ML4S: Learning Causal Skeleton from Vicinal Graphs

Pingchuan Ma*
HKUST
Hong Kong, China

Rui Ding†
Microsoft Research
Beijing, China

Haoyue Dai
Carnegie Mellon University
Pittsburgh, USA

Yuanyuan Jiang
Renmin University of China
Beijing, China

Shuai Wang
HKUST
Hong Kong, China

Shi Han
Dongmei Zhang
Microsoft Research
Beijing, China

ABSTRACT

Causal skeleton learning aims to identify the undirected graph of the underlying causal Bayesian network (BN) from observational data. It plays a pivotal role in causal discovery and many other downstream applications. The methods for causal skeleton learning fall into three primary categories: constraint-based, score-based, and gradient-based methods. This paper, for the first time, advocates for learning a causal skeleton in a supervision-based setting, where the algorithm learns from additional datasets associated with the ground-truth BNs (complementary to input observational data). Concretizing a supervision-based method is non-trivial due to the high complexity of the problem itself, and the potential "domain shift" between training data (i.e., additional datasets associated with ground-truth BNs) and test data (i.e., observational data) in the supervision-based setting. First, it is well-known that skeleton learning suffers worst-case exponential complexity. Second, conventional supervised learning assumes an independent and identical distribution (i.i.d.) on test data, which is not easily attainable due to the divergent underlying causal mechanisms between training and test data. Our proposed framework, ML4S, adopts order-based cascade classifiers and pruning strategies that can withstand high computational overhead without sacrificing accuracy. To address the "domain shift" challenge, we generate training data from *vicinal graphs* w.r.t. the target BN. The associated datasets of *vicinal graphs* share similar joint distributions with the observational data. We evaluate ML4S on a variety of datasets and observe that it remarkably outperforms the state of the arts, demonstrating the great potential of the supervision-based skeleton learning paradigm.

CCS CONCEPTS

• **Mathematics of computing → Bayesian networks**.

KEYWORDS

causal discovery, structure learning, Bayesian network

ACM Reference Format:
Pingchuan Ma, Rui Ding, Haoyue Dai, Yuanyuan Jiang, Shuai Wang, Shi Han, and Dongmei Zhang. 2022. ML4S: Learning Causal Skeleton from Vicinal Graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539447

1 INTRODUCTION

Causal skeleton learning aims to identify the undirected graph (i.e., skeleton) of the underlying causal Bayesian network (BN) from observational data. It forms the basis of many popular downstream applications, including causal discovery [15], optimal feature selection [35, 36] and experimental design [18, 30].

Table 1: Four categories for skeleton learning.

Category	Formulation
Constraint-based	$\exists \boldsymbol{z}. v_i \perp\!\!\!\perp v_j \mid \boldsymbol{z} \iff \neg v_i - v_j$
Score-based	maximize $f(G)$ subject to G is DAG
Gradient-based	maximize $f(W)$ subject to $h(W) = 0$
Supervision-based (ML4S)	estimate $Pr(v_i - v_j \mid \boldsymbol{x}_{ij}, \{(D_1, G_1), \cdots\})$

v_i, v_j: vertices in the skeleton;
$\boldsymbol{z}$: a set of vertices in the skeleton;
$v_i - v_j$: the two vertices are adjacent; ($\neg$ denotes negation)
G: Bayesian network;
W: adjacency matrix;
f: score function;
h: acyclicity constraint function;
$\boldsymbol{x}_{ij}$: feature for estimating $Pr(v_i - v_j)$ (probability of $v_i - v_j$);
(D, G): generated data and associated Bayesian networks for training;

As illustrated in Table 1, the methods of skeleton learning primarily fall into three categories: constraint-based, score-based, and gradient-based methods. Each of these methods takes observational data as input and produces a skeleton but with different strategies. Constraint-based methods progressively eliminate spurious adjacencies based on conditional independence relations between variables. By using either combinatorial or continuous optimization techniques, score-based and gradient-based methods search for a directed acyclic graph (DAG) to maximize a predefined score function, and the skeleton is obtained as the undirected graph of the DAG. In summary, these methods do not have access to any

*The work was done during his internship at Microsoft Research Asia.
†Corresponding author: juding@microsoft.com.

KDD '22, August 14–18, 2022, Washington, DC, USA.

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539447

additional datasets associated with ground-truth BNs or skeletons, which can be viewed as unsupervised.

In contrast, in this paper, we advocate a new aspect of skeleton learning that learns a skeleton in a supervision-based setting: an algorithm can access additional datasets associated with their ground-truth BNs. A machine learning (ML) model is trained on these datasets (labels are derived from the associated ground-truth BNs), and then it is used to predict adjacencies on the observational data. The supervision-based method enjoys the benefit of "free" acquisition of training data: with forward sampling techniques [2], we can generate additional datasets from as many synthetic BNs in the DAG space as needed. In fact, many existing algorithms can be deemed as a binary classifier with hardcoded rules.

Remark. *Given observational data D sampled from a BN with vertices $V = \{v_1, \cdots, v_p\}$, the following feature and binary classifier predict the adjacency between each pair of vertices. Given v_i, v_j, the feature can be extracted as*

$$\boldsymbol{x}_{ij} = \min_{\boldsymbol{z} \subseteq V \setminus \{v_i, v_j\}} \{v_i \sim v_j \mid \boldsymbol{z}\} \tag{1}$$

where $\{v_i \sim v_j \mid \boldsymbol{z}\}$ is a scalar value that measures the conditional dependency between v_i, v_j given $\boldsymbol{z}$. The classifier C_D is defined as

$$C_D(v_i, v_j) = \begin{cases} \textit{adjacent} & \boldsymbol{x}_{ij} \neq 0 \\ \textit{non-adjacent} & \boldsymbol{x}_{ij} = 0 \end{cases} \tag{2}$$

Under the canonical assumption, C_D yields the correct skeleton by enumerating all pairs of vertices. Appendix A contains the proof.

Concretizing supervision-based skeleton learning is non-trivial. First, skeleton learning suffers exponential complexity [32]. For instance, Eqn. 1 encapsulates all conditional dependencies as the feature associated with each node pair, whose number is combinatorial. Further, due to divergent underlying causal mechanisms between synthetic BNs and target BNs, supervision-based methods encounter potential "domain shifts" between training and test data. We introduce how we address the two challenges in the following.

Order-based Cascade Classifiers. Overall, one CI test can only refute adjacency. That is, adjacency can only be established after exhaustively enumerating all possible condition sets (i.e., $\boldsymbol{z}$ of constraint-based methods in Table 1), imposing super-exponential complexity. Typical constraint-based methods, for example, the PC algorithm [32], conduct hypothesis tests *incrementally* w.r.t. order size (i.e., cardinality of condition set). Once conditional independence is observed, it is sufficient to establish non-adjacency and no further tests are needed. Inspired by this, ML4S adopts a novel learning paradigm over a sequence of cascade classifiers, with each model (i.e., a classifier) corresponding to a particular order taking the results of the CI tests up to that order. Higher order models are applied only when lower order models are unable to refute adjacency. Finally, adjacencies are confirmed when no model can refute them. We formulate the problem and show that it asymptotically converges to the ground-truth BN's skeleton. We introduce informative features to enhance the supervision-based learning quality (Sec. 4.1) and pruning strategies to reduce costly CI tests (Sec. 4.2).

Generate Training Data from Vicinal Graphs. Given a parameterized BN, we can generate associated dataset by forward sampling [2]. Thus, training data can be obtained by randomly generated BNs (e.g., use Erdős-Rényi Model) [12]. However, as shown in Sec. 6, it is challenging for machine learning models trained only on random BNs to predict quality skeletons on real-world benchmarks, which may due to "domain shift" between training and test data. Motivated by this, it would be desirable to use $(\boldsymbol{X}^v, G^v)$ (i.e., the pair of generated dataset $\boldsymbol{X}^v$ and its associated BN G^v) such that $\boldsymbol{X}^v$ has a "similar" distribution w.r.t. the distribution P_X of the input observational data (i.e., test data). To do so, we propose a novel approach to generating training data from vicinal graphs. Each vicinal graph corresponds to a BN that is Markov compatible to a similar distribution P'_X of P_X (i.e., instance of the Markov equivalence class). The node adjacency of vicinal graphs forms the label for training. Note that generating vicinal graphs does not necessitate *ground-truth DAG*, which is unattainable. Instead, we propose a practical method for generating vicinal graphs by mutating a pseudo BN from a proxy algorithm (e.g., PC algorithm) and parameterizing them via estimated Dirichlet distributions (Sec. 5).

