assume causal sufficiency for the remainder of this paper.

3 Integrating Overlapping Bayes Nets

3.1 Edge Removal Rules

In this section, we attempt to determine whether we can learn anything about the structure of (the unknown) $Patt_{\boldsymbol{V}}$ simply through consideration of $Patt_{\boldsymbol{A}}$ and $Patt_{\boldsymbol{B}}$. At first glance, we might naturally think that nothing at all could be said about the pattern for the full causal structure. Answering the question requires making claims about the presence or absence of edges (i.e., causal relationships) without having *any* datapoints with values for every variable in $\boldsymbol{V}$. There seems to be no obvious reason why we should be able to determine anything at all, since the unobserved values seemingly might provide the crucial information for determining whether a particular edge should be present or absent.

This intuition is quite reasonable, and does apply to whether an edge should be present; it does *not*, however, apply to the absence of an edge. Removal of

an edge between two variables U and W indicates that they are independent conditional on some set $\boldsymbol{T}$ (namely, the parents of at least one of the variables). Hence, these variables will be non-adjacent in any graph for variables $\boldsymbol{R}$ with $\boldsymbol{T} \subseteq \boldsymbol{R}$. Since $\boldsymbol{V}$ contains both $\boldsymbol{A}$ and $\boldsymbol{B}$, we can use the following rule:

Edge Removal Rule 1 If U and W are not adjacent in $Patt_{\boldsymbol{A}}$ or $Patt_{\boldsymbol{B}}$, then they must also be not adjacent in $Patt_{\boldsymbol{V}}$.

In addition to enabling us to remove some edges from $Patt_{\boldsymbol{V}}$ without checking independencies, we can also use this rule to resolve conflicts when $U, W \in \boldsymbol{M}$ and U and W are adjacent in $Patt_{\boldsymbol{A}}$, but not $Patt_{\boldsymbol{B}}$ (or vice versa), since absence in either sub-pattern implies absence in the full pattern.

The above result essentially allows us to import the independencies encoded in $Patt_{\boldsymbol{A}}$ and $Patt_{\boldsymbol{B}}$. More surprisingly, we can sometimes remove edges between variables $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$, even though we do not have a single datapoint that contains values for both variables. Before showing how this process can work, we first prove a more general result. We define the "reachable ancestors" for a variable X relative to a "blocking set" $\boldsymbol{T}$ as:

$$RA\left(X, \boldsymbol{T}\right) = \{Y : \exists \boldsymbol{Z} = \{Z_1, ..., Z_n\} \text{ (possibly empty) s.t. } Y \rightarrow Z_1 \rightarrow ... \rightarrow Z_n \rightarrow X \wedge \forall i \left(Z_i \notin \boldsymbol{T}\right)\}$$

That is, $RA\left(X, \boldsymbol{T}\right)$ contains just those variables that have a directed path to X containing no variables in $\boldsymbol{T}$. Using "$X \perp Y | \boldsymbol{T}$" as shorthand for "$X$ is independent of Y conditional on set $\boldsymbol{T}$," we can state the following theorem:

Theorem 1. *Assume faithful and Markov data for graph $\boldsymbol{G}$. If $X \perp Y | \boldsymbol{T}$, then X and $Z \in RA\left(Y, \boldsymbol{T}\right)$ are non-adjacent in $\boldsymbol{G}$.*

Proof. Proof of contrapositive. Suppose some $Z \in RA\left(Y, \boldsymbol{T}\right)$ is adjacent to X in $\boldsymbol{G}$. Then since (by definition of $RA\left(Y, \boldsymbol{T}\right)$) there is an unblocked (by $\boldsymbol{T}$) directed path from Z to Y, there is an unblocked (by $\boldsymbol{T}$) trek between X and Y. Therefore, X and Y are associated conditional on $\boldsymbol{T}$. Q.E.D.

