International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-017-0085-7

REGULAR PAPER

Scoring Bayesian networks of mixed variables

Bryan Andrews[1] · **Joseph Ramsey**[2] · **Gregory F. Cooper**[1]

Received: 26 May 2017 / Accepted: 9 December 2017

Abstract
In this paper we outline two novel scoring methods for learning Bayesian networks in the presence of both continuous and discrete variables, that is, mixed variables. While much work has been done in the domain of automated Bayesian network learning, few studies have investigated this task in the presence of both continuous and discrete variables while focusing on scalability. Our goal is to provide two novel and scalable scoring functions capable of handling mixed variables. The first method, the Conditional Gaussian (CG) score, provides a highly efficient option. The second method, the Mixed Variable Polynomial (MVP) score, allows for a wider range of modeled relationships, including nonlinearity, but it is slower than CG. Both methods calculate log likelihood and degrees of freedom terms, which are incorporated into a Bayesian Information Criterion (BIC) score. Additionally, we introduce a structure prior for efficient learning of large networks and a simplification in scoring the discrete case which performs well empirically. While the core of this work focuses on applications in the search and score paradigm, we also show how the introduced scoring functions may be readily adapted as conditional independence tests for constraint-based Bayesian network learning algorithms. Lastly, we describe ways to simulate networks of mixed variable types and evaluate our proposed methods on such simulations.

Keywords Bayesian network structure learning · Mixed variables · Continuous and discrete variables

1 Introduction

Bayesian networks are a widely used graphical framework for representing probabilistic relationships among variables. In general, a Bayesian network consists of two components, a structure component and a distribution component. The structure component encodes conditional independence relationships between variables allowing for an efficient factorization of the joint distribution, while the distribution component parameterizes the probabilistic relationships among the variables. In this paper, our interests lie in learning the structure component of Bayesian networks, represented by a Directed Acyclic Graph (DAG). Learning a DAG over a set of variables is of particular interest, because under assumptions a DAG can be interpreted as a causal model [26].

Automated Bayesian network learning from data is an important and active area of research. However, relatively few researchers have investigated this task in the presence of both continuous and discrete variables [3,8,13,15,21,24,25]. In the limited work that has been done, researchers either ignore the case where continuous variables are parents of discrete variables, or do not provide solutions that scale much beyond 100 variables. The goal of this paper is to provide solutions for researchers working with datasets containing hundreds of variables.

Most methods for learning Bayesian networks fall into one of two categories: search and score or constraint-based. Search and score methods heuristically search the space

Research reported in this publication was supported by Grant U54HG008540 from the National Human Genome Research Institute through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative, by Grant R01LM012087 from the National Library of Medicine, by Grant IIS-1636786 from the National Science Foundation, and by Grant #4100070287 from the Pennsylvania Department of Health (PA DOH). The PA DOH specifically disclaims responsibility for any analyses, interpretations, or conclusions. The content of this paper is solely the responsibility of the authors and does not necessarily represent the official views of the granting agencies.

✉ Gregory F. Cooper
gfc@pitt.edu

Bryan Andrews
bja43@pitt.edu

Joseph Ramsey
jdramsey@andrew.cmu.edu

[1] University of Pittsburgh, Pittsburgh, PA 15260, USA

[2] Carnegie Mellon University, Pittsburgh, PA 15213, USA

Published online: 11 January 2018

of possible structures using an objective function to evaluate fitness while constraint-based methods use conditional independence tests find patterns of independence that are consistent with a set of DAGs. The core of this paper focuses on the search and score paradigm; however, we also show how the scoring functions we propose may be readily adapted as conditional independence tests for constraint-based methods. For additional background information on Bayesian networks and learning their structures, see [6].

The remainder of this paper is organized as follows. Section 2 discusses general properties of scoring functions and the Bayesian Information Criterion (BIC). Sections 3 and 4 introduce the Conditional Gaussian (CG) score and the Mixed Variable Polynomial (MVP) score, respectively. Section 5 details several adaptations of the introduced methods. Section 6 reports empirical results of the CG and MVP methods on data generated using simulation. Section 7 provides discussion and conclusions.