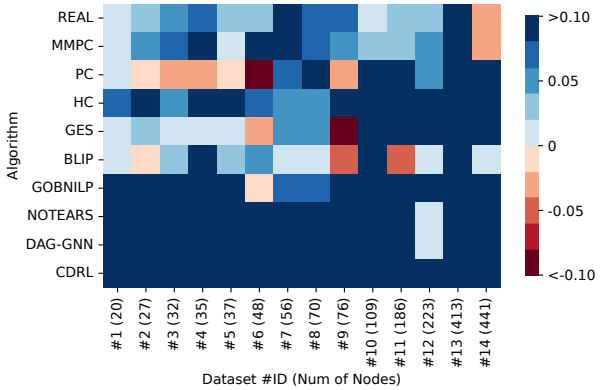


Figure 1: Comparison of ML4S and other algorithms on F1 Score. Blue implies that ML4S is superior whereas red denotes the compared algorithm is superior.

Evaluation Highlight. We compare ML4S and other algorithms in Fig. 1 based on the F1 Score for output skeletons on 14 datasets from the bnlearn benchmark [29]. Each grid in the heatmap represents the amount by which ML4S improves or degrades the F1 Score when compared to other algorithms. Overall, ML4S is among the top-3 best algorithms in terms of F1 Score on 12 of the 14 datasets and shows superior performance to all algorithms on the overall rankings. We interpret the findings as very encouraging. Particularly, ML4S outperforms other algorithms by a remarkable margin on the diabetes dataset (#13), a highly challenging large causal graph with 413 nodes (see details in Sec. 6). In summary, we conclude our contribution as follows:

- We, for the first time, formulate causal skeleton learning in a supervision-based paradigm. A trained machine learning model decides the node adjacency in Bayesian networks.
- We instantiate supervision-based skeleton learning into a novel procedure of learning cascade classifiers. A set of features and pruning strategies have been proposed to boost this procedure.
- To prepare training data, we introduce a novel approach to generating dataset-specific vicinal graphs; this enables applying supervision-based schemes in a practical and robust manner to diverse real-world datasets.
- We evaluate ML4S using 14 challenging datasets from the bnlearn repository. ML4S remarkably outperforms and consistently improves the state of the arts. ML4S is available at [1].

2 RELATED WORK

Causal Skeleton Learning. Causal skeleton learning serves the cornerstone of many causal discovery algorithms. Algorithms from various categories typically employ different strategies for learning skeletons. Constraint-based algorithms infer adjacency from conditional independence based on observational data. PC begins with a complete graph and deletes edges using conditional independence from hypothesis tests [32]. There are also PC-derived algorithms, including PC-stable [8] and Consistent-PC [20], which improve the robustness of skeleton learning. GS adopts grow-shrink strategy to learn skeleton [25]. It begins with an edgeless graph with edges being added in the growing phase; it then removes spurious edges in the shrinking phase. Many algorithms adopt a similar strategy to learn the skeleton, including TPDA [6] and MMPC [33]. REAL enhances skeleton learning by mitigating faithfulness violations induced by deterministic relations among variable dependencies [11]. Score-based algorithms recover skeleton graphs by maximizing a pre-defined scoring criterion, such as BDe(u) [16]. Representative score-based algorithms includes HC [33], GES [7], BLIP [28] and GOBNILP [9]. Recently, gradient-based algorithms have been proposed. For example, NOTEARS [39] and DAG-GNN [37] seek to recover the data generation process while adhering to acyclicity constraints. In summary, these methods do not have access to any additional datasets associated with ground-truth BNs or skeletons, which can be viewed as unsupervised approaches.

Supervised Causal Discovery. Supervised causal discovery has become an emerging topic, with the goal of leveraging the power of machine learning models to distinguish causes and effects. The majority of them focus on predicting pairwise causal directions [3, 13, 22, 23]. Li et al. aim to learn the whole DAG from data generated by linear Structural Equation Model [21]. ML4C takes a skeleton as input and predicts whether an unshielded triple is a v-structure [10]. In short, existing works focus on predicting directions on undirected causal relationships and form the second phase of causal discovery. Therefore, they are *complementary* to ML4S, which recovers skeletons by supervision-based learning.

Downstream Applications of Skeleton Learning. Various downstream applications and algorithms make use of skeletons. First, skeletons can be used to decouple the learning of skeletons from orientations in the majority of causal discovery algorithms. There are several well-known optimizations for existing algorithms in case skeletons are already known [33]. For constraint-based algorithms, its orientation rules can be applied seamlessly on ML4S-identified skeletons. For score-based algorithms, instead of starting from scratch, searching a DAG from a skeleton would notably reduce overall computational overheads. Gradient-based algorithms can also be boosted by skeleton learning: we show this combination through a case study (see Sec. 6 and Appendix F). Skeletons also serve as the cornerstone of other downstream applications, such as optimal feature selection [35, 36] and experimental design [18, 30].

3 PRELIMINARIES

Notations. We use X_i to represent a variable and use $\boldsymbol{X}$ to represent a set of variables. The node (vertex) in a graph (BN or skeleton) is represented by the lowercase x_i of the variable X_i. We use V_G, E_G denote the set of nodes and edges in a graph G, respectively. $v_i - v_j \in E_G$ denotes that v_i and v_j are adjacent in G and $v_i \to v_j \in E_G$ denotes that v_i is a parent of v_j. $N_G(v) \coloneqq \{v' \mid v' - v \in E_G\}$ denotes all neighbors of v in skeleton or BN G (in BN, $v' - v$ denotes either $v' \to v$ or $v' \leftarrow v$) and $Pa_G(v) \coloneqq \{v' \mid v' \to v \in E_G\}$ denotes all parents of v in BN G.

3.1 Markov Assumption and Faithfulness

Suppose the observational data follows a joint distribution $P_{\boldsymbol{X}}$ and a BN G. Markov assumption implies a recursive decomposition to $P_{\boldsymbol{X}}$ given G such that

$$P_{\boldsymbol{X}} = \prod P(X_i \mid Pa_G(x_i)) \tag{3}$$

Global Markov Property [19] holds true when Markov assumption holds true. It states that

$$x_i \perp\!\!\!\perp_G x_j \mid \boldsymbol{z} \implies x_i \perp\!\!\!\perp x_j \mid \boldsymbol{z} \tag{4}$$

where $\perp\!\!\!\perp_G$ stands for d-separation in G and $\perp\!\!\!\perp$ stands for statistical conditional independence in $P_{\boldsymbol{X}}$. We refer to [32] for the detailed definition on d-separation.

Faithfulness assumption states that conditional independence on the joint distribution implies d-separation on the BN. Formally,

$$x_i \perp\!\!\!\perp x_j \mid \boldsymbol{z} \implies x_i \perp\!\!\!\perp_G x_j \mid \boldsymbol{z} \tag{5}$$

Under the Markov assumption and faithfulness assumption, conditional independence and d-separation are equivalent. In the remainder of the paper, we use *canonical assumption* to denote Markov assumption and faithfulness assumption.

3.2 Skeleton, Identifiability and k-partial Graph

Under causal sufficiency [32], causal skeleton stands for the undirected graph of BN, whose formal definition is as follows.

Definition 3.1 (Skeleton). *A undirected graph $S = (V_S, E_S)$ represents the skeleton of a causal DAG $G = (V_G, E_G)$ if*

$$(x \to y) \in E_G \vee (y \to x) \in E_G \iff (x - y) \in E_S \tag{6}$$

When the Markov and faithfulness assumptions are satisfied, the following theorem [32] holds, which also forms the basis of many constraint-based approaches, as illustrated in Table 1.

Theorem 3.1. *Under the canonical assumption, x_i, x_j is adjacent in S if and only if $\forall \boldsymbol{z} \in V_G, x_i \not\perp\!\!\!\perp x_j \mid \boldsymbol{z}$.*

Given Theorem. 3.1, the skeleton is *identifiable* when complete information regarding conditional independence is provided. In practice, enumerating all possible conditional independence information is costly. In a more realistic scenario, conditional independence is tested sequentially. k-order conditional independence statements of x_i, x_j subsumes the conditional independence between x_i, x_j of $\forall |\boldsymbol{z}| \leq k$. *$k$-order identifiability* on a limited order of conditional independence is defined as follows.

Definition 3.2 (k-Order Identifiability). *Let $|V_G| = n$. Under canonical assumption, the adjacency between x_i, x_j is k-order identifiable if 1) $\exists |\boldsymbol{z}| \leq k$ such that $x_i \perp\!\!\!\perp x_j \mid \boldsymbol{z}$, or 2) $k \geq n - 2$.*

Def. 3.2 specifies the circumstances under which we can detect the adjacency. As a result, we can gradually expand the order of CI tests and perform CI tests solely on currently unidentifiable edges. At k-order, we at most recover a k-partial graph due to the remaining unidentifiable edges [4].