We can use Theorem 1 for our integration problem, because a special case of this theorem is:

Edge Removal Rule 2 If there exists $X \in \boldsymbol{X}$, $Y \in \boldsymbol{Y}$, $M \in \boldsymbol{M}$ such that (i) there is a directed path from X to M in $Patt_{\boldsymbol{A}}$ involving only variables in $\boldsymbol{X}$; and (ii) M and Y are not adjacent in $Patt_{\boldsymbol{B}}$, then X and Y are not adjacent in $Patt_{\boldsymbol{V}}$.

The lack of an edge between Y and M in $Patt_{\boldsymbol{B}}$ indicates an independence conditional on some subset of $\boldsymbol{Y} \cup \boldsymbol{M}$ (since $Patt_{\boldsymbol{B}}$ is faithful and Markov to the generating process marginalized to $\boldsymbol{Y} \cup \boldsymbol{M}$). Therefore, any ancestor of M that is connected only through elements of $\boldsymbol{X}$ (and so only variables in $\boldsymbol{X}$ are candidates) is reachable, and so there cannot be an edge between Y and those

variables. Moreover, Edge Removal Rule 2 results in the removal of as many edges as possible based solely on Theorem 1.

If we know something about the parameterization of the Bayes net, then we can sometimes remove more edges. For example, correlations in linear systems obey the generalized trek rule: $\rho_{i.j} = \sum_{t \in \boldsymbol{T}} \prod_{k=0}^{t_n - 1} \rho_{k.k+1}$, where $\rho_{i.j}$ is the correlation between V_i and V_j, $\boldsymbol{T}$ is the set of all treks between V_i and V_j, and k (in the product) indexes over the variables on a particular trek t. Bayes nets with only binary variables have a similar, though not identical, decomposition [4]. In these cases, we can attempt to remove edges between variables $M_1, M_2 \in \boldsymbol{M}$ by determining whether the observed correlation matches the correlations predicted by the two patterns. In the interest of limiting the assumptions about the underlying processes, we do not pursue these "special case" rules further.

3.2 A Toy Example

Suppose the true underlying graph is: $U \rightarrow X \rightarrow Y \leftarrow Z$, and that $\boldsymbol{A} = \{U, X, Y\}$ and $\boldsymbol{B} = \{X, Y, Z\}$. If we assume that the data are Markov and faithful to this underlying graph, then we can analytically determine the patterns for $\boldsymbol{A}$ and $\boldsymbol{B}$. Specifically, $Patt_{\boldsymbol{A}} = U - X - Y$, whose equivalence class is $\{U \rightarrow X \rightarrow Y,\ U \leftarrow X \rightarrow Y,\ U \leftarrow X \leftarrow Y\}$, and $Patt_{\boldsymbol{B}} = X \rightarrow Y \leftarrow Z$, which has a singleton equivalence class (namely, itself).

For $Patt_{\boldsymbol{V}}$, we start with the complete pattern, in which every pair of variables is connected by an undirected edge. By Edge Removal Rule 1, we can remove two edges: $U - Y$ and $X - Z$. We can also use Edge Removal Rule 2, since there is a directed path from U to X (in this case, involving no other variables), and X and Z are not adjacent in $Patt_{\boldsymbol{B}}$. Hence, there cannot be an edge between U and Z. Thus, by applying both of the edge removal rules (and incorporating orientation information), we get the output pattern: $U - X \rightarrow Y \leftarrow Z$, which is the correct pattern for the underlying graph (i.e., it describes the equivalence class of which the underlying graph is a member).

3.3 Piecewise Causal Learning

The example in Section 3.2 is, as the section title suggests, simply a toy example. It does, however, point towards situations in which these edge removal rules have non-trivial consequences. In particular, these rules can significantly advance causal learning by computationally limited agents, such as people.

