Learning Causal Trees with Latent Variables via Controlled Experimentation

Prasad Tadepalli
Oregon State University
Corvallis, Oregon, 97331

Cameron Barrie
Oregon State University
Corvallis, Oregon, 97331

Stuart J. Russell
University of California
Berkeley, California, 94720

Abstract

Learning causal knowledge often requires interventional experiments due to the non-determinacy of causality. In this paper we motivate the need for experiments from the perspective of computational tractability when there are latent variables. We present a polynomial-time *controlled experimentation* algorithm to learn the parameters of a known tree structured causal network when all nodes but the root and the leaves are latent and only the leaves are controllable. Our empirical results on several synthetic datasets show the superiority of algorithmic experimentation compared to other passive approaches such as neural network learning.

Introduction

Structural causal Models (SCMs) are attractive models of causality with intuitive semantics and mathematical rigor. It is well known that learning the structure of causal networks often requires interventional experiments, as the observational data alone cannot identify the true causal structure from its Markov equivalence class (Pearl 2009; Spirtes, Glymour, and Scheines 2000; Reichenbach 1991). There have been several approaches to learn the structure of the causal networks by combining observational and interventional data (Cooper and Yoo 1999; Peters, Bühlmann, and Meinshausen 2016; Silander and Myllymaki 2012; Eaton and Murphy 2007). There are also some sophisticated approaches that actively design optimal experiments to learn the structure of the network (He and Geng 2008). In recent work, optimal active learning strategies for identifying causal network structures with minimal number of single-node and multi-node interventions have been studied (Hauser and Bühlmann 2014).

There has been considerable work on learning the structure of SCMs in the presence of latent variables including IC* (Pearl 2009), FCI (Spirtes, Glymour, and Scheines 2000), and RFCI (Colombo et al. 2012). Recent methods have explored the use of path queries (Bello and Honorio 2017), experimental design (Kocaoglu, Shanmugam, and Bareinboim 2017), and constraint satisfaction to learn the structure (Hyttinen et al. 2013). In this work, we address the complementary problem of efficiently learning the parameters of a network with a known structure in the presence of a large number of latent variables. Previous work on learning the parameters of Bayesian networks are based on expectation maximization and gradient descent (Binder et al. 1997), a variety of imputation methods and others such as robust Bayesian estimator (Ramoni and Sebastiani 2001). However, none of these approaches make any formal guarantees of performance. Indeed, it appears that learning in the presence of latent variables is intractable without experiments, which is not considered by these methods.

We consider the problem of learning the parameters of a structural causal model (SCM) from observations and experiments in the framework of PAC-learning of probabilistic concepts or P-concepts (Kearns and Schapire 1994). This framework assumes that the inputs are generated from a natural distribution, and the goal is to learn, with a high probability, an approximate conditional distribution of the target variable given the inputs. In PAC-prediction, the learner is allowed to express the target function in any polynomially evaluatable representation equivalent to the target concept. Thus PAC-learning of a concept is in general a harder problem than PAC-prediction as the learner has greater freedom to represent its target. When all variables are Boolean, and the CPTs are deterministic, an SCM is equivalent to a Boolean circuit. Unfortunately PAC-prediction of Boolean circuits from random examples is cryptographically hard, i.e., as hard as solving cryptographic problems such as integer factoring (Pitt and Valiant 1988). Surprisingly, the problem remains hard even when the circuit is tree-structured and the structure of the tree is given (Pitt and Warmuth 1990). Since probabilistic Boolean SCMs, where only the inputs and the final output of the SCM are observed, generalize Boolean circuits, it follows that learning them is as hard as factoring. Just as in the Boolean circuits, the hardness remains even when the SCM has a known tree structure.

It was previously shown that the functions that correspond to the latent internal nodes of a tree structured deterministic Boolean circuit can be learned from random examples and membership queries (Tadepalli and Russell 1998). A membership query asks the output of the circuit for any given input. Interestingly both membership queries and the random examples are needed for learning to be successful. On the other hand, deviations from the tree structure make the problem cryptographically hard even when both random ex-

amples and membership queries are allowed. In this paper, we extend this previous work to probabilistic setting and show that rooted probabilistic causal trees can be efficiently learned from observational and experimental data.

While there have been many approaches to learning causal networks that employ a combination of observational and experimental data, in many cases, the experiments are offline and predetermined, which limits them to be small in number and non-adaptive. Previous work in active learning of Bayesian networks also explored the idea of experimentation where the relevant variables to be intervened are picked using greedy heuristics such as structure entropy and expected posterior loss over a query distribution (Li and Leong 2009; Tong and Koller 2001; He and Geng 2008). In contrast, we study the problem of algorithmic experimentation with *provable guarantees*, where the experiments are selected by the algorithm automatically for the specific informational need of learning. Also unlike the model of learning from membership queries where each query returns a label, each experiment in our setting returns the conditional probability distribution of output given the input.

