ML4C: Seeing Causality Through Latent Vicinity

Haoyue Dai*† Rui Ding‡§ Yuanyuan Jiang¶ Shi Han‡ Dongmei Zhang‡

Abstract

Supervised Causal Learning (SCL) aims to learn causal relations from observational data by accessing previously seen datasets associated with ground truth causal relations. This paper presents a first attempt at addressing a fundamental question: *What are the benefits from supervision and how does it benefit?* Starting from seeing that SCL is not better than random guessing if the learning target is non-identifiable a priori, we propose a two-phase paradigm for SCL by explicitly considering structure identifiability. Following this paradigm, we tackle the problem of SCL on discrete data and propose ML4C. The core of ML4C is a binary classifier with a novel learning target: it classifies whether an Unshielded Triple (UT) is a v-structure or not. Specifically, starting from an input dataset with the corresponding skeleton provided, ML4C orients each UT once it is classified as a v-structure. These v-structures are together used to construct the final output. To address the fundamental question of SCL, we propose a principled method for ML4C featurization: we exploit the vicinity of a given UT (i.e., the neighbors of UT in the skeleton), and derive features by considering the conditional dependencies and structural entanglement within the vicinity. We further prove that ML4C is asymptotically correct. Thorough experiments conducted on benchmark datasets demonstrate that ML4C remarkably outperforms other state-of-the-art algorithms in terms of accuracy, reliability, robustness and tolerance. In summary, ML4C shows promising results on validating the effectiveness of supervision for causal learning. Our codes are publicly available at https://github.com/microsoft/ML4C.

Keywords: Causal discovery, supervised causal learning, identifiability, learnability

1 Introduction

The problem of causal learning is to learn causal relations from observational data [14]. The learned causal relations are typically represented in the form of a Directed Acyclic Graph (DAG), where each edge in the DAG indicates direct cause-effect relation between the parent node and child node.

The methods of causal learning mostly fall into four categories: constraint-based, score-based, continuous optimization method and functional causal models. Each of these methods takes a given dataset as input and outputs a DAG but with different criteria. For instance, the DAG should be consistent with conditional independencies in the data (constraint-based); or it is optimal w.r.t. a pre-defined score function under either combinatorial constraint (score-based) or continuous equality constraint (continuous optimization). In a nutshell, these methods can be viewed as *unsupervised* since they do not access additional datasets associated with ground truth causal relations.

A new line of research called *Supervised* Causal Learning (SCL), on the other hand, aims to learn causal relations in the supervised fashion: the algorithm has access to datasets associated with ground truth causal relations, in the hope that learning causal relations on newly unseen datasets benefits from such supervision. Despite several existing works on this direction (see Related Work), a fundamental question remains unanswered: *How is supervised causal learning possible*? Specifically, compared with unsupervised causal learning methods, can we gain additional benefits from supervision? If the answer is yes, then what are the benefits?

We tackle the problem by first seeing crucial connections between SCL and causal structure identifiability. Considering the problem of causal learning on discrete data, the theorem in [29] states that, under standard assumptions (i.e., Markov assumption, faithfulness and causal sufficiency), we can only identify a graph up to its Markov equivalence class. Markov equivalence class is the set of DAGs having same skeleton and same v-structures, which can be represented by CPDAG (Completed Partially Directed Acyclic Graph). Thus, the (un)directed edges in the CPDAG indicate (non-)identifiable causal relations. Each non-identifiable edge in CPDAG can be oriented by either direction to equivalently fit the observational data. Given an SCL algorithm with the learning target as the orientation of an edge, we see that it is not better than random guessing (or could be worse due to sample bias in training data) to predict any non-identifiable edge since we can assign either $X \to Y$ or $X \leftarrow Y$ with the same input dataset.

Observation 1.1. *Considering the learning target is the orientation of an edge. If the edge is non-identifiable*

*Carnegie Mellon University. hyda@cmu.edu
†Work done during an internship at Microsoft Research Asia.
‡Microsoft Research Asia. juding@microsoft.com
§Corresponding author.
¶Renmin University of China.

a priori, then supervised causal learning is no better than random guessing.

All proofs of observations, propositions, lemmas and theorems are available in the supplementary material. Consequently, we propose and advocate a two-phase paradigm for SCL, as depicted in Figure 2(a): phase one corresponds to determining the identifiability of a specific learning target; only if it is determined as identifiable, we go to phase two to classify the specific orientation of the learning target. Following this paradigm, we tackle the problem of SCL on discrete data and propose an algorithm ML4C. The core of ML4C is a binary classifier with a novel learning target: it classifies whether an Unshielded Triple (UT: a triple of variables $\langle X, T, Y\rangle$ where X and Y are adjacent to T but are not adjacent to each other) is a v-structure or not. Starting from an input dataset with the corresponding skeleton provided, ML4C orients each UT once it is classified as a v-structure. These v-structures are further used to construct a CPDAG as output. Such a single classifier facilitates both learning tasks in the two phases, since standard theory shows that an identifiable UT implies that it is a v-structure [37], i.e., up to the partial DAG before applying Meek rules [28].

Operationally, we propose a principled method for ML4C featurization. We exploit the *vicinity* of a given UT (i.e., the neighbors of UT in the skeleton), and derive features by considering the conditional *dependencies* and structural *entanglement* within the vicinity. ML4C's featurization is appealing from both theoretical and empirical perspectives. We prove that ML4C is asymptotically correct, and ML4C shows remarkable performance on benchmark datasets (with finite samples).

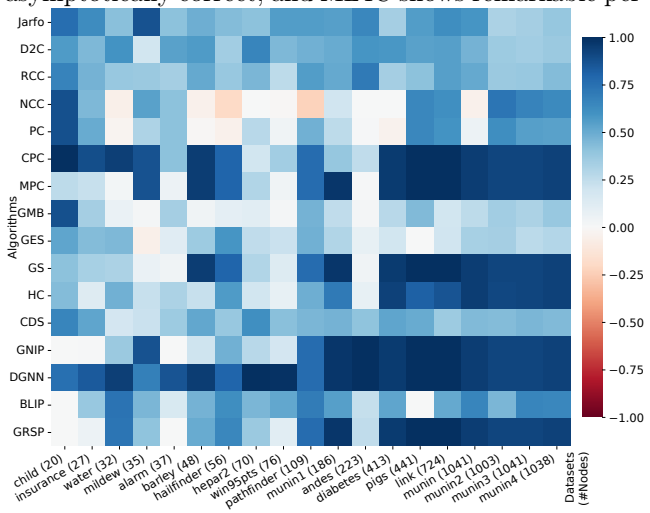

Figure 1: Comparison of ML4C and other algorithms (difference between ML4C's and others' F1 scores) on benchmark datasets. Blue indicates that ML4C is superior whereas red indicates that the competitor is better.