Suppose we have a causal system with a relatively large number of variables. Directly learning the structure of this system requires (in some sense) the ability to keep all of the variables "in mind" at one point in time. For agents with limited memory or computational abilities, we might consider a piecewise learning procedure. Specifically, we might consider only single variables, and attempt to learn that variable's parents (i.e., direct causes) with a standard learning algorithm, without worrying about larger-scale structure. We then must integrate the local causal information, but notice that the above edge removal rules will apply to any case in which the focus variable is later found to be a parent of

another variable. That is, we might not have to consider every variable as a possible effect in order to learn substantial portions of the causal structure.

To make the applicability more obvious, consider a concrete (though abstract) example. Suppose we have some large set of variables, we choose a variable Y in that set (perhaps at random), and we learn its direct causes: $X_1, ..., X_n$. Now suppose that we learn that the parents of some other variable Z are: $Y, A_1, ..., A_m$. We can use the edge removal rules to conclude immediately that there are no edges between the X's and any of the A's that are independent of Y. Hence, we can potentially get quite close to the true causal structure simply by learning local causal "families" (a child and its parents).

This rough strategy is, of course, not fully fleshed out. We must give a better account of exactly how the composition procedure works, as well as what to do when it *doesn't* work. In addition, this strategy is designed to aid computationally limited agents, but currently-developed methods for learning the structure of causal families are not practical for those types of agents. Nevertheless, some theoretical psychological work suggests that humans might actually learn large-scale causal structures in this compositional manner (e.g., [1]).

4 Efficient Learning through Prior Learning

4.1 SLPR Algorithm

We will not typically be as fortunate as in the examples in Sections 3.2 and 3.3, and we will thus be unable to reach the correct answer using only the edge removal rules. Therefore, we now turn to the second problem: what algorithm will most efficiently learn $Patt_{\boldsymbol{V}}$ when given $Patt_{\boldsymbol{A}}$ and $Patt_{\boldsymbol{B}}$, as well as complete data over $\boldsymbol{V}$ as inputs? Since we have complete data, we know that we can learn the Markov and faithful pattern for the generating process; this question asks how efficiently we can do it. In this section, we will focus on describing an algorithm that is more efficient than learning $Patt_{\boldsymbol{V}}$ completely from scratch, and not attempt to prove that it is the most efficient algorithm possible.[2]

At the very least, we can (sometimes) improve the efficiency of a learning algorithm by using the rules from Section 3.1 as a preprocessor that removes some edges without checking conditional independencies. This change, however, does not take advantage of all of the information in the input patterns. We could also use the algorithms described in [3] (which enable one to add variables to a graph after learning) by treating the variables in either $\boldsymbol{X}$ or $\boldsymbol{Y}$ (whichever is smaller) as the variables added to the graph. Even this more efficient algorithm is almost certainly sub-optimal, however, since it only uses the edge information in one of $Patt_{\boldsymbol{A}}$ and $Patt_{\boldsymbol{B}}$.

We propose below the SLPR (Structure Learning using Prior Results) algorithm as more efficient than either of the above proposals. Before describing the algorithm, we must define one term and two variables: (i) $\boldsymbol{T}$ is a *separating*

[2] The discussion below will give us reason to think that this algorithm is close to optimal; we have no proof, however, showing that it actually is optimal.

set for A and B iff $A \perp B|\boldsymbol{T}$; (ii) $SepSet\,(A,B)$ stores any separating sets for variables A and B discovered in earlier learning; and (iii) $Adj\,(\boldsymbol{G},X,Y)$ is the set of variables adjacent to either X or Y in the graph $\boldsymbol{G}$, excluding X and Y themselves. The SLPR algorithm takes an "oracle" as input to (i) enable the algorithm to be applicable to any data over which (conditional) independence is defined; and (ii) avoid discussions of particular statistical tests.

SLPR Algorithm

Inputs: (i) $Patt_{\boldsymbol{A}=\boldsymbol{X}\cup\boldsymbol{M}}$; (ii) $Patt_{\boldsymbol{B}=\boldsymbol{Y}\cup\boldsymbol{M}}$; (iii) the $SepSet$ functions from prior learning; and (iv) an "oracle" for determining independence relations.