The root of the tree represents the target variable whose value we are interested in predicting and the leaves of the tree represent the input variables whose values can be directly set or controlled by the algorithm. The causality flows from the leaf nodes to the root node, i.e., from the input variables to the target. Following the terminology of causal networks, we call the causal antecedents of a node, its parents, rather than its children as is normally referred to in the tree literature. We assume that the tree-structure of the network is given and the conditional probabilities at each node are represented explicitly in the form of tables. Learning the conditional probability tables (CPTs) is straightforward if we observe all nodes in the tree. What makes the problem difficult and interesting is that only the root and the leaves of the tree are observable and the rest of the nodes are latent.

Our top level algorithm is similar to that of (Tadepalli and Russell 1998) and is inspired by the idea of "controlled experimentation" in science. The algorithm, called ControlExp, creates an appropriate context – or experimental setting – that controls a latent variable and observes its effect on another variable. The result of the "experiment" is a conditional probability of the target variable given the input. While in the real world, the experiment will have to be repeated many times to estimate the probability, it is directly computed from synthetic causal models in our experiments.

The results show that our algorithm robustly and accurately learns the target models from a combination of random examples and automatically designed experiments. Deep neural networks trained on the same data or on the same amount of randomly generated data fail to learn as well. We believe that it strongly suggests the need for learning algorithms that exploit the known domain structure and adaptively design and conduct their own experiments.

Problem Setup

A Structural Causal Model (SCM) is a 4-tuple (V, D, G, P), where V is a set of random variables over the domain D, G is a directed acyclic graph over V, and P is the conditional distribution of variables $V_i \in V$ given the values of their parents (causal antecedents) $Pa(V_i)$ in G. A Tree-structured causal network or *causal tree* is an SCM in the form of a rooted tree, where the root represents the only target variable and the direction of causality is from the leaf nodes that represent the input variables to the root. Recall that we reverse the usual convention of parent-child relationship in trees to be consistent with the terminology of causal networks. In particular we assume that each node has a single child and has multiple parents which are its causal antecedents.

We assume that only the input variables and the target are observable. The rest of the variables, called the internal nodes, are latent. We often use the word 'node' to mean the variable represented at that node. In this work, we assume that the tree structure of an SCM, i.e., V and G are given and seek to learn the probability model $P(.|Pa(.))$. We restrict ourselves to the Boolean domain for all our variables, which allows P to be represented by a conditional probability table (CPT) given by $P(V_i = 1|Pa(V_i))$ for each variable V_i. The CPT has one entry for every possible value tuple of the parents of V_i. The generative process of the SCM induces a conditional distribution $P(target = 1|input = x)$, abbreviated as $P(target|x)$, which we seek to learn from examples.

We assume that the inputs x are generated from a natural distribution $D(x)$. Following the PAC framework of (Kearns and Schapire 1994), we seek an $\epsilon-\gamma$-approximate probability model of P, which is a model $\tilde{P}$ that guarantees that $\mathbf{P}r_{x\in D}[|P(target|x) - \tilde{P}(target|x)| > \gamma] \leq \epsilon$. Learning algorithms find such models by considering a hypothesis space H of possible models and returning a hypothesis $h \in H$ with the least empirical loss on a sufficiently large set of labeled samples. The PAC-learning theory of (Kearns and Schapire 1994; Haussler 1992) and others (Pollard 1984) define a combinatorial measure called pseudo-dimension d for the hypothesis space where a sample size polynomial in $d, 1/\epsilon, 1/\gamma$, and $1/\delta$ is sufficient to guarantee that any hypothesis that minimizes the empirical loss on the sample is going to be an $\epsilon-\gamma$-approximation of the true model with probability $1 - \delta$. The pseudo-dimension of the hypothesis space of Bayesian networks of a given structure is given by the sum of the sizes of all its CPTs, i.e., $n2^k$, where n is the number of nodes and the degree k is an upper bound on the number of parents of any node (Dasgupta 1997). Hence causal trees of small degree have low sample complexity.

Turning now to the time complexity of learning, in previous work it is shown that the parameters of deterministic causal trees with known structure can be learned in polynomial-time when both random examples and membership queries are available (Tadepalli and Russell 1998). Both random examples and membership queries are necessary for successful learning. Without membership queries the problem is equivalent to learning arbitrary Boolean functions from random examples, which is cryptographically hard (Pitt and Valiant 1988). Without random examples, the trees can encode arbitrary passwords so that it takes exponentially many guesses to get any useful information in the worst case.

The goal of this paper is to generalize the above result to

that of learning ϵ–γ approximate models of causal trees from random observational examples $(x, P(target|x))$ where x is chosen according to the natural distribution D, and experimental queries EXP(x) which are responded with the conditional probability $P(target|x)$ for any x. The learning algorithm works by drawing a sufficiently large sample of random examples, and then using the experimental queries around these examples to find the least loss causal tree consistent with the data. The crux of the problem is to do this in a way that only takes time polynomial in the size of the target tree and the size of the random sample.