2 Scoring Bayesian networks

Search and score methods utilize an objective function to evaluate the fitness of DAGs on a given dataset $\mathcal{D}$. Let S be a score function and $\mathcal{G}$ be a DAG containing m variables. Let Y_i be the ith variable with parents Pa_i for $i \in \{1, 2, \ldots, m\}$. When scoring $\mathcal{G}$, most search algorithms require that S decomposes into local components involving only Y_i and Pa_i. This property is known as decomposability. Given a score is decomposable, we need only compare the differing local components to decide which of any two DAGs is better. To solidify this concept, we say a score S is decomposable if it can be represented as a sum of local components. We score DAG $\mathcal{G}$ on dataset $\mathcal{D}$ using score S as,

$$S(\mathcal{G}, \mathcal{D}) = \sum_{i=1}^{m} s(Y_i, Pa_i),$$

where $s(Y_i, Pa_i)$ is the score for the ith local component.

Note that several DAGs can encode the same set of conditional independence relationships. A set of DAGs which encodes the same independencies is known as a Markov Equivalence Class (MEC). If a scoring function S scores all DAGs in the same MEC equally, then S is score equivalent. To clarify, let $\mathcal{G}$ and $\mathcal{G}'$ be DAGs over the variables in dataset $\mathcal{D}$. If $\mathcal{G}$ and $\mathcal{G}'$ encode the same conditional independence relationships and S is score equivalent, then $S(\mathcal{G}, \mathcal{D}) = S(\mathcal{G}', \mathcal{D})$. This can be a desirable trait because it allows search algorithms, such as Greedy Equivalent Search (GES) [5], to search over MECs directly.

Another common trait for scoring functions that algorithms such as GES require for optimality is consistency. Let $\mathcal{D}$ be a dataset and $\mathcal{G}$ and $\mathcal{G}'$be DAGs. A scoring function $\mathcal{S}$ is consistent if in the large sample limit the following two conditions imply $\mathcal{G}$ will score higher than $\mathcal{G}'$, i.e., $S(\mathcal{G}, \mathcal{D}) > S(\mathcal{G}', \mathcal{D})$: (1) There exists a parameterization θ which allows $\mathcal{G}$ to represent the generating distribution of $\mathcal{D}$ and no such parameterization θ' exists for $\mathcal{G}'$ or (2) there exist parameterizations θ and θ' which allow $\mathcal{G}$ and $\mathcal{G}'$ each to represent the generating distribution of $\mathcal{D}$, but $\mathcal{G}$ contains fewer parameters.

2.1 The Bayesian information criterion

The Bayesian Information Criterion (BIC) is a well studied and widely applied marginal log likelihood approximation that is generally used for model selection. Let M be a model we wish to score given a dataset $\mathcal{D}$. We can write the probability of model M given $\mathcal{D}$ using Bayes' rule as,

$$p(M|\mathcal{D}) = \frac{p(\mathcal{D}|M)p(M)}{p(\mathcal{D})}.$$

However, since the data are fixed, $p(\mathcal{D})$ will remain constant across different model choices. Thus, for model selection, we use,

$$p(M|\mathcal{D}) \propto p(\mathcal{D}|M)p(M). \tag{1}$$

BIC aims to approximate $p(M|\mathcal{D})$ in (1). For now, we assume $p(M)$ is distributed uniformly and thus drop it. Later in Sect. 5.1, we introduce an alternative distribution for $p(M)$, which we find performs well in practice. Raftery [17] shows that when assuming a flat prior over the parameters, the logarithm of $p(\mathcal{D}|M)$ can be approximated as:

$$\log p(\mathcal{D}|M) \approx -2\ell(\boldsymbol{\theta}) + df(\boldsymbol{\theta}) \times 2\log n, \tag{2}$$

where $\ell(\boldsymbol{\theta})$ is the maximum log likelihood of the data, $df(\boldsymbol{\theta})$ are the degrees of freedom, and n is the sample size. The approximation on the right hand side of (2) characterizes the BIC, introduced by Schwarz [23]. BIC is decomposable and can be readily applied to score Bayesian networks. In Sects. 3 and 4, we detail how to calculate the log likelihood and degrees of freedom terms for BIC using our proposed scoring methods. We score DAGs using BIC given such log likelihood and degrees of freedom calculations.

3 The conditional Gaussian score

In general, the Conditional Gaussian (CG) score calculates conditional Gaussian mixtures using the ratios of joint distributions. Since CG uses BIC as a framework to evaluate its approximations, the score is decomposable into a sum of

parent-child relationships. In order to outline such a relationship, we introduce continuous variables C_1, C_2 and discrete variables D_1, D_2. Below we detail the CG score using these four variables, however, this procedure straightforwardly generalizes to any number of variables. We compute the conditional distribution where C_1 is a child with parents C_2, D_1, and D_2 as,

$$\begin{aligned} p(C_1|C_2, D_1, D_2) &= \frac{p(C_1, C_2, D_1, D_2)}{p(C_2, D_1, D_2)} \\ &= \frac{p(C_1, C_2|D_1, D_2)}{p(C_2|D_1, D_2)} \end{aligned} \tag{3}$$

and the conditional distribution where D_1 is a child with parents C_1, C_2, and D_2 as,

$$\begin{aligned} p(D_1|C_1, C_2, D_2) &= \frac{p(C_1, C_2, D_1, D_2)}{p(C_1, C_2, D_2)} \\ &= \frac{p(C_1, C_2|D_1, D_2)p(D_1, D_2)}{p(C_1, C_2|D_2)p(D_2)}. \end{aligned} \tag{4}$$