Definition 3.3 (k-partial Graph). *An undirected graph $G_k = (V_k, E_k)$ represents the k-partial graph of the ground-truth BN G if*

$$\forall x, y, (\exists |Z| \leq k, x \perp\!\!\!\perp_G y \mid Z) \iff (x - y) \notin E_k \tag{7}$$

G_k is an undirected graph, and it is clear that for $k < n-2$, all edges in G_k are unidentifiable according to Def. 3.2 under the canonical assumption. By assessing the size of minimal d-separator for each pair of nodes in a BN, we can construct the k-partial graph.

Proposition 3.1. *Let $S = (V_S, E_S)$ be the true skeleton of a BN G and $|V_G| = n$. $\forall 0 < k < n-1, E_S = E_{n-2} \subseteq E_k \subseteq E_{k-1}$.*

When the order increases, more edges become identifiable by refuting its adjacency. Therefore, k-partial graph is a subgraph to all k'-partial graph if $k' < k$. Suppose CI tests are correct. When all CI tests are given, if no CI tests can reject an adjacency, the adjacency is confirmed at the $(n-2)$-order, as all possible CI tests have been enumerated. Then, we can find the true skeleton. In practice, a threshold of maximal order is usually set to avoid combinatorial explosions. If the maximal in degree of the ground-truth BN is known, we obtain stronger results on convergence. Note that the proof to Proposition. 3.1 and 3.2 is trivial. We omit it here.

Proposition 3.2. *Let $S = (V_S, E_S)$ be the true skeleton and d be the maximal in degree of the ground-truth BN. $E_S = E_d$, where E_d is the edges of d-partial graph.*

4 ML4S

Fig. 2a depicts the workflow of ML4S. It begins by using marginal independence to construct a 0-partial graph G_0. The 0-partial graph and 1-order CI tests are then passed to the 1-order model to obtain both training and test data (we instantiate this supervision-based setting as "cascade procedure"). The output — the 1-partial graph — is further given to train the 2-order model and so on.

Remark. *Under the canonical assumption and correct CI tests, when k is sufficiently large, iteratively repeating the cascade process converges the output of the k-order model to the real skeleton, according to Proposition. 3.1 and Proposition. 3.2.*

We explain how ML4S generates a k-partial graph from a $(k-1)$-partial graph G_{k-1} in Alg. 1. Alg. 1 trains a model M_k (line 1). Then, Alg. 1 enumerates each edge in G_{k-1} (line 2), generates a feature vector for this edge (line 3, see details in Sec. 4.1), and uses the trained model to predict whether the edge exists in the k-partial graph G_k (line 4). G_k is initialized as an edgeless graph containing full nodes from G_{k-1}. If M_k predicts adjacency, the edge is added to G_k (line 5). Recall that we can at most recover the k-partial graph using k-order CI tests. With canonical assumption and correct CI tests, each model in ML4S yields the k-partial graph.

We illustrate how a k-order model is trained in Fig. 2b (lines 8–17 in Alg. 1). Here, a set of parameterized vicinal graphs are provided (see Sec. 5 for generating vicinal graphs). Given a vicinal graph, we employ forward sampling to obtain the data X^v associated to the BN, which forms a set $\mathcal{V}_{k-1}$ containing generated datasets and their $(k-1)$-partial graphs (corresponding to the vicinal graphs) estimated by the preceding model M_{k-1} (line 10). For each $(k-1)$-partial graph G^v_{k-1}, we iterate its edges (line 11), and generate features between two nodes x_i, x_j using the same *GenerateFeature* function (line 12), whose label is defined by the adjacency of x_i, x_j

Algorithm 1: Generate k-partial graph

Input: $(k-1)$-partial graph estimated by M_{k-1}: G_{k-1}, observational data: $\boldsymbol{X}$, a set of generated datasets and associated $(k-1)$-partial graphs of vicinal graphs: $\mathcal{V}_{k-1} = \{(X^v, G^v_{k-1}), \cdots\}$
Output: k-partial graph estimated by M_k: G_k

1 $M_k \leftarrow$ `Train()`; // train k-order model M_k
2 **foreach** $(x_i, x_j) \in E_{k-1}$ **do**
3 $\boldsymbol{x}_{ij} \leftarrow$ *GenerateFeature*$(x_i, x_j, \boldsymbol{X})$;
4 **if** $M_k(\boldsymbol{x}_{ij}) =$ *adjacency* **then**
5 add $x_i - x_j$ to G_k
6 **return** G_k;
7
8 **Function** `Train()`:
9 // prepare training data for M_k
10 **foreach** $(X^v, G^v_{k-1}) \in \mathcal{V}_{k-1}$ **do**
11 **foreach** $(x_i, x_j) \in E^v_{k-1}$ **do**
12 $\boldsymbol{x}_{ij} \leftarrow$ *GenerateFeature*(x_i, x_j, X^v);
13 **if** x_i, x_j *are adjacent* **then**
14 add $\boldsymbol{x}_{ij}$ to training data as a positive sample;
15 **else** add $\boldsymbol{x}_{ij}$ to training data as a negative sample;
16 train a model M_k with all training data;
17 **return** M_k;

(line 13). The feature and label are inserted into the training data (lines 14–15). Then, M_k is trained by fitting the training data (line 16), which is further used to infer adjacency on test data (lines 2–5).

Sec. 4.1 describes how we extract a set of informative features to form the outputs of *GenerateFeature* in Alg. 1. Sec. 4.2 introduces pruning techniques to eliminate unnecessary CI tests during feature generation. We also elaborate on some implementation-level optimizations in Appendix D due to space limitations.

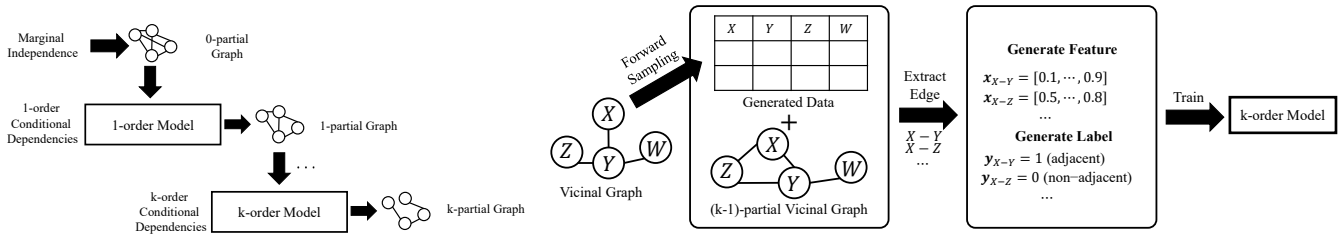
4.1 Feature Generation

Using only qualitative conditional independence suffices to recover skeletons from data. However, in practice, such qualitative conditional independence is often error-prone and sensitive to thresholds. Thus, we augment the qualitative conditional independence to quantitative conditional dependency derived from hypothesis tests. We also consider neighborhoods of target node pair to enable more accurate learning. Holistically, we propose features from two aspects: quantitative k-order conditional dependencies and local structural information. The former includes two classes of features (Sec. 4.1.1); and the latter contains three classes of features (Sec. 4.1.2), which together form the output of *GenerateFeature* in Alg. 1.

4.1.1 Quantitative Conditional Dependency. We view the k-order conditional dependency as a finer-grained version of the qualitative conditional independence; we form the following two features.

k-order Conditional Dependencies. We first conduct k-order CI tests between a pair of nodes and use the quantitative k-order conditional dependencies to form a numeric vector as features of this node pair. Note that the quantitative condition dependencies are derived from the p-values (see Appendix D).

Residual of Conditional Dependencies. We also consider how the conditional dependencies between a node pair x_i, x_j are changed, when a new variable y is added to the current condition set $\boldsymbol{z}$ (we term the change as residual of conditional dependencies of x_i, x_j).



(a) Cascade classifiers of ML4S. **(b) Training k-order model.**

Figure 2: Overview

Intuitively, when x_i and x_j are adjacent in the ground-truth BN, their dependencies would exhibit certain "resistance" regardless of whether y is in z. In that sense, the dependencies should not vanish in higher-orders. In contrast, when the two nodes are non-adjacent in the ground-truth BN, their dependencies will vanish if y together with z d-separates x_i and x_j. A relevant notion called "Monotone DAG-Faithfulness Assumption" [6] entails that the conditional dependencies of x_i and x_j is monotonic w.r.t. the number of unblocked paths between them in the ground-truth BN. With y adding to the condition set z, if we observe a notable drop in conditional dependencies (i.e., a large residual), x_i and x_j are likely to be non-adjacent as one path between them may be blocked by y. Operationally, we calculate the residual of conditional dependencies as follows:

$$\{x_i \sim x_j \mid z\} - \min_y \{x_i \sim x_j \mid z \cup \{y\}\} \tag{8}$$

where $|z| = k - 1$. Here, for each $(k-1)$-order conditional dependency that is computed on $x_i, x_j \mid z$, we find a new node y to minimize the k-order conditional dependency on $x_i, x_j \mid z$.