Output: $Patt_{\boldsymbol{V}}$, the Markov and faithful pattern for the generating causal structure for $\boldsymbol{V}$

1. Form the complete (undirected) graph $\boldsymbol{G}$ over $\boldsymbol{V}$. For $Patt_{\boldsymbol{A}}$, if some U and W are non-adjacent, then remove the edge between them in $\boldsymbol{G}$; otherwise, orient the edge in $\boldsymbol{G}$ as it is oriented in $Patt_{\boldsymbol{A}}$. Perform the same operations for $Patt_{\boldsymbol{B}}$ and resolve conflicts by: (a) excluding an edge if at least one pattern excludes it; and (b) making an edge unoriented if the two patterns disagree about its orientation.
2. For all $X \in \boldsymbol{X}$, $Y \in \boldsymbol{Y}$, $M \in \boldsymbol{M}$ such that (i) there is a directed path from X (or Y) to M involving only nodes in $\boldsymbol{X}$ ($\boldsymbol{Y}$); and (ii) Y (X) and M are non-adjacent in $\boldsymbol{G}$, remove the edge between X and Y.
3. $n = 0$
   repeat TEST_SETS_OF_SIZE_N {
   repeat SELECT_PAIR_AND _CHECK_INDEP {
   Select an ordered pair of variables (X,Y) such that (i) $X \in \boldsymbol{X}, Y \in \boldsymbol{Y}$; (ii) X and Y are adjacent in $\boldsymbol{G}$; and (iii) $|Adj\,(\boldsymbol{G},X,Y)| \leq n$.
   If $\exists \boldsymbol{T} \subseteq Adj\,(\boldsymbol{G},X,Y)$ such that (i) $|\boldsymbol{T}| = n$; and (ii) $X \perp Y|\boldsymbol{T}$, then (i) remove $X - Y$ from $\boldsymbol{G}$; (ii) record $\boldsymbol{T}$ in $SepSet\,(X,Y)$ and $SepSet\,(Y,X)$; and (iii) remove all edges between X and $Z \in RA\,(Y,\boldsymbol{T})$ (and similarly for Y and $RA\,(X,\boldsymbol{T})$).
   } until all ordered pairs (X,Y) that satisfy the preconditions have been tested.
   $n = n + 1$
   } until $|Adj\,(\boldsymbol{G},X,Y)| < n$ for all appropriate ordered pairs (X,Y).
4. For each triple A, B, C such that (i) A, B are adjacent; (ii) B, C are adjacent; and (iii) A, C are not adjacent, orient $A - B - C$ as $A \rightarrow B \leftarrow C$ iff $B \notin SepSet\,(A,C)$.
5. For each pair of adjacent variables U, W (except $U \in \boldsymbol{X}$ and $W \in \boldsymbol{Y}$, or *vice versa*) define

$$\boldsymbol{C} = \bigcup \begin{cases} \boldsymbol{X}, \text{ if } U \in \boldsymbol{X} \text{ or } W \in \boldsymbol{X} \\ \boldsymbol{Y}, \text{ if } U \in \boldsymbol{Y} \text{ or } W \in \boldsymbol{Y} \\ \boldsymbol{M}, \text{ if } U \in \boldsymbol{M} \text{ or } W \in \boldsymbol{M} \end{cases}$$

   I.e., $\boldsymbol{C}$ is the union of the containing sets (out of $\boldsymbol{X},\boldsymbol{Y},\boldsymbol{M}$) for U and W.
   If there exists a path Q between U and W such that (i) $Q = \{C_1,...,C_m\} \cup$

$\{A_1, ..., A_n\}$, with $\forall i\,(C_i \in \boldsymbol{C})$, $\forall j\,(A_j \in \boldsymbol{V} \setminus \boldsymbol{C})$; (ii) $\{A_j\} \neq \emptyset$ (though it is possible that $\{C_i\} = \emptyset$); (iii) every C_i (if any) is possibly a collider and every A_j is possibly a non-collider on Q; and (iv) every C_i (if any) is possibly an ancestor of either U or W.
Then if $\exists \boldsymbol{T} \subseteq Adj\,(\boldsymbol{G}, U, W)$ such that $\exists i\,(A_i \in \boldsymbol{T})$ and $U \perp W | \boldsymbol{T}$, then remove $U - W$ from $\boldsymbol{G}$ and record $\boldsymbol{T}$ in $SepSet\,(U, W)$ and $SepSet\,(W, U)$.

