Learning Sparse Fixed-Structure Gaussian Bayesian Networks

Arnab Bhattacharyya
National University of Singapore
arnabb@nus.edu.sg

Davin Choo
National University of Singapore
davin@u.nus.edu

Rishikesh Gajjala
Indian Institute of Science, Bangalore
rishikeshg@iisc.ac.in

Sutanu Gayen
National University of Singapore
sutanugayen@gmail.com

Yuhao Wang
National University of Singapore
yohanna.wang0924@gmail.com

Abstract

Gaussian Bayesian networks are widely used to model causal interactions among continuous variables. In this work, we study the problem of learning a fixed-structure Gaussian Bayesian network up to a bounded error in total variation distance. We analyze the commonly used node-wise least squares regression `LeastSquares` and prove that it has the near-optimal sample complexity. We also study a couple of new algorithms for the problem:

• `BatchAvgLeastSquares` takes the *average* of several batches of least squares solutions at each node, so that one can interpolate between the batch size and the number of batches. We show that `BatchAvgLeastSquares` has near-optimal sample complexity.

• `CauchyEst` takes the *median* of solutions to several batches of linear systems at each node. We show that the algorithm specialized to polytrees, `CauchyEstTree`, has near-optimal sample complexity.

Experimentally, `LeastSquares` performs best for uncontaminated data generated from the given DAG. However, with data contamination or DAG misspecification, `CauchyEst`, `CauchyEstTree` and `BatchAvgLeastSquares` perform significantly better.

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

1 INTRODUCTION

Linear structural equation models (SEMs) are widely used to model interactions among multiple variables, and they have found applications in diverse areas, including economics, genetics, sociology, psychology and education; see [Mue99, Mul09] for pointers to the extensive literature in this area.

Gaussian Bayesian networks are a common form of SEMs where (i) there are no hidden/unobserved variables, (ii) each variable is a linear combination of its parents and a noise term, and (iii) all noise terms are independent Gaussians. The *structure* of a Bayesian network refers to the directed acyclic graph (DAG) prescribing parent nodes for each node in the graph.

This work studies the fundamental task of learning a Gaussian Bayesian network given its structure which has been subject to extensive prior work. The usual formulation of this problem is in terms of *parameter estimation*, where one wants a consistent estimator that *exactly* recovers the parameters of the Bayesian network in the limit, as the number of samples approaches infinity. In contrast, we consider the problem from the viewpoint of *distribution learning* [KMR+94]. Analogous to the PAC model for learning Boolean functions [Val84], the goal here is to learn a distribution $\widehat{\mathcal{P}}$ that is close to the ground-truth distribution $\mathcal{P}$ with high probability using an efficient algorithm. In this setting, pointwise convergence of the parameters is no longer a requirement; the aim is rather to approximately learn the induced distribution. Indeed, this relaxed objective may be achievable when the former may not be (e.g., for ill-conditioned systems) and can be the more relevant requirement for downstream inference tasks. Diakonikolas [Dia16] surveys the current state-of-the-art in distribution learning from an algorithmic perspective.

Consider a Gaussian Bayesian network $\mathcal{P}$ with n variables with parameters in the form of coefficients[1] $a_{i \leftarrow j}$'s and variances σ_i's. For each $i \in [n]$, the *linear structural equation* for variable X_i, with parent indices $\pi_i \subseteq [n]$, is $X_i = \eta_i + \sum_{j \in \pi_i} a_{i \leftarrow j} X_j,\ \eta_i \sim N(0, \sigma_i^2)$ for some unknown parameters $a_{i \leftarrow j}$'s and σ_i's. If a variable X_i has no parents, then $X_i \sim N(0, \sigma_i^2)$ is simply an independent Gaussian. Our goal is to efficiently learn a distribution $\mathcal{Q}$ such that $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$ while minimizing the number of samples we draw from $\mathcal{P}$ as a function of n, ε and relevant structure parameters. Here, $\mathrm{d_{TV}}$ denotes the total variation or statistical distance between two distributions: $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) = \frac{1}{2} \int_{\mathbb{R}^n} |\mathcal{P}(x) - \mathcal{Q}(x)|\, dx$.

One way to approach the problem is to simply view $\mathcal{P}$ as an n-dimensional multivariate Gaussian. In this case, it is known that the Gaussian $\mathcal{Q} \sim N(0, \widehat{\Sigma})$ defined by the empirical covariance matrix $\widehat{\Sigma}$, computed with $\mathcal{O}(n^2/\varepsilon^2)$ samples from $\mathcal{P}$, is ε-close in TV distance to $\mathcal{P}$ with constant probability [ABDH+20, Appendix C]. This sample complexity is also necessary for learning general n-dimensional Gaussians and hence general Gaussian Bayesian networks on n variables.

We focus therefore on the setting where the structure of the network is *sparse*.

Assumption 1.1. *Each variable in the Bayesian network has at most d parents. i.e. $|\pi_i| \leq d, \forall i \in [n]$.*

Sparsity is a common and very useful assumption for statistical learning problems; see the book [HTW19] for an overview of the role of sparsity in regression. More specifically, in our context, the assumption of bounded in-degree is popular (e.g., see [Das97, BCD20]) and also very natural, as it reflects the belief that in the correct model, each linear structural equation involves a small number of variables[2].

1.1 Our contributions

1. Analysis of MLE `LeastSquares` and a distributed-friendly generalization `BatchAvgLeastSquares`

The maximum likelihood estimator for parameter estimation of Gaussian Bayesian networks is to perform node-wise least squares regression. However, there does not exist an explicit sample complexity bound for this estimator. We show that the sample complexity for learning $\mathcal{P}$ to within TV distance ε using `LeastSquares` requires only $\widetilde{\mathcal{O}}(nd/\varepsilon^2)$ samples, which is nearly optimal.

We also give a generalization which solves multiple batches of least squares problems (on smaller systems of equations), and then returns their mean. We call this algorithm `BatchAvgLeastSquares`. As each batch is independent from the others, they can be solved separately before their solutions are combined. Our analysis will later show that we can essentially interpolate between "batch size" and "number of batches" while maintaining a similar total sample complexity – `LeastSquares` is the special case of a single batch. Notably, we do not require any additional assumptions about the coefficients or variance terms in the analyses of `LeastSquares` and `BatchAvgLeastSquares`.

2. New algorithms based on Cauchy random variables: `CauchyEst` and `CauchyEstTree`

We develop a new algorithm `CauchyEst`. At each node, `CauchyEst` solves several small linear systems of equations and takes the component-wise median of the obtained solution to obtain an estimate of the coefficients for the corresponding structural equation. In the special case of bounded-degree *polytree* structures, where the underlying undirected Bayesian network is acyclic, we specialize the algorithm to `CauchyEstTree` for which we give theoretical guarantees. Polytrees are of particular interest because inference on polytree-structured Bayesian networks can be performed efficiently [KP83, Pea86]. On polytrees, we show that the sample complexity of `CauchyEstTree` is also $\widetilde{\mathcal{O}}(nd/\varepsilon^2)$. Somewhat surprisingly, our analysis (as the name of the algorithm suggests) involves Cauchy random variables which usually do not arise in the context of regression.

3. Hardness results

We show that our sample complexity is nearly optimal in terms of the dependence on the parameters n, d, and ε. In particular, we show that learning the coefficients and noises of a linear structural equation model (on n variables, with in-degree at most d) up to an ε-error in $\mathrm{d_{TV}}$-distance with probability 2/3 requires $\Omega(nd\varepsilon^{-2})$ samples in general. We use a packing argument based on Fano's inequality to achieve this result.

4. Experiments on synthetic and real-world networks

We experimentally investigate how the KL distance[3] between the true and learned distributions changes

[1] We write $i \leftarrow j$ to emphasize that X_j is the parent of X_i in the Bayesian network.

[2] More generally, one would want to assume a bound on the "complexity" of each equation. For linear equations, their complexity are proportional to the in-degree.

[3] Since it's hard to compute TV distance, we report KL

with the number of samples when using the algorithms studied here, as well as other well-known methods for undirected Gaussian graphical model regression. We find that the LeastSquares estimator performs the best among all algorithms on uncontaminated datasets. However, CauchyEst and CauchyEstTree outperform LeastSquares and BatchAvgLeastSquares by a large margin when a fraction of the samples are contaminated. In non-realizable/agnostic learning case, BatchAvgLeastSquares, CauchyEst and CauchyEstTree outperform other algorithms.

1.2 Outline of paper

In Section 2, we relate KL divergence with TV distance and explain how to decompose the KL divergence into n terms so that it suffices for us to estimate the parameters for each variable independently. We also give an overview of our two-phased recovery approach and explain how to use recovered coefficients to estimate the variances via VarianceRecovery. For estimating coefficients, Section 3 covers the estimators based on linear least squares (LeastSquares and BatchAvgLeastSquares) while Section 4 presents our new Cauchy-based algorithms (CauchyEst and CauchyEstTree). To complement our algorithmic results, we provide hardness results in Section 5 and some experimental evaluation in Section 6. Proofs and further experiments are deferred to the appendix.

1.3 Further related work

Bayesian networks were formally introduced by Pearl [Pea88] in 1988 to model uncertainty in AI systems. For the continuous case, Pearl considered the case when each node is a linear function of its parents added with an independent Gaussian noise [Pea88, Chapter 7]. The parameter learning problem – recovering the distribution of nodes conditioned on its parents from data – is well-studied in practice, and maximum likelihood estimators are known for various simple settings such as when the conditional distribution is Gaussian or the variables are discrete-valued.

The focus of our paper is to give formal guarantees for the parameter learning in the PAC framework introduced by Valiant [Val84] in 1984. Subsequently, Haussler [Hau18] generalized this framework for studying parameter and density estimation problems of continuous distributions. Dasgupta [Das97] first looked at the problem of parameter learning for fixed structure Bayesian networks and gave finite sample complexity bounds for these problems based on the VC-dimensions of the hypothesis classes. In particular, he gave an algorithm for learning the parameters of a Bayes net on n binary variables of bounded in-degree in $\mathrm{d_{KL}}$ distance using a quadratic in n samples. Subsequently, tight (linear) sample complexity upper and lower bounds were shown for this problem [BGMV20, BGPV20, CDKS20]. To the best of our knowledge, a finite PAC-style bound for fixed-structure Gaussian Bayesian networks was not known previously.

distance instead.

The question of *structure learning* for Gaussian Bayesian networks has been extensively studied. A number of works [PB14, GH17, CDW19, PK20, Par20, GDA20] have proposed increasingly general conditions for ensuring identifiability of the network structure from observations. Structure learning algorithms that work for high-dimensional Gaussian Bayesian networks have also been proposed by others (e.g., see [AZ15, AGZ19, GZ20]). In this work, our main focus is *parameter learning on a given graph structure.*

2 PRELIMINARIES

In this section, we discuss why we upper bound the total variational distance using KL divergence and give a decomposition of the KL divergence into n terms, one associated with each variable in the Bayesian network. This decomposition motivates why our algorithms and analysis focus on recovering parameters for a single variable. We also present our general two-phased recovery approach and explain how to estimate variances using recovered coefficients in Section 2.4. Proofs and derivation details are deferred to the appendix.

2.1 Notation

A Bayesian network (Bayes net in short) $\mathcal{P}$ is a joint distribution $\mathcal{P}$ over n variables $X_1, \ldots, X_n$ defined by the underlying directed acyclic graph (DAG) G. The DAG $G = (V, E)$ encodes the dependence between the variables where $V = \{X_1, \ldots, X_n\}$ and $(X_j, X_i) \in E$ if and only if X_j is a parent of X_i. For any variable X_i, we use the set $\pi_i \subseteq [n]$ to represent the indices of X_i's parents. Under this notation, each variable X_i of $\mathcal{P}$ is independent of X_i's non-descendants conditioned on π_i. Therefore, using Bayes rule in the topological order of G, we have $\mathcal{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \Pr_{\mathcal{P}}(X_i \mid \pi_i)$

Without loss of generality, by renaming variables, we may assume that each variable X_i only has ancestors with indices smaller than i. We also define $p_i = |\pi_i|$ as the number of parents of X_i and $d_{avg} = \frac{1}{n} \sum_{i=1}^{n} p_i$ to be average in-degree. Furthermore, a DAG G is a *polytree* if the undirected version of G is a acyclic.

We study the *realizable* setting where our unknown probability distribution $\mathcal{P}$ is Markov with respect to the given Bayesian network. We denote the true (hidden) parameters associated with $\mathcal{P}$ by $\alpha^* = (\alpha_1^*, \ldots, \alpha_n^*)$.

Our algorithms recover parameter estimates $\widehat{\alpha} = (\widehat{\alpha}_1, \dots, \widehat{\alpha}_n)$ such that the induced probability distribution $\mathcal{Q}$ is close in total variational distance to $\mathcal{P}$. For each $i \in [n]$, $\alpha_i^* = (A_i, \sigma_i)$ is the set of ground truth parameters associated with variable X_i, A_i is the coefficients associated to π_i, σ_i^2 is the variance of η_i, $\widehat{\alpha}_i^* = (\widehat{A}_i, \widehat{\sigma}_i)$ is our estimate of α_i^*.

In the course of the paper, we will often focus on the perspective a single variable of interest. This allows us to drop a subscript for a cleaner discussion. Let us denote such a variable of interest by $Y \in V$ and use the index $y \in [n]$. Without loss of generality, by renaming variables, we may further assume that the parents of Y are $X_1, \dots, X_p$. By Assumption 1.1, we know that $p \le d$. We can write $Y = \eta_y + \sum_{i=1}^{p} a_{y \leftarrow i} X_i$ where $\eta_y \sim N(0, \sigma_y^2)$. We use matrix $M \in \mathbb{R}^{p \times p}$ to denote the covariance matrix defined by the parents of Y, where $M_{i,j} = \mathbb{E}[X_i X_j]$ and $M = LL^\top$ is the Cholesky decomposition of M. Under this notation, we see the vector $(X_1, \dots, X_p) \sim N(0, M)$ is distributed as a multivariate Gaussian. Our goal is then to produce estimates $\widehat{a}_{y \leftarrow i}$ for each $a_{y \leftarrow i}$. For notational convenience, we can group the coefficients into $A = [a_{y \leftarrow 1}, \dots, a_{y \leftarrow p}]^\top$ and $\widehat{A} = [\widehat{a}_{y \leftarrow 1}, \dots, \widehat{a}_{y \leftarrow p}]^\top$. The vector $\Delta = (\widehat{A} - A)^\top$ captures the entry-wise gap between our estimates and the ground truth.

We write $[n]$ to mean $\{1, 2, \dots, n\}$ and $|S|$ to mean the size of a set S. For a matrix M, $M_{i,j}$ denotes its (i,j)-th entry. We use $\|\cdot\|$ to both mean the operator/spectral norm for matrices and L_2-norm for vectors, which should be clear from context. We hide absolute constant multiplicative factors and multiplicative factors poly-logarithmic in the argument using standard notations: $\mathcal{O}(\cdot)$, $\Omega(\cdot)$, and $\widetilde{\mathcal{O}}(\cdot)$.

2.2 Decomposing the KL divergence

For a set of parameters $\alpha = (\alpha_1, \dots, \alpha_n)$, denote α_i as the subset of parameters that are relevant to the variable X_i. Following the approach of [Das97], we decompose $d_{\mathrm{KL}}(\mathcal{P}, \mathcal{Q})$ into n terms that can be computed by analyzing the quality of recovered parameters for each variable X_i.

For notational convenience, we write x to mean $(x_1, \dots, x_n)$, $\pi_i(x)$ to mean the values given to parents of variable X_i by x, and $\mathcal{P}(x)$ to mean $\mathcal{P}(X_1 = x_1, \dots, X_n = x_n)$. Let us define $d_{\mathrm{CP}}(\alpha_i^*, \widehat{\alpha}_i) = \int_{x_i, \pi_i(x)} \mathcal{P}(x_i, \pi_i(x)) \log\left(\frac{\mathcal{P}(x_i | \pi_i(x))}{\mathcal{Q}(x_i | \pi_i(x))}\right) dx_i \, d\pi_i(x)$ where each $\widehat{\alpha}_i$ and α_i^* represent the parameters that relevant to variable X_i from $\widehat{\alpha}$ and α^* respectively. By the Bayesian network decomposition of joint probabilities and marginalization, one can show that $d_{\mathrm{KL}}(\mathcal{P}, \mathcal{Q}) = \sum_{i=1}^{n} d_{\mathrm{CP}}(\alpha_i^*, \widehat{\alpha}_i)$.

2.3 Bounding d_{CP} for an arbitrary variable

We now analyze $d_{\mathrm{CP}}(\alpha_i^*, \widehat{\alpha}_i)$ with respect to the our estimates $\widehat{\alpha}_i = (\widehat{A}_i, \widehat{\sigma}_i)$ and the hidden true parameters $\alpha_i^* = (A_i, \sigma_i)$ for any $i \in [n]$.

With respect to variable X_i, one can derive $d_{\mathrm{CP}}(\alpha_i^*, \widehat{\alpha}_i) = \ln\left(\frac{\widehat{\sigma}_i}{\sigma_i}\right) + \frac{\sigma_i^2 - \widehat{\sigma}_i^2}{2\widehat{\sigma}_i^2} + \frac{\Delta_i^\top M_i \Delta_i}{2\widehat{\sigma}_i^2}$. Thus,

$$d_{\mathrm{KL}}(\mathcal{P}, \mathcal{Q}) = \sum_{i=1}^{n} d_{\mathrm{CP}}(\alpha_i^*, \widehat{\alpha}_i) = \sum_{i=1}^{n} \ln\left(\frac{\widehat{\sigma}_i}{\sigma_i}\right) + \frac{\sigma_i^2 - \widehat{\sigma}_i^2}{2\widehat{\sigma}_i^2} + \frac{\Delta_i^\top M_i \Delta_i}{2\widehat{\sigma}_i^2} \quad (1)$$

where M_i is the covariance matrix associated with variable X_i, $\alpha_i^* = (A_i, \sigma_i)$ is the coefficients and variance associated with variable X_i, $\alpha_i = (\widehat{A}_i, \widehat{\sigma}_i)$ are the estimates for α_i^*, and $\Delta_i = \widehat{A}_i - A_i$.

Proposition 2.1 (Implication of KL decomposition)**.** *Let $\varepsilon \le 0.17$ be a constant. For each $i \in [n]$, define $\gamma_i = \frac{\varepsilon \cdot p_i}{n \cdot d_{avg}}$ and suppose $\widehat{\alpha}_i$ has the following properties:*

$$\left|\Delta_i^\top M_i \Delta_i\right| \le \sigma_i^2 \cdot \gamma_i \quad \text{(Condition 1)}$$

$$(1 - \sqrt{\gamma_i}) \cdot \sigma_i^2 \le \widehat{\sigma}_i^2 \le (1 + \sqrt{\gamma_i}) \cdot \sigma_i^2 \quad \text{(Condition 2)}$$

Then, $d_{\mathrm{CP}}(\alpha_i^*, \widehat{\alpha}_i) \le 3\gamma_i = 3 \cdot \frac{\varepsilon \cdot p_i}{n \cdot d_{avg}}$ for all $i \in [n]$. Thus, $d_{\mathrm{KL}}(\mathcal{P}, \mathcal{Q}) = \sum_{i=1}^{n} d_{\mathrm{CP}}(\alpha_i^*, \widehat{\alpha}_i) \le 3\varepsilon$.[4]

2.4 Two-phased recovery approach

Algorithm 1 states our two-phased recovery approach. We estimate the coefficients of the Bayesian network in the first phase and use them to recover the variances in the second phase.