Evaluation Highlight: We compare ML4C and other SOTA algorithms on the bnlearn benchmark [33] thoroughly (as shown in Figure 1). Overall, ML4C significantly outperforms all competitors, with the highest average F1-score and consistent performance across all datasets (i.e., most of the blocks in Figure 1 are blue-colored). Also, ML4C shows high accuracy (F1-score$>$ 0.9) on very large-scale datasets ($>$ 1000 nodes) while max(others)$\sim$ 0.6 (last three rows of '`munin*`' in Table 1). Our main contributions are:

1. ML4C is a supervised approach for causal learning, and to the best of our knowledge, it is the first such approach to tackle discrete data systematically.

2. ML4C is with the following novelties: i) **Learning Target**: The core of ML4C is a binary classifier with the orientation of a UT as its learning target to address the two-phase tasks simultaneously. ii) **Featurization**: A principled method to exploit vicinity information in terms of dependencies and entanglement of a given UT. iii) **Learnability**: We prove that ML4C is asymptotically correct. iv) **Empirical Performance**: Experiments conducted on benchmark datasets demonstrate that ML4C remarkably outperforms other SOTAs.

2 Related Work

We divide literature on causal learning into supervised and unsupervised approaches, depending on whether additional datasets (associated with ground truth causal relations) are accessed (supervised) or not (unsupervised). In the literature of unsupervised causal learning, constraint-based methods aim to identify a DAG which is consistent with conditional independencies. The learning procedure of constraint-based methods first identifies the corresponding skeleton and then conducts orientation based on v-structure identification [39]. The typical algorithm is PC [36], and there are also PC-derived algorithms such as Conservative-PC [31] and PC-stable [10] which improve the robustness of v-structure identification. Score-based methods aim to find the DAG which is optimal w.r.t. a pre-defined score function under combinatorial constraints by a specific search procedure, such as forward-backward search GES [7], hill-climbing [19], integer programming [9], or by order search [32, 20]. Continuous optimization methods transform the discrete search procedure into continuous equality constraint: NOTEARS [41] formulates the acyclic constraint as a continuous equality constraint, it is further extended by DAG-GNN [40] and MCSL [30] to support learning non-linear causal relations. [26] proposes to estimate the posterior distribution of causal DAGs.

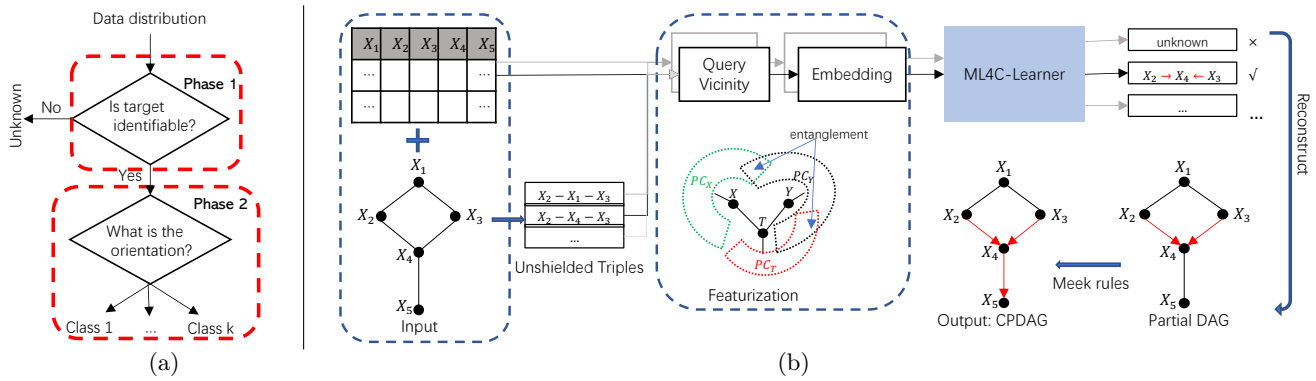

Figure 2: (a) Two-phase paradigm for supervised causal learning. (b) ML4C's workflow.

SCL emerges from the task of orienting edge in the continuous, non-linear bivariate case under Functional Causal Model (FCM) formalism. Given a collection of cause-effect samples (dataset $\sim$ binary label indicating whether $X \to Y$ or $X \leftarrow Y$), supervised approaches such as RCC [23, 24], NCC [25], D2C [5] and Jarfo [13] achieve better performance on predicting pairwise relations (i.e., the orientation of an edge) than unsupervised approaches such as ANM [16] or IGCI [17]. Differently, [22] sets the learning target as the whole DAG structure instead of pairwise relation and it is applied on data which is generated by the linear Structural Equation Model (SEM). We summarize the differences in the problem space between ML4C and the other SCL approaches as follows: i) We advocate a two-phase learning paradigm and emphasize the relationship between identifiability and learnability. Specifically, presuming additive noise model [16] or linear SEM with non-Gaussian noise [34] provides license to identifiability thus the aforementioned approaches can be viewed as specific tasks in phase two. ii) We tackle SCL's learnability not only via empirical evaluation but also by theoretical analysis to shed light on the fundamental question of learnability. iii) ML4C deals with discrete data systematically while other approaches mainly focus on continuous data.

3 Background

3.1 Basic Notations A discrete dataset D_i consists of n_i records and d_i categorical columns, which represents n_i instances drawn i.i.d. from d_i discrete variables $X_1, X_2, \cdots, X_{d_i}$ by a joint probability distribution P_i, which is entailed by an underlying data generating process, denoted as DAG G_i.

Causal sufficiency: There are no latent common causes of any of the variables in the graph.

Markov factorization property: Given a joint probability distribution P and a DAG G, P is said to satisfy Markov factorization property w.r.t. G if $P := P(X_1, X_2, \cdots, X_d) = \prod_{i=1}^{d} P\left(X_i | \mathrm{pa}_i^G\right)$, where pa_i^G is the parent set of X_i in G.

Global Markov Property (GMP): P is said to satisfy GMP (or Markovian) w.r.t. a DAG G if $X \perp_G Y | Z \Rightarrow X \perp Y | Z$. Here $\perp_G$ denotes d-separation, and $\perp$ denotes statistical independence. GMP indicates that any d-separation in graph G implies conditional independence in distribution P. GMP is equivalent to the Markov factorization property [21].