Control Normal Form of Causal Trees

In this section, we show that the conditional probability tables of causal trees with latent internal nodes can be transformed into a special normal form which will be exploited by our learning algorithm.

Definition 1 *A causal tree is in control normal form (CNF) if for every internal variable $V_i \in V$, there are some values d for its parents such that $P(V_i=1|Pa(V_i)=d)=1$ and some other values z for its parents such that $P(V_i=1|Pa(V_i)=z)=0$.*

In a causal tree in a control normal form, there are inputs that can deterministically set any internal variable to 0 or 1. We can find these inputs by choosing appropriate values for each parent of the internal node to set it to 0 or 1 and recursing over the parents to do the same. For any internal node n of a target tree in CNF, let $d(n)$ be the input assignment for its leaf nodes that deterministically sets the node to a value of 1. Similarly let $z(n)$ set its value to 0. We call the dual inputs $z(n)$ and $d(n)$ the *control inputs* of i. We will now consider the problem of transforming any given causal tree to CNF.

Definition 2 *The* weighted average *of S and T with respect to w, denoted by $A(w,S,T)$, is $(1-w)S+wT$.*

The following lemma is useful to justify the normalization algorithm.

Lemma 1 *Let $0 \le lo \le w \le hi \le 1$. Then, $A(\frac{w-lo}{hi-lo}, A(lo,S,T), A(hi,S,T)) = A(w,S,T)$.*

Proof: $A(\frac{w-lo}{hi-lo}, A(lo,S,T), A(hi,S,T))$
$= (1-\frac{w-lo}{hi-lo})A(lo,S,T) + \frac{w-lo}{hi-lo}A(hi,S,T)$
$= \frac{hi-w}{hi-lo}((1-lo)S+loT) + \frac{w-lo}{hi-lo}((1-hi)S+hiT)$
$= \frac{(hi-lo)-w(hi-lo)}{hi-lo}S + \frac{hi-lo}{hi-lo}wT$
$= (1-w)S+wT = A(w,S,T)$ □ Table 1 describes the algorithm "Normalize" that converts any rooted tree-structured causal network to its normal form. The algorithm starts from the leaf CPTs and proceeds toward the root. In the algorithm, we let P and P' denote the old and new entries in the CPTs respectively. In Table 1 and elsewhere, we assume $Pa(n)$ represents the parents (direct causes) of n, n_i is i^{th} parent of n, and $n_{\bar{i}}$ is the set of all co-parents of n_i. i.e., parents of n other than n_i. Let z be a vector of 0s and 1s of appropriate dimensions. Step 2 of the algorithm linearly rescales the probability tables of n_i so that the lowest probability lo maps to 0 and the highest probability hi maps to 1. Every other probability w maps to $\frac{w-lo}{hi-lo}$. Step 3 compensates for this change at node n by adjusting its CPT with respect to node n_i. Lemma 2 shows that the compensation preserves the distribution of n for any values of its children.

1. Pick an internal node n_i, which is a parent of n, where all its parents are leaves or have already been normalized.
2. Let the smallest probability in the CPT of n_i be lo, and the highest probability be hi. Replace the entries in the CPT of n_i as follows.
   $P'(n_i=1|Pa(n_i)=z) \leftarrow \frac{P(n_i=1|pa(n_i)=z)-lo}{hi-lo}$.
3. Change the CPT of n as follows. Here y is an assignment of 0s and 1s to co-parents of n_i, i.e., nodes in $n_{\bar{i}}$.
   $P'(n=1|n_i=0, n_{\bar{i}}=y) \leftarrow$
   $A(lo, P(n=1|n_i=0, n_{\bar{i}}=y),\ P(n=1|n_i=1, n_{\bar{i}}=y))$
   $P'(n=1|n_i=1, n_{\bar{i}}=y) \leftarrow$
   $A(hi, P(n=1|n_i=0, n_{\bar{i}}=y),\ P(n=1|n_i=1, n_{\bar{i}}=y))$

Table 1: The Normalize procedure repeatedly executes steps 1-3 by picking nodes in bottom-up manner.