Algorithm 1 Two-phased recovery algorithm

```
1: Input: DAG $G$ and parameters $m_1$ and $m_2$
2: Draw $m_1 + m_2$ independent samples from $G$
3: Run a coefficient recovery algorithm using first $m_1$
   samples to obtain coefficients $\widehat{A}_1, \dots, \widehat{A}_n$
4: Run VarianceRecovery using the next $m_2$ samples
   and $\widehat{A}_1, \dots, \widehat{A}_n$ to obtain $\widehat{\sigma}_1^2, \dots, \widehat{\sigma}_n^2$
5: return $\widehat{A}_1, \dots, \widehat{A}_n, \widehat{\sigma}_1^2, \dots, \widehat{\sigma}_n^2$
```

Motivated by Proposition 2.1, we will estimate parameters for each variable in an independent fashion[5]. We will provide various coefficient recovery algorithms in

[4]For a cleaner argument, we will bound $d_{\mathrm{KL}}(\mathcal{P}, \mathcal{Q}) \le 3\varepsilon$. This is qualitatively the same as showing $d_{\mathrm{KL}}(\mathcal{P}, \mathcal{Q}) \le \varepsilon$ since one can repeat the entire analysis with $\varepsilon' = \varepsilon/3$.

[5]Given the samples, parameters related to each variable can be estimated in parallel. Furthermore, it suffices to use *one* batch of samples for all the nodes as we can obtain high-probability bounds on the error events at each node.

the subsequent sections. These algorithms will recover coefficients $\widehat{A}_i$ that satisfy Condition 1 for each variable X_i. For variance recovery, we use `VarianceRecovery` for each variable Y by computing the empirical variance[6] of $Y - X\widehat{A}$ such that the recovered variance satisfies Condition 2. Note that if Y has no parents, then $p = 0$ and $\widehat{A} = 0$.

Algorithm 2 `VarianceRecovery`: Variance recovery algorithm given coefficient estimates

```
1: Input: DAG $G$, estimates $\widehat{A}$, and $m_2$ samples
2: for variable $Y$ with parents $X_1, \ldots, X_p$ do
3:     for $s = 1, \ldots, m_2$ do
4:         Compute $Z^{(s)} = (Y^{(s)} - X^{(s)}\widehat{A})^2$, where $Y^{(s)}$ and $X^{(s)} = [X_1^{(s)}, \ldots, X_p^{(s)}]$ are the $s^{th}$ sample of $Y$ and $X_1, \ldots, X_p$ (as a row vector) respectively.
5:     end for
6:     Estimate $\widehat{\sigma}_y^2 = \frac{1}{m_2} \sum_{i=1}^{m_2} Z^{(s)}$
7: end for
```

To analyze `VarianceRecovery`, we first prove guarantees for an arbitrary variable and then take union bound over n variables.

Lemma 2.2. *Consider Algorithm 2. Fix any arbitrary variable of interest Y with p parents, parameters (A, σ_y), and associated covariance matrix M. Define $\gamma = \frac{\varepsilon \cdot p}{n \cdot d_{avg}}$. Suppose we have coefficient estimates $\widehat{A}$ such that $\left|\Delta^\top M \Delta\right| \leq \sigma_y^2 \gamma$. Suppose $0 \leq \varepsilon \leq 3 - 2\sqrt{2} \leq 0.17$. With $k = 32\gamma^{-1} \cdot \log(2\delta^{-1})$ samples, we recover variance estimate $\widehat{\sigma}_y$ such that*

$$\Pr\left((1 - \sqrt{\gamma}) \cdot \sigma_y^2 \leq \widehat{\sigma}_y^2 \leq (1 + \sqrt{\gamma}) \cdot \sigma_y^2\right) \geq 1 - \delta.$$

Corollary 2.3 (Guarantees of `VarianceRecovery`). *Consider Algorithm 2. Define $\gamma = \frac{\varepsilon \cdot p}{n \cdot d_{avg}}$. Suppose $0 \leq \varepsilon \leq 3 - 2\sqrt{2} \leq 0.17$ and we have coefficient estimates $\widehat{A}_i$ such that $\left|\Delta_i^\top M_i \Delta_i\right| \leq \sigma_i^2 \gamma$ for all $i \in [n]$. With $m_2 \in \mathcal{O}(n d_{avg} \varepsilon^{-1} \cdot \log(n\delta^{-1}))$ samples, we recover variance estimates $\widehat{\sigma}_1, \ldots, \widehat{\sigma}_n$ such that*

$$\Pr\left(\forall i \in [n], (1 - \sqrt{\gamma}) \cdot \sigma_i^2 \leq \widehat{\sigma}_i^2 \leq (1 + \sqrt{\gamma}) \cdot \sigma_i^2\right) \geq 1 - \delta$$

The total running time is $\mathcal{O}(n^2 d_{avg}^2 \varepsilon^{-1} \cdot \log(\delta^{-1}))$.

In Section 5, we show that this sample complexity is nearly optimal with respect to n and ε.

[6]Except in our experiments with contaminated data in Section 6 where we use the classical median absolute deviation (MAD) estimator, which we describe in the appendix.

3 COEFFICIENT ESTIMATORS BASED ON LINEAR LEAST SQUARES

In this section, we provide algorithms `LeastSquares` and `BatchAvgLeastSquares` for recovering the coefficients in a Bayesian network using linear least squares. As discussed in Section 2.4, we will recover the coefficients for each variable such that Condition 1 is satisfied. To do so, we estimate the coefficients associated with each individual variable using independent samples. At each node, `LeastSquares` computes an estimate by solving the linear least squares problem with respect to a collection of sample observations. `BatchAvgLeastSquares` generalizes this approach by allowing any interpolation between "batch size" and "number of batches" – `LeastSquares` is a special case of a single batch. Since each solution to batch can be computed independently before their results are combined, `BatchAvgLeastSquares` facilitates parallelism.

3.1 Vanilla least squares

Consider an arbitrary variable Y with p parents. Using m_1 independent samples, we form *column* vector $B = [Y^{(1)}, \ldots, Y^{(m_1)}]^\top \in \mathbb{R}^{m_1}$ and matrix $X \in \mathbb{R}^{m_1 \times p}$, where the r^{th} *row* consists of sample values $X_1^{(r)}, \ldots, X_p^{(r)}$. Then, we define $\widehat{A} = (X^\top X)^{-1} X^\top B$ as the solution to the least squares problem $X\widehat{A} = B$. Algorithm 3 shows the pseudocode of `LeastSquares` and Theorem 3.1 states its guarantees.

Algorithm 3 `LeastSquares`: Coefficient recovery algorithm for general Bayesian networks

```
1: Input: DAG $G$ and $m_1$ samples
2: for variable $Y$ with parents $X_1, \ldots, X_p$ do
3:     Compute $\widehat{A} = (X^\top X)^{-1} X^\top B$ as the solution to the least squares problem $X\widehat{A} = B$, where column vector $B = [Y^{(1)}, \ldots, Y^{(m_1)}]^\top \in \mathbb{R}^{m_1}$ and the $r^{th}$ row of matrix $X \in \mathbb{R}^{m_1 \times p}$ consists of sample values $X_1^{(r)}, \ldots, X_p^{(r)}$
4: end for
```

Theorem 3.1 (Distribution learning using `LeastSquares`). *Let $\varepsilon, \delta \in (0, 1)$. Suppose G is a fixed DAG on n variables with in-degree at most d. Given $\mathcal{O}(n d_{avg} \varepsilon^{-2} \cdot \log(n\delta^{-1}))$ samples from an unknown Bayesian network $\mathcal{P}$ over G, if we use `LeastSquares` for coefficient recovery in Algorithm 1, then with probability at least $1 - \delta$, we recover a Bayesian network $\mathcal{Q}$ over G in $\mathcal{O}(n^2 d_{avg}^2 d \varepsilon^{-2} \cdot \log(\delta^{-1}))$ time such that* $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$.

In the proof of Theorem 3.1, we show that we obtain a $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) \leq \epsilon$ guarantee and then appeal to Pinsker's inequality to relate KL with TV. As a consequence of our result, we can learn centered multivariate Gaussians using $\widetilde{O}(n^2/\epsilon)$ samples up to $d_{\mathrm{KL}}(\mathcal{P},\mathcal{Q}) \leq \epsilon$. For an analogous $d_{\mathrm{KL}}(\mathcal{Q},\mathcal{P}) \leq \epsilon$ guarantee, see [ABDH+20].

3.2 Interpolating between batch size and number of batches

We now generalize `LeastSquares`. For each variable with p parents, `BatchAvgLeastSquares` solves b batches of linear systems made up of k samples and then uses the *mean* of the recovered solutions as an estimate for the coefficients. Note that one can interpolate between different values of k and b, as long as $k \geq p$ (so that batch solutions are correlated to true parameters). Algorithm 4 describes `BatchAvgLeastSquares` and Theorem 3.2 states its guarantees.

Algorithm 4 `BatchAvgLeastSquares`: Coefficient recovery for general Bayesian networks

```
1: Input: DAG $G$ and $m_1$ samples
2: Let $k \in \Omega(d + \log(n\varepsilon^{-1}\delta^{-1}))$ and $b = m_1/k$
3: for variable $Y$ with parents $X_1, \ldots, X_p$ do
4:     for $s = 1, \ldots, b$ do
5:         Compute $\widetilde{A}^{(s)} = (X^\top X)^{-1} X^\top B$ as the solution to the least squares problem $X\widetilde{A}^{(s)} = B$, where column vector $B = [Y^{(s,1)}, \ldots, Y^{(s,k)}]^\top \in \mathbb{R}^k$ and the $r^{th}$ row of matrix $X \in \mathbb{R}^{k \times p}$ consists of sample values $X_1^{(s,r)}, \ldots, X_p^{(s,r)}$
6:     end for
7:     Define $\widehat{A} = \frac{1}{b}\sum_{s=1}^{b} \widetilde{A}^{(s)}$
8: end for
```

In Section 6, we also experimented on a variant of `BatchAvgLeastSquares` where $\widehat{A}$ is defined to be the coordinate-wise *median* of the $\widetilde{A}^{(s)}$ vectors. However, we only provide theoretical analysis for `BatchAvgLeastSquares` and not this variant.

Theorem 3.2 (Distribution learning using `BatchAvgLeastSquares`). *Let $\varepsilon, \delta \in (0,1)$. Suppose G is a fixed DAG on n variables with in-degree at most d. Given $\mathcal{O}(nd_{avg}\varepsilon^{-2} \cdot (d + \log(n\varepsilon^{-1}\delta^{-1})))$ samples from an unknown Bayesian network $\mathcal{P}$ over G, if we use `BatchAvgLeastSquares` for coefficient recovery in Algorithm 1, then with probability at least $1-\delta$, we recover a Bayesian network $\mathcal{Q}$ over G in $\mathcal{O}(n^2 d_{avg}^2 d\varepsilon^{-2} \cdot (d + \log(n\varepsilon^{-1}\delta^{-1})))$ time such that $d_{\mathrm{TV}}(\mathcal{P},\mathcal{Q}) \leq \varepsilon$.*

4 COEFFICIENT RECOVERY ALGORITHM BASED ON CAUCHY RANDOM VARIABLES

In this section, we provide novel algorithms `CauchyEst` and `CauchyEstTree` for recovering the coefficients in polytree Bayesian networks. We will show that `CauchyEstTree` has near-optimal sample complexity. Later in Section 6, we will see that both these algorithms outperform `LeastSquares` and `BatchAvgLeastSquares` on randomly generated Bayesian networks. Of technical interest, our analysis involves Cauchy random variables, which are somewhat of a rarity in statistical learning. As in `LeastSquares` and `BatchAvgLeastSquares`, `CauchyEst` and `CauchyEstTree` use independent samples to recover the coefficients associated to each individual variable in an independent fashion.

Consider an arbitrary variable Y with p parents. The intuition is as follows: if $\eta_y = 0$, then one can form a linear system of equations using p samples to solve for the coefficients $a_{y \leftarrow i}$ *exactly* for each $i \in \pi(y)$. Unfortunately, η_y is non-zero in general. Instead of exactly recovering A, we partition the m_1 independent samples into $k = \lfloor m_1/p \rfloor$ batches of p fresh samples and solve systems of linear equations (see Algorithm 5) to obtain intermediate estimates $\widetilde{A}^{(1)}, \ldots, \widetilde{A}^{(k)}$. Then, we fuse these intermediate estimates to obtain our estimate $\widehat{A}$ using appropriate median operations.

Algorithm 5 Batch coefficient recovery algorithm for variable with p parents

```
1: Input: DAG $G$, variable $Y$, and $|\pi_Y| = p$ samples
2: Suppose the parents of $Y$ are $X_1, \ldots, X_p$ and the $r^{th}$ sample involves $X_1^{(r)}, \ldots, X_p^{(r)}, Y^{(r)}$ for $r \in [p]$
3: Compute $\widetilde{A} = [\widehat{a}_{y\leftarrow 1}, \ldots, \widehat{a}_{y\leftarrow p}]^\top$ as any solution to $X\widetilde{A} = \left[Y^{(1)}, \ldots, Y^{(p)}\right]^\top$ where the $r^{th}$ row of matrix $X \in \mathbb{R}^{p\times p}$ is $[X_1^{(r)}, \ldots, X_p^{(r)}]$
4: return $\widetilde{A}$
```

Consider an arbitrary copy of recovered coefficients $\widetilde{A}$. Let $\Delta = [\Delta_1, \ldots, \Delta_p]^\top = \widetilde{A} - A$ be a vector measuring the gap between these recovered coefficients and the ground truth, where $\Delta_i = \widetilde{a}_{y\leftarrow i} - a_{y\leftarrow i}$. Lemma 4.1 gives a condition where a vector is term-wise Cauchy. Using this, Lemma 4.2 shows that each entry of the vector $L^\top \Delta$ is distributed according to $\sigma_y \cdot \mathrm{Cauchy}(0,1)$, although the entries may be correlated with each other in general.

Lemma 4.1. *Consider the matrix equation $AB = E$ where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$, and $E \in R^{n\times 1}$ such that entries of A and E are independent Gaussians, elements in each* column *of A have the same variance, and all entries in E have the same variance. That is, $A_{\cdot,j} \sim N(0,\sigma_i^2)$ and $E_i \sim N(0,\sigma_{n+1}^2)$. Then, we have that $B_i \sim \frac{\sigma_{n+1}}{\sigma_i} \cdot \mathrm{Cauchy}(0,1)$ for all $i \in [n]$.*

Lemma 4.2. *Consider a batch estimate $\widetilde{A}$ from Algorithm 5 and define $\Delta = \widetilde{A} - A$. Then, $L^\top\Delta$ is entry-wise distributed as $\sigma_y \cdot \mathrm{Cauchy}(0,1)$. Note that the entries of $L^\top\Delta$ may be correlated in general.*

If we have direct access to the matrix L, then one can do the following: for each $i \in [p]$, take *medians*[7] of $\left(L^\top [\widetilde{a}_{y\leftarrow 1}, \ldots, \widetilde{a}_{y\leftarrow n}]^\top\right)_i$ to form $\texttt{MED}_i$ and then estimate $[\widehat{a}_{y\leftarrow 1}, \ldots, \widehat{a}_{y\leftarrow 1}] = (L^\top)^{-1}[\texttt{MED}_1, \ldots, \texttt{MED}_n]^\top$. By the convergence of Cauchy random variables to their median, one can show that each $\widehat{a}_{y\leftarrow i}$ converges to the true coefficient $a_{y\leftarrow i}$ as before. Unfortunately, we do not have L and can only hope to estimate it with some matrix $\widehat{L}$ using the *empirical* covariance matrix $\widehat{M}$ (see Algorithm 6).

Algorithm 6 `CauchyEst`: Coefficient recovery algorithm for general Bayesian networks

```
1: Input: DAG $G$ and $m$ samples
2: for variable $Y$ with parents $X_1, \ldots, X_p$ do
3:     Let $\widehat{M}$ be the empirical covariance matrix with respect to $X_1, \ldots, X_p$
4:     Compute Cholesky decomposition $\widehat{M} = \widehat{L}\widehat{L}^\top$
5:     for $s = 1, \ldots, \lfloor m/p \rfloor$ do        ▷ Use Algorithm 5
6:         Compute $\widetilde{A}^{(s)} \in \mathbb{R}^p$ using $p$ fresh samples
7:     end for
8:     for $i = 1, \ldots, p$ do
9:         Compute $\texttt{MED}_i$ as the median of the value at the $i^{th}$ coordinates $(\widehat{L}^\top \widetilde{A}^{(1)})_i, \ldots, (\widehat{L}^\top \widetilde{A}^{(\lfloor m/p \rfloor)})_i$
10:    end for
11:    Define $\widehat{A}^\top = (\widehat{L}^\top)^{-1} [\texttt{MED}_1, \ldots, \texttt{MED}_n]^\top$
12: end for
```

4.1 Polytree Bayesian networks

If the Bayesian network is a polytree, then L is diagonal. In this case, we specialize `CauchyEst` to `CauchyEst-Tree` and are able to give theoretical guarantees. We begin with simple corollary which tells us that the i^{th} entry of Δ is distributed according to $\sigma_y/\sigma_i \cdot \mathrm{Cauchy}(0,1)$.

Corollary 4.3. *Consider a batch estimate $\widetilde{A}$ from Algorithm 5 and define $\Delta = \widetilde{A} - A$. If the Bayesian network is a polytree, then $\Delta_i = (\widetilde{A} - A)_i \sim \frac{\sigma_y}{\sigma_i} \cdot \mathrm{Cauchy}(0,1)$ for all $i \in [n]$.*

For each $i \in \pi(y)$, we combine the k independent copies of $\widetilde{a}^{(1)}_{y\leftarrow i}, \ldots, \widetilde{a}^{(k)}_{y\leftarrow i}$ using the *median*. Observe that $a_{y\leftarrow i}$ is just an unknown *constant*. So, for any sample $s \in [k]$ and parent index $i \in \pi(y)$, we have $\Delta^{(s)}_i = \widetilde{a}^{(s)}_{y\leftarrow i} - a_{y\leftarrow i}$ and $\widehat{a}_{y\leftarrow i} = \mathrm{median}\{\widetilde{a}^{(1)}_{y\leftarrow i}, \ldots, \widetilde{a}^{(k)}_{y\leftarrow i}\} = \mathrm{median}\{\Delta^{(1)}_i, \ldots, \Delta^{(k)}_i\} + a_{y\leftarrow i}$. Since each $\Delta^{(s)}_i$ term is i.i.d. distributed as $\sigma_y \cdot \mathrm{Cauchy}(0,1)$, the term $\mathrm{median}\{\Delta^{(1)}_i, \ldots, \Delta^{(k)}_i\}$ converges to 0 with sufficiently large k, and thus $\widehat{a}_{y\leftarrow i}$ converges to the true $a_{y\leftarrow i}$.

[7]The typical strategy of averaging independent estimates fails here as the variance of a Cauchy variable is unbounded.

Our specialized algorithm `CauchyEstTree` is given in Algorithm 7 with its guarantees stated in Theorem 4.4.