6. Unorient all of the edges in $\boldsymbol{G}$ and reorient using the following two steps (in order):
   (a) For each triple A, B, C such that (i) A, B are adjacent; (ii) B, C are adjacent; and (iii) A, C are not adjacent, orient $A - B - C$ as $A \rightarrow B \leftarrow C$ iff $B \notin SepSet\,(A, C)$.
   (b) Repeatedly apply the following two rules until no more edges can be oriented:
      i. If (i) $A \rightarrow B - C$; and (ii) A, C are not adjacent, then orient $B - C$ as $B \rightarrow C$.
      ii. If there is a directed path from A to B, and A and B are adjacent, then orient $A - B$ as $A \rightarrow B$.

Step 2 is simply an application of Rule 2. Further note that despite the apparent complexity of the preconditions in that rule, it is computationally quite simple. We simply iterate through the nodes in $\boldsymbol{M}$ and, for each node $M \in \boldsymbol{M}$, determine the non-adjacent nodes $X \in \boldsymbol{X}$ (and $Y \in \boldsymbol{Y}$), and then recursively remove edges between the non-adjacent nodes and variable parents (not in $SepSet\,(M, X)$), starting with adjacent (to M) variables in $\boldsymbol{Y}$ ($\boldsymbol{X}$). The path-checking in the preconditions for Step 2 occurs implicitly in this procedure.

Step 3 tests the variables that have not previously appeared in the same dataset. For these pairs of variables, $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$, we need to determine whether they are independent, conditional on some (sub)set of the adjacent variables. We only need to consider adjacent variables because, if X and Y are not adjacent in the true generating structure, then they must be independent conditional on one of their parents. Since the edges between a variable and its parents are never removed, the variables currently adjacent to X (and similarly for Y) must be a superset of its parents. Hence, we do not need to condition on all variables, but only those that could possibly be parents of one of the variables.[3]

Step 5 is necessary because of the possibility that some edges in $Patt_{\boldsymbol{A}}$ are present only because of the existence of common causes in $\boldsymbol{V}$ that are outside of $\boldsymbol{A}$ (similarly for $Patt_{\boldsymbol{B}}$ and $\boldsymbol{B}$). For example, if the true generating structure contains $X_1 \leftarrow Y \rightarrow X_2$, then there will be an incorrect edge between X_1 and X_2 in $Patt_{\boldsymbol{A}}$ (since they are associated regardless of conditioning set). This step

[3] It is actually possible to make this step slightly more efficient, by considering subsets of variables adjacent to X, followed by subsets of variables adjacent to Y (since we only need to condition on the parents of one of the variables). However, this modification is more difficult to express, more difficult to implement, and makes no difference for the "worst-case" scenario discussed in the complexity analysis of Section 4.2.

removes those edges that remained because of the restricted variable sets of the initial learning.

We do not necessarily need to check all pairs of adjacent variables, but only those with at least one (other) path between them that might explain the presence of the edge. The characteristics of such a possible inducing path are rather complicated, but all necessary [3]. However, we only need to find one such path to trigger the conditional independence checks. Therefore, the added computational burden of path-checking is relatively minor.

4.2 Complexity Analysis

Recall that the original question for Section 4 centered on finding a more efficient algorithm. In this section, we determine the conditions (for the worst-case) under which the SLPR algorithm is more efficient than a comparable constraint-based algorithm. This complexity analysis focuses only on the number of conditional independence tests we must perform. Orientation is a computationally simple task compared to the conditional independence tests, and so we should expect the latter to dominate the algorithmic complexity.