Lemma 2 *Consider a rooted tree-structured causal network for which* Normalize *has been applied at node n_i. Then for any vectors x and y of appropriate dimensions, $P(n=1|Pa(n_i)=x, n_{\bar{i}}=y) = P'(n=1|Pa(n_i)=x, n_{\bar{i}}=y)$*

Proof:
$P(n=1|Pa(n_i)=x, n_{\bar{i}}=y)$
$= P(n_i=1|Pa(n_i)=x, n_{\bar{i}}=y)$
$\quad P(n=1|n_i=1, Pa(n_i)=x, n_{\bar{i}}=y)$
$\quad + P(n_i=0|Pa(n_i)=x, n_{\bar{i}}=y)$
$\quad P(n=1|n_i=0, Pa(n_i)=x, n_{\bar{i}}=y)$
$= P(n_i=1|Pa(n_i)=x)P(n=1|n_i=1, n_{\bar{i}}=y)$
$\quad + (1-P(n_i=1|Pa(n_i)=x))P(n=1|n_i=0, n_{\bar{i}}=y)$
$= A(P(n_i=1|Pa(n_i)=x), P(n=1|n_i=0, n_{\bar{i}}=y),$
$\quad P(n=1|n_i=1, n_{\bar{i}}=y))$
$= A(\frac{P(n_i=1|Pa(n_i)=x)-lo}{hi-lo},$
$\quad A(lo, P(n=1|n_i=0, n_{\bar{i}}=y),$
$\quad\quad P(n=1|n_i=1, n_{\bar{i}}=y)),$
$\quad A(hi, P(n=1|n_i=0, n_{\bar{i}}=y),$
$\quad\quad P(n=1|n_i=1, n_{\bar{i}}=y)))$
$= A(P'(n_i=1|Pa(n_i)=x), P'(n=1|n_i=0, n_{\bar{i}}=y),$
$\quad\quad P'(n=1|n_i=1, n_{\bar{i}}=y))$
$= P'(n_i=1|Pa(n_i)=x)P'(n=1|n_i=1, n_{\bar{i}}=y)$
$\quad + (1-P'(n_i=1|Pa(n_i)=x))P'(n=1|n_i=0, n_{\bar{i}}=y)$
$= P'(n=1|Pa(n_i)=x, n_{\bar{i}}=y)$ □

Since each transformation at n_i preserves the conditional probability $P(n|Pa(n_i), n_{\bar{i}})$, a sequence of such transformations at each n_i preserves it as well. Repeating this argument for each internal node n in bottom-up manner shows that the probability of the target given any input is preserved. The following theorem formalizes the argument.

Theorem 1 *Let P^* be the probabilities that result by doing the above transformation on all hidden nodes in bottom up manner on a rooted tree-structured Bayesian net.*

Let z be any binary vector the leaf nodes are set to. Then, $P(target|z) = P^*(target|z)$.

Proof: We show that each transformation preserves the above conditional probability. By induction it follows that it is preserved by a sequence of transformations.

$$P(target|z) = \sum_w P(target|n=w,z) \sum_{x,y} P(n=w|Pa(n_i)=x, n_{\bar{i}}=y) P(Pa(n_i)=x|z) P(n_{\bar{i}}=y|z)$$

Note that none of the three values – $P(target|n=w,z)$, $P(Pa(n_i)=x|z)$, and $P(n_{\bar{i}}=y|z)$ – change as a result of changing the CPTs of nodes n and n_i. By Lemma 2, the changes at nodes n and n_i are such that $P(n=w|Pa(n_i)=x, n_{\bar{i}}=y)$ is preserved for $w=1$ or 0. Hence the value of the whole expression is preserved. □

For example, consider the 4-input tree-structured causal network in Figure 1. We assume all variables are Boolean valued. Only the input variables D, E, F, G at the leaf nodes and the target A at the root are observable. B and C are latent. The figure shows how the CPTs of A, B, and C are transformed to normalize the causal network and preserve the distribution $P(A|D,E,F,G)$. For example, the notation $0.3/0$ on node B indicates that the $P(B=1|D=0,E=0,F=1,G=0) = 0.3$ before normalization and 0.0 after normalization. The conditional probability of node C similarly changes from 0.1 to 0. The conditional probability of A on the other hand remains the same as before at 0.36.

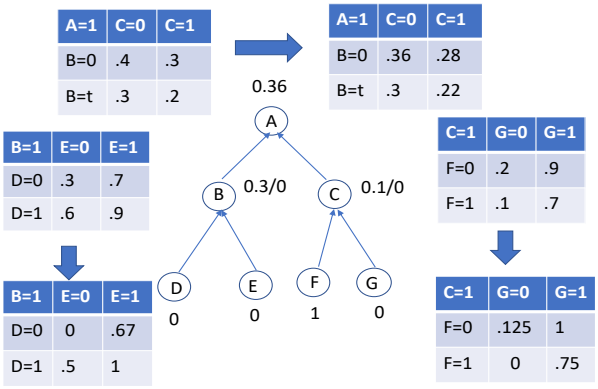

Figure 1: CPT normalization. The CPTs show the probability that a variable = 1, given its parents. After normalization both B and C have a 0 and a 1 in their CPTs. The figure shows the probabilities of different variables given an input before and after normalization.