Algorithm 7 `CauchyEstTree`: Coefficient recovery algorithm for polytree Bayesian networks

```
1: Input: A polytree $G$ and $m_1$ samples
2: for variable $Y$ with parents $X_1, \ldots, X_p$ do
3:     for $s = 1, \ldots, \lfloor m_1/p \rfloor$ do    ▷ Use Algorithm 5
4:         Compute $\widetilde{A}^{(s)}$ using $p$ fresh samples
5:     end for
6:     for $i = 1, \ldots, p$ do
7:         Compute $\widehat{a}_{y\leftarrow i}$ as the median of the value at the $i^{th}$ coordinates $\widetilde{A}^{(1)}_i, \ldots, \widetilde{A}^{(\lfloor m_1/p \rfloor)}_i$
8:     end for
9:     Define $\widehat{A} = [\widehat{a}_{y\leftarrow 1}, \ldots, \widehat{a}_{y\leftarrow p}]^\top$
10: end for
```

Theorem 4.4 (Distribution learning using `CauchyEst-Tree`)**.** *Let $\varepsilon, \delta \in (0,1)$. Suppose G is a fixed DAG on n variables with degree at most d. Given $\mathcal{O}(nd_{avg}d\varepsilon^{-1} \cdot \log(n\delta^{-1}))$ samples from an unknown Bayesian network $\mathcal{P}$ over G, if we use `CauchyEstTree` for coefficient recovery in Algorithm 1, then with probability at least $1-\delta$, we recover a Bayesian network $\mathcal{Q}$ over G such that $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$ in $\mathcal{O}(n^2 d^2_{avg} d^{\omega-1} \varepsilon^{-1} \cdot \log(n\delta^{-1}))$ time.*

Note that for polytrees, d_{avg} is just a constant.

5 HARDNESS FOR LEARNING GAUSSIAN BAYESIAN NETWORKS

In this section, we present our two hardness results. Theorem 5.1 illustrates a tight lower-bound for the simpler case of learning *Gaussian product distributions* in total variational distance while Theorem 5.2 shows a tight lower-bound for learning *Gaussian Bayesian networks* with respect to total variational distance.

Theorem 5.1. *Given samples from a n-fold Gaussian product distribution P, learning a $\widehat{P}$ such that in $\mathrm{d_{TV}}(P, \widehat{P}) = O(\varepsilon)$ with success probability 2/3 needs $\Omega(n\varepsilon^{-2})$ samples in general.*

Theorem 5.2. *For any $0 < \varepsilon < 1$ and n, d such that $d \leq n/2$, there exists a DAG G over $[n]$ of in-degree d such that learning a Gaussian Bayesian network $\widehat{P}$ on G such that $\mathrm{d_{TV}}(P, \widehat{P}) \leq \varepsilon$ with success probability 2/3 needs $\Omega(nd\varepsilon^{-2})$ samples in general.*

Both hardness results apply to the problems of learning the covariance matrix of a centered multivariate Gaussian, which is equivalent to recovering the coefficients and noises of the underlying linear structural equation.

6 EXPERIMENTS

General Setup For each experiment, we report the average (over 20 random instantiations) KL divergence between the ground truth and our learned distribution using Eq. (1). All experiments were conducted on an Intel Core i7-9750H 2.60GHz CPU. Our code is available at `https://github.com/YohannaWANG/CauchyEst`.

DAGs considered For our experiments, we tested algorithms on Erdős-Rényi graphs $G(n, p)$ with bounded *expected* degrees (i.e. $p = d/n$ for some degree parameter d) using the Python package `networkx` [HSSC08]. In the supplementary material, we also present experimental results for synthetic polytree networks (generated using random Prüfer sequences), four real Gaussian Bayesian networks from R package `bnlearn` [Scu09], and the non-realizable/agnostic setting where the data is not generated according to the input graph given to the algorithms.

Synthetic data generation For each edge in the DAG, we uniformly draw a value from the range $(-2, -1] \cup [1, 2)$ as our coefficient weight. Since there are no self-loops, this corresponds a strictly upper triangular matrix B. Then, for sample size $m \in \{1000, 2000, \ldots, 5000\}$, we generate our i.i.d. samples $X \in \mathbb{R}^{m \times n}$ by computing $X = B^\top X + \eta$, where $\eta \sim N(\mathbf{0}, I_{n \times n})$.[8] To simulate contamination of data samples, we randomly choose $m/20$ samples (5%) and changed the value of some nodes to either $N(1000, 1)$ or Cauchy$(1000, 1)$. In our appendix, we also experimented on graphs where we assign $N(0, 10^{-20})$ additive noise instead of $N(0, 1)$ noise to certain nodes.

Algorithms We evaluated `LeastSquares`, `BatchAvgLeastSquares`, `BatchMedLeastSquares`, `CauchyEstTree`, `CauchyEst`, Graphical Lasso [FHT08], CLIME [CLL11], and the empirical MLE estimator. We use `BatchAvg_LS+`x and `BatchMed_LS+`x to represent the `BatchAvgLeastSquares` and `BatchMedLeastSquares` algorithms respectively with a batch size of $p + x$ at each node, where p is the number of parents of that node. Here, we only give results for $p + 20$. For other batch sizes, please refer to Appendix F.

Analysis of selected experiment results Fig. 1 shows our experimental results for graphs with 100 variables, under the *realizable* setting. We show results for Erdős-Rényi graphs with *uncontaminated* data in Fig. 1(a) and *contaminated* data in Fig. 1(b). In both experiments, the GLASSO, CLIME and empirical MLE estimators perform very poorly, with MLE and CLIME have similar performance. For this reason, Fig. 1 only displays the plots for `LeastSquares`, `BatchMedLeastSquares`, `BatchAvgLeastSquares`, `CauchyEstTree`, and `CauchyEst` for a clearer comparison. In the uncontaminated setting (Fig. 1(a)), `LeastSquares` outperforms all the other estimators for small sample sizes but we expect the gap to narrow as the number of sample size increases (since we have theoretical guarantees for some of these estimators). Meanwhile, in the contaminated setting (Fig. 1(b)), we see that the median-based estimators significantly outperforms the other estimators. This is because the median operator makes the estimators more robust against such contamination.

[8]We do not report the results over the varied variance synthetic data, because their performance are close to the performance of the equal variance synthetic data.

Takeaways Across all of our experiments, the `LeastSquares` estimator runs the fastest and performs the best among all algorithms on *uncontaminated* datasets (both real and synthetic). This holds even when some nodes are assigned very small additive Gaussian noise. However, if the data for a small fraction of the samples are contaminated, then `CauchyEst`, `CauchyEstTree` and `BatchMedLeastSquares` outperform `LeastSquares` and `BatchAvgLeastSquares` by a large margin. This observation holds under different noise and graph types. Lastly, in the non-realizable/agnostic learning setting where the data is not generated according to the input graph, then `BatchAvgLeastSquares`, `CauchyEst`, and `CauchyEstTree` empirically outperform other algorithms. However, we do not have a theoretical explanation why this happens.

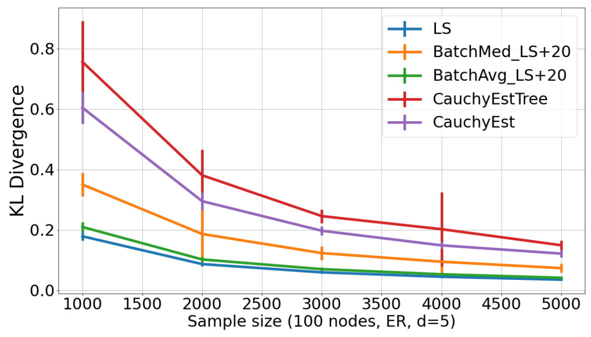
Acknowledgements

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-PhD/2021-08-013).

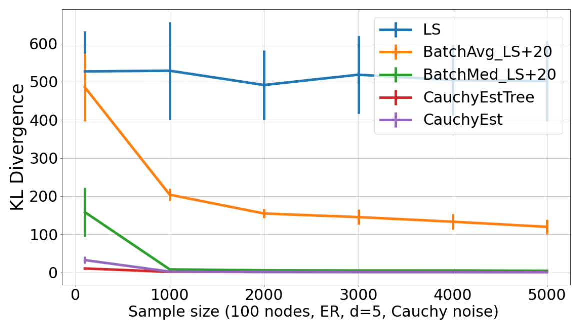
References

[ABDH+20] Hassan Ashtiani, Shai Ben-David, Nicholas JA Harvey, Christopher Liaw, Abbas Mehrabian, and Yaniv Plan. Near-optimal sample complexity bounds for robust learning of gaussian mixtures via compression schemes. *Journal of the ACM (JACM)*, 67(6):1–42, 2020. 2, 6, 16

[AGZ19] Bryon Aragam, Jiaying Gu, and Qing Zhou. Learning large-scale bayesian networks with the sparsebn package. *Journal of Statistical Software*, 91(1):1–38, 2019. 3



(a) Algorithms evaluated on ER graph with $d = 5$ on *uncontaminated* data



(b) Algorithms evaluated on ER graph with $d = 5$ on *contaminated* data, where 5 random node values are set to Cauchy$(1000, 1)$ instead of following the linear structural equations in each contaminated sample

Figure 1: Select experiments on uncontaminated and contaminated data (lower KL divergence is better)

[AZ15] Bryon Aragam and Qing Zhou. Concave penalized estimation of sparse gaussian bayesian networks. *The Journal of Machine Learning Research*, 16(1):2273–2328, 2015. 3

[BCD20] Johannes Brustle, Yang Cai, and Constantinos Daskalakis. Multi-item mechanisms without item-independence: Learnability via robustness. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 715–761, 2020. 2

[BGMV20] Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S Meel, and NV Vinodchandran. Efficient distance approximation for structured high-dimensional distributions via learning. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[BGPV20] Arnab Bhattacharyya, Sutanu Gayen, Eric Price, and NV Vinodchandran. Near-optimal learning of tree-structured distributions by chow-liu. *arXiv preprint arXiv:2011.04144*, 2020. ACM STOC 2021. 3

[BVH$^+$16] Afonso S Bandeira, Ramon Van Handel, et al. Sharp nonasymptotic bounds on the norm of random matrices with independent entries. *Annals of Probability*, 44(4):2479–2506, 2016. 12

[CDKS20] C. L. Canonne, I. Diakonikolas, D. M. Kane, and A. Stewart. Testing bayesian networks. *IEEE Transactions on Information Theory*, 66(5):3132–3170, 2020. 3

[CDW19] Wenyu Chen, Mathias Drton, and Y Samuel Wang. On causal discovery with an equal-variance assumption. *Biometrika*, 106(4):973–980, 2019. 3

[CLL11] Tony Cai, Weidong Liu, and Xi Luo. A constrained l1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011. 8, 25

[Das97] Sanjoy Dasgupta. The sample complexity of learning fixed-structure bayesian networks. *Mach. Learn.*, 29(2-3):165–180, 1997. 2, 3, 4

[Dia16] Ilias Diakonikolas. Learning structured distributions. *Handbook of Big Data*, 267, 2016. 2

[DMR18] Luc Devroye, Abbas Mehrabian, and Tommy Reddad. The total variation distance between high-dimensional gaussians. *arXiv preprint arXiv:1810.08693*, 2018. 14

[Duc07] John Duchi. Derivations for linear algebra and optimization. *Berkeley, California*, 3(1):2325–5870, 2007. 25

[FHT08] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008. 8, 25

[GDA20] Ming Gao, Yi Ding, and Bryon Aragam. A polynomial-time algorithm for learning nonparametric causal graphs. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[GH17] Asish Ghoshal and Jean Honorio. Learning identifiable Gaussian bayesian networks in polynomial time and sample complexity. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6460–6469, 2017. 3

[Gut09] Allan Gut. *An Intermediate Course in Probability*. Springer New York, 2009. 12

[GZ20] Jiaying Gu and Qing Zhou. Learning big gaussian bayesian networks: Partition, estimation and fusion. *Journal of Machine Learning Research*, 21(158):1–31, 2020. 3

[Hau18] David Haussler. *Decision theoretic generalizations of the PAC model for neural net and other learning applications*. CRC Press, 2018. 3

[HSSC08] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. 8

[HTW19] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2019. 2

[Hub04] Peter J Huber. *Robust statistics*, volume 523. John Wiley & Sons, 2004. 15, 27

[JNG$^+$19] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan. A short note on concentration inequalities for random vectors with subgaussian norm. *arXiv preprint arXiv:1902.03736*, 2019. 13

[KMR$^+$94] Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E Schapire, and Linda Sellie. On the learnability of discrete distributions. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 273–282, 1994. 1

[KP83] H Kim and J Perl. 'a computational model for combined causal and diagnostic reasoning in inference systems', 8th ijcai, 1983. 2

[Mue99] Ralph O Mueller. *Basic principles of structural equation modeling: An introduction to LISREL and EQS*. Springer Science & Business Media, 1999. 1

[Mul09] Stanley A Mulaik. *Linear causal modeling with structural equations*. CRC press, 2009. 1

[ORS07] Rainer Opgen-Rhein and Korbinian Strimmer. From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC systems biology*, 1(1):1–10, 2007. 25

[Par20] Gunwoong Park. Identifiability of additive noise models using conditional variances. *Journal of Machine Learning Research*, 21(75):1–34, 2020. 3

[PB14] Jonas Peters and Peter Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2014. 3

[Pea86] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288, 1986. 2

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, San Francisco, Calif, 2nd edition edition, 1988. 3

[PK20] Gunwoong Park and Youngwhan Kim. Identifiability of Gaussian linear structural equation models with homogeneous and heterogeneous error variances. *Journal of the Korean Statistical Society*, 49(1):276–292, 2020. 3

[RV09] Mark Rudelson and Roman Vershynin. Smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 62(12):1707–1739, 2009. 12

[Scu09] Marco Scutari. Learning bayesian networks with the bnlearn r package. *arXiv preprint arXiv:0908.3817*, 2009. 8, 25

[SHBM14] Marco Scutari, Phil Howell, David J Balding, and Ian Mackay. Multiple quantitative trait analysis using bayesian networks. *Genetics*, 198(1):129–137, 2014. 25

[SS05] Juliane Schafer and Korbinian Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical applications in genetics and molecular biology*, 4(1), 2005. 25

[Tsy08] Alexandre B Tsybakov. *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008. 13

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 1, 3

[Wai19] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019. 12

Supplementary Material: Learning Sparse Fixed-Structure Gaussian Bayesian Networks

A Preliminaries

Appendix A.1 states facts and standard results such as some standard concentration bounds and Pinsker's inequality which relates TV distance with KL divergence. Appendix A.2 provides the full derivation of Eq. (1) which decomposes the KL divergence via d_{CP}. In Appendix A.3, we give the pseudocode of the Median Absolute Deviation (MAD) estimator.

A.1 Basic facts and results

Fact A.1. *Suppose $X \sim N(0, \sigma^2)$. Then, for any $t > 0$, $\Pr(X > t) \leq \exp(-\frac{t^2}{2\sigma^2})$.*

Fact A.2. *Consider any matrix $B \in \mathbb{R}^{n \times m}$ with rows $B_1, \ldots, B_n \in \mathbb{R}^m$. For any $i \in [m]$ and any vector $v \in \mathbb{R}^m$ with i.i.d. $N(0, \sigma^2)$ entries, we have that $(Bv)_i = B_i v \sim N(0, \sigma^2 \cdot \|B_i\|^2)$.*

Fact A.3 (Theorem 2.2 in [Gut09])**.** *Let $X_1, \ldots, X_p \sim N(0, LL^\top)$ be p i.i.d. n-dimensional multivariate Gaussians with covariance $LL^\top \in \mathbb{R}^{n \times n}$ (i.e. $L \in \mathbb{R}^{n \times p}$). If $X \in \mathbb{R}^{p \times n}$ is the matrix formed by stacking $X_1, \ldots, X_p$ as rows of X, then $X = GL^\top$ where $G \in \mathbb{R}^{p \times p}$ is a random matrix with i.i.d. $N(0,1)$ entries.*[9]

Lemma A.4 (Equation 2.19 in [Wai19])**.** *Let $Y = \sum_{k=1}^{n} Z_k^2$, where each $Z_k \sim N(0,1)$. Then, $Y \sim \chi_n^2$ and for any $0 < t < 1$, we have $\Pr(|Y/n - 1| \geq t) \leq 2\exp\left(-nt^2/8\right)$.*

Lemma A.5 (Consequence of Corollary 3.11 in [BVH+16])**.** *Let $G \in \mathbb{R}^{n \times m}$ be a matrix with i.i.d. $N(0,1)$ entries where $n \leq m$. Then, for some universal constant C, $\Pr(\|G\| \geq 2(\sqrt{n} + \sqrt{m})) \leq \sqrt{n} \cdot \exp(-C \cdot m)$.*

Lemma A.6 ([RV09]; Theorem 6.1 and Equation 6.10 in [Wai19])**.** *Let $\ell \geq d$ and $G \in \mathbb{R}^{\ell \times d}$ be a matrix with i.i.d. $N(0,1)$ entries. Denote $\sigma_{\min}(G)$ as the smallest singular value of G. Then, for any $0 < t < 1$, we have $\Pr\left(\sigma_{\min}(G) \geq \sqrt{\ell}(1-t) - \sqrt{d}\right) \leq \exp\left(-\ell t^2/2\right)$.*

Lemma A.7. *Let $G \in \mathbb{R}^{k \times d}$ be a matrix with i.i.d. $N(0,1)$ entries. Then, for any constant $0 < c_1 < 1/2$ and $k \geq d/c_1^2$,*

$$\Pr\left(\left\|(G^\top G)^{-1}\right\|_{op} \leq \frac{1}{(1-2c_1)^2 k}\right) \geq 1 - \exp\left(-\frac{kc_1^2}{2}\right)$$

Proof of Lemma A.7. Observe that $G^\top G$ is symmetric, thus $(G^\top G)^{-1}$ is also symmetric and the eigenvalues of $G^\top G$ equal the singular values of $G^\top G$. Also, note that event that $G^\top G$ is singular has measure 0.[10]

By definition of operation norm, $\left\|(G^\top G)^{-1}\right\|$ equals the square root of *maximum* eigenvalue of

$$((G^\top G)^{-1})^\top((G^\top G)^{-1}) = ((G^\top G)^{-1})^2,$$

where the equality is because $(G^\top G)^{-1}$ is symmetric. Since $G^\top G$ is invertible, we have $\|(G^\top G)^{-1}\| = 1/\|G^\top G\|$, which is equal to the inverse of *minimum* eigenvalue $\lambda_{\min}(G^\top G)$ of $G^\top G$, which is in turn equal to the square of minimum singular value $\sigma_{\min}(G)$ of G.

Therefore, the following holds with probability at least $1 - \exp\left(-kc_1^2/2\right)$:

$$\left\|\left(G^\top G\right)^{-1}\right\| = \frac{1}{\|G^\top G\|} = \frac{1}{\lambda_{\min}(G^\top G)} = \frac{1}{\sigma_{\min}^2(G)} \leq \frac{1}{\left(\sqrt{k}(1-c_1) - \sqrt{d}\right)^2} \leq \frac{1}{(1-2c_1)^2 k}$$

[9] The transformation stated in [Gut09, Theorem 2.2, page 120] is for a single multivariate Gaussian vector, thus we need to take the transpose when we stack them in rows. Note that G and $G^\top$ are identically distributed.

[10] Consider fixing all but one arbitrary entry of G. The event of this independent $N(0,1)$ entry making $\det(G^\top G) = 0$ has measure 0.

where the second last inequality is due to Lemma A.6 and the last inequality holds when $k \geq d/c_1^2$. □

Lemma A.8. *Let $G \in \mathbb{R}^{k \times p}$ be a matrix with i.i.d. $N(0,1)$ entries and $\eta \in \mathbb{R}^k$ be a vector with i.i.d. $N(0,\sigma^2)$ entries, where G and η are independent. Then, for any constant $c_2 > 0$,*

$$\Pr\left(\left\|G^\top \eta\right\|_2 < 2\sigma c_2 \sqrt{kp}\right) \geq 1 - 2p\exp\left(-2k\right) - p\exp\left(-\frac{c_2^2}{2}\right)$$

Proof of Lemma A.8. Let us denote $g_r \in \mathbb{R}^k$ as the r^{th} row of $G^\top$. Then, we see that $\|G^\top \eta\|_2^2 = \sum_{r=1}^p \langle g_r, \eta\rangle^2$. For any row r, we see that $\langle g_r, \eta\rangle = \|\eta\|_2 \cdot \langle g_r, \eta/\|\eta\|_2\rangle$. We will bound values of $\|\eta\|_2$ and $|\langle g_r, \eta/\|\eta\|_2\rangle|$ separately.