Faithfulness: Distribution P is faithful w.r.t. a DAG G if $X \perp Y | Z \Rightarrow X \perp_G Y | Z$.

Canonical dataset: We say a discrete dataset D is canonical if its underlying probability distribution P is Markovian and faithful w.r.t. some DAG G.

3.2 Causal Structure Identifiability Below are the established identifiability results on discrete data [29].

Definition 3.1. (Markov equivalence) *Two graphs are Markov equivalent if and only if they have the same skeleton and same v-structures. A Markov equivalence class can be represented by a CPDAG having both directed and undirected edges. A CPDAG can be derived from a DAG G, denoted as CPDAG(G).* The theorem of Markov completeness in [29] states that, under causal sufficiency, we can only identify a causal graph up to its Markov equivalence class on canonical data. Therefore, the (non-)identifiable causal relations are the (un)directed edges in the CPDAG. Formally,

Definition 3.2. (Identifiability) *Assuming P is Markovian and faithful w.r.t. a DAG G and causal sufficiency, then each (un)directed edge in CPDAG(G) indicates (non-)identifiable causal relation.*

3.3 ML4C Related Notations

Definition 3.3. (Skeleton) *A skeleton E defined over distribution $P(X_1, X_2, \cdots, X_d)$ is an undirected graph such that there is an edge between X_i and X_j if and only if X_i and X_j are always dependent, i.e., $\nexists Z \subseteq \{X_1, X_2, \cdots, X_d\}$ s.t. $X_i \perp X_j | Z$.* Skeleton is a statistical concept, which can be obtained prior to facilitating various downstream tasks. Recently, there have

been some novel skeleton learning algorithms such as [11]. In particular, skeleton can be used for causal learning: the theorem in [37] states that if distribution P is Markovian and faithful w.r.t. a DAG G, then skeleton E is the same as the undirected graph of G.

DEFINITION 3.4. (UT) *A triple of variables $\langle X, T, Y\rangle$ in a skeleton is an unshielded triple, or short for UT, if X and Y are adjacent to T but are not adjacent to each other. $\langle X, T, Y\rangle$ can be further oriented to become a v-structure $X \to T \leftarrow Y$, in which T is called the collider.*

DEFINITION 3.5. (PC) *Denote the set of parents and children of X in a skeleton as PC_X, in other words, PC_X are the neighbors of X in the skeleton. For convenience, if we discuss PC_X in the context of a UT $\langle X, T, Y\rangle$, we intentionally mean the set of parents and children of X but exclude T. Similarly, PC_T excludes X, Y.*

DEFINITION 3.6. (VICINITY) *We define the vicinity of a UT $\langle X, T, Y\rangle$ as $V_{\langle X,T,Y\rangle} := \{X, T, Y\} \cup PC_X \cup PC_Y \cup PC_T$. Vicinity is a generalized version of PC, i.e., the neighbors of $\{X, T, Y\}$ in the skeleton.*

DEFINITION 3.7. (SEPSETS) *Denoted as* $\mathcal{S} := \{S : X \perp Y | S,\ S \subset PC_X \cup T,\ \text{or } S \subset PC_Y \cup T\}$. Under faithfulness assumption, sepsets $\mathcal{S}$ is an ensemble where each item is a subset of variables within the vicinity that d-separates X and Y.

4 Approach

4.1 Viewing PC and MPC as Special Classifiers

The motivation of ML4C is from the observation, that the orientation logics of both PC [37] and MPC [10] can be interpreted as explicit featurization and static classification mechanisms. Starting from PC, it first learns the skeleton from data, and then conducts orientation by taking skeleton as input. Specifically, for each UT $\langle X, T, Y\rangle$ (queried from skeleton), it finds a sepset S such that $X \perp Y | S$, and then simply checks if $T \in S$ or $T \notin S$: if $T \notin S$, orient $\langle X, T, Y\rangle$ as a v-structure, and otherwise undetermined. From ML perspective, we re-formulate PC's logic as follows:

Task: To classify if a UT $\langle X, T, Y\rangle$ is a v-structure.

Featurization: Finds a sepset S s.t. $X \perp Y | S$, and defines a Boolean feature $x_{\texttt{PC}}(\langle X, T, Y\rangle) := T \in S$.

Classifier: $C_{\texttt{PC}}(x_{\texttt{PC}}) := \begin{cases} \text{non-v } x_{\texttt{PC}} = \text{TRUE} \\ \text{v-struc } x_{\texttt{PC}} = \text{FALSE} \end{cases}$.

Majority-PC (MPC) [10] is a sample version enhancement of PC's orientation, which achieves better performance on finite samples. Instead of finding only one sepset S, MPC finds all the sepsets $\mathcal{S}$ and counts the number of sepsets $S_i \in \mathcal{S}$ that contains T. We summarize its logic in the ML formulae:

Featurization: Finds all sepsets $\mathcal{S}$ of X, Y, and defines a real-value feature $x_{\texttt{MPC}}(\langle X, T, Y\rangle) := \frac{|\{S_i | T \in S_i \in \mathcal{S}\}|}{|\mathcal{S}|}$.

Classifier: $C_{\texttt{MPC}}(x_{\texttt{MPC}}) := \begin{cases} \text{non-v } x_{\texttt{MPC}} > 0.5 \\ \text{v-struc } x_{\texttt{MPC}} \leq 0.5 \end{cases}$.

OBSERVATION 4.1. *Both the "hand-crafted" classifiers of PC and MPC are asymptotically correct. Furthermore, $C_{\texttt{MPC}}$ is more "sophisticated" than $C_{\texttt{PC}}$ due to its more complex classification mechanism, and thus $C_{\texttt{MPC}}$ achieves better performance empirically. However, from the ML perspective, either $C_{\texttt{PC}}$ or $C_{\texttt{MPC}}$ is simple, so we are encouraged to provide more systematic featurization, and to learn a better classification mechanism.*

4.2 ML4C's Overview Figure 2(b) depicts the overall workflow of ML4C, which is composed of ML4C-Learner with other standard logics. Similar to [37], ML4C-Learner classifies whether a UT is a v-structure or not. Specifically, featurization is conducted to represent each UT as an embedded vector, which is fed into ML4C-Learner, a binary classifier. In the inference stage, we obtain all the v-structures which are classified by ML4C-Learner and reconstruct a partial DAG and then, a CPDAG is output by applying Meek rules on the partial DAG. In the training stage, the label of each UT is obtained by querying from ground truth DAG G_i. We collect labeled training data from synthesis.