The Learning Algorithm

Given that any target causal tree can be put in CNF, we seek to learn a tree in CNF and call it the target tree. Recall that for trees in CNF, every internal node n has control inputs $d(n)$ and $z(n)$ to its leaves that set it to 1 and 0 respectively. In addition, if there is also an input context, i.e., an assignment to the input variables other than the leaves of n, that makes this internal node have an impact on the distribution of the target variable, we call the node distinguishable. Distinguishability allows us to infer the distribution of the latent internal node from the distribution of the target variable when the node's context is set appropriately. If a node is not distinguishable, it has little impact on the target distribution for any input, and hence can be permanently set to a constant value, significantly simplifying learning.

Formally, a node n is *γ-distinguishable* if there is an assignment $c(\overline{n})$ to the inputs other than the leaves of n, called the "context input" of n such that $|P(target|d(n),c(\overline{n})) - P(target|z(n),c(\overline{n}))| > \gamma$. In this case we assume $P(target|c(\overline{n}),n=1) > P(target|c(\overline{n}),n=0)$ so that $P(target|c(\overline{n}),d(n)) > P(target|c(\overline{n}),z(n))$. This is w.l.o.g. since the internal nodes are latent, which allows redefining 0 and 1 values for them.

We describe our algorithm in three stages. First observing a distinguishable node, second filling in the CPT of a distinguishable node whose parents are also distinguishable, and third finding the distinguishable assignments. We then put it all together as a top-down recursive algorithm.

Inferring the distribution of a distinguishable node

Suppose n is a distinguishable node with control inputs $d(n)$ and $z(n)$ and context input $c(\overline{n})$. Let $y(n)$ represent some assignment to the leaf (input) nodes under the subtree rooted at node n. The following suggests how we might infer the conditional distribution of n given $y(n)$ from the conditional distribution of the target given $y(n)$ and $c(\overline{n})$.

$$\begin{aligned} &P(target|y(n),c(\overline{n})) = \\ &\quad P(target|y(n),c(\overline{n}),n=1)P(n=1|l(n)=y) \\ &\quad +P(target|y(n),c(\overline{n}),n=0)(1-P(n=1|y(n))) \\ &\quad = P(target|d(n),c(\overline{n}),n=1)P(n=1|y(n)) \\ &\quad +P(target|z(n),c(\overline{n}))(1-P(n=1|y(n))) \end{aligned}$$

From the last equation, it follows that

$$P(n=1|y(n)) = \frac{P(target|y(n),c(\overline{n})) - P(target|z(n),c(\overline{n}))}{P(target|d(n),c(\overline{n})) - P(target|z(n),c(\overline{n}))}.$$

We can implement this by using the EXP oracle that returns the probability of the target for various inputs. In particular, we have:

$$P(n=1|y(n)) = \frac{EXP(y(n),c(\overline{n})) - EXP(z(n),c(\overline{n}))}{EXP(d(n),c(\overline{n})) - EXP(z(n),c(\overline{n}))}.$$

The above shows that finding the context and control inputs would make a latent node effectively observable and controllable. If a node is not distinguishable, it means that it has no impact on the target, and hence can be effectively ignored by setting it to a constant function, say **0**.

Filling in the CPT of a distinguishable node

Assuming that n and its parents are distinguishable with known context and control inputs, it is straightforward to fill in the CPT of n using appropriate experimental queries. We consider each possible assignment vector x to the parents of node n, where x_i is the bit we would like to assign to the i^{th} parent n_i. To fill in $P(n=1|Pa(n)=x)$, for each parent n_i of n, set its leaf inputs $y(n_i)$ according to $d(n_i)$ if

Algorithm 1 ControlExp

1: Procedure LearnTree($S, n, c(\overline{n})$)
2: **for** each parent n_i of n **do**
3: find $c(\overline{n_i}), d(n_i), z(n_i)$ using S
4: $hi \leftarrow$ EXP($d(n_i), c(\overline{n_i})$)
5: $lo \leftarrow$ EXP($z(n_i), c(\overline{n_i})$)
6: **end for**
7: **for** each tuple of values x for the parents of n **do**
8: **for** each distinguishable parent n_i of n **do**
9: if $x_i = 1$ set inputs $y(n_i) \leftarrow d(n_i)$
10: else set inputs $y(n_i) \leftarrow z(n_i)$
11: **end for**
12: w = EXP($y(n_1), \ldots, y(n_k), c(\overline{n})$)
13: $P(n = 1|Pa(n) = x) \leftarrow \frac{w-lo}{hi-lo}$
14: **end for**
15: **for** each distinguishable parent n_i of n **do**
16: call LearnTree($S, n_i, c(\overline{n_i})$)
17: **end for**
18: End LearnTree

Table 2: The learning algorithm recursively finds the distinguishing assignments of each node and learns its CPT in top down fashion.