It is well-known (e.g. see [JNG$^+$19, Lemma 2]) that the norm of a Gaussian vector concentrates around its mean. So, $\Pr\left(\|\eta\|_2 \geq 2\sigma\sqrt{k}\right) \leq 2\exp\left(-2k\right)$. Since $g_r \sim N(0, I_k)$ and η are independent, we see that $\langle g_r, \eta/\|\eta\|_2\rangle \sim N(0,1)$. By standard Gaussian bounds, we have that $\Pr\left[|\langle g_r, \eta/\|\eta\|_2\rangle| \geq c_2\right] \leq \exp\left(-c_2^2/2\right)$.

By applying a union bound over these two events, we see that $\|\langle g_r, \eta\rangle\| \geq 2\sigma c_2\sqrt{k}$ for any row with probability at most $2\exp\left(-2k\right) + \exp\left(-c_2^2/2\right)$. The claim follows from applying a union bound over all p rows. □

The next result gives the non-asymptotic convergence of medians of Cauchy random variables. We use this result in the analysis of `CauchyEst`, and it may be of independent interest.

Lemma A.9 (Non-asymptotic convergence of Cauchy median)**.** *Consider a collection of m i.i.d.* Cauchy$(0,1)$ *random variables $X_1, \ldots, X_m$. Given a threshold $0 < \tau < 1$, we have*

$$\Pr\left(\text{median}\left\{X_1, \ldots, X_m\right\} \notin [-\tau, \tau]\right) \leq 2\exp\left(-\frac{m\tau^2}{8}\right)$$

Proof of Lemma A.9. Let $S_{>\tau} = \sum_{i=1}^m \mathbb{1}_{X_i > \tau}$ be the number of values that are larger than τ, where $\mathbb{E}[\mathbb{1}_{X_i > \tau}] = \Pr(X \geq \tau)$. Similarly, let $S_{<-\tau}$ be the number of values that are smaller than $-\tau$. If $S_{>\tau} < m/2$ *and* $S_{<-\tau} < m/2$, then we see that median$\{X_1, \ldots, X_m\} \in [-\tau, \tau]$.

For a random variable $X \sim \text{Cauchy}(0,1)$, we know that $\Pr(X \leq x) = 1/2 + \arctan(x)/\pi$. For $0 < \tau < 1$, we see that $\Pr(X \geq \tau) = 1/2 - \arctan(\tau)/\pi \leq 1/2 - \tau/4$. By additive Chernoff bounds, we see that

$$\Pr\left(S_{>\tau} \geq \frac{m}{2}\right) \leq \exp\left(-\frac{2m^2\tau^2}{16m}\right) = \exp\left(-\frac{m\tau^2}{8}\right)$$

Similarly, we have $\Pr\left(S_{<-\tau} \geq m/2\right) \leq \exp\left(-m\tau^2/8\right)$. The claim follows from a union bound over the events $S_{>\tau} \geq m/2$ and $S_{<-\tau} \geq m/2$. □

Recall that we are given sample access to an unknown probability distribution $\mathcal{P}$ and the corresponding structure of a Bayesian network G on n variables. In this work, we aim to recover parameters such that our induced probability distribution $\mathcal{Q}$ is as close as possible to $\mathcal{P}$ in total variational distance.

Definition A.10 (Total variational (TV) distance)**.** *Given two probability distributions $\mathcal{P}$ and $\mathcal{Q}$ over $\mathbb{R}^n$, the total variational distance between them is defined as* $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) = \sup_{A \in \mathbb{R}^n} |\mathcal{P}(A) - \mathcal{Q}(A)| = \frac{1}{2}\int_{\mathbb{R}^n} |\mathcal{P} - \mathcal{Q}|\, dx$*.*

Instead of directly dealing with total variational distance, we will instead bound the Kullback–Leibler (KL) divergence and then appeal to the Pinsker's inequality [Tsy08, Lemma 2.5, page 88] to upper bound $\mathrm{d_{TV}}$ via $\mathrm{d_{KL}}$. We will later show algorithms that achieve $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$.

Definition A.11 (Kullback–Leibler (KL) divergence)**.** *Given two probability distributions $\mathcal{P}$ and $\mathcal{Q}$ over $\mathbb{R}^n$, the KL divergence between them is defined as* $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) = \int_{A \in \mathbb{R}^n} \mathcal{P}(A) \log\left(\frac{\mathcal{P}(A)}{\mathcal{Q}(A)}\right) dA$*.*

Fact A.12 (Pinsker's inequality)**.** *For distributions $\mathcal{P}$ and $\mathcal{Q}$,* $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) \leq \sqrt{\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q})/2}$*.*

Thus, if $s(\varepsilon)$ samples are needed to learn a distribution $\mathcal{Q}$ such that $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$, $s(\varepsilon^2)$ samples are needed to ensure $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$.

For our hardness results, we will need the following fact about the variation distance between multivariate Gaussians and a Frobenius norm $\|\cdot\|_F$ between the covariance matrices.

Fact A.13 ([DMR18]). *There exists two universal constants $\frac{1}{100} \leq c_1 \leq c_2 \leq \frac{3}{2}$, such that for any two covariance matrices Σ_1 and Σ_2,*

$$c_1 \leq \frac{\mathrm{d_{TV}}(N(0,\Sigma_1), N(0,\Sigma_2))}{\left\| \Sigma_1^{-1}\Sigma_2 - I \right\|_F} \leq c_2.$$

A.2 Details on decomposition of KL divergence

For notational convenience, we write x to mean $(x_1, \ldots, x_n)$, $\pi_i(x)$ to mean the values given to parents of variable X_i by x, and $\mathcal{P}(x)$ to mean $\mathcal{P}(X_1 = x_1, \ldots, X_n = x_n)$. Observe that

$$\begin{aligned}
&\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) \\
&= \int_x \mathcal{P}(x) \log\left(\frac{\mathcal{P}(x)}{\mathcal{Q}(x)}\right) dx \\
&= \int_x \mathcal{P}(x) \log\left(\frac{\Pi_{i=1}^n \mathcal{P}(x_i \mid \pi_i(x))}{\Pi_{i=1}^n \mathcal{Q}(x_i \mid \pi_i(x))}\right) dx && (\star) \\
&= \sum_{i=1}^n \int_x \mathcal{P}(x) \log\left(\frac{\mathcal{P}(x_i \mid \pi_i(x))}{\mathcal{Q}(x_i \mid \pi_i(x))}\right) dx \\
&= \sum_{i=1}^n \int_{x_i, \pi_i(x)} \mathcal{P}(x_i, \pi_i(x)) \log\left(\frac{\mathcal{P}(x_i \mid \pi_i(x))}{\mathcal{Q}(x_i \mid \pi_i(x))}\right) dx_i d\pi_i(x) && \text{Marginalization}
\end{aligned}$$

where $(\star)$ is due to the Bayesian network decomposition of joint probabilities. Let us define

$$\mathrm{d_{CP}}(\alpha_i^*, \widehat{\alpha}_i) = \int_{x_i, \pi_i(x)} \mathcal{P}(x_i, \pi_i(x)) \log\left(\frac{\mathcal{P}(x_i \mid \pi_i(x))}{\mathcal{Q}(x_i \mid \pi_i(x))}\right) dx_i d\pi_i(x)$$

where each $\widehat{\alpha}_i$ and α_i^* represent the parameters that relevant to variable X_i from $\widehat{\alpha}$ and α^* respectively. Under this notation, we can write $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) = \sum_{i=1}^n \mathrm{d_{CP}}(\alpha_i^*, \widehat{\alpha}_i)$.

Fix a variable of interest Y with parents $X_1, \ldots, X_p$, each with coefficient c_i, and variance σ^2. That is, $Y = \eta_y + \sum_{i=1}^p c_i X_i$ for some $\eta_y \sim N(0, \sigma^2)$ that is independent of $X_1, \ldots, X_p$. By denoting $X = x$ (i.e. $X_1 = x_1, \ldots, X_p = x_p$) and $c = (c_1, \ldots, c_p)$, we can write the conditional distribution density of Y as

$$\Pr(y \mid x, c, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \cdot \left(y - \sum_{i=1}^p c_i X_i\right)^2\right)$$

We now analyze $\mathrm{d_{CP}}(\alpha_y^*, \widehat{\alpha}_y)$ with respect to the our estimates $\widehat{\alpha}_y = (\widehat{A}, \widehat{\sigma}_y)$ and the hidden true parameters $\alpha_y^* = (A, \sigma_y)$, where $\widehat{A} = (\widehat{a}_{y\leftarrow 1}, \ldots, \widehat{a}_{y\leftarrow p})$ and $A = (a_{y\leftarrow 1}, \ldots, a_{y\leftarrow p})$.

With respect to variable Y, we see that

$$\begin{aligned}
&\mathrm{d_{CP}}(\alpha_y^*, \widehat{\alpha}_y) \\
&= \int_{x,y} \mathcal{P}(x,y) \ln\left(\frac{\frac{1}{\sigma_y\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \cdot (y - \sum_{i=1}^p a_{y\leftarrow i} X_i)^2\right)}{\frac{1}{\widehat{\sigma}_y\sqrt{2\pi}} \exp\left(-\frac{1}{2\widehat{\sigma}_y^2} \cdot (y - \sum_{i=1}^p \widehat{a}_{y\leftarrow i} X_i)^2\right)}\right) dy\ dx \\
&= \ln\left(\frac{\widehat{\sigma}_y}{\sigma_y}\right) - \frac{1}{2\sigma_y^2} \cdot \mathbb{E}_{x,y}\left(y - \sum_{i=1}^p a_{y\leftarrow i} X_i\right)^2 + \frac{1}{2\widehat{\sigma}_y^2} \cdot \mathbb{E}_{x,y}\left(y - \sum_{i=1}^p \widehat{a}_{y\leftarrow i} X_i\right)^2 \\
&= \ln\left(\frac{\widehat{\sigma}_y}{\sigma_y}\right) - \frac{1}{2\sigma_y^2} \cdot \mathbb{E}_{x,y}\left(y - A^\top X\right)^2 + \frac{1}{2\widehat{\sigma}_y^2} \cdot \mathbb{E}_{x,y}\left(y - \widehat{A}^\top X\right)^2
\end{aligned}$$

By defining $\Delta = \widehat{A} - A$, we can see that for any instantiation of $y, X_1, \ldots, X_p$,

$$\begin{aligned}
&\left(y - \widehat{A}^\top X\right)^2 \\
&= \left(y - (\Delta + A)^\top X\right)^2 && \text{By definition of } \Delta \\
&= \left((y - A^\top X) - \Delta^\top X\right)^2 \\
&= (y - A^\top X)^2 - 2(y - A^\top X)(\Delta^\top X) + \left(\Delta^\top X\right)^2 \\
&= (y - A^\top X)^2 - 2\left(y\Delta^\top X - A^\top X \Delta^\top X\right) + \left(\Delta^\top X\right)^2 \\
&= (y - A^\top X)^2 - 2\left(y X^\top \Delta - A^\top X X^\top \Delta\right) + \Delta^\top X X^\top \Delta && \text{Since } \Delta^\top X \text{ is just a number}
\end{aligned}$$

Denote the covariance matrix with respect to $X_1, \ldots, X_p$ as $M \in \mathbb{R}^{p \times p}$, where $M_{i,j} = \mathbb{E}[X_i X_j]$. Then, we can further simplify $\mathrm{d}_{\mathrm{CP}}(\alpha_y^*, \widehat{\alpha}_y)$ as follows:

$$\begin{aligned}
&\mathrm{d}_{\mathrm{CP}}(\alpha_y^*, \widehat{\alpha}_y) \\
&= \ln\left(\frac{\widehat{\sigma}_y}{\sigma_y}\right) - \frac{1}{2\sigma_y^2} \cdot \mathbb{E}_{x,y}\left(y - A^\top X\right)^2 + \frac{1}{2\widehat{\sigma}_y^2} \cdot \mathbb{E}_{x,y}\left(y - \widehat{A}^\top X\right)^2 && \text{From above} \\
&= \ln\left(\frac{\widehat{\sigma}_y}{\sigma_y}\right) - \frac{1}{2\sigma_y^2} \cdot \mathbb{E}_{x,y}\left(y - A^\top X\right)^2 \\
&\quad + \frac{1}{2\widehat{\sigma}_y^2} \cdot \mathbb{E}_{x,y}\left[(y - A^\top X)^2 - 2\left(y X^\top \Delta - A^\top X X^\top \Delta\right) + \Delta^\top X X^\top \Delta\right] && \text{From above} \\
&= \ln\left(\frac{\widehat{\sigma}_y}{\sigma_y}\right) - \frac{1}{2\sigma_y^2} \cdot \mathbb{E}_{x,y}\eta_y^2 + \frac{1}{2\widehat{\sigma}_y^2} \cdot \mathbb{E}_{x,y}\left[\eta_y^2 - 2\left(\eta_y X^\top \Delta\right) + \Delta^\top X X^\top \Delta\right] && (\dagger) \\
&= \ln\left(\frac{\widehat{\sigma}_y}{\sigma_y}\right) - \frac{1}{2\sigma_y^2} \cdot \sigma_y^2 + \frac{1}{2\widehat{\sigma}_y^2} \cdot \left[\sigma_y^2 - 0 + \Delta^\top M \Delta\right] && (*) \\
&= \ln\left(\frac{\widehat{\sigma}_y}{\sigma_y}\right) - \frac{1}{2} + \frac{\sigma_y^2}{2\widehat{\sigma}_y^2} + \frac{\Delta^\top M \Delta}{2\widehat{\sigma}_y^2} \\
&= \ln\left(\frac{\widehat{\sigma}_y}{\sigma_y}\right) + \frac{\sigma_y^2 - \widehat{\sigma}_y^2}{2\widehat{\sigma}_y^2} + \frac{\Delta^\top M \Delta}{2\widehat{\sigma}_y^2}
\end{aligned}$$

where $(\dagger)$ is because $y = \eta_y + A^\top X$ while $(*)$ is because $\eta_y \sim N(0, \sigma_y^2)$, $\mathbb{E}_{x,y}\left(\eta_y X^\top \Delta\right) = \mathbb{E}_{x,y}\eta_y \cdot \mathbb{E}_{x,y} X^\top \Delta = 0$, and $\mathbb{E}_{x,y}\Delta^\top X X^\top \Delta = \Delta^\top (\mathbb{E}_{x,y} X X^\top) \Delta = \Delta^\top M \Delta$.

In conclusion, we have

$$\mathrm{d}_{\mathrm{KL}}(\mathcal{P}, \mathcal{Q}) = \sum_{i=1}^{n} \mathrm{d}_{\mathrm{CP}}(\alpha_i^*, \widehat{\alpha}_i) = \sum_{i=1}^{n} \ln\left(\frac{\widehat{\sigma}_i}{\sigma_i}\right) + \frac{\sigma_i^2 - \widehat{\sigma}_i^2}{2\widehat{\sigma}_i^2} + \frac{\Delta_i^\top M_i \Delta_i}{2\widehat{\sigma}_i^2} \tag{2}$$

where M_i is the covariance matrix associated with variable X_i, $\alpha_i^* = (A_i, \sigma_i)$ is the coefficients and variance associated with variable X_i, $\alpha_i = (\widehat{A}_i, \widehat{\sigma}_i)$ are the estimates for α_i^*, and $\Delta_i = \widehat{A}_i - A_i$.

A.3 Median absolute deviation

Algorithm 8 states a pseudocode of the well-known Median Absolute Deviation (MAD) estimator (see [Hub04] for example) which we use for the component-wise variance recovery in the contaminated setting. The scale factor, $1/\Phi^{-1}(3/4) \approx 1.4826$ below, is needed to make the estimator unbiased.

B Appendix material for Section 2

In this section, we give the missing proofs of Proposition 2.1, Lemma 2.2, and Corollary 2.3.

Algorithm 8 `MAD`: Variance recovery in the contaminated setting

```
1: Input: Contaminated samples $\{x_1, x_2, \ldots, x_m\}$ from a univariate Gausssian
2: $\widehat{\mu} \leftarrow$ median $\{x_1, x_2, \ldots, x_m\}$.
3: $\widehat{\sigma} \leftarrow 1.4826 \cdot$ median $\{|x_1 - \widehat{\mu}|, |x_2 - \widehat{\mu}|, \ldots, |x_m - \widehat{\mu}|\}$.
4: return $\widehat{\sigma}$
```

Proposition 2.1 (Implication of KL decomposition)**.** *Let $\varepsilon \leq 0.17$ be a constant. For each $i \in [n]$, define $\gamma_i = \frac{\varepsilon \cdot p_i}{n \cdot d_{avg}}$ and suppose $\widehat{\alpha}_i$ has the following properties:*

$$\left|\Delta_i^\top M_i \Delta_i\right| \leq \sigma_i^2 \cdot \gamma_i \qquad \text{(Condition 1)}$$

$$(1 - \sqrt{\gamma_i}) \cdot \sigma_i^2 \leq \widehat{\sigma}_i^2 \leq (1 + \sqrt{\gamma_i}) \cdot \sigma_i^2 \qquad \text{(Condition 2)}$$

Then, $\mathrm{d_{CP}}(\alpha_i^*, \widehat{\alpha}_i) \leq 3\gamma_i = 3 \cdot \frac{\varepsilon \cdot p_i}{n \cdot d_{avg}}$ for all $i \in [n]$. Thus, $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) = \sum_{i=1}^n \mathrm{d_{CP}}(\alpha_i^*, \widehat{\alpha}_i) \leq 3\varepsilon$.[11]

Proof of Proposition 2.1. Consider an arbitrary fixed $i \in [n]$. Denote $\gamma = \frac{\sigma_i^2}{\widehat{\sigma}_i^2}$. Observe[12] that $\gamma - 1 - \ln(\gamma) \leq (\gamma - 1)^2$ for $\gamma \geq 0.3$. Under Eq. (Condition 2), $\gamma \geq 0.3$ always holds since $0 \leq \epsilon p_i \leq n d_{avg}$. Then,

$$\begin{aligned}
\ln\left(\frac{\widehat{\sigma}_i}{\sigma_i}\right) + \frac{\sigma_i^2 - \widehat{\sigma}_i^2}{2\widehat{\sigma}_i^2} &= \frac{1}{2} \cdot \left(\ln\left(\frac{\sigma_i}{\widehat{\sigma}_i}\right) + \frac{\sigma_i^2}{\widehat{\sigma}_i^2} - 1\right) \\
&= \frac{1}{2} \cdot (\ln(\gamma) + \gamma - 1) \\
&\leq \frac{1}{2} \cdot (\gamma - 1)^2 && \text{By Condition 2} \\
&\leq \frac{1}{2} \cdot \left(\frac{1}{1 - \sqrt{\frac{\epsilon p_i}{n d_{avg}}}} - 1\right)^2 && \text{Since } \left(1 - \sqrt{\frac{\epsilon p_i}{n d_{avg}}}\right) \cdot \sigma_i^2 \leq \widehat{\sigma}_i^2 \\
&\leq \frac{2\epsilon p_i}{n d_{avg}} && \text{Holds when } 0 \leq \frac{\epsilon p_i}{n d_{avg}} \leq \frac{1}{4}
\end{aligned}$$

Meanwhile,

$$\begin{aligned}
\frac{\Delta_i^\top M_i \Delta_i}{2\widehat{\sigma}_i^2} &\leq \frac{\left|\Delta_i^\top M_i \Delta_i\right|}{2\widehat{\sigma}_i^2} \\
&\leq \frac{p_i \epsilon}{n d_{avg}} \cdot \frac{\sigma_i^2}{2\widehat{\sigma}_i^2} && \text{By Condition 1} \\
&\leq \frac{p_i \epsilon}{2 n d_{avg}} \cdot \frac{1}{1 - \sqrt{\frac{\epsilon p_i}{n d_{avg}}}} && \text{Since } \left(1 - \sqrt{\frac{\epsilon p_i}{n d_{avg}}}\right) \cdot \sigma_i^2 \leq \widehat{\sigma}_i^2 \\
&\leq \frac{\epsilon p_i}{n d_{avg}} && \text{Holds when } 0 \leq \frac{\epsilon p_i}{n d_{avg}} \leq \frac{1}{4}
\end{aligned}$$

Putting together, we see that $d_{CP}(\alpha_i^*, \widehat{\alpha}_i) \leq \frac{3\epsilon p_i}{n d_{avg}}$. □

Lemma 2.2. *Consider Algorithm 2. Fix any arbitrary variable of interest Y with p parents, parameters (A, σ_y), and associated covariance matrix M. Define $\gamma = \frac{\varepsilon \cdot p}{n \cdot d_{avg}}$. Suppose we have coefficient estimates $\widehat{A}$ such that $\left|\Delta^\top M \Delta\right| \leq \sigma_y^2 \gamma$. Suppose $0 \leq \varepsilon \leq 3 - 2\sqrt{2} \leq 0.17$. With $k = 32\gamma^{-1} \cdot \log(2\delta^{-1})$ samples, we recover variance estimate $\widehat{\sigma}_y$ such that*

$$\Pr\left((1 - \sqrt{\gamma}) \cdot \sigma_y^2 \leq \widehat{\sigma}_y^2 \leq (1 + \sqrt{\gamma}) \cdot \sigma_y^2\right) \geq 1 - \delta.$$

[11]For a cleaner argument, we will bound $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) \leq 3\varepsilon$. This is qualitatively the same as showing $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$ since one can repeat the entire analysis with $\varepsilon' = \varepsilon/3$.