By Markov completeness, the set of v-structures is invariant across all Markov equivalent DAGs for a canonical dataset, and it can fully recover the CPDAG, provided that the skeleton is given. Thus, besides its dedicated role in phase 2, ML4C-Learner also facilitates learning task in phase 1 since an identifiable UT implies that it is a v-structure (up to the partial DAG before applying Meek rules).

4.3 Featurization We propose a principled method for ML4C-Learner's featurization. We further prove that ML4C-Learner is asymptotically correct.

Design Principles: Our key aspect of featurization is to broaden the focus from a specific UT to its *vicinity* and seeking conditional *dependencies* and structural *entanglement* within the vicinity, to reveal reliable asymmetry that distinguishes v-strucs and non-v-strucs.

• ***Dependencies within Vicinity***

Conditional dependency: Denoted as $X \sim Y | \mathbf{Z}$, which is a non-negative scalar that measures the dependence between two random variables X and Y given variable set $\mathbf{Z}$. Operationally, $X \sim Y | \mathbf{Z}$ is composed of two parts, bivariable $X \sim Y$, and conditional $\mathbf{Z}$. We further extend the definition to allow a set of variables in bivariable, and an ensemble (i.e., a set of set) as conditional:

Extended conditional dependency: Denoted as

$\mathbf{A} \sim \mathbf{B}|\mathcal{Z} := \{X \sim Y|\mathbf{Z} : X \in \mathbf{A}, Y \in \mathbf{B}, \mathbf{Z} \in \mathcal{Z}\}$, where $\mathbf{A}$ and $\mathbf{B}$ are set of variables, and $\mathcal{Z}$ is an ensemble. Thus, extended conditional dependency is a set of scalars.

Within the vicinity of $\langle X, T, Y\rangle$, we start from measuring dependencies between $\{X,\ PC_X\}$ and $\{Y,\ PC_Y\}$ by conditioning on $\{T,\ PC_T\}$. Intuitively, if $\langle X, T, Y\rangle$ is a v-structure, conditioning on T or T's descendants tends to strengthen the dependency between PC_X and PC_Y since the paths passing $X - T - Y$ are unblocked; otherwise, conditioning on T tends to weaken the dependency between PC_X and PC_Y because T blocks the paths passing $X - T - Y$. Therefore, such conditional dependencies reflect potential asymmetry to distinguish v-structure and non-v-structure. Formally,

Definition 4.1. (Domain of bivariable) *Denoted as* $\mathbb{B} := \{X,\ PC_X\} \times \{Y,\ PC_Y\} \equiv \{X \sim Y, X \sim PC_Y, PC_X \sim Y, PC_X \sim PC_Y\}$, *here symbol* $\times$ *is Cartesian product.*

Definition 4.2. (Domain of conditional) *Denoted as* $\mathbb{C} := \{\varnothing, T, \mathcal{PC}_T\} \vee \{\varnothing, S\} \equiv \{\varnothing, T, \mathcal{PC}_T, \mathcal{S}, \mathcal{S} \vee T, \mathcal{S} \vee \mathcal{PC}_T\}$, *where* $\mathcal{PC}_T := \{\{I\} : I \in PC_T\}$ *which is an ensemble version of* PC_T, *and* $\mathcal{S} \vee \mathcal{PC}_T := \{S \cup I : S \in \mathcal{S}, I \in \mathcal{PC}_T\}$. *Here symbol* $\vee$ *is element-wise union.*

We exploit the extended conditional dependencies from $\mathbb{B} \times \mathbb{C}$, i.e., we pick a bivariable from $\mathbb{B}$ and a conditional from $\mathbb{C}$, and calculate the extended conditional dependency. There are in total $|\mathbb{B}| \times |\mathbb{C}| = 24$ extended conditional dependencies.

Lemma 4.1. *Sepsets* $\mathcal{S}$ *of a UT is non-empty.*

We intend to restrict the sepsets within the vicinity of $\langle X, T, Y\rangle$. Lemma 4.1 shows the existence of such d-separation sets within vicinity. Furthermore, searching for all d-separation sets is highly time-consuming, thus significant computational cost is also saved.

• ***Entanglement within Vicinity***

Structural entanglement reflects complex structure within the vicinity of $\langle X, T, Y\rangle$. Variables X, Y and T can mutually share common neighbors, and their neighbors may also overlap with sepsets $\mathcal{S}$. We call such overlaps structural entanglement. Specifically, we exploit the overlap coefficient [38] to measure the entanglement:

Definition 4.3. (Overlap coefficient) $\textsc{olp}(\mathbf{A}, \mathbf{B}) := |\mathbf{A} \cap \mathbf{B}| / \min(|\mathbf{A}|, |\mathbf{B}|)$, *where* $\mathbf{A}$ *and* $\mathbf{B}$ *are two sets of variables.* We extend this formula to support the ensemble as input:

(Extended) Overlap coefficient: $\textsc{olp}(\mathbf{A}, \mathcal{S}) := \sum_{i=1}^{|\mathcal{S}|} \textsc{olp}(\mathbf{A}, S_i) / |\mathcal{S}|$. Naturally, we consider the entanglement in terms of the overlap coefficient on each pair of items in domain $\{PC_X, PC_Y, PC_T, \mathcal{S}\}$. Thus, use 7 scalars (including $\textsc{olp}(\{T\}, \mathcal{S})$) to represent the entanglement within the vicinity of a UT.

Embedding: We aim to represent the dependencies and entanglement by a feature vector with fixed dimensionality, which can be used to train ML4C-Learner. We adopt the standard kernel mean embedding technique in [35]. Details are available in Appendix (section B).

4.4 Learnability We have presented ML4C's featurization and see that conditional dependencies and structural entanglement have the potential to reveal asymmetry to distinguish v-structure and non-v-structure UTs. Now we provide rigorous analysis to show that, for a canonical dataset with sufficient samples, ML4C-Learner is asymptotically correct. We first propose a surrogate object called discriminative predicate:

Definition 4.4. (Discriminative predicate) *A discriminative predicate is a binary predicate function with domain as ML4C's feature vector.* A discriminative predicate can be viewed as a special classifier with pre-specified form of mechanism (e.g., $C_{\texttt{PC}}$ or $C_{\texttt{MPC}}$ is a discriminative predicate).