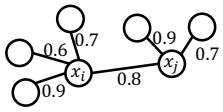


Figure 3: Example of model's prediction on different edges.

4.1.2 Local Structural Information. Local structural information depicts the sparsity around the checked pair of nodes through various lenses. Three features are accordingly formed as follows.

Superiority. Note that the preceding model provides a numerical prediction on each edge in the $(k-1)$-partial graph. Therefore, the superiority of the target edge in the local structure would be a strong indicator to its true adjacency. Consider Fig. 3, and let the neighbors of x_i, x_j on the $(k-1)$-partial graph be N_i, N_j, the superiority of the adjacency between x_i, x_j is defined by their neighbors, respectively. Particularly, on x_i, it is defined as

$$\frac{|\{x_m \mid x_m \in N_i, M_{k-1}(x_i, x_j) > M_{k-1}(x_i, x_m)\}|}{|N_{G_{k-1}}(x_i)| - 1} \tag{9}$$

which counts the number of edges of x_i's neighbors having a lower confidence compared with $x_i - x_j$. In Fig. 3, there are two edges with lower predictions on adjacency (i.e., 0.6 and 0.7), and therefore, the relative confidence on x_i is $2/3 = 0.67$. Likewise, the relative confidence on x_j is $1/2 = 0.5$. In addition, we also use the absolute superiority on x_i, x_j (0.8 in Fig. 3) as one feature.

Degrees of Target Nodes. Many skeleton learning algorithms make additional assumptions about the sparsity or minimality of the ground-truth BN [11, 34]. In practice, BNs are frequently sparse. Rather than making explicit assumptions, we use the degrees of x_i, x_j on the $(k-1)$-partial graph (i.e., number of neighbors) as part of features to train our model. The example in Fig. 3 generates this feature as 3, 3, denoting the degrees of x_i, x_j, respectively.

Density. We consider a local structure as dense if two nodes x_i, x_j have a large proportion of common neighbors. When higher-order CI tests are performed, these common neighbors may have the adjacency of x_i, x_j blocked. To quantify such density, we compute the overlapping ratio [10] as follows:

$$\frac{|N_i \cap N_j|}{\min(|N_i|, |N_j|)} \tag{10}$$

4.2 Pruning Strategy

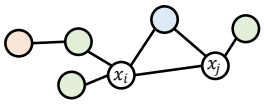


Figure 4: Example of an intermediate k-partial graph.

In addition to the effort in the learning process, we also explore pruning unnecessary CI tests in the following two aspects.

P1: Pruning CI Tests for General Cases. Inspired by PC [32], it should generally be more efficient to enumerate size-k combinations on the neighbors of x_i, x_j, rather than picking conditional set z from the entire graph. For instance, given an intermediate graph estimated by model M_{k-1} at the $(k-1)$ order, let N_i, N_j (blue and green nodes in Fig. 4) represent the set of neighbors of x_i, x_j. The k-order model requires only that all $z \subseteq N_i \cup N_j$ with $|z| = k$.

P2: Pruning CI Tests for 1-Order Models. Pearl's axioms on conditional independence [27] (Appendix C) allow us to infer high-order conditional independence from a set of known low-order conditional independence. In particular, we observe a pruning opportunity derived from the following lemma.

Lemma 4.1. *Let G_k be the k-partial graph to the BN G. If x_i, x_j are adjacent in G_k, for any node $z \notin N_i \cap N_j$, there exist a set of nodes $\boldsymbol{w}$ with $|\boldsymbol{w}| \leq k$ such that $x_i \not\perp\!\!\!\perp x_j \mid \{z\} \cup \boldsymbol{w}$. (see proof in Appendix B)*

Lemma. 4.1 entails a useful conditional dependence based on the local structure of the 0-partial graph. Formally,

Proposition 4.1. *Let G_0 be the 0-partial graph to the BN G. If x_i, x_j are adjacent in G_0, for any node $z \notin N_i \cap N_j$, $x_i \not\perp\!\!\!\perp x_j \mid \{z\}$.*

Proposition. 4.1 allows us to only conduct CI tests on the shared neighbors (the blue node in Fig. 4), as x_i, x_j are always conditionally dependent given the other neighbors (green nodes in Fig. 4).

In practice, we find Proposition. 4.1 highly useful, because 0-partial graphs are generally dense and conducting 1-order CI tests

even within the neighborhood (by P1) is costly. As shown in Sec. 6, this pruning strategy alleviates a notable proportion of CI tests.

5 VICINAL GRAPHS

Quality training data (i.e., graphs associated with generated datasets via forward sampling) is essential to model training. Therefore, generating a set of adequate graphs is the key task. Instead of generating graphs randomly in the DAG space (e.g., by Erdős-Rényi Model), we propose to generate vicinal graphs, where each vicinal graph is Markov compatible to a joint distribution sharing certain "similarities" with the distribution of observational data. In the following, we introduce how vicinal graphs are generated and parameterized such that we can use forward sampling to obtain generated data.

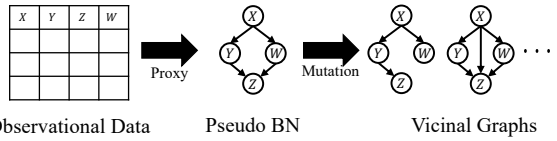


Figure 5: Generate vicinal graphs from observational data.

The workflow of generating vicinal graphs is outlined in Fig. 5. ML4S first employs a *proxy algorithm*, representing an existing causal discovery algorithm like PC, to learn a pseudo BN w.r.t. the observational data. Then, it forms the seed of generating vicinal graphs. Specifically, we mutate the pseudo BN with two schemes to obtain vicinal graphs: 1) randomly inserting edges, and 2) randomly deleting edges. After multiple rounds of mutations, the structure of vicinal graphs are generated. However, in this stage, the vicinal graphs are not parameterized (without CPTs (conditional probability tables)), on which forward sampling cannot be applied. In the following, we introduce how CPTs are prepared.

As a starting point, we may parameterize vicinal graphs with randomly generated CPTs. However, the underlying joint distribution represented by random CPTs may drastically differ to the true joint distribution of observational data, undermining the training process. As a desiderata, we would expect the vicinal graphs' distributions to retain a reasonable degree of similarity to the one of ground-truth BN. To accomplish this, we generate vicinal graphs' CPTs by inheriting pseudo BN's CPTs. First, we employ MLE (Maximum Likelihood Estimation) to estimate the CPTs of a pseudo BN. For each node in a vicinal graph, its CPT is directly inherited from the pseudo BN if its parents are identical to its counterpart node in the pseudo BN. When its parents are modified by edge insertions or deletions during mutations, its CPT would be adjusted accordingly. The rest of this section describes how we generate CPTs for such cases while preserving essential probabilistic properties.

Edge Deletion. Adjusting CPTs for edge deletion is straightforward using marginalization, as in Alg. 2 (lines 1–4). Consider the left vicinal graph shown in Fig. 5, which deletes the edge between Z and W in the pseudo BN. The CPT of Z on the pseudo BN (CPT_{old} in line 2) encodes the distribution of $P(Z \mid Y, W)$. In the new vicinal graph, Z only has one parent node, whose CPT shall encode $P(Z \mid Y)$. By the law of total probability, $P(Z \mid Y) = \sum_w P(Z \mid Y, W = w)$, which forms the new CPT (line 3).

Edge Insertion. Generating CPTs for edge insertion is complex. Consider the right vicinal graph in Fig. 5, where an extra edge between X and Z is inserted. On the pseudo BN, Z's CPT encodes $P(Z \mid Y, W)$. If we simply let $P(Z \mid Y, W, X) = P(Z \mid Y, W)$ on each $X = x$ to retain a similar distribution, X would be independent of Z given Y, W, implying that X should be non-adjacent to Z in accordance with the faithfulness assumption. To resolve the conflict, we employ the Dirichlet-multinomial distribution (Dirichlet distribution for brevity) to sample diverse CPTs while retaining the essences of observational data. The method is outlined in Alg. 2 (lines 5–13). Here, Dirichlet distribution can be seen as a family distribution of categorical-valued distributions. For each concrete conditional probability distribution $P(Z \mid Y = y, W = w)$ (line 8), we use MLE to estimate the parameters of its Dirichlet distribution $\boldsymbol{\alpha}_i$ (line 9). Then, for each possible value x of variable X, we sample $P(Z \mid Y = y, W = w, X = x)$ from $Dirichlet(\beta\boldsymbol{\alpha}_i)$ (line 11), where β is a hyper-parameter tuning "variance" of the Dirichlet distribution. After enumerating all possible combinations of conditional probability distribution $P(Z \mid Y = y, W = w)$ and x, the new CPT is generated. Furthermore, since $P(Z \mid Y = y, W = w, X)$ is sampled from the same Dirichlet distribution, the joint distribution will be similar but the conflict is resolved. We refer readers to [17] for MLE on Dirichlet distributions.