In (3) and (4), we can readily calculate $p(C_1, C_2|D_1, D_2)$ and $p(C_1, C_2|D_2)$ using Gaussian distributions partitioned on the discrete variables and $p(D_1, D_2)$, $p(D_2)$ using multinomial distributions. This raises the first of CG's assumptions.

Assumption 1 The data were generated from a Gaussian mixture where each Gaussian component exists for a particular setting of the discrete variables.

This assumption allows for efficient calculations, but also assumes that the discrete variables take part in generating the continuous variables by defining the Gaussian mixture components, e.g., $p(C_1, C_2, D_1, D_2)$ is a Gaussian mixture with a Gaussian component for each setting of D_1 and D_2. Therefore, when scoring a discrete variable as the child of a continuous variable, our model assumption will inherently encode the reverse relationship. In Sect. 6, we see that even with this assumption, CG performs quite well.

Assumption 2 The instances in the data are independent and identically distributed.

The data are assumed to be i.i.d. so that we can calculate the log likelihood as a sum over the marginal log probabilities for each instance in the data.

It is important to note that if we treat $p(C_1, C_2, D_1, D_2)$ as a mixture distribution with a Gaussian component for each setting of D_1 and D_2, to calculate $p(C_1, C_2, D_2)$ correctly, we must marginalize D_2 out and treat $p(C_1, C_2, D_2)$ as a mixture distribution with a Gaussian mixture for each setting of D_2.

Assumption 3 All Gaussian mixtures are approximately Gaussian.

For computational efficiency, we approximate all Gaussian mixtures resulting from marginalizing discrete variables out as single Gaussian distributions. In Sect. 6.1, we evaluate this approximation experimentally and find that it performs well.

Under mild conditions, BIC is consistent for Gaussian mixture models [10]. Since CG assumes the data are generated according to a Gaussian mixture, under the same mild assumptions, CG is consistent. Additionally, CG is score equivalent; see Appendix A for a proof.

In the remainder of the current section, we provide a high-level overview of the CG method; Sects. 3.1 and 3.2 provide details. Let Y_i be the ith variable in a DAG $\mathcal{G}$ with the set Pa_i containing the parents of Y_i. Furthermore, let Pa_i consist of two mutually exclusive subsets Pc_i and Pd_i such that Pc_i and Pd_i hold the continuous and discrete parents of Y_i, respectively. To evaluate the parent–child relationship between a variable Y_i and its parents Pa_i, CG calculates the log likelihood and degrees of freedom for the joint distributions of two sets of variables, $Y_i \cup Pa_i$ and Pa_i. The log likelihood of Y_i given its parents Pa_i is computed as the difference between the log likelihood terms for $Y_i \cup Pa_i$ and Pa_i. Similarly, the degrees of freedom are calculated as the difference in parameters used to fit $Y_i \cup Pa_i$ and Pa_i.

When evaluating the two sets of variables, the dataset $\mathcal{D}$ is first partitioned according to the discrete variables in each set. That is, we divide $\mathcal{D}$ using a partitioning set Π_i over all the instances in $\mathcal{D}$. Π_i contains a partition for each combination of values the discrete variables take on in $\mathcal{D}$. Further, we form a design matrix $\boldsymbol{X}_p$ for each partition $p \in \Pi_i$. $\boldsymbol{X}_p$ holds the data corresponding to the instances of the continuous variables in partition p. Gaussian and multinomial distributions are fit according to the continuous and discrete variables, respectively, to calculate log likelihood and degrees of freedom terms which BIC uses to compute the score.

3.1 Modeling a set of variables

When using CG, we have three different kinds of sets to model: $Y_i \cup Pa_i$ where Y_i is continuous, $Y_i \cup Pa_i$ where Y_i is discrete, and Pa_i. They all follow the same generic format so we will describe the process in general while pointing out any subtle differences where they apply.