$x_i = 1$, and to $z(n_i)$ if $x_i = 0$. If n_i is not distinguishable, its leaves $y(n_i)$ are set according to $z(n)$. We set the context of n according to its context input $c(\overline{n})$. We use the previous equation to infer the distribution $P(n = 1|y(n), c(\overline{n}))$ and assign it to the CPT entry $P(n=1|Pa(n) = x)$.

Finding control and context inputs

We now describe the crucial step of finding the control inputs $d(n)$ and $z(n)$ and the context input $c(\overline{n})$ for any node n. It is here that we need random example set S chosen according to the natural distribution D. Without random examples from a natural distribution, the search for control and context inputs will have to be exhaustive and impractical.

For any random example x and internal node n, let x_n be the part of the input at the leaves of node n and $x_{\overline{n_i}}$ be the input under all parents of n other than n_i. To find $c(\overline{n})$, we find an $x \in S$ which maximizes the absolute difference between $P(target|x_n, c(\overline{n}))$ and $P(target|z(n)_{n_i}, c(\overline{n}), x_{\overline{n_i}})$, where $z(n)_{n_i}$ is the part of the control input $z(n)$ of node n which is under the node n_i. If this difference is greater than a small threshold γ, we set $c(\overline{n_i})$ to $x_{\overline{n_i}}$. If it is not, we conclude that the node n_i is not distinguishable. Finally, we set $d(n_i) = \arg\max_{x \in S} P(target|x_{n_i}, c(\overline{n_i}))$ and $z(n_i) = \arg\min_{x \in s} P(target|x_{n_i}, c(\overline{n_i}))$ to achieve maximum difference between the target probabilities for the two control inputs and to make sure that the probabilities are within the range of $[0, 1]$ after transformation.

Putting it all together

We now put it all together in Algorithm 1. Our recursive algorithm ControlExp takes as input the random example set s, the current node n and its context input $c(\overline{n})$ (line 1). It starts from the root node of the causal tree and proceeds top-down by finding distinguishing inputs and the context for the children of the current node n (line 3). The calls to EXP return the probability that the target variable is 1 given the inputs (lines 4-5). It then systematically sets the values of the parents to each k-tuple x. To set a parent n_i to 1, it sets the inputs under that node to $d(n_i)$ (line 9). To set it to 0, it sets its inputs to $z(n_i)$ (line 10). It sets the context of the current node n to $c(\overline{n})$ and calls EXP (line 12). The value returned by EXP which represents the probability of the target is used to infer the probability of the current node n when its parents are set according to x (line 13). This probability will be used to fill the CPT entry of node n and k-tuple x. It then recursively calls itself on the parents of the current node and their context vectors (lines 15–17).

ControlExp makes $O(n2^k)$ experimental queries, where n is the number of nodes and k is the maximum number of parents of any node in the causal tree. The number of random examples is bounded by $m = O(\frac{kn}{\epsilon} \log \frac{n}{\delta})$, which guarantees that all CPT entries used in computing the conditional probabilities of examples x of $D(x) > \epsilon$ are learned with a probability at least $1 - \delta$. The time complexity of the algorithm is $O(nm + n2^k)$.

Experimental Results

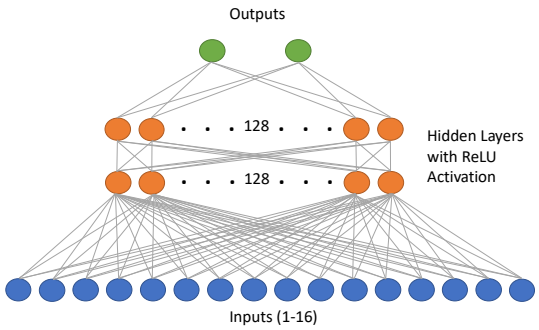

Figure 2: Diagram of MLP network. The blue nodes are the 16 inputs, the 256 orange nodes are divided into two equally sized layers of ReLU units, and the green nodes comprise the softmax output layer.

Experimental Setup

We evaluated our results on synthetic data generated using a target causal tree, which is a full binary tree with depth 4 and 16 leaf nodes. At each node we randomly selected a CPT, where each probability in the table is chosen independently with 50% chance the probability is either close to 1 or to 0 by less than $\epsilon = 0.1$. The extreme probabilities are chosen to make the learning problem challenging. We show the results of our program on 9 causal trees, i.e., 9 different sets of CPTs. We compare our results to a neural network baseline which is described later.