In a slight abuse of notation (that greatly simplifies our formulae), we define $|\boldsymbol{X}| = x$, $|\boldsymbol{Y}| = y$, and $|\boldsymbol{M}| = m$. Hopefully there will be no confusion with the notation introduced in Section 2.1, since we will not be referring to the values of individual variables in this section.

Recall that we are interested in determining whether the SLPR algorithm is more efficient than *de novo* structure learning. An appropriate reference algorithm is the PC algorithm ([14, pp. 84-88]). If we assume that the degree of the true graph is k (i.e., each variable has at most k adjacent variables), then the "worst-case" complexity of the PC algorithm on $\boldsymbol{S}$ is:

$$2\binom{x+y+m}{2}\sum_{i=0}^{k}\binom{x+y+m-2}{i} \tag{2}$$

To determine the complexity of the SLPR algorithm, we consider each step individually. Since they do not involve any conditional independence tests, we ignore Steps 1, 2, 4, and 6 of the algorithm. Furthermore, since we want to provide the worst case for the SLPR algorithm in this analysis, we will assume that no edges are removed in Step 2, and that no edges are removed in the last stage of Step 3 (in which the removal of one edge can trigger the removal of other edges).

Step 3, in which we determine whether there are edges between $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$, has the following worst-case complexity:

$$xy\sum_{i=0}^{k}\binom{x+y+m-2}{i} \tag{3}$$

The complexity of Step 5 is a bit trickier, since it depends crucially on the particular structure of the graph being learned. The worst-case scenario occurs

when every pair of variables we consider is involved in a triangle with a variable outside of that pair's $\boldsymbol{C}$, since that maximizes the number of checks we must perform. It is unclear whether such a structure can exist for many values of k. Nevertheless, we want to put SLPR in the most difficult position possible, so we assume that this graph structure is possible. We further define a and b to be the maximum degrees of $Patt_{\boldsymbol{A}}$ and $Patt_{\boldsymbol{B}}$, respectively. For this graph structure, an upper bound on the number of conditional independence checks in Step 5 is:

$$[ax + by + m\,(a + b)] \sum_{i=0}^{k-1} \binom{x + y + m - 3}{i} \tag{4}$$

The SLPR algorithm is computationally less complex than the PC algorithm if and only if (2) $>$ (3) + (4). Since the summation in (4) is always strictly less than the summation in the other two equations, SLPR is less complex than PC if (but not only if):

$$2 \binom{x + y + m}{2} > xy + ax + by + m\,(a + b) \tag{5}$$

Surprisingly, this equality always hold. Clearly, it is least likely to hold when a and b each have their maximal values ($x + m - 1$ and $y + m - 1$, respectively). In fact, inequality (5) does not hold if we simply substitute in these values. However, the $m\,(a + b)$ term on the right-hand side corresponds to the number of possible pairs (in Step 5) involving a variable in $\boldsymbol{M}$. Therefore, we cannot simply substitute in the maximal values for a and b, since there is an upper limit to the number of adjacent variables: namely, $(x + y + m - 1)$. When we substitute this lesser value in for $(a + b)$, then (5) reduces to $xy > 0$, which clearly always holds (since we assume that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are always non-empty). Therefore, the SLPR algorithm is always less computationally complex than the PC algorithm in the worst-case, regardless of the specific underlying parameterization.

5 Conclusion

Databases now regularly reach terabyte sizes, and their size arises from both large numbers of datapoints, and large numbers of variables. Hence, for practical reasons, our analyses are often restricted to a subset of the variables in the dataset. Moreover, multiple databases can be more usefully shared if we have efficient algorithms for integrating machine learning outputs for the datasets considered in isolation. Multiple datasets might also face practical barriers to integration (e.g., privacy issues). Hence, there are practical (in addition to purely theoretical) reasons to consider the problems associated with integrating and using learning outputs for multiple overlapping sets of variables.