Definition 4.5. (Weak/Strong predicate) *Whenever a discriminative predicate takes the feature vector of a UT as input, a weak (discriminative) predicate satisfies one of the following two criteria; a strong (discriminative) predicate satisfies both: i) it is evaluated to* TRUE *if the UT is a v-structure; ii) it is evaluated to* FALSE *if the UT is not a v-structure.*

By definition, a weak predicate exhibits discriminative power since its evaluation as false implies the UT is a non-v-structure (or true implies v-structure). A strong predicate is sound and complete (e.g., $C_{\texttt{MPC}}$). Denote $\{\mathbf{A} \sim \mathbf{B}|\mathcal{Z}\} > \delta := X \sim Y|\mathbf{Z} > \delta : \forall X \in \mathbf{A}, Y \in \mathbf{B}, \mathbf{Z} \in \mathcal{Z}$, then we have:

Lemma 4.2. (Existence of strong predicate) *For a canonical dataset with infinite samples, the following are three strong discriminative predicates: i)* $\textsc{olp}(T, \mathcal{S}) = 0$, *ii)* $\textsc{olp}(T, \mathcal{S}) < 0.5$, *iii)* $\textsc{olp}(T, \mathcal{S}) < 1 \wedge \min\{X \sim Y|T \cup \mathcal{S}\} > 0$.

For example, *ii)* is the orientation logic of $C_{\texttt{MPC}}$.

Lemma 4.3. (Existence of weak predicate) *For a canonical dataset with infinite samples, the following are three weak discriminative predicates: i)* $\{X \sim Y|T\} > 0$, *ii)* $\{X \sim Y|\mathcal{PC}_T\} = 0$, *iii)* $\{PC_X \sim PC_Y|\mathcal{S} \cup T\} > 0$.

Take *i)* $\{X \sim Y|T\} > 0$ for example, if $\langle X, T, Y\rangle$ is a v-structure, then T is a collider that unblocks X and Y

through path $X - T - Y$, so $\{X \sim Y|T\} > 0$. Similarly, *ii)* if a UT is not a v-structure, then $\forall pc_T \in \mathcal{PC}_T$, X and Y must be d-connected by pc_T. Weak predicates exploit graphical implications within the vicinity. They are valid in one direction, i.e., either sufficient or necessary but not both. In addition to strong predicates, such one-direction-asymmetry exhibits additional discriminative power from the ML perspective.

THEOREM 4.1. *ML4C-Learner is asymptotically correct on classifying a canonical dataset with sufficient samples.*

5 Evaluation

Benchmark Datasets: We use discrete datasets of all 24 networks from bnlearn repository [33] for evaluation. For each dataset we sample 1k, 5k, 10k, 15k, 20k records for use.

ML4C's Training and Inference: We generate ML4C's training data synthetically (which is also used for other SCL competitors). Specifically, 400 unique DAGs are randomly generated by two models: Erdős-Rényi (ER) model [12] and Scale-Free (SF) model [2], with the number of nodes ranging from 10 to 1,000. A standard random forward data generation process is applied to obtain 10k observational samples for each graph. We further extract UTs from the 400 DAGs, consisting of 97,010 v-structures (label = 1) and 195,691 non-v-structures (label = 0). We use these instances to train ML4C-Learner, which is implemented by a XGBoost [6] binary classifier with default hyper-parameters and we use binary cross-entropy as the loss function. Details on our synthesis procedure, configurations and implementation of ML4C-Learner are available in the supplementary material.

Competitors: We categorize state-of-the-art causal learning algorithms from two aspects, supervised vs. unsupervised, and can or cannot take a skeleton as input. We choose Jarfo [13], D2C [5], RCC [23], and NCC [25] as SCL competitors. Same as ML4C, all these algorithms require a skeleton as input. All these algorithms use ML4C's training set for training and with appropriate learning target extracted. Regarding unsupervised algorithms, we choose PC [36], Conservative-PC (CPC) [31], Majority-rule PC (MPC) [10], GLL-MB (GMB) [3], GES [7], Grow-Shrink (GS) [27], Hill-Climbing (HC) [19], Conditional Distribution Similarity (CDS) [13] and GOBNILP (GNIP) [9]. which can also take skeleton as input. Lastly, we also compare with DAG-GNN (DGNN) [40], BLIP [32], and GRaSP (GRSP) [20], which are unsupervised algorithms but cannot take skeletons as input. All these competitors are capable of dealing with discrete data. All experiments are done in a Windows Server with 2.8GHz Intel E5-2680 CPU and 256G RAM. Details are in the appendix (section B).

Design: Our evaluation mainly consists of two parts: end-to-end comparison with competitors on benchmark datasets, and in-depth experiments on ML4C's learnability. The latter is further divided into four aspects: i) **Towards a perfect classifier**. ML4C-Learner is asymptotically correct, and we would like to know far it is from such a perfect classifier in real-world settings. ii) **Reliability** (against weak/strong predicates). As stated in lemma 4.3 and 4.2, there exist weak and strong predicates, which provide prediction power of ML4C's feature vector. Thus, we would like to see how ML4C-Learner takes advantage of machine learning, to learn a more reliable classification mechanism than individual weak/strong predicates. iii) **Robustness** (against varied sample size). It is known that many causal learning algorithms lack robustness w.r.t sample noise for finite datasets [23], especially CI tests are error-prone on small samples for constraint-based algorithms. We want to evaluate the robustness of ML4C (i.e., the classification mechanism) against varied sample sizes. iv) **Tolerance** (to imperfect skeleton input). Besides using ground-truth skeleton as input, it is also interesting to see how ML4C's performance changes when the input skeleton is misspecified. This also provides a fairer comparison with the baseline approaches which do not take skeleton as input.

Metrics: We use two standard metrics for performance evaluation: Structural Hamming Distance (SHD) and F1-score. For each dataset, we measure the SHD / F1-score of the output CPDAG (learned by a specific algorithm) against the ground truth CPDAG. Specifically, SHD is calculated at CPDAG level, which is the smallest number of edge additions, deletions, direction reversals and type changes (directed vs. undirected) to convert the output CPDAG to ground truth CPDAG. F1-score is calculated over identifiable edges. Roughly, F1-score can be viewed as a normalized version of SHD. Now we present the experiment results:

End-to-End Comparison: Due to page limits, we report SHD and F1-score of all algorithms on top-19 large-scale datasets (results of the other 5 smallest datasets are omitted due to triviality), as depicted in Table 1. '-' means the algorithm fails on the dataset (either out-of-memory/exceeds 24 hours execution time/break caused by unknown errors). ML4C significantly outperforms all other competitors. The average F1-score of ML4C is the highest (0.92, first column in Table 2). Moreover, ML4C exhibits the most stable performance across all datasets, its average ranking is 1.5 ± 0.7, while the second best is GLL-MB (GMB), with average ranking 4.4 ± 2.4. NCC is the strongest SCL competitor. It performs well on some datasets but overall its per-

Table 1: Experiment results for end-to-end comparison with SOTA causal learning algorithms on benchmark datasets. Algorithm names are abbreviated. SHD and F1-score are reported. The last two rows show statistics of rank by SHD and F1-score for all competitors (Note: F1-score is at UT level).