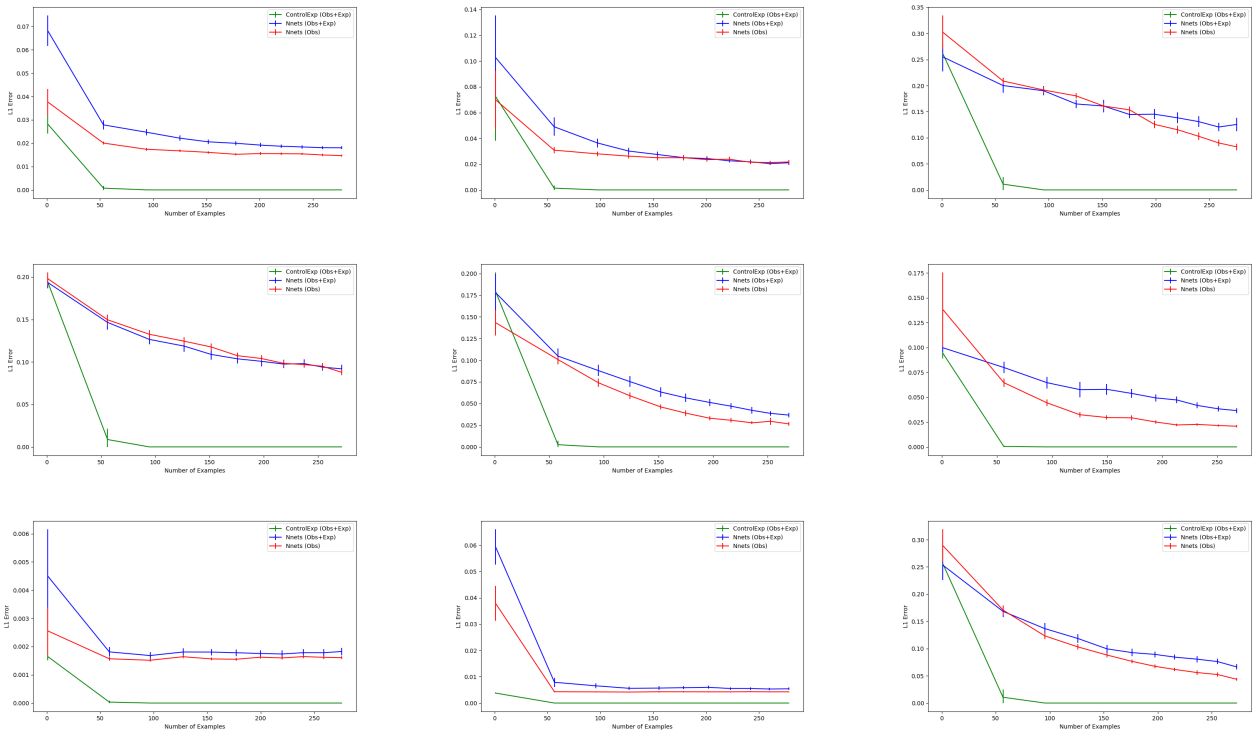

Figure 3: L1-error for ControlExp and Neural Nets. Obs+Exp is based on training the neural net on the same observational and experimental data as ControlExp. Obs is based on choosing the same amount of data drawn from the observational distribution.

Each evaluation consists of 40 runs on different training sets. Each training input is chosen uniformly randomly. The training was performed on batches of increasing sizes of random examples from 0 to 40 in increments of 4. During the training of each batch, the algorithm also asks experimental queries, which are answered by referring to the target tree. When a query is answered, its response is stored so that it is never asked again. After each batch of training the learned tree was evaluated on 100 test examples which were chosen randomly from the pool of examples not used for training.

The baseline neural network we compared to is shown in Figure 2. Specifically, it is a multi-layered perceptron (MLP) with two hidden layers, each with 128 nodes, which use the ReLU activation function. The network's output takes the form of a two-valued probability distribution, as defined by the softmax function over the last layer of two outputs. The first output value indicates the learned probability of the Bayesian network outputting a 0 and the second value indicates the learned probability of it outputting a 1.

The neural network was trained using stochastic gradient descent to minimize the Kullback-Leibler divergence between the learned probability distribution and the ground truth distribution. We found that a learning rate of 1.0, a weight decay of 0.001, and a momentum of 0.9 produced the best results. Training of the neural network was done with 1000 epochs of each batch of examples.

Discussion of Results

The learning curves shows the average L1-error of the conditional probability $P(target|x)$ for the test example x compared to the true probability plotted against the number of unique experiments or random examples in each training batch. Each subfigure is based on a different target tree. The green plot labeled ControlExp shows the performance of our algorithm. The blue plot labeled NNets(Obs+Exp) is the performance of our neural network on the same observed examples and experiments as used by our algorithm. The red plot is the performance of the neural network on the same amount of data as the other two, but chosen using the observational distribution (uniformly random). As is evident from the plots in Figure 3, ControlExp decisively outperforms the two neural network approaches in all cases.

Interestingly, in most cases the L1 error of the MLP network is smaller and much less noisy when training on only the observational data than when training on both the observational and the experimental data. This is because the distribution of the mixture of observational and experimental data is very different from that of the test distribution. However, the test distribution is identical to that of the observational data. Since the neural network learning optimizes its performance to the training distribution, it is misled by mixing in the experimental data and performs better with the purely observational data.