Algorithm 2: Generate CPTs for Vicinal Graphs

```
1  Function DeleteEdge(start_node, end_node):
2      CPT_old ← end_node's CPT;
3      CPT_new ← Marginalize(CPT_old, start_node);
4      update end_node's CPT with CPT_new;
5  Function InsertEdge(start_node, end_node):
6      CPT_old ← end_node's CPT;
7      CPT_new ← ∅;
8      foreach cpt_i ∈ CPT_old do
9          α_i ← MLE_Dirichlet(cpt_i);
10         foreach val ∈ start_node do
11             cpt_i^val ← Dirichlet(β α_i);
12             CPT_new ← CPT_new ∪ {cpt_i^val};
13     update end_node's CPT with CPT_new;
```

6 EVALUATION

In evaluation, we aim to answer the following research questions:

(1) *how good are the skeletons discovered by ML4S compared with other de facto algorithms?*
(2) *does ML4S provides a general and unique improvement to all algorithms?*
(3) *how do different techniques (e.g., vicinal graphs and pruning strategies) improve ML4S?*
(4) *how good is ML4S in terms of boosting downstream applications?*

In the following subsections, we explore each research question.

6.1 End-to-end Comparison

We collect 14 datasets from the bnlearn repository [29] of diverse scales. Each dataset corresponds with 10,000 observational data samples. We compare ML4S with a wide range of learning algorithms, including constraint-based algorithms (REAL [11], MMPC [33] and PC [32]), score-based algorithms (HC [33], GES [7], BLIP [28] and GOBNILP [9]), and gradient-based algorithms (NOTEARS [39], DAG-GNN [37] and CDRL [40]). ML4S uses BLIP to learn a pseudo

Table 2: Comparison of different skeleton learning algorithms on the bnlearn benchmarks. We highlight the best, second best and third best algorithms in each dataset. ML4S is among the top-3 on 12 out of 14 datasets.

Dataset #nodes/#edges	Metric	ML4S	Constraint-based			Score-based				Gradient-based		
			REAL	MMPC	PC	HC	GES	BLIP	GOBNILP	NOTEARS	DAG-GNN	CDRL
child 20/25	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.57	0.72	0.88
	Recall	1.00	1.00	1.00	1.00	0.89	1.00	1.00	0.52	0.84	0.52	0.60
	F1 Score	1.00	1.00	1.00	1.00	0.94	1.00	1.00	0.68	0.68	0.60	0.71
insurance 27/52	Precision	0.98	1.00	0.97	1.00	0.69	0.93	0.98	0.97	0.40	0.47	0.82
	Recall	0.77	0.71	0.69	0.79	0.81	0.75	0.81	0.56	0.73	0.35	0.27
	F1 Score	0.86	0.83	0.81	0.88	0.74	0.83	0.88	0.71	0.52	0.40	0.41
water 32/66	Precision	1.00	1.00	1.00	1.00	0.68	0.97	0.86	-	0.66	0.91	0.75
	Recall	0.45	0.42	0.41	0.52	0.55	0.48	0.48	-	0.47	0.32	0.27
	F1 Score	0.65	0.60	0.58	0.68	0.61	0.65	0.62	-	0.55	0.47	0.40
mildew 35/46	Precision	0.92	1.00	0.89	0.92	0.76	0.94	0.82	-	0.52	0.59	0.53
	Recall	0.72	0.59	0.52	0.76	0.67	0.70	0.61	-	0.52	0.59	0.53
	F1 Score	0.80	0.74	0.66	0.83	0.71	0.80	0.70	-	0.52	0.59	0.53
alarm 37/46	Precision	0.98	0.96	0.98	1.00	0.69	0.98	0.94	1.00	0.43	0.55	0.74
	Recall	0.96	0.93	0.93	0.96	0.96	0.96	0.96	0.59	0.89	0.61	0.57
	F1 Score	0.97	0.95	0.96	0.98	0.80	0.97	0.95	0.74	0.58	0.58	0.64
barley 48/84	Precision	0.83	0.90	0.84	1.00	0.66	0.91	0.88	0.96	0.25	0.41	0.64
	Recall	0.68	0.62	0.55	0.73	0.70	0.69	0.60	0.64	0.45	0.23	0.17
	F1 Score	0.75	0.73	0.66	0.84	0.68	0.78	0.71	0.77	0.32	0.30	0.26
hailfinder 56/66	Precision	0.22	0.15	0.08	0.16	0.14	0.16	0.20	0.14	0.10	0.07	0.06
	Recall	0.18	0.11	0.11	0.11	0.18	0.15	0.19	0.12	0.15	0.03	0.08
	F1 Score	0.20	0.12	0.09	0.13	0.16	0.16	0.19	0.13	0.12	0.04	0.07
hepar2 70/123	Precision	0.97	0.97	0.97	0.91	0.74	1.00	0.96	0.94	0.61	1.00	-
	Recall	0.69	0.60	0.60	0.57	0.80	0.62	0.69	0.61	0.53	0.12	-
	F1 Score	0.81	0.74	0.74	0.70	0.77	0.76	0.80	0.74	0.57	0.22	-
win95pts 76/112	Precision	0.91	0.89	0.94	0.96	0.32	0.86	0.78	-	0.44	0.81	-
	Recall	0.69	0.59	0.59	0.70	0.90	0.87	0.88	-	0.81	0.52	-
	F1 Score	0.78	0.71	0.73	0.81	0.48	0.86	0.82	-	0.57	0.63	-
pathfinder 109/195	Precision	0.85	0.98	0.81	-	-	-	0.40	-	0.17	0.71	-
	Recall	0.46	0.41	0.43	-	-	-	0.35	-	0.43	0.43	-
	F1 Score	0.59	0.58	0.56	-	-	-	0.37	-	0.24	0.54	-
munin1 186/273	Precision	0.62	0.86	0.77	-	-	-	0.67	-	0.19	0.55	-
	Recall	0.47	0.38	0.38	-	-	-	0.51	-	0.34	0.18	-
	F1 Score	0.54	0.51	0.51	-	-	-	0.58	-	0.24	0.27	-
andes 223/338	Precision	0.96	0.98	0.96	0.94	-	-	0.90	-	0.84	0.90	-
	Recall	0.86	0.79	0.78	0.82	-	-	0.90	-	0.40	0.25	-
	F1 Score	0.91	0.88	0.86	0.87	-	-	0.90	-	0.54	0.39	-
diabetes 413/602	Precision	0.76	0.71	0.80	-	-	-	0.73	-	0.05	0.19	-
	Recall	0.80	0.60	0.53	-	-	-	0.65	-	0.43	0.56	-
	F1 Score	0.78	0.65	0.64	-	-	-	0.69	-	0.08	0.29	-
pigs 441/592	Precision	0.93	1.00	1.00	-	-	-	0.93	-	0.60	0.90	-
	Recall	1.00	1.00	1.00	-	-	-	1.00	-	0.99	0.69	-
	F1 Score	0.97	1.00	1.00	-	-	-	0.96	-	0.75	0.79	-
Overall Rank	Mean	2.00	4.00	4.93	3.93	6.21	4.07	3.07	7.57	7.57	8.14	8.86
	Std	1.07	1.77	2.34	2.79	1.52	2.49	1.79	2.19	2.13	2.45	1.46

BN and then generates vicinal graphs using the pseudo BN; their ground-truth adjacencies form the labels of training data. ML4S is trained with 5,000 vicinal graphs by default (and smaller on large datasets). ML4S employs XGBoost [5] as classifiers with default parameters. Other implementation details are presented in Appendix D. We consider a case as failed (denoted as "-") if an algorithm takes more than eight hours or crashed. For algorithms allowing a time limit, we use the best-so-far results within eight hours.

We report the full comparison on Precision, Recall and F1 Score of different algorithms in Table 2. A succinct comparison has been given in Fig. 1. Overall, we interpret the results as highly promising. Holistically, ML4S is the best algorithms among all these de facto algorithms with an average rank of 2.00 ± 1.07. The second best algorithm, BLIP, has an overall lower and less stable performance of 3.07 ± 1.79. We also observe that different algorithms usually have distinct performance on different datasets. In particular, we observe that classical algorithms such as PC and GES manifest competitive

performance on small datasets (child, insurance, etc.). We presume it is primarily because we adopt the latest implementations of the algorithms with good engineering quality and useful tricks (e.g., PC stable strategy). Nevertheless, they generally fail to scale to larger datasets with more than 100 variables, which impedes their overall performance. In addition, we observe that gradient-based algorithms are usually sub-optimal in comparison to algorithms from other categories. We presume it is because they are originally designed for continuous data with stronger assumptions enforced.