In this paper, we provide two rules for edge presence and absence when integrating two Bayes nets. These rules almost certainly do not exhaust the possible rules; the existence of others, however, remains an open question. Also, as noted earlier, there are additional rules if we have some prior knowledge of the

parameterization of the network. Given that we often have some domain-specific knowledge about the types of causation under investigation, further investigation of these rules could have substantial practical impact.

The SLPR algorithm provided in Section 4.1 also supports the goal of integrating multiple datasets. However, the practical usefulness of the algorithm awaits a more adequate "expected-case" complexity analysis. The usefulness of the algorithm also depends on whether it is in fact faster when path-checking and orientation steps are taken into consideration. Although those steps are much simpler, they might nonetheless add sufficient time to make the PC algorithm faster. In addition, we can ask whether this is the best we can do; are there more efficient algorithms than SLPR?

Most importantly, the robustness of the SLPR algorithm should be fully checked using real-world datasets. The algorithm assumes that the data independencies match the independencies in the true underlying generating structure, and this assumption is often violated by real-world data. Further empirical validation is necessary to determine both the scope of the problems that arise when the above assumption is violated, and also the magnitude of "speed-up" benefit provided by the SLPR algorithm. This further analysis might also suggest an algorithm that drops the Causal Sufficiency Assumption.

Despite these remaining open problems, the rules and algorithm presented in this paper provide an important start on the problem of integrating the causal learning of multiple, overlapping datasets.

References

1. Cheng, Patricia. 1997. "From Covariation to Causation: A Causal Power Theory." *Psychological Review*, 104: 367-405.
2. Cooper, Gregory F. 2000. "A Bayesian Method for Causal Modeling and Discovery Under Selection." In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000).*
3. Danks, David. 2002. "Efficient Integration of Novel Variables in Bayes Net Learning." Technical report: Institute for Human & Machine Cognition, University of West Florida.
4. Danks, David, and Clark Glymour. 2001. "Linearity Properties of Bayes Nets with Binary Variables." In J. Breese & D. Koller, eds. *Uncertainty in Artificial Intelligence: Proceedings of the 17th Conference (UAI-2001).* San Francisco: Morgan Kaufmann. pp. 98-104.
5. Fienberg, Stephen E., and Sung-Ho Kim. 1999. "Combining Conditional Log-Linear Structures." *Journal of the American Statistical Association*, 94 (445): 229-239.
6. Geiger, Dan, David Heckerman, and Christopher Meek. 1996. "Asymptotic Model Selection for Directed Networks with Hidden Variables." Microsoft Research Technical Report: MSR-TR-96-07.
7. Glymour, Clark, and Gregory F. Cooper, eds. 1999. *Computation, Causation, and Discovery.* Cambridge, Mass.: AAAI Press and The MIT Press.
8. Heckerman, David, Dan Geiger, and David M. Chickering. 1994. "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data." Microsoft Research Technical Report: MSR-TR-94-09.

9. Jordan, Michael I., ed. 1998. *Learning in Graphical Models*. Cambridge, Mass.: The MIT Press.
10. McKim, Vaughn R., and Stephen P. Turner. 1997. *Causality in Crisis? Statistical Methods and the Search for Causal Knowledge in the Social Sciences*. Notre Dame, Ind.: University of Notre Dame Press.
11. Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann Publishers.
12. Pearl, Judea. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge: Cambridge University Press.
13. Seidenfeld, Teddy, Joseph B. Kadane, and Mark J. Schervish. 1989. "On the Shared Preferences of Two Bayesian Decision Makers." *The Journal of Philosophy*, 86 (5): 225-244.
14. Spirtes, Peter, Clark Glymour, and Richard Scheines. 1993. *Causation, Prediction, and Search*. 2nd edition, 2001. Cambridge, Mass.: AAAI Press and The MIT Press.
15. Williamson, Jon. 2001. "Foundations for Bayesian Networks." Forthcoming in D. Corfield & J. Williamson, eds. *Foundations of Bayesianism*. Kluwer Applied Logic Series.