[12]This inequality is also used in [ABDH+20, Lemma 2.9].

Proof of Lemma 2.2. We first argue that $\widehat{\sigma}_y^2 \sim \left(\sigma_y^2 + \Delta^\top M \Delta\right) \cdot \chi_k^2$, then apply standard concentration bounds for χ^2 random variables (see Lemma A.4).

For any sample $s \in [k]$, we see that $Y^{(s)} - X^{(s)}\widehat{A} = X^{(s)}A + \eta_y^{(s)} - X^{(s)}\widehat{A} = \eta_y^{(s)} - X^{(s)}\Delta$, where $\Delta = \widehat{A} - A \in \mathbb{R}^p$ is an unknown constant vector (because we do not actually know A). For fixed Δ, we see that $X^{(s)}\Delta \sim N(0, \Delta^\top M \Delta)$. Since $\eta_y^{(s)} \sim N(0, \sigma_y^2)$ and $X^{(s)}$ are independent, we have that $Y^{(s)} - X^{(s)}\widehat{A} \sim N(0, \sigma_y^2 + \Delta^\top M \Delta)$. So, for any sample $s \in [k]$, $Z^{(s)} = (Y^{(s)} - X^{(s)}\widehat{A})^2 \sim \left(\sigma_y^2 + \Delta^\top M \Delta\right) \cdot \chi_1^2$. Therefore, $\widehat{\sigma}_y = \frac{1}{k}\sum_{s=1}^k Z^{(s)} \sim (\sigma_y^2 + \Delta^\top M \Delta)/k \cdot \chi_k^2$. Let us define

$$\gamma = \frac{\widehat{\sigma}_y^2}{\sigma_y^2} \cdot \left(\frac{1}{1 + \frac{\Delta^\top M \Delta}{\sigma_y^2}} \right) \sim \frac{\chi_k^2}{k}$$

Since $p \leq nd_{avg}$, if $\varepsilon \leq 3 - 2\sqrt{2}$, then $\frac{\varepsilon p}{nd_{avg}} \leq 3 - 2\sqrt{2} \leq 3 + 2\sqrt{2}$. We first make two observations:

1. For $0 \leq \frac{\varepsilon p}{nd_{avg}} \leq 3 - 2\sqrt{2}$, $\left(1 + \sqrt{\frac{\varepsilon p}{nd_{avg}}}\right) \cdot \left(\frac{1}{1 + \frac{\Delta^\top M \Delta}{\sigma_y^2}}\right) \geq 1 + \sqrt{\frac{\varepsilon p}{4nd_{avg}}}$.
2. For $0 \leq \frac{\varepsilon p}{nd_{avg}} \leq 3 + 2\sqrt{2}$, $\left(1 - \sqrt{\frac{\varepsilon p}{nd_{avg}}}\right) \cdot \left(\frac{1}{1 + \frac{\Delta^\top M \Delta}{\sigma_y^2}}\right) \leq 1 - \sqrt{\frac{\varepsilon p}{4nd_{avg}}}$.

Using Lemma A.4 with the above discussion, we have

$$\begin{aligned}
&\Pr\left(\frac{\widehat{\sigma}_y^2}{\sigma_y^2} \geq 1 + \sqrt{\frac{\varepsilon p}{nd_{avg}}} \quad \text{or} \quad \frac{\widehat{\sigma}_y^2}{\sigma_y^2} \leq 1 - \sqrt{\frac{\varepsilon p}{nd_{avg}}}\right) \\
=\ &\Pr\left(\gamma \geq \left(1 + \sqrt{\frac{\varepsilon p}{nd_{avg}}}\right) \cdot \left(\frac{1}{1 + \frac{\Delta^\top M \Delta}{\sigma_y^2}}\right) \quad \text{or} \quad \gamma \leq \left(1 - \sqrt{\frac{\varepsilon p}{nd_{avg}}}\right) \cdot \left(\frac{1}{1 + \frac{\Delta^\top M \Delta}{\sigma_y^2}}\right)\right) \\
\leq\ &\Pr\left(\gamma \geq 1 + \sqrt{\frac{\varepsilon p}{4nd_{avg}}} \quad \text{or} \quad \gamma \leq 1 - \sqrt{\frac{\varepsilon p}{4nd_{avg}}}\right) \\
=\ &\Pr\left(|\gamma - 1| \geq \sqrt{\frac{\varepsilon p}{4nd_{avg}}}\right) \\
\leq\ &2\exp\left(-\frac{k\varepsilon p}{32nd_{avg}}\right)
\end{aligned}$$

The claim follows by setting $k = \frac{32nd_{avg}}{\varepsilon p}\log\left(\frac{2}{\delta}\right)$. □

Corollary 2.3 (Guarantees of `VarianceRecovery`)**.** *Consider Algorithm 2. Define* $\gamma = \frac{\varepsilon \cdot p}{n \cdot d_{avg}}$*. Suppose* $0 \leq \varepsilon \leq 3 - 2\sqrt{2} \leq 0.17$ *and we have coefficient estimates* $\widehat{A}_i$ *such that* $\left|\Delta_i^\top M_i \Delta_i\right| \leq \sigma_i^2 \gamma$ *for all* $i \in [n]$*. With* $m_2 \in \mathcal{O}(nd_{avg}\varepsilon^{-1} \cdot \log(n\delta^{-1}))$ *samples, we recover variance estimates* $\widehat{\sigma}_1, \ldots, \widehat{\sigma}_n$ *such that*

$$\Pr\left(\forall i \in [n], (1 - \sqrt{\gamma}) \cdot \sigma_i^2 \leq \widehat{\sigma}_i^2 \leq (1 + \sqrt{\gamma}) \cdot \sigma_i^2\right) \geq 1 - \delta$$

The total running time is $\mathcal{O}(n^2 d_{avg}^2 \varepsilon^{-1} \cdot \log(\delta^{-1}))$*.*

Proof of Corollary 2.3. For each $i \in [n]$, apply Lemma 2.2 with $\delta' = \delta/n$ and $m_2 = \frac{32nd_{avg}}{\varepsilon}\log\left(\frac{2}{\delta}\right) \geq \max_{i \in [n]} \frac{32nd_{avg}}{\varepsilon p_i}\log\left(\frac{2}{\delta}\right)$, then take the union bound over all n variables.

The computational complexity for a variable with p parents is $\mathcal{O}(m_2 \cdot p)$. Since $\sum_{i=1}^n p_i = nd_{avg}$, the total runtime is $\mathcal{O}(m_2 \cdot n \cdot d_{avg})$. □

C Appendix material for Section 3

The missing proofs of Theorem 3.1 and Theorem 3.2 are given in Appendix C.1 and Appendix C.2 respectively.

C.1 Proof of Theorem 3.1

Our analysis begins by proving guarantees for an arbitrary variable.

Lemma C.1. *Consider Algorithm 3. Fix an arbitrary variable Y with p parents, parameters (A, σ_y), and associated covariance matrix M. With $k \geq \frac{4c_2^2}{(1-c_1)^4} \cdot \frac{nd_{avg}}{\varepsilon}$ samples, for any constants $0 < c_1 < 1/2$ and $c_2 > 0$, we recover the coefficients $\widehat{A}$ such that*

$$\Pr\left(\left|\Delta^\top M \Delta\right| \geq \sigma_y^2 \cdot \frac{p\varepsilon}{nd_{avg}}\right) \leq \exp\left(-\frac{kc_1^2}{2}\right) + 2p\exp\left(-2k\right) + p\exp\left(-\frac{c_2^2}{2}\right)$$

Proof. Since $\left|\Delta^\top M \Delta\right| = \left|\Delta^\top L L^\top \Delta\right| = \left\|L^\top \Delta\right\|^2$, it suffices to bound $\left\|L^\top \Delta\right\|$.

Without loss of generality, the parents of Y are $X_1, \ldots, X_p$. Define $X \in \mathbb{R}^{k \times p}$, $B \in \mathbb{R}^k$, and $\widehat{A} \in \mathbb{R}^p$ as in Algorithm 3. Let $\eta = [\eta_y^{(1)}, \ldots, \eta_y^{(k)}] \in \mathbb{R}^k$ be the instantiations of Gaussian η_y in the k samples. By the structural equations, we know that $B = XA + \eta$. So,

$$\widetilde{A} = (X^\top X)^{-1} X^\top B = (X^\top X)^{-1} X^\top (XA + \eta) = A + (X^\top X)^{-1} X^\top \eta$$

By Fact A.3, we can express $X = GL^\top$ where matrix $G \in \mathbb{R}^{k \times p}$ is a random matrix with i.i.d. $N(0,1)$ entries. Since $\Delta = \widehat{A} - A$, we see that $\Delta = (L^\top)^{-1}(G^\top G)^{-1} G^\top \eta$. Rearranging, we have $L^\top \Delta = (G^\top G)^{-1} G^\top \eta$ and so $\|L^\top \Delta\| \leq \|(G^\top G)^{-1}\| \cdot \|G^\top \eta\|$. Combining Lemma A.7 and Lemma A.8, which bound $\|(G^\top G)^{-1}\|$ and $\|G^\top \eta\|$ respectively, we get

$$\Pr\left(\|L^\top \Delta\| > \frac{2\sigma_y c_2 \sqrt{p}}{(1-2c_1)^2 \sqrt{k}}\right) \leq \exp\left(-\frac{kc_1^2}{2}\right) + 2p\exp\left(-2k\right) + p\exp\left(-\frac{c_2^2}{2}\right) \tag{3}$$

for any constants $0 < c_1 < 1/2$ and $c_2 > 0$. The claim follows by setting $k = \frac{4c_2^2}{(1-c_1)^4} \cdot \frac{nd_{avg}}{\varepsilon}$. □

We can now establish Condition 1 of Proposition 2.1 for `LeastSquares`.

Lemma C.2. *Consider Algorithm 3. With $m_1 \in \mathcal{O}\left(\frac{nd_{avg}}{\varepsilon} \cdot \ln\left(\frac{n}{\delta}\right)\right)$ samples, we recover the coefficients $\widehat{A}_1, \ldots, \widehat{A}_n$ such that*

$$\Pr\left(\forall i \in [n], \left|\Delta_i^\top M_i \Delta_i\right| \geq \sigma_i^2 \cdot \frac{\varepsilon p_i}{nd_{avg}}\right) \leq \delta$$

The total running time is $\mathcal{O}\left(\frac{n^2 d_{avg}^2 d}{\varepsilon} \ln\left(\frac{1}{\delta}\right)\right)$.

Proof. By setting $c_1 = 1/4$, $c_2 = \sqrt{2\ln(3n/\delta)}$, and $k = \frac{32nd_{avg}}{\varepsilon} \ln\left(\frac{3n}{\delta}\right) \geq \frac{4c_2^2}{(1-c_1)^4} \cdot \frac{nd_{avg}}{\varepsilon}$ in Lemma C.1, we have

$$\Pr\left(\left|\Delta_i^\top M_i \Delta_i\right| \geq \sigma_i^2 \cdot \frac{p_i \varepsilon}{nd_{avg}}\right) \leq \exp\left(-\frac{kc_1^2}{2}\right) + p\exp\left(-2k\right) + p\exp\left(-\frac{c_2^2}{2}\right) \leq \frac{\delta}{3n} + \frac{\delta}{3n} + \frac{\delta}{3n} = \frac{\delta}{n}$$

for any $i \in [n]$. The claim holds by a union bound over all n variables.

The computational complexity for a variable with p parents is $\mathcal{O}(p^2 \cdot m_1)$. Since $\max_{i \in [n]} p_i \leq d$ and $\sum_{i=1}^n p_i = nd_{avg}$, the total runtime is $\mathcal{O}(m_1 \cdot n \cdot d_{avg} \cdot d)$. □

Theorem 3.1 follows from combining the guarantees of `LeastSquares` and `VarianceRecovery` (given in Lemma C.2 and Corollary 2.3 respectively) via Proposition 2.1.

Theorem 3.1 (Distribution learning using `LeastSquares`)**.** *Let $\varepsilon, \delta \in (0,1)$. Suppose G is a fixed DAG on n variables with in-degree at most d. Given $\mathcal{O}(nd_{avg}\varepsilon^{-2} \cdot \log(n\delta^{-1}))$ samples from an unknown Bayesian network $\mathcal{P}$ over G, if we use `LeastSquares` for coefficient recovery in Algorithm 1, then with probability at least $1-\delta$, we recover a Bayesian network $\mathcal{Q}$ over G in $\mathcal{O}(n^2 d_{avg}^2 d\varepsilon^{-2} \cdot \log(\delta^{-1}))$ time such that $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$.*

Proof of Theorem 3.1. We will show sample and time complexities before giving the proof for the $\mathrm{d_{TV}}$ distance. Let $m_1 \in \mathcal{O}\left(\frac{nd_{avg}}{\varepsilon} \cdot \ln\left(\frac{n}{\delta}\right)\right)$ and $m_2 \in \mathcal{O}\left(\frac{nd_{avg}}{\varepsilon} \log\left(\frac{n}{\delta}\right)\right)$. Then, the total number of samples needed is $m = m_1 + m_2 \in \mathcal{O}\left(\frac{nd_{avg}}{\varepsilon} \log\left(\frac{n}{\delta}\right)\right)$. `LeastSquares` runs in $\mathcal{O}\left(\frac{n^2 d_{avg}^2 d}{\varepsilon} \ln\left(\frac{1}{\delta}\right)\right)$ time while `VarianceRecovery` runs in $\mathcal{O}\left(\frac{n^2 d_{avg}^2}{\varepsilon} \log\left(\frac{1}{\delta}\right)\right)$ time. Therefore, the overall running time is $\mathcal{O}\left(\frac{n^2 d_{avg}^2 d}{\varepsilon} \log\left(\frac{1}{\delta}\right)\right)$.

By Lemma C.2, `LeastSquares` recovers coefficients $\widehat{A}_1, \ldots, \widehat{A}_n$ such that

$$\Pr\left(\forall i \in [n], \left|\Delta_i^\top M_i \Delta_i\right| \geq \sigma_i^2 \cdot \frac{\varepsilon p_i}{nd_{avg}}\right) \leq \delta$$

By Corollary 2.3 and using the recovered coefficients from `LeastSquares`, `VarianceRecovery` recovers variance estimates $\widehat{\sigma}_i^2$ such that

$$\Pr\left(\forall i \in [n], \left(1 - \sqrt{\frac{\varepsilon p_i}{nd_{avg}}}\right) \cdot \sigma_i^2 \leq \widehat{\sigma}_i^2 \leq \left(1 + \sqrt{\frac{\varepsilon p_i}{nd_{avg}}}\right) \cdot \sigma_i^2\right) \geq 1 - \delta$$

As our estimated parameters satisfy Condition 1 and Condition 2, Proposition 2.1 tells us that $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) \leq 3\varepsilon$. Thus, $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) \leq \sqrt{\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q})/2} \leq \sqrt{3\varepsilon/2}$. The claim follows by setting $\varepsilon' = \sqrt{3\varepsilon/2}$ throughout. □

C.2 Proof of Theorem 3.2

Our approach for analyzing `BatchAvgLeastSquares` is the same as our approach for `LeastSquares`: we prove guarantees for an arbitrary variable and then take union bound over n variables. At a high-level, for each node Y, for every fixing of the randomness in generating $X_1, \ldots, X_p$, we show that each $\widetilde{A}^{(s)}$ is a Gaussian. Since the b iterations are independent, $\frac{1}{b} \sum_{s=1}^{b} \widetilde{A}^{(s)}$ is also a Gaussian. Its variance is itself a random variable but can be bounded with high probability using concentration inequalities.