6.2 Generality and Uniqueness

We further explore ML4S through the lens of improvement on the chosen proxy algorithms. Here, we focus on two orthogonal aspects: 1) Is ML4S a general improvement to different proxy algorithms? 2) Is ML4S a unique improvement that cannot be trivially replaced by existing approaches? For the first question, we employ the pseudo BNs learned by different algorithms to generate vicinal graphs on the pathfinder dataset, where different algorithms show a distinct performance (see Table 2). For the second question, we employ the "Backward" phase described in MMPC [33] as a standard post-process on skeletons discovered by the BLIP algorithm.

Table 3: F1 Score of different algorithms on pathfinder.

	REAL	MMPC	BLIP	NOTEARS	DAG-GNN
w/o ML4S	0.58	0.56	0.37	0.24	0.54
w/ ML4S	0.65	0.63	0.59	0.48	0.69
Improvement	+12.1%	+12.5%	+59.5%	+100.0%	+27.8%

Generality. ML4S improves existing proxy algorithms in general. To demonstrate this, we create vicinal graphs on the pathfinder dataset using several proxy algorithms and provide the (improved) F1 Score in Table 3. On the one hand, we see that ML4S enhances F1 Score significantly when compared to the original results. On the other hand, we find that the quality of the training data has a notable effect on performance. The greater agreement between the pseudo BN and the ground-truth BN (i.e., a higher accuracy in the 2nd row), the better ML4S performs to improve the overall accuracy (i.e., the accuracy in the 3rd row).

Table 4: F1 Score of BLIP (B), BLIP with Backward (BB) and ML4S (M). #1-14 denote the datasets evaluated in Table 2. We highlight the best for each comparison.

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14
B	1.0	.88	.62	.70	.95	.71	.19	.80	.82	.37	.58	.90	.69	.96
BB	1.0	.82	.57	.62	.96	.69	.19	.72	.74	.34	.53	.88	.59	1.0
M	1.0	.86	.65	.80	.97	.75	.20	.81	.78	.59	.54	.91	.78	.97

Uniqueness. We demonstrate how ML4S is distinct from existing post-processing techniques to skeleton learning. Table 4 summarizes F1 Score for BLIP, BLIP with Backward, and BLIP with ML4S on 14 datasets. While "Backward" is highly useful in MMPC [33], it generally fails to improve BLIP on skeleton learning. In contrast, ML4S delivers an unique improvement to existing algorithms.

6.3 Ablation Study and Feature Impact

We now study how techniques proposed in the paper improve ML4S. We explore the impact of training data, features and pruning.

Table 5: F1 Score on different training data.

	Vicinal	Random (SF)	Random (ER)
insurance	0.86	0.74 (-14%)	0.74 (-14%)
water	0.65	0.63 (-3%)	0.61 (-6%)
mildew	0.80	0.68 (-15%)	0.74 (-8%)
barley	0.75	0.62 (-16%)	0.63 (-16%)

Vicinal Graph vs. Random Graph. Table 5 assesses the performance of ML4S when trained on data of varying quality. In particular, we examine how ML4S would perform on purely random BNs. These random graphs are constructed using two classical random graph models, namely the Erdős-Rényi Model and the Scale-Free (SF) Model. Random graphs are on a par with the ground-truth BN in terms of scale. We observe that replacing vicinal graphs with random graphs notably degrades the performance of ML4S. We see it as reasonable, given that real-world BNs exhibit distinct characteristics that are hard to capture using random graphs.

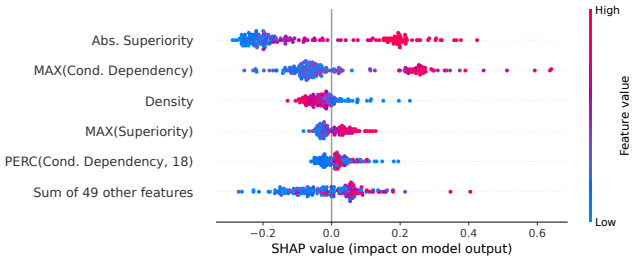


Figure 6: Feature impact computed using Shapley values (SHAP) for a 2-order model on the barley dataset. "PERC" denotes percentiles with 20 splits.

Feature Impact. We also explore the impact of different features (introduced in Sec. 4.1) on model predictions. We use Shapley values (SHAP [24]) to illustrate feature importance for a 2-order model on the barley dataset in Fig. 6, with 0.0 on the x-axis denoting "no impact." Features are ranked by their Shapley values, where features with higher importance have their positive (red dots) and negative (blue dots) impact farther away from 0.0. Overall, we find that all features described in Sec. 4.1 make reasonable contributions to model predictions. Among these features, we observe that the feature of superiority is the most important and positively correlated with the prediction of the 2-order model. We consider it as intuitive; the superiority of an edge in the $(k-1)$-model would most likely increase the confidence of k-model in their adjacency. We also observe that the feature of k-order conditional dependencies is a strong predictor of adjacencies, as it reflects conditional independence relations which are the basis of constraint-based methods. See Appendix E for a detailed feature impact analysis.

Table 6: Number and Proportion of Omitted CI Tests.

#1	#2	#3	#4	#5	#6	#7
714 (-53%)	1370 (-57%)	410 (-76%)	4620 (-44%)	1712 (67%)	7058 (-18%)	5162 (-70%)
#8	#9	#10	#11	#12	#13	#14
3130 (-85%)	1506 (-84%)	22242 (-73%)	28608 (-70%)	6194 (-89%)	326130 (-78%)	174122 (-56%)

Pruning Strategies. We give the number and proportion of CI tests that our pruning algorithms facilitate to omit in Table 6. We do not examine **P1** because its effectiveness is self-evident and it has been used in many existing constraint-based methods. We focus on **P2** here. Table 6 reports the results for 14 datasets. We interpret

the results as promising. On average, it eliminates 65.8% of the 1-order CI tests for **P2**, especially on large datasets. For instance, 78% 1-order CI tests are pruned by **P2** on the diabetes dataset.

6.4 Downstream Application

We illustrate the effectiveness of ML4S in terms of boosting downstream applications through a case study on NOTEARS [39]. In particular, we design a crafted NOTEARS [39], which takes skeletons as a prior. Following [39], we use structural Hamming distance (SHD) between the output DAG and the ground-truth DAG as the performance metrics (lower is better). The the detailed algorithm and full results are reported in Appendix F. Overall, ML4S substantially improves NOTEARS on *all datasets*. In particular, on the diabetes dataset, the original NOTEARS yields 5666 erroneous edges (most are false positives), while ML4S, by supplying the skeleton, substantially eliminates errors. We presume skeletons produced by ML4S can be used as general enhancements to a variety of gradient-based algorithms, including DAG-GNN [37] and CDRL [40]. For other algorithms (e.g., constraint-based), their combinations with skeletons are evidently explored in [33].

7 CONCLUSION AND FUTURE WORK

Causal skeleton learning uncovers the underlying undirected graph (skeleton) of causal BNs using observational data. We proposed the first supervision-based paradigm for causal skeleton learning. The proposed technique features an effective cascade procedure for skeleton learning, with training data derived from vicinal graphs. Evaluation over various datasets illustrates the promising capability of ML4S by notably outperforming the state of the arts.

This paper primarily focuses on analyzing discrete data. Having that said, both the supervision-based paradigm and ML4S are capable of handling a wide variety of data types. For instance, we envision that users of ML4S can use kernel-based CI test [38] to conduct statistical tests over continuous data, allowing ML4S to be applied smoothly. Also, we have presented a case study to illustrate that causal skeleton enhances the accuracy of downstream gradient-based causal discovery tasks in Sec. 6.4. We leave boosting other downstream tasks (e.g., optimal feature selection) with our proposed technique as future work.

ACKNOWLEDGMENTS

In this work, Pingchuan Ma is jointly supervised by Dr. Shuai Wang at HKUST and Rui Ding at MSRA. The authors would like to thank anonymous reviewers for their valuable comments and Yang Zhang for refactoring the code, substantially improving the engineering quality and valuable discussion.