First we partition the data with respect to a partitioning set Π_i generated according to the discrete variables Pd_i. Note that if our set includes a discrete child Y_i, then the discrete variables are comprised of $Y_i \cup Pd_i$ and we partition according to these variables. Π_i contains a partition for every combination of values in the discrete variables. We define the partitioning set Π_i using a Cartesian product of the discrete variables. Let $|Pd_i| = d$, then partitioning set $\Pi_i = (Y_i) \times Pd_i(1) \times Pd_i(2) \times \cdots \times Pd_i(d)$ where Y_i is the set of values for the child (included only if Y_i is discrete), $Pd_i(1)$ is the

set of values for the first discrete parent, $Pd_i(2)$ is the set of values for the second discrete parent, and so forth.

Let $|Pc_i| = c$, then for each partition $p \in \Pi_i$ we define a design matrix $\boldsymbol{X}_p$ with n_p observations and c variables. Here, if our set includes a continuous child Y_i, then we instead define $\boldsymbol{X}_p$ with $c+1$ variables corresponding to the variables in $Y_i \cup Pc_i$. That is,

$$\boldsymbol{X}_p = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1c} & (y_1) \\ x_{21} & x_{22} & \dots & x_{2c} & (y_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n_p 1} & x_{n_p 2} & \cdots & x_{n_p c} & (y_{n_p}) \end{bmatrix},$$

where x_{jk} is the jth value with respect to partition p of the kth variable in Pc_i and y_j is the jth value with respect to p of the child Y_i (included only if Y_i is continuous) for $j \in \{1, 2, \dots, n_p\}$ and $k \in \{1, 2, \dots, c\}$.

3.2 Calculating the log likelihood and degrees of freedom

The calculations for the three aforementioned sets are identical in formulation, so without loss of generality, we demonstrate the log likelihood and degrees of freedom calculations for the set $Y_i \cup Pa_i$. The log likelihood for a set is calculated component-wise over each partition and summed together as follows,

$$\ell_{Y_i \cup Pa_i}(\boldsymbol{\theta}|\boldsymbol{X}) = \sum_{p \in \Pi_i} \ell_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p). \tag{5}$$

The degrees of freedom are calculated in a similar manner,

$$df_{Y_i \cup Pa_i}(\boldsymbol{\theta}) = \sum_{p \in \Pi_i} df_p(\boldsymbol{\theta}_p) - 1, \tag{6}$$

where the minus 1 term accounts for the redundant mixing component. Let n be the number of observations in the unpartitioned dataset. For each partition $p \in \Pi_i$, let d be the number of variables in $\boldsymbol{X}_p$ and $\boldsymbol{x}_{p,j}$ be the jth observation from $\boldsymbol{X}_p$. From [2], we calculate the Gaussian log likelihood for partition p as,

$$\ell(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p|\boldsymbol{X}_p) = -\frac{n_p d}{2}\log 2\pi - \frac{n_p}{2}\log|\boldsymbol{\Sigma}_p| - \frac{1}{2}\sum_{j=1}^{n_p}(\boldsymbol{x}_{p,j} - \boldsymbol{\mu}_p)^T \boldsymbol{\Sigma}_p^{-1}(\boldsymbol{x}_{p,j} - \boldsymbol{\mu}_p), \tag{7}$$

where $\boldsymbol{\mu}_p$, $\boldsymbol{\Sigma}_p$ are the mean and variance of the Gaussian distribution, respectively. The maximum likelihood estimate $\hat{\boldsymbol{\Sigma}}_p$ is computed as,

$$\hat{\boldsymbol{\Sigma}}_p = \frac{1}{n_p}\sum_{j=1}^{n_p}(\boldsymbol{x}_{p,j} - \bar{\boldsymbol{x}_p})(\boldsymbol{x}_{p,j} - \bar{\boldsymbol{x}_p})^T. \tag{8}$$

Let $\boldsymbol{\mu}_p = \bar{\boldsymbol{x}_p}$. Note that $\bar{\boldsymbol{x}_p}$ will converge quickly to $\boldsymbol{\mu}_p$. Using the estimate in (8), the log likelihood in (7) simplifies to,

$$\ell(\hat{\boldsymbol{\Sigma}}_p|\boldsymbol{X}_p) = -\frac{n_p}{2}(\log|\hat{\boldsymbol{\Sigma}}_p| + d\log 2\pi + d). \tag{9}$$