Lemma C.3. *Consider Algorithm 4. Fix any arbitrary variable of interest Y with p parents, parameters (A, σ_y), and associated covariance matrix M. With $k > C_k \cdot \left(p + \ln\left(\frac{n}{\varepsilon\delta}\right)\right)$ and $kb = C_{kb} \cdot \left(\frac{nd_{avg}}{\varepsilon}\left(d + \ln\left(\frac{n}{\varepsilon\delta}\right)\right)\right)$, for some universal constants C_k and C_{kb}, we recover coefficients estimates $\widehat{A}$ such that*

$$\Pr\left(\left|\Delta^\top M \Delta\right| \leq \sigma_y^2 \cdot \frac{\varepsilon p}{nd_{avg}}\right) \geq 1 - \delta$$

Proof. Without loss of generality, the parents of Y are $X_1, \ldots, X_p$. For $s \in [n]$, define $X^{(s)} \in \mathbb{R}^{k \times p}$, $B^{(s)} \in \mathbb{R}^k$, and $\widetilde{A}^{(s)} \in \mathbb{R}^p$ as the quantities involved in the s^{th} batch of Algorithm 4. Let $\eta^{(s)} = [\eta_y^{(s,1)}, \ldots, \eta_y^{(s,k)}] \in \mathbb{R}^k$ be the instantiations of Gaussian η_y in the k samples for the s^{th} batch. By the structural equations, we know that $B^{(s)} = X^{(s)} A + \eta^{(s)}$. So,

$$\begin{aligned}\widetilde{A}^{(s)} &= ((X^{(s)})^\top X^{(s)})^{-1} (X^{(s)})^\top B \\ &= ((X^{(s)})^\top X^{(s)})^{-1} (X^{(s)})^\top (X^{(s)} A + \eta^{(s)}) \\ &= A + ((X^{(s)})^\top X^{(s)})^{-1} (X^{(s)})^\top \eta^{(s)}\end{aligned}$$

By Fact A.3, we can express $X^{(s)} = G^{(s)} L^\top$ where matrix $G^{(s)} \in \mathbb{R}^{k \times p}$ is a random matrix with i.i.d. $N(0,1)$ entries. So, we see that

$$L^\top \Delta = \frac{1}{b} \sum_{s=1}^{b} ((G^{(s)})^\top G^{(s)})^{-1} (G^{(s)})^\top \eta^{(s)}$$

For any $i \in [p]$, Fact A.2 tells us that

$$\left(L^\top \Delta\right)_i = \frac{1}{b} \sum_{s=1}^{b} \left(((G^{(s)})^\top G^{(s)})^{-1} (G^{(s)})^\top \eta^{(s)}\right)_i \sim N\left(0, \frac{\sigma_y^2}{b^2} \sum_{s=1}^{b} \left\| \left(((G^{(s)})^\top G^{(s)})^{-1} (G^{(s)})^\top\right)_i \right\|^2\right)$$

We can upper bound each $\left\|\left(((G^{(s)})^\top G^{(s)})^{-1}(G^{(s)})^\top\right)_i\right\|$ term as follows:

$$\left\|\left(((G^{(s)})^\top G^{(s)})^{-1}(G^{(s)})^\top\right)_i\right\| \leq \left\|((G^{(s)})^\top G^{(s)})^{-1}(G^{(s)})^\top\right\| \leq \left\|((G^{(s)})^\top G^{(s)})^{-1}\right\| \cdot \left\|G^{(s)}\right\|$$

When $k \geq 4p$, Lemma A.7 tells us that $\Pr\left(\left\|((G^{(s)})^\top G^{(s)})^{-1}\right\| \geq \frac{4}{k}\right) \leq \exp\left(-\frac{k}{32}\right)$. Meanwhile, Lemma A.5 tells us that $\Pr\left(\left\|G^{(s)}\right\| \geq 2(\sqrt{k}+\sqrt{p})\right) \leq \sqrt{p}\cdot\exp\left(-C\cdot k\right)$ for some universal constant C. Let $\mathcal{E}$ be the event that $\left\|\left(((G^{(s)})^\top G^{(s)})^{-1}(G^{(s)})^\top\right)_i\right\| < \frac{8(\sqrt{k}+\sqrt{p})}{k}$ for any $s \in [b]$. Applying union bound with the conclusions from Lemma A.7 and Lemma A.5, we have

$$\begin{aligned}\Pr\left(\overline{\mathcal{E}}\right) &= \Pr\left(\exists s \in [b], \left\|\left(((G^{(s)})^\top G^{(s)})^{-1}(G^{(s)})^\top\right)_i\right\| \geq \frac{8(\sqrt{k}+\sqrt{p})}{k}\right)\\ &\leq b\cdot\exp\left(-\frac{k}{32}\right) + b\cdot\sqrt{p}\cdot\exp\left(-C\cdot k\right)\end{aligned}$$

Conditioned on event $\mathcal{E}$, standard Gaussian tail bounds (e.g. see Fact A.1) give us

$$\begin{aligned}\Pr\left(\left|L^\top\Delta\right|_i > \sigma_y\cdot\sqrt{\frac{\varepsilon}{nd_{avg}}} \;\middle|\; \mathcal{E}\right) &\leq \exp\left(-\frac{\sigma_y^2\cdot\frac{\varepsilon}{nd_{avg}}}{2\cdot\frac{\sigma_y^2}{b^2}\sum_{s=1}^{b}\left\|\left(((G^{(s)})^\top G^{(s)})^{-1}(G^{(s)})^\top\right)_i\right\|^2}\right)\\ &\leq \exp\left(-\frac{\varepsilon\cdot b\cdot k^2}{128\cdot n\cdot d_{avg}\cdot(\sqrt{k}+\sqrt{p})^2}\right)\\ &\leq \exp\left(-\frac{\varepsilon\cdot b\cdot k}{512\cdot n\cdot d_{avg}}\right)\end{aligned}$$

where the second last inequality is because of event $\mathcal{E}$ and the last inequality is because $(\sqrt{k}+\sqrt{p})^2 \leq (2\sqrt{k})^2 = 4k$, since $k \geq p$.

Thus, applying a union bound over all p entries of $L^\top\Delta$ and accounting for $\Pr(\overline{\mathcal{E}})$, we have

$$\begin{aligned}&\Pr\left(\left\|L^\top\Delta\right\| > \sigma_y\cdot\sqrt{\frac{\varepsilon p}{nd_{avg}}}\right)\\ \leq\ &\Pr\left(\left\|L^\top\Delta\right\| > \sigma_y\cdot\sqrt{\frac{\varepsilon p}{nd_{avg}}} \;\middle|\; \mathcal{E}\right) + \Pr\left(\overline{\mathcal{E}}\right)\\ \leq\ & p\cdot\exp\left(-\frac{\varepsilon\cdot b\cdot k}{512\cdot n\cdot d_{avg}}\right) + b\cdot\exp\left(-\frac{k}{32}\right) + b\cdot\sqrt{p}\cdot\exp\left(-C\cdot k\right)\end{aligned}$$

for some universal constant C.

The claim follows by observing that $\left|\Delta^\top M\Delta\right| = \left|\Delta^\top LL^\top\Delta\right| = \left\|L^\top\Delta\right\|^2$ and applying the assumptions on k and b. □

We can now establish Condition 1 of Proposition 2.1 for `BatchAvgLeastSquares`.

Lemma C.4 (Coefficient recovery guarantees of `BatchAvgLeastSquares`). *Consider Algorithm 4. With $m_1 \in \mathcal{O}\left(\frac{nd_{avg}}{\varepsilon}\left(d+\ln\left(\frac{n}{\varepsilon\delta}\right)\right)\right)$ samples, where $k \in \Omega\left(d+\ln\left(\frac{n}{\varepsilon\delta}\right)\right)$ and $b = \frac{m_1}{k}$, we recover coefficient estimates $\widehat{A}_1,\ldots,\widehat{A}_n$ such that*

$$\Pr\left(\forall i \in [n], \left|\Delta_i^\top M_i\Delta_i\right| \geq \sigma_i^2\cdot\frac{\varepsilon p_i}{nd_{avg}}\right) \leq \delta$$

The total running time is $\mathcal{O}(m_1\cdot n\cdot d_{avg}\cdot d)$.

Proof. For each $i \in [n]$, apply Lemma C.3 with $\delta' = \delta/n$, then take the union bound over all n variables.

The computational complexity for a variable with p parents is $\mathcal{O}(b\cdot p^2\cdot k) = \mathcal{O}(p^2\cdot m_1)$. Since $\max_{i\in[n]} p_i \leq d$ and $\sum_{i=1}^{n} p_i = nd_{avg}$, the total runtime is $\mathcal{O}(m_1\cdot n\cdot d_{avg}\cdot d)$. □

Theorem 3.2 follows from combining the guarantees of `BatchAvgLeastSquares` and `VarianceRecovery` (given in Lemma C.4 and Corollary 2.3 respectively) via Proposition 2.1.

Theorem 3.2 (Distribution learning using `BatchAvgLeastSquares`)**.** *Let $\varepsilon, \delta \in (0,1)$. Suppose G is a fixed DAG on n variables with in-degree at most d. Given $\mathcal{O}(nd_{avg}\varepsilon^{-2} \cdot (d + \log(n\varepsilon^{-1}\delta^{-1})))$ samples from an unknown Bayesian network $\mathcal{P}$ over G, if we use `BatchAvgLeastSquares` for coefficient recovery in Algorithm 1, then with probability at least $1-\delta$, we recover a Bayesian network $\mathcal{Q}$ over G in $\mathcal{O}(n^2 d_{avg}^2 d\varepsilon^{-2} \cdot (d + \log(n\varepsilon^{-1}\delta^{-1})))$ time such that $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$.*

Proof of Theorem 3.2. We will show sample and time complexities before giving the proof for the $\mathrm{d_{TV}}$ distance. Let $m_1 \in \mathcal{O}\left(\frac{nd_{avg}}{\varepsilon}\left(d + \ln\left(\frac{n}{\varepsilon\delta}\right)\right)\right)$ and $m_2 \in \mathcal{O}\left(\frac{nd_{avg}}{\varepsilon}\log\left(\frac{n}{\delta}\right)\right)$. Then, the total number of samples needed is $m = m_1 + m_2 \in \mathcal{O}\left(\frac{nd_{avg}}{\varepsilon}\left(d + \ln\left(\frac{n}{\varepsilon\delta}\right)\right)\right)$. `BatchAvgLeastSquares` runs in $\mathcal{O}(m_1 n d_{avg} d)$ time while `VarianceRecovery` runs in $\mathcal{O}\left(\frac{n^2 d_{avg}^2}{\varepsilon}\log\left(\frac{1}{\delta}\right)\right)$ time. Therefore, the overall running time is $\mathcal{O}\left(\frac{n^2 d_{avg}^2 d}{\varepsilon}\left(d + \ln\left(\frac{n}{\varepsilon\delta}\right)\right)\right)$.

By Lemma C.4, `BatchAvgLeastSquares` recovers coefficients $\widehat{A}_1, \ldots, \widehat{A}_n$ such that

$$\Pr\left(\forall i \in [n], \left|\Delta_i^\top M_i \Delta_i\right| \geq \sigma_i^2 \cdot \frac{\varepsilon p_i}{nd_{avg}}\right) \leq \delta$$

By Corollary 2.3 and using the recovered coefficients from `BatchAvgLeastSquares`, `VarianceRecovery` recovers variance estimates $\widehat{\sigma}_i^2$ such that

$$\Pr\left(\forall i \in [n], \left(1 - \sqrt{\frac{\varepsilon p_i}{nd_{avg}}}\right) \cdot \sigma_i^2 \leq \widehat{\sigma}_i^2 \leq \left(1 + \sqrt{\frac{\varepsilon p_i}{nd_{avg}}}\right) \cdot \sigma_i^2\right) \geq 1 - \delta$$

As our estimated parameters satisfy Condition 1 and Condition 2, Proposition 2.1 tells us that $\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q}) \leq 3\varepsilon$. Thus, $\mathrm{d_{TV}}(\mathcal{P}, \mathcal{Q}) \leq \sqrt{\mathrm{d_{KL}}(\mathcal{P}, \mathcal{Q})/2} \leq \sqrt{3\varepsilon/2}$. The claim follows by setting $\varepsilon' = \sqrt{3\varepsilon/2}$ throughout. □

D Appendix material for Section 4

In this section, we give the missing proofs of Lemma 4.1, Lemma 4.2, Corollary 4.3, and Theorem 4.4. As the proof of Theorem 4.4 requires a few steps, we dedicate a subsection Appendix D.1 to it for a cleaner presentation.

Lemma 4.1. *Consider the matrix equation $AB = E$ where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times 1}$, and $E \in R^{n \times 1}$ such that entries of A and E are independent Gaussians, elements in each* column *of A have the same variance, and all entries in E have the same variance. That is, $A_{\cdot,j} \sim N(0, \sigma_i^2)$ and $E_i \sim N(0, \sigma_{n+1}^2)$. Then, we have that $B_i \sim \frac{\sigma_{n+1}}{\sigma_i} \cdot \mathrm{Cauchy}(0,1)$ for all $i \in [n]$.*

Proof of Lemma 4.1. As the event that A is singular has measure zero, we can write $B = A^{-1}E$. By Cramer's rule,

$$A^{-1} = \frac{1}{\det(A)} \cdot \mathrm{adj}(A) = \frac{1}{\det(A)} \cdot C^\top$$

where $\det(A)$ is the determinant of A, $\mathrm{adj}(A)$ is the adjugate/adjoint matrix of A, and C is the cofactor matrix of A. Recall that the $\det(A)$ can defined with respect to elements in C: For any *column* $i \in [n]$,

$$\det(A) = A_{1,i} \cdot C_{1,i} + A_{2,i} \cdot C_{2,i} + \ldots + A_{n,i} \cdot C_{n,i}$$

So, $\det(A) \sim N\left(0, \sigma_i^2\left(C_{1,i} + \ldots + C_{n,i}\right)\right)$. Thus, for any $i \in [n]$,

$$B_i = \left(\frac{1}{\det(A)} C^\top E\right)_i \sim \frac{N\left(0, \sigma_{n+1}^2\left(C_{1,i} + \ldots + C_{n,i}\right)\right)}{N\left(0, \sigma_i^2\left(C_{1,i} + \ldots + C_{n,i}\right)\right)} = \frac{\sigma_{n+1}}{\sigma_i} \cdot \mathrm{Cauchy}(0,1)$$

□

Lemma 4.2. *Consider a batch estimate $\widetilde{A}$ from Algorithm 5 and define $\Delta = \widetilde{A} - A$. Then, $L^\top \Delta$ is entry-wise distributed as $\sigma_y \cdot \mathrm{Cauchy}(0,1)$. Note that the entries of $L^\top \Delta$ may be correlated in general.*

Proof of Lemma 4.2. Observe that each row of X is an independent sample drawn from a multivariate Gaussian $N(0, M)$. By denoting $\eta = \left[\eta_y^{(1)}, \ldots, \eta_y^{(p)}\right]^\top$ as the p samples of η_y, we can write $X\widetilde{A} = XA + \eta$ and thus $X\Delta = \eta$ by rearranging terms. By Fact A.3, we can express $X = GL^\top$ where matrix $G \in \mathbb{R}^{p \times p}$ is a random matrix with i.i.d. $N(0,1)$ entries. By substituting $X = GL^\top$ into $X\Delta = \eta$, we have $L^\top \Delta = G^{-1}\eta$.[13]

By applying Lemma 4.1 with the following parameters: $A = G, B = L^\top \Delta, E = \eta$, we conclude that each entry of $L^\top \Delta$ is distributed as $\sigma_y \cdot \mathrm{Cauchy}(0,1)$. However, note that *these entries are generally correlated.* □

Corollary 4.3. *Consider a batch estimate $\widetilde{A}$ from Algorithm 5 and define $\Delta = \widetilde{A} - A$. If the Bayesian network is a polytree, then $\Delta_i = (\widetilde{A} - A)_i \sim \frac{\sigma_y}{\sigma_i} \cdot \mathrm{Cauchy}(0,1)$ for all $i \in [n]$.*

Proof of Corollary 4.3. Observe that each row of X is an independent sample drawn from a multivariate Gaussian $N(0, M)$. By denoting $\eta = \left[\eta_y^{(1)}, \ldots, \eta_y^{(p)}\right]^\top$ as the p samples of η_y, we can write $X\widetilde{A} = XA + \eta$ and thus $X\Delta = \eta$ by rearranging terms. Since the parents of any variable in a polytree are not correlated, each element in the i^{th} column of X is a $N(0, \sigma_i^2)$ Gaussian random variable.

By applying Lemma 4.1 with the following parameters: $A = X, B = \Delta E = \eta$, we conclude that $\Delta_i = (\widetilde{A} - A)_i \sim \frac{\sigma_y}{\sigma_i} \cdot \mathrm{Cauchy}(0,1)$. □

D.1 Proof of Theorem 4.4

We will first prove guarantees for an arbitrary variable and then take union bound over n variables.

Lemma D.1. *Consider Algorithm 7. Fix an arbitrary variable of interest Y with p parents, parameters (A, σ_y), and associated covariance matrix M. With $k = \frac{8nd_{avg}}{\varepsilon}\log\left(\frac{2}{\delta}\right)$ samples, we recover coefficient estimates $\widehat{A}$ such that*

$$\Pr\left(\left|\Delta^\top M \Delta\right| \le \sigma_y^2 \cdot \frac{\varepsilon p}{nd_{avg}}\right) \ge 1 - \delta$$

Proof. Since $M = LL^\top$, it suffices to bound $\|L^\top \Delta\|$. Lemma 4.2 tells us that each entry of the vector $L^\top\Delta$ is the median of k copies of $\mathrm{Cauchy}(0,1)$ random variables multiplied by σ_y. Setting $k = \frac{8nd_{avg}}{\varepsilon}\log\left(\frac{2}{\delta}\right)$ and $0 < \tau = \sqrt{\varepsilon/(nd_{avg})} < 1$ in Lemma A.9, we see that

$$\Pr\left(\text{median of } k \text{ i.i.d. } \mathrm{Cauchy}(0,1) \text{ random variables} \notin \left[-\sqrt{\frac{\varepsilon}{nd_{avg}}}, \sqrt{\frac{\varepsilon}{nd_{avg}}}\right]\right) \le \delta$$

That is, each entry of $L^\top\Delta$ has absolute value at most $\sigma_y \cdot \sqrt{\frac{\varepsilon}{nd_{avg}}}$. By summing across all p entries of $L^\top\Delta$, we see that

$$|\Delta^\top M \Delta| = |\Delta^\top L L^\top \Delta| = \|L^\top \Delta\|^2 \le p \cdot \sigma_y^2 \cdot \frac{\varepsilon}{nd_{avg}} = \sigma_y^2 \cdot \frac{\varepsilon p}{nd_{avg}}$$

□

We can now establish Condition 1 of Proposition 2.1 for `CauchyEstTree`.

Lemma D.2. *Consider Algorithm 7. Suppose the Bayesian network is a polytree. With $m_1 \in \mathcal{O}\left(\frac{nd_{avg}d}{\varepsilon}\log\left(\frac{n}{\delta}\right)\right)$ samples, we recover coefficient estimates $\widehat{A}_1, \ldots, \widehat{A}_n$ such that*

$$\Pr\left(\forall i \in [n], \left|\Delta_i^\top M_i \Delta_i\right| \ge \sigma_i^2 \cdot \frac{\varepsilon p_i}{nd_{avg}}\right) \le \delta$$

The total running time is $\mathcal{O}\left(\frac{n^2 d_{avg}^2 d^{\omega-1}}{\varepsilon}\log\left(\frac{n}{\delta}\right)\right)$ where ω is the matrix multiplication exponent.

[13] Note that event that G is singular has measure 0.

Proof. For each $i \in [n]$, apply Lemma D.1 with $\delta' = \delta/n$ and $m_1 = \frac{8nd_{avg}}{\varepsilon} \log\left(\frac{2n}{\delta}\right)$, then take the union bound over all n variables.

The runtime of Algorithm 5 is the time to find the inverse of a $p \times p$ matrix, which is $\mathcal{O}(p^{\omega})$ for some $2 < \omega < 3$. Therefore, the computational complexity for a variable with p parents is $\mathcal{O}(p^{\omega-1} \cdot m_1)$. Since $\max_{i\in[n]} p_i \leq d$ and $\sum_{i=1}^{n} p_i = nd_{avg}$, the total runtime is $\mathcal{O}(m_1 \cdot n \cdot d_{avg} \cdot d^{\omega-2})$. □

We are now ready to prove Theorem 4.4.

Theorem 4.4 follows from combining the guarantees of `CauchyEstTree` and `VarianceRecovery` (given in Lemma D.2 and Corollary 2.3 respectively) via Proposition 2.1.

Theorem 4.4 (Distribution learning using `CauchyEstTree`)**.** *Let $\varepsilon, \delta \in (0,1)$. Suppose G is a fixed DAG on n variables with degree at most d. Given $\mathcal{O}(nd_{avg}d\varepsilon^{-1} \cdot \log(n\delta^{-1}))$ samples from an unknown Bayesian network $\mathcal{P}$ over G, if we use `CauchyEstTree` for coefficient recovery in Algorithm 1, then with probability at least $1-\delta$, we recover a Bayesian network $\mathcal{Q}$ over G such that $\mathrm{d_{TV}}(\mathcal{P},\mathcal{Q}) \leq \varepsilon$ in $\mathcal{O}(n^2 d_{avg}^2 d^{\omega-1}\varepsilon^{-1} \cdot \log(n\delta^{-1}))$ time.*

Proof of Theorem 4.4. We will show sample and time complexities before giving the proof for the $\mathrm{d_{TV}}$ distance. Let $m_1 \in \mathcal{O}\left(\frac{nd_{avg}d}{\varepsilon}\log\left(\frac{n}{\delta}\right)\right)$ and $m_2 \in \mathcal{O}\left(\frac{nd_{avg}}{\varepsilon}\log\left(\frac{n}{\delta}\right)\right)$. Then, the total number of samples needed is $m = m_1 + m_2 \in \mathcal{O}\left(\frac{nd_{avg}d}{\varepsilon}\log\left(\frac{n}{\delta}\right)\right)$. `CauchyEstTree` runs in $\mathcal{O}\left(\frac{n^2 d_{avg}^2 d^{\omega-1}}{\varepsilon}\log\left(\frac{n}{\delta}\right)\right)$ time while `VarianceRecovery` runs in $\mathcal{O}\left(\frac{n^2 d_{avg}^2}{\varepsilon}\log\left(\frac{1}{\delta}\right)\right)$ time, where ω is the matrix multiplication exponent. Therefore, the overall running time is $\mathcal{O}\left(\frac{n^2 d_{avg}^2 d^{\omega-1}}{\varepsilon}\log\left(\frac{n}{\delta}\right)\right)$.