REFERENCES

[1] [n.d.]. https://github.com/microsoft/reliableAI
[2] 2022. Sampling methods. https://ermongroup.github.io/cs228-notes/inference/sampling/.
[3] Gianluca Bontempi and Maxime Flauder. 2015. From dependency to causality: a machine learning approach. *JMLR* (2015).
[4] Robert Castelo and Alberto Roverato. 2006. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *JMLR* (2006).
[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *SIGKDD*.
[6] Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, and Weiru Liu. 2002. Learning Bayesian networks from data: An information-theory based approach. *Artificial intelligence* 137, 1-2 (2002), 43–90.
[7] David Maxwell Chickering. 2002. Learning equivalence classes of Bayesian-network structures. *JMLR* 2 (2002), 445–498.
[8] Diego Colombo, Marloes H Maathuis, et al. 2014. Order-independent constraint-based causal structure learning. *JMLR*. 15, 1 (2014), 3741–3782.
[9] James Cussens, Matti Järvisalo, Janne H Korhonen, and Mark Bartlett. 2017. Bayesian network structure learning with integer programming: Polytopes, facets and complexity. *JAIR* 58 (2017), 185–229.
[10] Haoyue Dai, Rui Ding, Yuanyuan Jiang, Shi Han, and Dongmei Zhang. 2021. ML4C: Seeing Causality Through Latent Vicinity. *arXiv:2110.00637* (2021).
[11] Rui Ding, Yanzhi Liu, Jingjing Tian, Zhouyu Fu, Shi Han, and Dongmei Zhang. 2020. Reliable and Efficient Anytime Skeleton Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 10101–10109.
[12] Paul Erdős, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5, 1 (1960), 17–60.
[13] José AR Fonollosa. 2019. Conditional distribution variability measures for causality detection. In *Cause Effect Pairs in Machine Learning*.
[14] Nir Friedman, Dan Geiger, and Moises Goldszmidt. 1997. Bayesian network classifiers. *Machine learning* (1997).
[15] Clark Glymour, Kun Zhang, and Peter Spirtes. 2019. Review of causal discovery methods based on graphical models. *Frontiers in genetics* 10 (2019), 524.
[16] David Heckerman, Dan Geiger, and David M Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *ML* (1995).
[17] Jonathan Huang. 2005. Maximum likelihood estimation of Dirichlet distribution parameters. *CMU Technique Report* (2005).
[18] Murat Kocaoglu, Alex Dimakis, and Sriram Vishwanath. 2017. Cost-optimal learning of causal graphs. In *ICML*. PMLR, 1875–1884.
[19] Steffen L Lauritzen. 1996. *Graphical models*. Clarendon Press.
[20] Honghao Li, Vincent Cabeli, Nadir Sella, and Hervé Isambert. 2019. Constraint-based causal structure learning with consistent separating sets. *NIPS* (2019).
[21] Hebi Li, Qi Xiao, and Jin Tian. 2020. Supervised Whole DAG Causal Discovery. *arXiv:2006.04697* (2020).
[22] David Lopez-Paz, Krikamol Muandet, and Benjamin Recht. 2015. The Randomized Causation Coefficient. *JMLR* 16 (2015), 2901–2907.
[23] David Lopez-Paz, Krikamol Muandet, Bernhard Schölkopf, and Iliya Tolstikhin. 2015. Towards a learning theory of cause-effect inference. In *ICML*.
[24] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *NeurIPS* 30 (2017).
[25] Dimitris Margaritis and Sebastian Thrun. 1999. Bayesian network induction via local neighborhoods. *NIPS* 12 (1999).
[26] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. 2017. Kernel Mean Embedding of Distributions: A Review and Beyond. *Machine Learning* 10, 1-2 (2017), 1–144.
[27] Judea Pearl, Dan Geiger, and Thomas Verma. 1989. Conditional independence and its representations. *Kybernetika* 25, 7 (1989), 33–44.
[28] Mauro Scanagatta, Cassio P de Campos, Giorgio Corani, and Marco Zaffalon. 2015. Learning Bayesian Networks with Thousands of Variables.. In *NIPS*. 1864–1872.
[29] Marco Scutari. 2010. Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software* (2010).
[30] Karthikeyan Shanmugam, Murat Kocaoglu, Alexandros G Dimakis, and Sriram Vishwanath. 2015. Learning causal graphs with small interventions. *NIPS* (2015).
[31] Carl-Johann Simon-Gabriel and Bernhard Schölkopf. 2018. Kernel distribution embeddings: Universal kernels, characteristic kernels and kernel metrics on distributions. *JMLR* 19, 1 (2018), 1708–1736.
[32] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. 2000. *Causation, prediction, and search*. MIT press.
[33] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning* 65, 1 (2006), 31–78.
[34] Jing Xiang and Seyoung Kim. 2013. A\Ast Lasso for Learning a Sparse Bayesian Network Structure for Continuous Variables. In *Advances in neural information processing systems*. Citeseer, 2418–2426.
[35] Sandeep Yaramakala and Dimitris Margaritis. 2005. Speculative Markov blanket discovery for optimal feature selection. In *ICDM*. IEEE, 4–pp.
[36] Kui Yu, Xianjie Guo, Lin Liu, Jiuyong Li, Hao Wang, Zhaolong Ling, and Xindong Wu. 2020. Causality-based feature selection: Methods and evaluations. *ACM Computing Surveys (CSUR)* (2020).
[37] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. 2019. Dag-gnn: Dag structure learning with graph neural networks. In *ICML*. PMLR, 7154–7163.
[38] Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. 2011. Kernel-based conditional independence test and application in causal discovery. In *UAI*.
[39] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. 2018. DAGs with NO TEARS: Continuous Optimization for Structure Learning. *NIPS* (2018).
[40] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. 2019. Causal Discovery with Reinforcement Learning. In *ICLR*.

A PROOF OF CLASSIFIER CORRECTNESS

We construct a classifier C_D at the remark in Sec. 1. We prove that C_D yields the correct skeleton w.r.t. observational data.

Proof. We discuss C_D with two types of errors (i.e., false positive and false negative) and prove the absence of these errors under the canonical assumption. Here, false positive (FP) denotes C_D predicts a non-adjacent node pair as adjacent and false negative (FN) denotes C_D predicts an adjacent node pair as non-adjacent.

False Positive. Suppose v_i, v_j are non-adjacent. Under the canonical assumption, there exists a set of nodes $\boldsymbol{z}$ such that $v_i \perp\!\!\!\perp v_j \mid \boldsymbol{z}$. In that sense, their conditional dependency $\{v_i \sim v_j \mid \boldsymbol{z}\} = 0$ and $\boldsymbol{x}_{ij} = 0$. According to Eqn. 2, C_D will always predicts them as non-adjacent.

False Negative. Suppose v_i, v_j are adjacent. Under the canonical assumption, for any $\boldsymbol{z} \in V \setminus \{v_i, v_j\}$, $v_i \not\perp\!\!\!\perp v_j \mid \boldsymbol{z}$, which implies the conditional dependencies for any $\boldsymbol{z}$ is higher than zero. According to Eqn. 2, C_D will always predicts them as adjacent.

In sum, when the canonical assumption holds, C_D would not yield any FP or FN. Thus, C_D recovers the correct skeleton. □

B PROOF OF LEMMA. 4.1

Proof. We prove Lemma. 4.1 by contradictions. Suppose there exists a node $z \notin N_i \cap N_j$ such that $\forall |\boldsymbol{w}| = k, x_i \perp\!\!\!\perp x_j \mid \{z\} \cup \boldsymbol{w}$. Without loss of generality, let $z \notin N_i$. According to the adjacency on the k-partial graph, we have $\exists |\boldsymbol{w}| \leq k, x_i \perp\!\!\!\perp z \mid \boldsymbol{w}$ and $x_i \not\perp\!\!\!\perp x_j \mid \boldsymbol{w}$. According to Pearl's axioms on conditional independence [27] (see Appendix C), $x_i \perp\!\!\!\perp z \mid \boldsymbol{w}$ and $x_i \perp\!\!\!\perp x_j \mid \{z\} \cup \boldsymbol{w}$ imply that $x_i \perp\!\!\!\perp x_j \cup \{z\} \mid \boldsymbol{w}$ by Contraction rule. Then, by Decomposition rule, $x_i \perp\!\!\!\perp x_j \cup \{z\} \mid \boldsymbol{w}$ implies that $x_i \perp\!\!\!\perp x_j \mid \boldsymbol{w}$ and $x_j \perp\!\!\!\perp \{z\} \mid \boldsymbol{w}$, which contracts the fact that $x_i \not\perp\!\!\!\perp x_j \mid \boldsymbol{w}$. □

C PEARL'S AXIOMS ON CONDITIONAL INDEPENDENCE

We introduce Pearl's axioms [27] used in Lemma. 4.1 as follows. The uppercase symbols denote a set of variables.

Decomposition.

$(X \perp\!\!\!\perp Y \cup W \mid Z) \implies (X \perp\!\!\!\perp Y \mid Z) \wedge (X \perp\!\!\!\perp W \mid Z)$

Contraction.

$(X \perp\!\!\!\perp Y \mid Z) \wedge (X \perp\!\!\!\perp W \mid Z \cup Y) \implies (X \perp\!\!\!\perp Y \cup W \mid Z)$

D IMPLEMENTATION DETAILS

We implement ML4S with about 5k lines of Python code. Our codebase is available at [1]. We use XGBoost [5] to form the basic models in each order with default parameters. We use at most four orders of conditional dependencies. If the adjacency cannot be rejected by 4-order model, we confirm it as an edge in the skeleton. We employ the following technique to generate conditional dependencies via a non-linear transformation to p-values and also confirm adjacent edges in an early stage.