Datasets		supervised					unsupervised									no skeleton input		
#nodes/#edges		ML4C	Jarfo	D2C	RCC	NCC	PC	CPC	MPC	GMB	GES	GS	HC	CDS	GNIP	DGNN	BLIP	GRSP
child	SHD	**0**	18	16	18	20	22	13	9	20	15	13	13	18	**0**	23	**0**	**0**
20/25	F1	**1.0**	.24	.43	.33	.12	.12	.00	.74	.12	.47	.59	.57	.34	**1.0**	.25	**1.0**	**1.0**
insurance	SHD	**5**	41	30	34	28	36	34	21	29	34	28	19	36	11	53	35	14
27/52	F1	**.89**	.26	.44	.42	.44	.39	.00	.66	.55	.46	.56	.76	.36	.88	.05	.51	.83
water	SHD	5	33	43	31	**0**	4	60	7	8	38	27	38	18	34	61	65	58
32/66	F1	.94	.52	.34	.56	**1.0**	.97	.00	.91	.87	.49	.62	.46	.76	.57	.00	.20	.21
mildew	SHD	6	-	17	25	34	21	-	-	7	**3**	9	23	18	-	52	36	37
35/46	F1	.87	-	.68	.50	.33	.56	-	-	.85	**.93**	.80	.64	.65	-	.19	.41	.47
alarm	SHD	**1**	21	26	18	20	20	20	6	17	8	3	21	18	2	46	17	2
37/46	F1	**.98**	.57	.44	.64	.57	.57	.57	.92	.64	.86	.94	.66	.62	**.98**	.12	.82	**.98**
barley	SHD	5	48	55	50	**0**	3	-	-	8	42	-	34	50	34	87	60	70
48/84	F1	.95	.46	.38	.44	**1.0**	.96	-	-	.91	.59	-	.72	.43	.74	.00	.48	.45
hailfinder	SHD	11	47	41	43	**0**	17	-	-	26	60	-	59	44	56	76	111	114
56/66	F1	.80	.37	.45	.42	**1.0**	.85	-	-	.70	.21	-	.23	.42	.32	.00	.18	.15
hepar2	SHD	**0**	54	81	59	**0**	35	27	37	14	46	40	35	75	48	123	79	64
70/123	F1	**1.0**	.59	.34	.54	**1.0**	.72	.81	.70	.89	.75	.70	.81	.39	.72	.00	.54	.63
win95pts	SHD	1	65	51	33	**0**	8	42	7	5	32	21	16	50	27	112	103	23
76/112	F1	.99	.43	.54	.73	**1.0**	.95	.64	.95	.97	.77	.85	.91	.57	.81	.00	.47	.88
pathfinder	SHD	25	157	145	151	**0**	150	-	-	147	158	-	168	148	119	196	241	-
109/195	F1	.77	.21	.29	.21	**1.0**	.29	-	-	.30	.29	-	.28	.31	.00	.00	.07	-
munin1	SHD	**10**	169	154	153	72	86	117	-	84	109	-	233	151	265	-	257	-
186/273	F1	**.97**	.42	.47	.46	.77	.71	.58	-	.72	.67	-	.26	.50	.00	-	.42	-
andes	SHD	**0**	226	209	246	**0**	4	83	4	5	47	15	38	149	328	-	175	146
223/338	F1	**1.0**	.35	.41	.29	**1.0**	.99	.75	.99	.98	.92	.96	.92	.60	.00	-	.76	.75
diabetes	SHD	25	220	395	237	48	**0**	-	-	204	146	-	592	368	-	-	534	-
413/602	F1	.96	.62	.38	.62	.96	**1.0**	-	-	.68	.77	-	.03	.43	-	-	.43	-
pigs	SHD	**0**	350	332	263	400	400	-	-	268	**0**	-	532	316	-	-	6	-
441/592	F1	**1.0**	.44	.46	.59	.35	.35	-	-	.56	**1.0**	-	.18	.50	-	-	**1.0**	-
link	SHD	**0**	731	630	638	749	737	-	-	204	324	-	1047	400	-	-	947	-
724/1125	F1	**1.0**	.38	.45	.45	.39	.40	-	-	.81	.80	-	.14	.64	-	-	.49	-
munin	SHD	72	967	790	816	**0**	156	-	-	458	661	-	1397	795	-	-	1599	-
1041/1397	F1	.95	.36	.48	.44	**1.0**	.89	-	-	.69	.62	-	.00	.51	-	-	.29	-
munin2	SHD	**118**	554	611	646	1052	898	-	-	536	632	-	1240	753	-	-	1321	-
1003/1244	F1	**.92**	.60	.56	.55	.19	.30	-	-	.57	.58	-	.01	.49	-	-	.46	-
munin3	SHD	**113**	616	629	688	1048	860	-	-	544	566	-	1306	819	-	-	1539	-
1041/1306	F1	**.92**	.58	.57	.54	.25	.37	-	-	.60	.65	-	.00	.46	-	-	.26	-
munin4	SHD	**126**	696	658	776	1058	876	-	-	649	618	-	1388	812	-	-	1627	-
1038/1388	F1	**.93**	.54	.56	.50	.29	.39	-	-	.55	.64	-	.00	.49	-	-	.28	-
rank(SHD)	mean	**1.5**	9.5	8.7	8.4	5.2	6.5	11.3	9.9	4.6	6.3	10.0	9.0	8.4	9.4	14.1	11.0	10.6
	±stdd	**0.7**	3.7	4.0	2.6	4.4	4.0	3.0	4.2	2.7	3.3	3.6	2.7	2.8	4.4	2.2	3.8	4.0
UT-F1	mean	**.90**	.22	.19	.27	.66	.50	.53	.87	.59	.54	.77	.47	.30	.55	.09	.36	.55
	±stdd	**.13**	.17	.13	.18	.40	.34	.33	.16	.32	.28	.24	.35	.22	.42	.07	.29	.38

Table 2: Reliability: average F1-score of ML4C vs. 8 discriminative predicates derived from ML4C features on benchmarks.

		strong predicates				weak predicates			
id	ML4C	1	2	3	4	1	2	3	4
F1	**.92±.20**	.77±.31	.52±.27	.38±.25	.66±.27	.72±.25	.61±.29	.73±.30	.55±.27

formance fluctuates significantly. Overall it only ranks 5.1 ± 4.2. Last but not least, ML4C shows high accuracy (F1>0.9) on very large-scale datasets (e.g., medicine datasets 'munin2/3/4' [4]) while max(others) ~ 0.6.