ControlExp is able to do better than the neural networks because it systematically targets its queries to reduce uncer-

tainty. It fully takes advantage of the tree structure of the causal network by exposing and controlling one node at a time. The neural network, on the other hand, is oblivious to the known structure and goes about its learning blindly.

We also tested other learning architectures including a sparser and deeper network that mirrors the structure of the true causal tree, other denser and deeper MLP networks, convolutional networks, and support vector machines (SVMs). While SVMs performed slightly better than the others, the prediction accuracies of all these methods were fairly similar and were considerably worse than the MLP network described in Figure 2. The fact that the best results were obtained with a relatively simple and shallow network with fully connected layers suggests that the depth of the networks is not their key strength in this domain.

Conclusions

While the need for experiments to uncover causal laws has been known for a long time, there appear to be more than one reason for this need. The classical reason is that the direction and structure of causality can only be discovered by changing the data distribution through interventions. While certainly true, even when the structure and the direction of causality are known, there are other obstacles. In this paper we argued that the latency of variables and the resulting computational intractability are other obvious hurdles, and showed that they too can be tackled to some extent through experimentation. However, the experimentation must be deliberate, targeted, and adaptive to mirror what scientists do in their everyday lives. It must be algorithmic to overcome human limitations and scale to large real world problems. We hope that this work paves the way for pushing the study of causality in these directions.

References

Bello, K., and Honorio, J. 2017. Learning bayes networks using interventional path queries in polynomial time and sample complexity.

Binder, J.; Koller, D.; Russell, S.; and Kanazawa, K. 1997. Adaptive probabilistic networks with hidden variables. *Machine Learning* 29:213–244.

Colombo, D.; Maathuis, M. H.; Kalisch, M.; and Richardson, T. S. 2012. Learning high dimensional direct acyclic graphs with latent and selection variables. *The Annals of Statistics* 40(1):294–321.

Cooper, G. F., and Yoo, C. 1999. Causal discovery from a mixture of experimental and observational data. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 116–125. Morgan Kaufmann Publishers Inc.

Dasgupta, S. 1997. The sample complexity of learning fixed structure bayesian networks. *Machine Learning* 29:165–180.

Eaton, D., and Murphy, K. 2007. Exact bayesian structure learning from uncertain interventions. In *Artificial Intelligence and Statistics*, 107–114.

Hauser, A., and Bühlmann, P. 2014. Two optimal strategies for active learning of causal models from interventional data. *International Journal of Approximate Reasoning* 55(4):926–939.

Haussler, D. 1992. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inf. Comput.* 100(1):78–150.

He, Y.-B., and Geng, Z. 2008. Active learning of causal networks with intervention experiments and optimal designs. *JMLR* 9:2523–2547.

Hyttinen, A.; Hoyer, P. O.; Eberhardt, F.; and Järvisalo, M. 2013. Discovering cyclic causal models with latent variables: A general SAT-based procedure. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Kearns, M. J., and Schapire, R. E. 1994. Efficient distribution-free learning of probabilistic concepts. *J. Comput. Syst. Sci.* 48(3):464–497.

Kocaoglu, M.; Shanmugam, K.; and Bareinboim, E. 2017. Experimental design for learning causal graphs with latent variables. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 7018–7028. Curran Associates, Inc.

Li, G., and Leong, T.-Y. 2009. Active learning for causal bayesian network structure with non-symmetrical entropy. In *PAKDD*, 290–301. Springer-Verlog.

Pearl, J. 2009. *Causality*. Cambridge university press.

Peters, J.; Bühlmann, P.; and Meinshausen, N. 2016. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 78(5):947–1012.

Pitt, L., and Valiant, L. G. 1988. Computational limitations on learning from examples. *J. ACM* 35(4):965–984.

Pitt, L., and Warmuth, M. K. 1990. Prediction-preserving reducibility. *J. Comput. Syst. Sci.* 41(3):430–467.

Pollard, D. 1984. *Convergence of Stochastic Processes*. New York, Berlin: Springer Verlog.

Ramoni, M., and Sebastiani, P. 2001. Robust learning with missing data. *Machine Learning* 45:147170.

Reichenbach, H. 1991. *The direction of time*, volume 65. Univ of California Press.

Silander, T., and Myllymaki, P. 2012. A simple approach for finding the globally optimal bayesian network structure. *arXiv preprint arXiv:1206.6875*.

Spirtes, P.; Glymour, C.; and Scheines, R. 2000. Causation, prediction, and search. adaptive computation and machine learning.

Tadepalli, P., and Russell, S. J. 1998. Learning from examples and membership queries with structured determinations. *Machine Learning* 32(3):245–295.

Tong, S., and Koller, D. 2001. Active learning for structure in Bayesian networks. In *IJCAI*, 863–869.

Acknowledgments

We acknowledge the support of DARPA under contract N66001-17-2-4030 and NSF under grant IIS-1619433.