Generate Conditional Dependencies. Following [10], we employ the non-linear transformation on p-values to derive conditional dependencies such that they can be smoothly processed by machine learning models. Formally,

$$g(z) = 1 - \frac{2}{\pi} \int_0^z e^{-x^2} dx \tag{11}$$

and we use the quantity $z = g^{-1}(p)$ as a re-scaled p-value, where g^{-1} is the inverse of the function defined in Eqn. 11. We call z as *conditional dependency*; and lower p-values correspond to higher z.

Handling Features of Flexible Size. Most machine learning models would anticipate receiving a fixed-size feature. However, k-order conditional dependencies and corresponding residuals may be of arbitrary size, as we have no control over the number of CI tests performed of this order. There are many well-established techniques to handle such features, such as kernel embedding [26, 31]. In ML4S, we employ a simple yet highly effective strategy to handle these cases. Suppose we have a collection of k-order conditional dependencies. We first estimate a distribution by treating these scalars as samples from an unknown linear distribution and then extracting percentiles of this distribution (e.g., 10%, 20%-percentile) to form a fixed-length vector. In addition, we compute several common statistics (e.g., max, min, mean and std). We then concatenate all these extracted features to form a vector for the model.

Early Edge Confirmation. We also spot an opportunity to confirm an adjacent edge in an early stage, by which we do not need to conduct high-order CI tests that are costly and error-prone. The optimization opportunity relies on the following lemma.

Lemma D.1. *Let G_k be a k-partial graph of ground-truth BN G, if there exists a pair of adjacent nodes x, y in G_k where x has at most $k+1-s$ neighbors in G_k and y has at most $k+1$ neighbors in G_k, x is adjacent to y in G, where s is the number of its spouses.*

Proof. We prove the above lemma by contradictions. Suppose y is adjacent to x in G_k but non-adjacent to x in G. Let MB_x be the Markov blanket of x. Let $\widehat{N_x}, N_x$ be the neighbors of x in G_k and G, respectively. We prove the lemma in three cases.

Case 1: y is not a spouse of x in G. According to Proposition. 3.1, $|MB_x| \leq (k+1-s) - 1 + s = k$. And MB_x d-separates all nodes in $V \setminus MB_x$ to x, which implies that $x \perp\!\!\!\perp y \mid MB_x$. Given that $|MB_x| \leq k$, x, y are non-adjacent in G_k, which contradicts.

Case 2: y is a non-descendant spouse to x in G. In this case, according to Markov property, $Pa_G(x)$ d-separates x with all non-descendant nodes. According to Proposition. 3.1, $|Pa_G(x)| < |\widehat{N_x}| \leq k+1-s$. Thus, $|Pa_G(x)| \leq k$, which further implies that x, y are non-adjacent in G_k. It contradicts.

Case 3: y is a descendant spouse to x in G. In this case, x is a non-descendant spouse to y. According to Markov property, $Pa_G(y)$ d-separates y with all non-descendant nodes. According to Proposition. 3.1, $|Pa_G(y)| < |\widehat{N_y}| \leq k+1$. Thus, $|Pa_G(y)| \leq k$, which further implies that x, y are non-adjacent in G_k. It contradicts. □

In theory, the size of spouse is unknown during skeleton learning, but is bounded by the maximal indegree b and maximal outdegree a of the BN, where $s \leq a(b-1)$. In the tree-based BN, e.g. TAN [14], the lemma would be largely eliminates computational costs, where $s = 0$. In practice, we employ $s = 0$ as a hyper-parameter that trades accuracy for efficiency. The empirical false rate of this optimization on normal datasets is 0.03 when $k = 2$ and 0.02 when $k = 3$, which we deem as negligible.

Algorithm 3: Boost Gradient-based Algorithms with ML4S

Input: Observational Data: $\boldsymbol{X}$
Output: Adjacency Matrix: W

1 generate skeleton S with ML4S on $\boldsymbol{X}$;
2 generate skeleton-induced adjacency matrix W^S by S;
3 initialize adjacency matrix $W = [0]_{n\times n}$;
4 **while** *not terminated* **do**
5 compute gradient ∇F;
6 $\nabla F^* \leftarrow \nabla F \odot W^S$;
7 update W with ∇F^*;
8 **end**
9 **return** W;

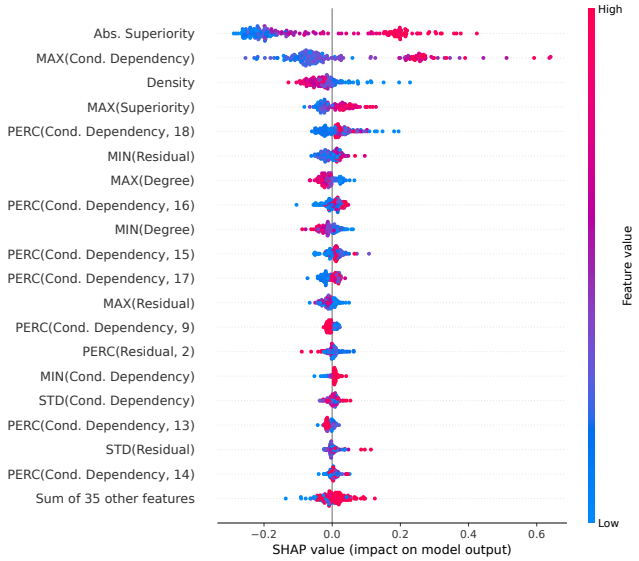


Figure 7: Detailed feature impact computed using Shapley values (SHAP) for a 2-order model on the insurance dataset.

E FEATURE ANALYSIS

We report the detailed feature impact in Fig. 7, which validates our statements Sec. 4.1. First, both superiority and maximal conditional dependencies show strong correlations to adjacencies and notably influence model's predictions. Density also manifests a clear correlations on the adjacencies, which is consistent with our hypothesis. If the two nodes shares a large proportion of common neighbors (i.e., high density; red in Fig. 7), they are very likely to be non-adjacent in the ground-truth skeleton (left to 0.0). Likewise, the maximal degree of focused nodes also negatively affects the model deciding adjacencies. Conversely, the impacts of residuals are subtle in terms of its maximum and minimum. Minimum residuals are positively correlated with adjacencies while maximum residuals are negative. We interpret it as reasonable. Since non-adjacent nodes would manifest low conditional dependencies, their minimal residuals would be small under a small baseline. However, their maximal residuals would be useful to estimate the "vanishment" in the monotonic assumption. The impacts of percentiles on conditional dependencies and residuals also align to our hypothesis but relatively minor compared to other statistics (e.g., min and max).

F BOOSTING NOTEARS WITH ML4S

Skeletons can be used to support a variety of downstream applications. It has shown that many constraint-based and score-based causal discovery algorithms decouple the phases of skeleton learning and orientation [32, 33]. Thus, the output of ML4S can be smoothly employed to enhance these algorithms by replacing their own skeleton learning module. Here, we demonstrate a case of boosting NOTEARS, which is a representative gradient-based algorithm, with skeletons generated by ML4S.

To the best of our knowledge, this is the first attempt for enhancing gradient-based algorithms with known skeletons, shedding a light on versatile usages of ML4S and other skeleton learning algorithms. We outline the general workflow for boosting gradient-based algorithms in Alg. 3, where the gradient is hooked. Alg. 3 first leverages ML4S to identify the skeleton (line 1) and then generate the skeleton-induced adjacency matrix W^S (line 2). When x_i, x_j are adjacent on S, $W^S_{i,j} = 1$. Otherwise, $W^S_{j,i} = 0$. Then, the adjacency matrix of DAG is initialized as a zero matrix. In the optimization phase of gradient-based algorithms (line 4–8), the original gradient is modified by W^S with Hadamard product (line 6). In that sense, the gradient on non-adjacent edges becomes zero such that the adjacency matrix is only updated on a limited edges (line 7). Therefore, the gradient-based algorithms is only responsible for orienting edges; skeleton learning is offloaded to ML4S which is generally more accurate, as already shown in Table 2.

We report the results of Alg. 3 in Table 7 and observe that Alg. 3 improves NOTEARS on all datasets.

Table 7: SHD of NOTEARS and NOTEARS with ML4S.

Dataset	NOTEARS	NOTEARS w/ ML4S
child	31	10 (-68%)
insurance	91	36 (-60%)
water	68	54 (-21%)
mildew	57	35 (-39%)
alarm	69	16 (-77%)
barley	184	71 (-61%)
hailfinder	153	96 (-37%)
hepar2	128	84 (-34%)
win95pts	162	55 (-66%)
pathfinder	576	186 (-68%)
munin1	612	227 (-63%)
andes	305	244 (-20%)
diabetes	5666	553 (-90%)
pigs	612	123 (-80%)