By Lemma D.2, `CauchyEstTree` recovers coefficients $\widehat{A}_1, \ldots, \widehat{A}_n$ such that

$$\Pr\left(\forall i \in [n], \left|\Delta_i^\top M_i \Delta_i\right| \geq \sigma_i^2 \cdot \frac{\varepsilon p_i}{nd_{avg}}\right) \leq \delta$$

By Corollary 2.3 and using the recovered coefficients from `CauchyEstTree`, `VarianceRecovery` recovers variance estimates $\widehat{\sigma}_i^2$ such that

$$\Pr\left(\forall i \in [n], \left(1 - \sqrt{\frac{\varepsilon p_i}{nd_{avg}}}\right) \cdot \sigma_i^2 \leq \widehat{\sigma}_i^2 \leq \left(1 + \sqrt{\frac{\varepsilon p_i}{nd_{avg}}}\right) \cdot \sigma_i^2\right) \geq 1-\delta$$

As our estimated parameters satisfy Condition 1 and Condition 2, Proposition 2.1 tells us that $\mathrm{d_{KL}}(\mathcal{P},\mathcal{Q}) \leq 3\varepsilon$. Thus, $\mathrm{d_{TV}}(\mathcal{P},\mathcal{Q}) \leq \sqrt{\mathrm{d_{KL}}(\mathcal{P},\mathcal{Q})/2} \leq \sqrt{3\varepsilon/2}$. The claim follows by setting $\varepsilon' = \sqrt{3\varepsilon/2}$ throughout. □

E Appendix material for Section 5

In this section, we give the missing proofs of Theorem 5.1 and Theorem 5.2.

Theorem 5.1. *Given samples from a n-fold Gaussian product distribution P, learning a $\widehat{P}$ such that in $\mathrm{d_{TV}}(P,\widehat{P}) = O(\varepsilon)$ with success probability 2/3 needs $\Omega(n\varepsilon^{-2})$ samples in general.*

Proof of Theorem 5.1. Let $\mathcal{C} \subseteq \{0,1\}^n$ be a set with the following properties: 1) $|\mathcal{C}| = 2^{\Theta(n)}$ and 2) every $i \neq j \in \mathcal{C}$ have a Hamming distance $\Theta(n)$. Existence of such a set is well-known. We create a class of Gaussian product distributions $\mathcal{P}_{\mathcal{C}}$ based on $\mathcal{C}$ as follows. For each $s \in \mathcal{C}$, we define a distribution $P_s \in \mathcal{P}_{\mathcal{C}}$ such that if the i-th bit of s is 0, we use the distribution $N(0,1)$ for the i-th component of P_s; else if the i-th bit is 1, we use the distribution $N(0, 1+\frac{\epsilon}{\sqrt{n}})$. Then for any $P_s \neq P_t$, $\mathrm{d_{KL}}(P_s,P_t) = O(\epsilon^2)$. Fano's inequality tells us that guessing a random distribution from $\mathcal{P}_{\mathcal{C}}$ correctly with 2/3 probability needs $\Omega(n\epsilon^{-2})$ samples.

Fact A.13 tells us that for any $P_s \neq P_t \in \mathcal{P}_{\mathcal{C}}$, $\mathrm{d_{TV}}(P_s,P_t) \geq c_3\epsilon$ for some constant c_3. Consider any algorithm for learning a random distribution $P = N(0,\Sigma)$ from $\mathcal{P}_{\mathcal{C}}$ in $\mathrm{d_{TV}}$-distance at most $c_4\epsilon$ for a sufficiently small constant c_4. Let the learnt distibution be $\widehat{P} = N(0,\widehat{\Sigma})$. Due to triangle inequality, Fact A.13, and an appropriate choice of c_4, P must be the unique distribution from $\mathcal{P}_{\mathcal{C}}$ satisfying $||\widehat{\Sigma}^{-1}\Sigma - I||_F \leq c_5\epsilon$ for an appropriate choice of c_5.

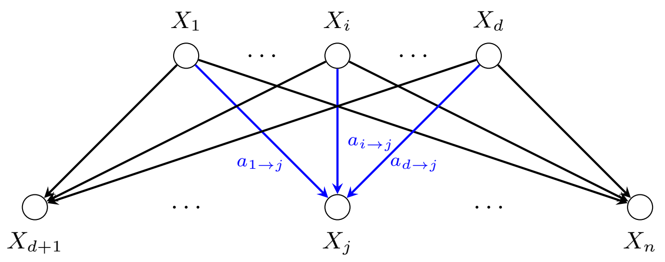


Figure 2: Bipartite DAG on n vertices with maximum degree d. For $i \in \{1,\dots,d\}$, $X_i = \eta_i$ where $\eta_i \sim N(0,1)$. For $j \in \{d+1,\dots,n\}$, $X_j = \eta_j + \sum_{i=1}^{d} a_{i\to j} X_i$ where $\eta_j \sim N(0,1)$. Furthermore, each X_j is associated with a d-bit string and each coefficients $a_{1\to j},\dots,a_{d\to j}$ is either $\frac{1}{\sqrt{d(n-d)}}$ or $\frac{1+\epsilon}{\sqrt{d(n-d)}}$, depending on the i^{th} bit in the associated d-bit string.

We can find this unique distribution by computing $||\widehat{\Sigma}^{-1}\Sigma' - I||_F$ for every covariance matrix Σ' from $\mathcal{P}_{\mathcal{C}}$ and guess the random distribution correctly. Hence, the lower-bound follows. $\square$

Now, we present the lower-bound for learning general Bayes nets.

Theorem 5.2. *For any $0 < \varepsilon < 1$ and n, d such that $d \le n/2$, there exists a DAG G over $[n]$ of in-degree d such that learning a Gaussian Bayesian network $\widehat{P}$ on G such that $\mathrm{d_{TV}}(P,\widehat{P}) \le \varepsilon$ with success probability 2/3 needs $\Omega(nd\varepsilon^{-2})$ samples in general.*

Let $\mathcal{C} \subseteq \{0,1\}^d$ be a set with the following properties: 1) $|\mathcal{C}| = 2^{\Theta(d)}$ and 2) every $i \neq j \in \mathcal{C}$ have a Hamming distance $\Theta(d)$. Existence of such a set is well-known. We define a class of distributions $\mathcal{P}_{\mathcal{C}}$ based on $\mathcal{C}$ and the graph G shown in Fig. 2 as follows. Each vertex of each distribution in $\mathcal{P}_{\mathcal{C}}$ has a $N(0,1)$ noise, and hence no learning is required for the noises. Each coefficient $a_{i\to j}$ takes one of two values $\left\{\frac{1}{\sqrt{d(n-d)}}, \frac{1+\epsilon}{\sqrt{d(n-d)}}\right\}$ corresponding to bits $\{0,1\}$ respectively. For each $s \in \mathcal{C}$, we define A_s to be the vector of coefficients corresponding to the bit-pattern of s as above. We have $2^{\Theta(d)}$ possible bit-patterns, which we use to define each conditional probability $(X_i \mid X_1, X_2, \dots, X_d)$. Then, we have a class $\mathcal{Q}_{\mathcal{P}}$ of $|\mathcal{C}|^{(n-d)}$ distributions for the overall Bayes net. We prune some of the distributions to get the desired subclass $\mathcal{P}_{\mathcal{C}} \subseteq \mathcal{Q}_{\mathcal{C}}$, such that $\mathcal{P}_{\mathcal{C}}$ is the largest-sized subset with any pair of distributions in the subset differing in at least $(n-d)/2$ bit-patterns (out of $(n-d)$ many) for the $(X_i \mid X_1, X_2, \dots, X_d)$'s.

Claim E.1. $|\mathcal{P}_{\mathcal{C}}| \ge 2^{\Theta(d(n-d))}$.

Proof. Let $\ell = |\mathcal{C}| = 2^{\Theta(d)}$ and $m = n - d$. Then, there are $N = \ell^m$ possible coefficient-vectors/distributions in $\mathcal{Q}_{\mathcal{C}}$. We create an undirectred graph G consisting of a vertex for each coefficient, and edges between any pair of vertices differing in at least $m/2$ bit-patterns. Note that G is r-regular for $r = \binom{m}{0.5m}(\ell-1)^{0.5m} + \cdots + \binom{m}{0.5m+j}(\ell-1)^{0.5m+j} + \cdots + (\ell-1)^m$. Turan's theorem says that there is a clique of size $\alpha = (1 - \frac{r}{N})^{-1}$. We define $\mathcal{P}_{\mathcal{C}}$ to be the vertices of this clique.

To show that α is as large as desired, it suffices to show $N - r \le N/2^{\Theta(dm)}$. The result follows by noting $N - r = 1 + \binom{m}{1}(\ell-1) + \cdots + \binom{m}{j}(\ell-1)^j + \cdots + \binom{m}{0.5m}(\ell-1)^{0.5m} \le m \cdot (4(\ell-1))^{0.5m}$. $\square$

We also get for any two distributions P_a, P_b in this model with the coefficients a, b respectively, $\mathrm{d_{KL}}(P_a, P_b) = \frac{1}{2}||a-b||_2^2$ from (1). Hence, any pair of $\mathcal{P}_{\mathcal{C}}$ has $\mathrm{d_{KL}} = \Theta(\epsilon^2)$. Then, Fano's inequality tells us that given a random distribution $P \in \mathcal{P}_{\mathcal{C}}$, it needs at least $\Omega(d(n-d)\epsilon^{-2})$ samples to guess the correct one with 2/3 probability. We need the following fact about the $\mathrm{d_{TV}}$-distance among the members of $\mathcal{P}_{\mathcal{C}}$.

Claim E.2. *Let $P_a, P_b \in \mathcal{P}_{\mathcal{C}}$ be two distinct distributions (i.e. $P_a \neq P_b$) with coefficient-vectors a, b respectively. Then, $\mathrm{d_{TV}}(P_a, P_b) \in \Theta(\epsilon)$.*

Proof. By Pinsker's inequality, we have $\mathrm{d_{TV}}(P_a, P_b) \in \mathcal{O}(\epsilon)$. Here we show $\mathrm{d_{TV}}(P_a, P_b) \in \Omega(\epsilon)$. Let $n - d = m$. By construction, a and b differ in $m' \ge m/2$ conditional probabilities. Let $a' \subseteq a, b' \subseteq b$ be the coefficient-vectors on the coordinates where they differ. Let $P_{a'}, \Sigma_{a'}$ and $P_{b'}, \Sigma_{b'}$ be the corresponding marginal distributions on $(m'+d)$ variables, and their covariance matrices. In the following, we show $||\Sigma_{a'}^{-1}\Sigma_{b'} - I||_F = \Omega(\epsilon)$. This proves the claim from Fact A.13.

Let $M_{a'} = \begin{bmatrix} 0_{m' \times m'} & 0_{m' \times d} \\ A_{d \times m'} & 0_{d \times d} \end{bmatrix}$ be the adjacency matrix for P_a, where the sources appear last in the rows and columns and in the matrix A, each $A_{ij} \in \{\frac{1}{\sqrt{md}}, \frac{1+\epsilon}{\sqrt{md}}\}$ denote the coefficient from source i to sink j. Similarly, we define $M_{b'}$ using a coefficient matrix $B_{d \times m'}$. Let $\{A_i : 1 \leq i \leq m'\}$ and $\{B_i : 1 \leq i \leq m'\}$ denote the columns of A and B. Then for every i, A_i and B_i differ in at least $\Theta(d)$ coordinates by construction.

By definition $\Sigma_{b'} = \begin{bmatrix} \bullet & B^T \\ B & I_{d \times d} \end{bmatrix}$ and $\Sigma_{a'}^{-1} = \begin{bmatrix} I_{m' \times m'} & -A^T \\ -A & \bullet \end{bmatrix}$, where $\bullet$ corresponds to certain matrices not relevant to our discussion[14]. Let $J = \Sigma_{a'}^{-1}\Sigma_{b'} = \begin{bmatrix} \bullet & X_{m' \times d} \\ \bullet & \bullet \end{bmatrix}$. It can be checked that $X_{ij} = (B_i(j) - A_i(j))$ for every $1 \leq i \leq m'$ and $m'+1 \leq j \leq m'+d$. Now for every i, each of the $\Theta(d)$ places that A_i and B_i differ, $X_{ij} \in \frac{\pm\epsilon}{\sqrt{md}}$. Hence, their total contribution in $||J - I||_F^2 = \Omega(\epsilon^2)$. □

Proof of Theorem 5.2. Consider any algorithm which learns a random distribution $P = N(0, \Sigma)$ from $\mathcal{P}_\mathcal{C}$ in $\mathrm{d_{TV}}$ distance $c_3\epsilon$, for a small enough constant c_3. Let the learnt distribution be $\widehat{P} = N(0, \widehat{\Sigma})$. Then, from Fact A.13, and triangle inequality of $\mathrm{d_{TV}}$, only the unique distribution P with $\Sigma' = \Sigma$ would satisfy $||\widehat{\Sigma}^{-1}\Sigma' - I||_F \leq c_4\epsilon$ for every covariance matrix Σ' from $\mathcal{P}_\mathcal{C}$, for an appropriate choice of c_4. This would reveal the random distribution, hence the lower-bound follows. □

F Detailed experiment results

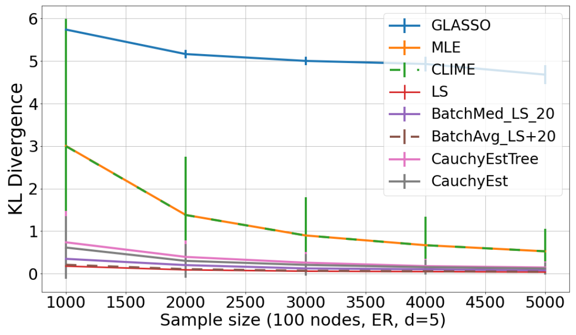
Algorithms The algorithms used in our experiments are as follows: `LeastSquares`, `BatchAvgLeastSquares`, `BatchMedLeastSquares`, `CauchyEstTree`, `CauchyEst`, Graphical Lasso [FHT08], MLE (empirical) estimator, and CLIME [CLL11]. Specifically, we use `BatchAvg_LS+`x and `BatchMed_LS+`x to represent the `BatchAvgLeastSquares` and `BatchMedLeastSquares` algorithms respectively with a batch size of $p + x$ at each node. p is the number of parents of that node.

Synthetic data Fig. 3 compares the KL divergence between the ground truth and our learned distribution over 100 variables between the eight estimators mentioned above. The first three estimators are for undirected graph structure learning. For this reason, we are not using Eq. (1) but the common equation in [Duc07, page 13] for calculating the KL divergence between multivariate Gaussian distributions. Fig. 3(a) and Fig. 3(b) shows the results on ER graphs while Fig. 3(c) shows the results for random tree graphs. The performances of MLE and CLIME are very close to each other, thus are overlapped in Fig. 3(a). In figure Fig. 3(b), we take a closer look at the results from Fig. 3(a) for `LeastSquares`, `BatchMedLeastSquares`, `BatchAvgLeastSquares`, `CauchyEstTree`, and `CauchyEst` estimators for a clear comparison. In our experiments, we find that the latter five outperform the GLASSO, CLIME and MLE (empirical) estimators. With a degree 5 ER graph, `CauchyEst` performs better than `CauchyEstTree`, while `LeastSquares` performs best. In our experiments for our random tree graphs with in-degree 1 (see Fig. 3(c), we find that the performances between `CauchyEstTree` and `CauchyEst` are very close to each other and their plots overlap.

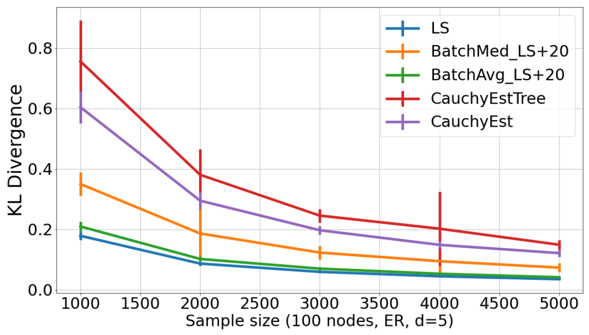
In our experiments, `CauchyEst` outperforms `CauchyEstTree` when the $G(n, p)$ graph is generated with a higher degree parameter d (e.g. $d > 5$) and the resultant graph is unlikely to yield a polytree structure.

Real-world datasets We also evaluated our algorithms on four real Gaussian Bayesian networks from R package `bnlearn` [Scu09]. The **ECOLI70** graph provided by [SS05] contains 46 nodes and 70 edges. The **MAGIC-NIAB** graph from [SHBM14] contains 44 nodes and 66 edges. The **MAGIC-IRRI** graph contains 64 nodes and 102 edges, and the **ARTH150** [ORS07] graph contains 107 nodes and 150 edges. Experimental results in Fig. 4 show that the error for `LeastSquares` is smaller than `CauchyEst` and `CauchyEstTree` for all the above datasets.

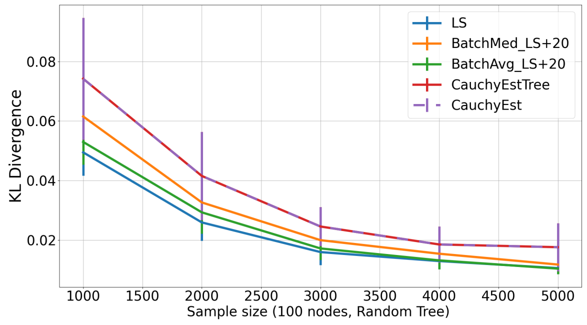
[14]The missing (symmetric) submatrix of $\Sigma_{b'}$ is the identity matrix added with the entries $\langle B_i, B_j \rangle$. The missing (symmetric) submatrix of $\Sigma_{a'}^{-1}$ is the identity matrix added with the inner products of the rows of A.