Towards a Perfect Classifier: The last row of Table 1 shows the performance of ML4C-Learner component at UT level by UT-F1 (i.e., F1-score of classifying UTs): such UT level accuracy is crucial for causal learning on discrete data, since the set of v-structures is invariant across all Markov equivalent DAGs and it can fully recover the CPDAG. The average F1-score of ML4C-Learner is 0.90 ± 0.13, which shows promising results towards a perfect classifier.

Reliability: We manually identify 4 strong predicates and 4 weak predicates and treat each one as a replacement of ML4C-Learner. Table 2 shows the performance of these predicates. Although most predicates show value on discriminating UTs (e.g., 5/8 predicates are with >0.6 F1-score), ML4C-Learner has higher performance (average F1-score $= 0.92$) than each individual predicate (best average F1-score $= 0.77$). Thus, it is evident that ML4C-Learner learns a more reliable clas-

Table 3: Robustness: ML4C is trained on synthetic datasets with sample size = 10k, but tested on benchmark datasets with different sample sizes ∈ {1k, 5k, 10k, 15k, 20k}.

	size	1k	5k	10k	15k	20k	size	1k	5k	10k	15k	20k	size	1k	5k	10k	15k	20k
SHD	insu	11	1	5	1	0	wate	12	11	5	8	6	mild	8	5	6	6	1
F1		.81	.97	.89	.97	1.0		.86	.87	.94	.89	.93		.83	.89	.87	.87	.98
SHD	alar	5	4	1	1	5	barl	13	9	5	8	6	hail	15	15	11	15	13
F1		.93	.95	.98	.98	.93		.88	.93	.95	.92	.94		.74	.72	.80	.72	.76
SHD	hepa	8	2	0	1	2	win9	7	1	1	1	1	path	1	7	25	7	1
F1		.96	.99	1.0	.99	.99		.96	.99	.99	.99	.99		.99	.92	.77	.92	.99
SHD	mun1	32	7	10	9	15	ande	3	2	0	2	0	diab	18	28	25	26	27
F1		.89	.98	.97	.97	.95		.99	.99	1.0	.99	1.0		.97	.95	.96	.96	.96
SHD	pigs	0	0	0	0	0	link	88	13	0	0	0	mun	107	76	72	93	87
F1		1.0	1.0	1.0	1.0	1.0		.93	.99	1.0	1.0	1.0		.93	.95	.95	.94	.94
SHD	mun2	117	95	118	110	97	mun3	151	119	113	99	62	mun4	165	130	126	146	133
F1		.92	.93	.92	.93	.93		.90	.92	.92	.94	.96		.90	.92	.93	.91	.93

Table 4: Tolerance: Use the imperfect skeleton identified by BLIP as ML4C's input. 'SA' denotes skeleton accuracy. 'M' is short for ML4C. 'B' is short for BLIP.

	data	SA	M	B	data	SA	M	B	data	SA	M	B	data	SA	M	B
SHD	asia	.82	6	6	canc	1.0	**0**	4	eart	.89	5	5	sach	.97	14	**1**
F1			.57	.57			**1.0**	.00			.00	.00			.00	.00
SHD	surv	.91	**2**	6	alar	.91	**13**	17	barl	.70	**52**	60	chil	1.0	0	0
F1			**.73**	.00			**.84**	.82			**.57**	.48			1.0	1.0
SHD	insu	.78	**31**	35	mild	.69	**31**	36	wate	.48	**63**	65	hail	.16	111	111
F1			**.59**	.51			**.56**	.41			**.25**	.20			.17	**.18**
SHD	hepa	.71	85	**79**	win9	.71	**83**	103	ande	.80	**158**	175	diab	.66	**522**	534
F1			.46	**.54**			**.63**	.47			**.78**	.76			**.44**	.43
SHD	link	.61	**916**	947	mun1	.57	**249**	257	path	.35	259	**241**	pigs	1.0	12	**6**
F1			**.53**	.49			**.49**	.42			**.12**	.07			.99	**1.0**
SHD	mun	.50	**1484**	1599	mun3	.49	**1410**	1539	mun4	.45	**1565**	1627				
F1			**.36**	.29			**.43**	.26			**.37**	.28				

sification mechanism, by taking advantage of machine learning techniques.

Robustness: To evaluate robustness, ML4C is trained on synthetic datasets with sample size = 10k, but it is tested on benchmark datasets with different sample sizes: 1k, 5k, 10k, 15k and 20k respectively. Table 3 shows that ML4C exhibits satisfactory robustness (decrease of F1-score is less than 0.1) against sample size on 17 out of 18 datasets.

Tolerance: BLIP [32] is the strongest competitor among the three algorithms without skeleton input. To evaluate ML4C's tolerance to imperfect skeletons, we use the skeleton identified by BLIP (i.e., convert its DAG output to the corresponding skeleton) as input for ML4C. The result is shown in Table 4. Among the 23 datasets, ML4C is better than BLIP on 16 datasets, tied on 4 datasets and BLIP is better than ML4C only on 3 datasets (even for these datasets, ML4C still has very close accuracy to BLIP). Moreover, for the datasets where BLIP produces skeletons with very low accuracy (such as munins, skeleton accuracies$\sim$ 0.5), ML4C has significantly better accuracy than BLIP, which shows ML4C's better ability for orientation and also tolerance w.r.t. skeleton misspecification.

6 Conclusion and Future Work

We have proposed a supervised causal learning algorithm ML4C, with theoretical guarantee on learnability and remarkable empirical performance. ML4C shows promising results on validating the effectiveness of supervision. To make SCL practical in real-world scenarios, one important direction for future work is to identify the reliable and accurate skeleton from data, considering ML4C requires the skeleton as additional input. Another important future work is to incorporate continuous data with the identifiability results from various parametric assumptions.

References

[1] A. Agresti, *Categorical data analysis*, vol. 482, John Wiley & Sons, 2003.