(a) Eight algorithms evaluated on ER graph with $d = 5$

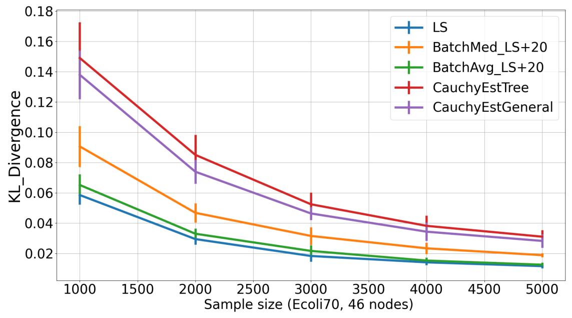


(b) A closer look at some of the algorithms in the plot of Fig. 3(a)

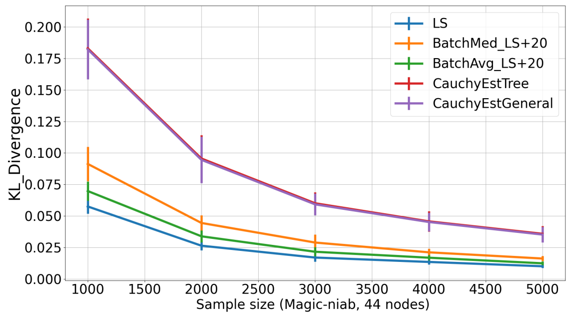


(c) A closer look at some of the algorithms evaluated on random tree graphs

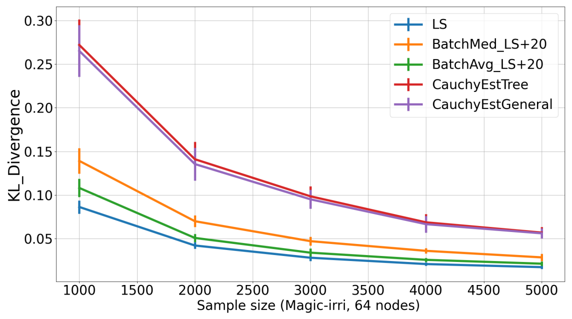
Figure 3: Experiment on well-conditioned uncontaminated data



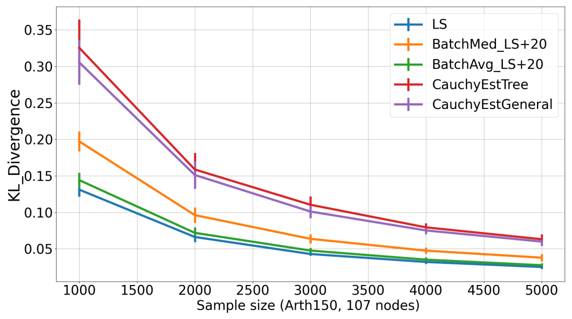
(a) Ecoli70 46 nodes



(b) Magic-niab 44 nodes

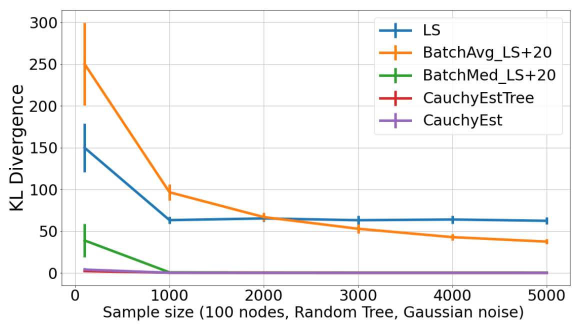


(c) Magic-irri 64 nodes

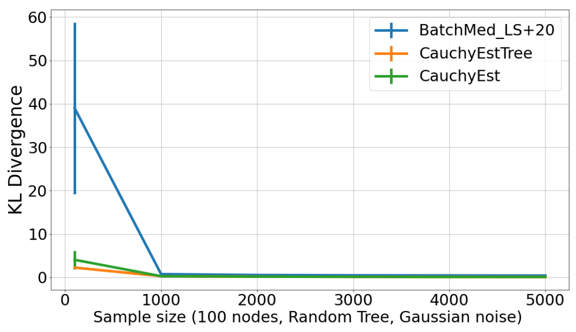


(d) Arth150 107 nodes

Figure 4: Experiment results over bnlearn real graph

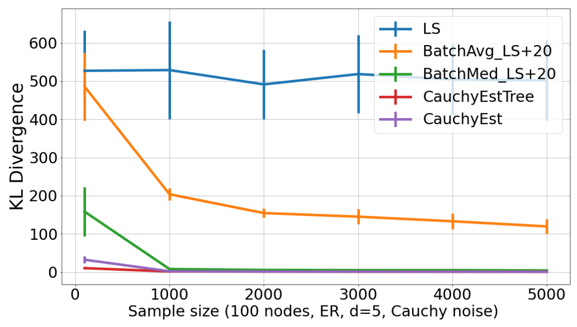


(a) all algorithms

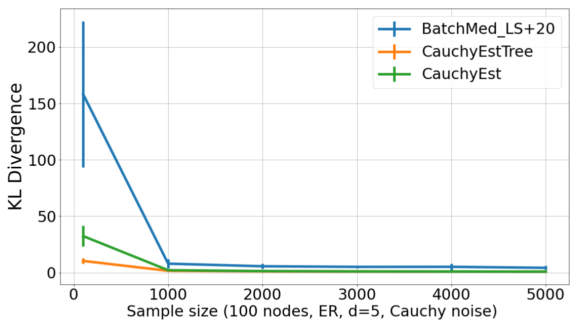


(b) BatchMed_LS, CauchyEst, and CauchyEstTree

Figure 5: Experiment results on contaminated data (random tree with Gaussian noise)



(a) all algorithms



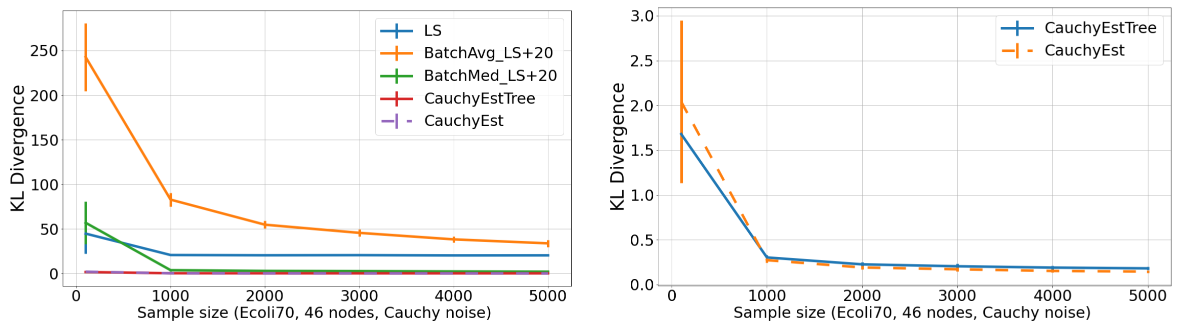
(b) BatchMed_LS, CauchyEst, and CauchyEstTree

Figure 6: Experiment results on contaminated data (ER graph with Cauchy noise)

Contaminated synthetic data The contaminated synthetic data is generated in the following way: we randomly choose 5% samples with 5 nodes to be contaminated from the well-conditioned data over $n = 100$ node graphs. The well-conditioned data has a $N(0, 1)$ noise for every variable, while the small proportion of the contaminated data has either $N(1000, 1)$ or Cauchy$(1000, 1)$. In our experiments in Fig. 5 and Fig. 6, `CauchyEst`, `CauchyEstTree`, and `BatchMedLeastSquares` outperform `BatchAvgLeastSquares` and `LeastSquares` by a large margin. With more than 1000 samples, `BatchMedLeastSquares` with a batch size of $p + 20$ performs similar to `CauchyEst` and `CauchyEstTree`, but performs worse with less samples. When comparing the performance between `LeastSqures` and `BatchAvgLeastSquares` over either a random tree or a ER graph, the experiment in Fig. 5(a) based on a random tree graph shows that `LeastSqures` performs better than `BatchAvgLeastSquares` when sample size is smaller than 2000, while `BatchAvgLeastSquares` performs relatively better with more samples. Experiment results in Fig. 6(a) based on ER degree 5 graphs is slightly different from Fig. 5(a). In Fig. 6(a), `BatchAvgLeastSquares` performs better than `LeastSqures` by a large margin. Besides, we can also observe that the performances of `CauchyEst`, `CauchyEstTree`, and `BatchMedLeastSquares` are better than the above two estimators and are consistent over different types of graphs. For all five algorithms, we use the median absolute devation for robust variance recovery [Hub04] in the contaminated case only (see Algorithm 8 in Appendix).

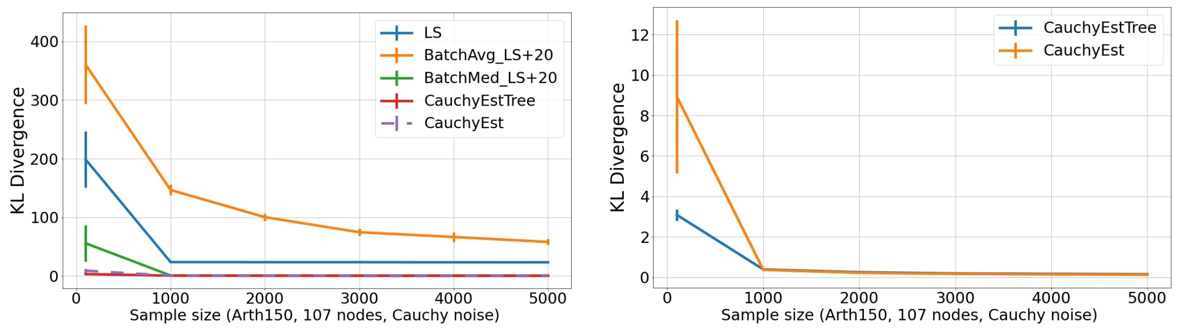
This is because both `LeastSquares` and `BatchAvgLeastSquares` use the sample covariance (of the entire dataset or its batches) in the coefficient estimators for the unknown distribution. The presence of a small proportion of outliers in the data can have a large distorting influence on the sample covariance, making them sensitive to atypical observations in the data. On the contrary, our `CauchyEstTree` and `BatchMedLeastSquares` estimators are developed using the sample median and hence are resistant to outliers in the data.

Contaminated real-world datasets To test the robustness of the real data in the contaminated condition, we manually contaminate 5% samples of 5 nodes from observational data in **ECOLI70** and **ARTH150**. The results are reported in Fig. 7 and Fig. 8. In our experiments, `CauchyEst` and `CauchyEstTree` outperforms `BatchAvgLeastSquares` and `LeastSquares` by a large margin, and therefore are stable in both contaminated and



(a) Ecoli70, 5/46 noisy nodes (b) CauchyEst and CauchyEstTree

Figure 7: Ecoli70 under contaminated condition



(a) Arth150, 5/107 noisy nodes (b) CauchyEst and CauchyEstTree

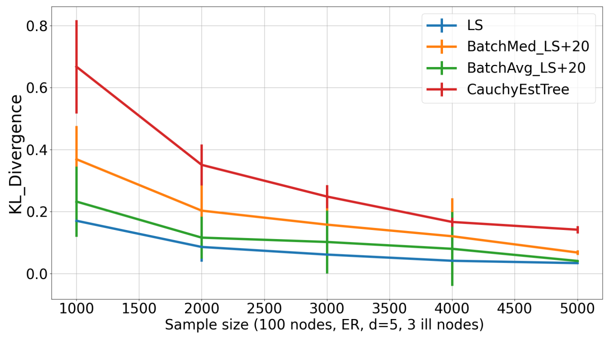
Figure 8: Arth150 under contaminated condition

well-conditioned case. Besides, note that different from the well-conditioned case, here `CauchyEstTree` performs slightly better than `CauchyEst`. This is because the Cholesky decomposition used in `CauchyEst` estimator takes contaminated-data into account.

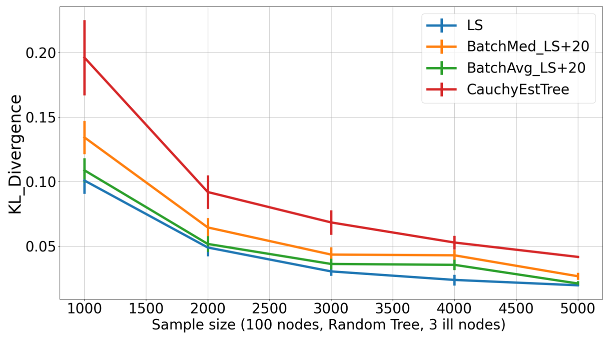
Ill-conditioned synthetic data The ill-conditioned data is generated in the following way: we classify the node sets V into either well-conditioned or ill-conditioned nodes. The well-conditioned nodes have a $N(0,1)$ noise, while ill-conditioned nodes have a $N(0,10^{-20})$ noise. In our experiments, we choose 3 ill-conditioned nodes over 100 nodes. Synthetic data is sampled from either a random tree or a Erdős Rényi (ER) model with an expected number of neighbors $d=5$. Experiments over ill-conditioned Gaussian Bayesian networks through 20 random repetitions are presented in Fig. 9. For the ill-conditioned settings, we sometimes run into numerical issues when computing the Cholesky decomposition of the empirical covariance matrix $\hat{M}$ in our `CauchyEst` estimator. Thus, we only show the comparision results between `LeastSquares`, `BatchAvgLeastSquares`, `BatchMedLeastSquares`, and `CauchyEstTree`. Here also, the error for `LeastSquares` decreases faster than the other three estimators. The performance of `BatchMedLeastSquares` is worse than `BatchAvgLeastSquares` but slightly better than `CauchyEstTree` estimator.

Agnostic Learning Our theoretical results treat the case that the data is realized by a distribution consistent with the given DAG. In this section, we explore learning of non-realizable inputs, so that there is a non-zero KL divergence between the input distribution P and any distribution consistent with the given DAG.

We conduct agnostic learning experiments by fitting a random sub-graph of the ground truth graph. To do this, we first generate a 100-node ground truth graph G, either a random tree with 4 random edges removed or a random ER graph with 9 random edges removed. Our algorithm will try to fit the data from the original Bayes net on G on the edited graph G^*. Fig. 10 reports the KL divergence learned over our five estimators. We find that `BatchAvgLeastSquares` estimator performs slightly better than all other estimators in both cases.

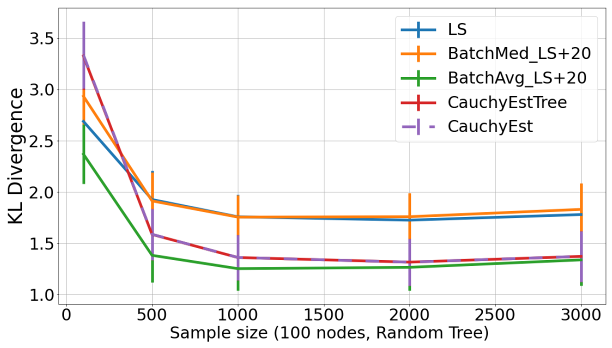


(a) ER graph, $d = 5$

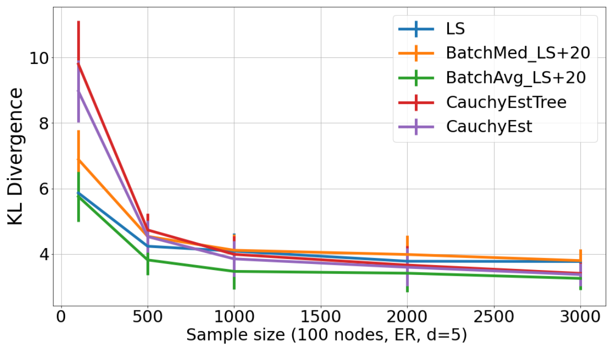


(b) Random tree

Figure 9: Experiment results on ill-conditioned data

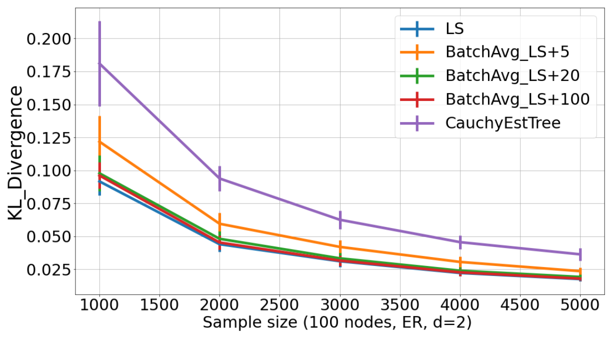


(a) Random tree: 4 edges removed

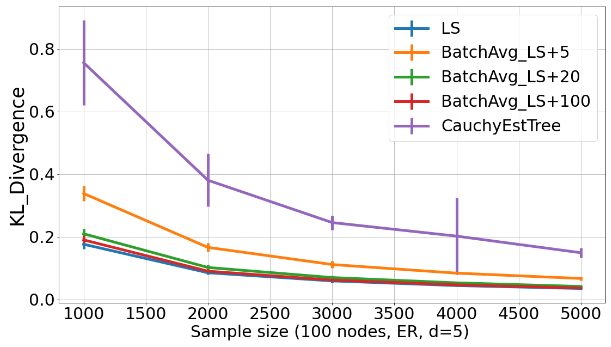


(b) ER graph, d=5: 9 edges removed

Figure 10: Agnostic learning

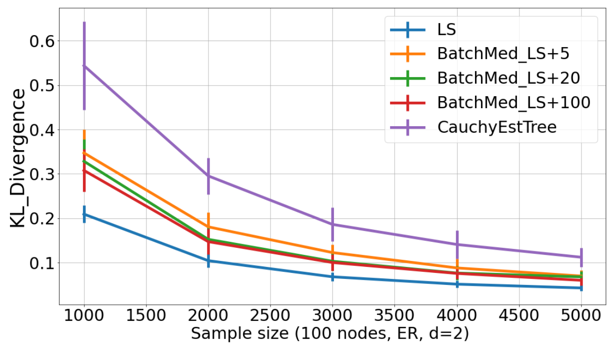


(a) ER graph, $d = 2$

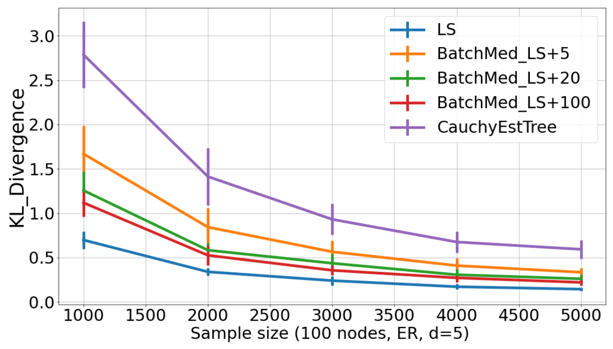


(b) ER graph, $d = 5$

Figure 11: Effect of changing batch size over Batch Average



(a) ER graph, $d = 2$



(b) ER graph, $d = 5$

Figure 12: Effect of changing batch size over Batch Median (ER)

Effect of changing batch size Next, we experiment the trade off between the batch-size (eg. batch size = [5, 20, 100][15]) and the KL-divergence of our `BatchAvgLeastSquares` and `BatchMedLeastSquares` estimators in detail. As shown in Fig. 11 and Fig. 12, when batch size increases, the results are closer to the `LeastSquares` estimator. In other words, `LeastSquares` can be seen as a special case of either `BatchAvgLeastSquares` or `BatchMedLeast-Squares` with one batch only. Thus, when batch size increases, the performances of `BatchAvgLeastSquares` and `BatchMedLeastSquares` are closer to the `LeastSquares` estimator. On the contrary, the `CauchyEstTree` estimator can be seen as the estimator with a batch size of p. Therefore, with smaller batch size (eg. batch size $= p + 5$), `BatchAvgLeastSquares` and `BatchMedLeastSquares` performs closer to the `CauchyEstTree` estimator.

Runtime comparison We now give the amount of time spent by each algorithm to process a degree-5 ER graph on 100 nodes with 5000 samples. `LeastSquares` algorithm takes 0.0186 seconds, `BatchAvgLeastSquares` with a batch size of $p + 20$ takes 0.9080 seconds, `BatchMedLeastSquares` with a batch size of $p + 20$ takes 0.94218 seconds, `CauchyEstTree` takes 10.4443 seconds, and `CauchyEst` takes 25.1891 seconds. The timings given above are representative of the relative running times of these algorithms across different graph sizes.

[15] We use batch size up to 100 to ensure there are enough batch size from simulation data, so that either mean and median can converge.