[2] R. Albert and A.-L. Barabási, *Statistical mechanics of complex networks*, Reviews of modern physics, 74 (2002), p. 47.

[3] C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. D. Koutsoukos, *Local causal and markov blanket induction for causal discovery and feature selection for classification part i: algorithms and empirical evaluation.*, Journal of Machine Learning Research, 11 (2010).

[4] S. Andreassen, M. Woldbye, B. Falck, and S. K. Andersen, *Munin: A causal probabilistic network for interpretation of electromyographic findings*, in Proceedings of the 10th international joint conference on Artificial intelligence-Volume 1, 1987, pp. 366–372.

[5] G. Bontempi and M. Flauder, *From dependency to causality: a machine learning approach.*, J. Mach. Learn. Res., 16 (2015), pp. 2437–2457.

[6] T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system*, in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[7] D. M. Chickering, *Optimal structure identification with greedy search*, Journal of machine learning research, 3 (2002), pp. 507–554.

[8] T. M. Cover, *Elements of information theory*, John Wiley & Sons, 1999.

[9] J. Cussens, *Bayesian network learning with cutting planes*, arXiv preprint arXiv:1202.3713, (2012).

[10] C. Diego and H. M. Marloes, *Order-independent constraint-based causal structure learning*, Journal of Machine Learning Research, 15 (2014), pp. 3921–3962.

[11] R. Ding, Y. Liu, J. Tian, Z. Fu, S. Han, and D. Zhang, *Reliable and efficient anytime skeleton learning*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020.

[12] P. Erdős and R. A., *On random graphs*, Publicationes, Mathematicae, 6 (1959), pp. 290–297.

[13] J. A. Fonollosa, *Conditional distribution variability measures for causality detection*, in Cause Effect Pairs in Machine Learning, Springer, 2019, pp. 339–347.

[14] C. Glymour, K. Zhang, and P. Spirtes, *Review of causal discovery methods based on graphical models*, Frontiers in genetics, 10 (2019), p. 524.

[15] A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, A. J. Smola, et al., *A kernel statistical test of independence.*, in Nips, vol. 20, Citeseer, 2007, pp. 585–592.

[16] P. Hoyer, D. Janzing, J. M. Mooij, J. Peters, and B. Schölkopf, *Nonlinear causal discovery with additive noise models*, Advances in neural information processing systems, 21 (2008), pp. 689–696.

[17] D. Janzing, J. Mooij, K. Zhang, J. Lemeire, J. Zscheischler, P. Daniušis, B. Steudel, and B. Schölkopf, *Information-geometric approach to inferring causal directions*, Artificial Intelligence, 182 (2012), pp. 1–31.

[18] D. Kalainathan and O. Goudet, *Causal discovery toolbox: Uncover causal relationships in python*, arXiv preprint arXiv:1903.02278, (2019).

[19] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*, MIT press, 2009.

[20] W.-Y. Lam, B. Andrews, and J. Ramsey, *Greedy relaxations of the sparsest permutation algorithm*, in The 38th Conference on Uncertainty in Artificial Intelligence, 2022.

[21] S. L. Lauritzen, *Graphical models*, vol. 17, Clarendon Press, 1996.

[22] H. Li, Q. Xiao, and J. Tian, *Supervised whole dag causal discovery*, arXiv preprint arXiv:2006.04697, (2020).

[23] D. Lopez-Paz, K. Muandet, and B. Recht, *The randomized causation coefficient.*, J. Mach. Learn. Res., 16 (2015), pp. 2901–2907.

[24] D. Lopez-Paz, K. Muandet, B. Schölkopf, and I. Tolstikhin, *Towards a learning theory of cause-effect inference*, in Proceedings of the 32nd International Conference on Machine Learning, PMLR, 2015, pp. 1452–1461.

[25] D. Lopez-Paz, R. Nishihara, S. Chintala, B. Scholkopf, and L. Bottou, *Discovering causal signals in images*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6979–6987.

[26] L. Lorch, J. Rothfuss, B. Schölkopf, and A. Krause, *Dibs: Differentiable bayesian structure learning*, Advances in Neural Information Processing Systems, (2021).

[27] D. Margaritis, *Learning bayesian network model structure from data*, tech. rep., Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2003.

[28] C. Meek, *Causal inference and causal explanation with background knowledge*, arXiv preprint arXiv:1302.4972, (2013).

[29] C. Meek, *Strong completeness and faithfulness in bayesian networks*, arXiv preprint arXiv:1302.4973, (2013).

[30] I. Ng, S. Zhu, Z. Fang, H. Li, Z. Chen, and J. Wang, *Masked gradient-based causal structure learning*, in Proceedings of the 2022 SIAM International Conference on Data Mining (SDM), 2022.

[31] J. Ramsey, J. Zhang, and P. Spirtes, *Adjacency-faithfulness and conservative causal inference*, arXiv preprint arXiv:1206.6843, (2012).

[32] M. Scanagatta, C. P. de Campos, G. Corani, and M. Zaffalon, *Learning bayesian networks with thousands of variables.*, in NIPS, 2015, pp. 1864–1872.

[33] M. Scutari, *Bayesian network repository*, URL http://www. bnlearn. com, (2012).

[34] S. Shimizu, P. O. Hoyer, A. Hyvärinen, A. Kerminen, and M. Jordan, *A linear non-gaussian acyclic model for causal discovery.*, Journal of Machine Learning Research, 7 (2006).

[35] A. Smola, A. Gretton, L. Song, and B. Schölkopf, *A hilbert space embedding for distributions*, in International Conference on Algorithmic Learning Theory, Springer, 2007, pp. 13–31.

[36] P. Spirtes and C. Glymour, *An algorithm for fast recovery of sparse causal graphs*, Social science computer review, 9 (1991), pp. 62–72.

[37] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman, *Causation, prediction, and search*, MIT press, 2000.

[38] M. Vijaymeena and K. Kavitha, *A survey on similarity measures in text mining*, Machine Learning and Applications: An International Journal, 3 (2016), pp. 19–28.

[39] K. Yu, J. Li, and L. Liu, *A review on algorithms for constraint-based causal discovery*, arXiv preprint arXiv:1611.03977, (2016).

[40] Y. Yu, J. Chen, T. Gao, and M. Yu, *Dag-gnn: Dag structure learning with graph neural networks*, in International Conference on Machine Learning, PMLR, 2019, pp. 7154–7163.

[41] X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing, *Dags with no tears: Continuous optimization for structure learning*, arXiv preprint arXiv:1803.01422, (2018).