We use (9) to compute the log likelihood of a Gaussian conditioned on discrete variables. However, we still must add the log probability of an instance being from partition p to calculate the desired joint log likelihood. These probabilities are computed using the maximum likelihood estimate of variables distributed according to a multinomial. This estimate is the count of instances in partition p denoted n_p over the total count n of all instances: $\frac{n_p}{n}$. Thus, we calculate the log likelihood for partition p as,

$$\ell_p(\hat{\boldsymbol{\theta}_p}|\boldsymbol{X}_p) = -\frac{n_p}{2}(\log|\hat{\boldsymbol{\Sigma}}_p| + d\log 2\pi + d) + n_p \log\frac{n_p}{n}. \tag{10}$$

We use (10) to calculate $\ell_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p)$ in (5). To find the number of parameters in partition p, we count the number of unique terms in $\hat{\boldsymbol{\Sigma}}_p$ plus one for the mixing component. Therefore,

$$df_p(\hat{\boldsymbol{\theta}_p}) = \frac{d(d+1)}{2} + 1. \tag{11}$$

We use (11) to calculate $df_p(\boldsymbol{\theta}_p)$ in (6).

Using the form of (3) and (4), we calculate the log likelihood and degrees of freedom terms as,

$$\ell_i(\hat{\boldsymbol{\theta}}|\boldsymbol{X}) = \ell_{Y_i \cup Pa_i}(\hat{\boldsymbol{\theta}}|\boldsymbol{X}) - \ell_{Pa_i}(\hat{\boldsymbol{\theta}}|\boldsymbol{X}), \tag{12}$$

$$df_i(\hat{\boldsymbol{\theta}}) = df_{Y_i \cup Pa_i}(\hat{\boldsymbol{\theta}}) - df_{Pa_i}(\hat{\boldsymbol{\theta}}). \tag{13}$$

BIC uses (12) and (13) to compute the score for the parent-child relationship of Y_i given Pa_i.

4 The mixed variable polynomial score

The Mixed Variable Polynomial (MVP) score uses higher order polynomial functions to approximate relationships between any number of continuous and discrete variables. Since MVP uses BIC as a framework to evaluate its approximations, the score is decomposable into a sum of parent–child relationships. The MVP method scores the decomposed local components of a DAG $\mathcal{G}$ using approximating polynomial functions. To motivate the ideas underlying this approach, we note the implications of the Weierstrass Approximation Theorem for consistency.

Weierstrass Approximation Theorem *Suppose f is a continuous real-valued function defined on the real interval $[a, b]$. For every $\epsilon > 0$, there exists a polynomial p such that for all $x \in [a, b]$, we have $|f(x) - p(x)| < \epsilon$.*

In short, as long as a function f is continuous and the contributing variables exist within a bounded interval, then there exists a polynomial function which approximates f to an arbitrary degree of accuracy [11]. This brings us to our first two assumptions.

Assumption 1 The sample space of each variable is finite.

To shed some light on this assumption, we note that MVP's approximations are functions of continuous variables in the data. Thus, the motivation for Assumption 1 becomes apparent as a prerequisite of the previously stated theorem; finite sample spaces are bounded.

Assumption 2 Each continuous variable is defined by continuous functions of their continuous parents plus additive Gaussian noise. The probability mass function of each discrete variable is defined by positive continuous functions of their continuous parents.

The motivation for this assumption follows from Weierstrass's approximation theorem since f, the function to be approximated, must be continuous. However, along with assuming continuity, we restrict the model class in the continuous child case to have additive Gaussian noise. This assumption allows us to use least squares regression to obtain efficient maximum likelihood estimates. Additionally, we assume positive functions in the discrete case since we are estimating probability mass functions. It is worth noting that we do not assume linearity unlike other commonly used scores.

Assumption 3 There are no interaction terms between continuous parents.

We make this assumption for tractability. Modeling all interactions among the continuous parents is a combinatorial problem. Thus, we forgo such interaction terms.

Assumption 4 The instances in the data are independent and identically distributed.

The data are assumed to be i.i.d. so that we can calculate the log likelihood as a sum over the marginal log probabilities for each instance in the data.

Under these assumptions, the MVP score is consistent in the large sample limit with an adequate choice of maximum polynomial degree; see Appendix A for a proof. However, due to the use of nonlinear functions, it is not score equivalent for any maximum polynomial degree greater than 1. In Sect. 6, we see that even without this property, the MVP score still performs quite well. Moreover, in general, we do not expect causal relationships to be score equivalent, so using a framework that requires score equivalence would not be desirable. As an example of previous work suggesting that asymmetric scores can be beneficial in inferring causation, see [12].

4.1 Partitioned regression

Let Y_i be the ith variable in a DAG $\mathcal{G}$ and Pa_i be the set containing the parents of Y_i in $\mathcal{G}$. Furthermore, let Pa_i consist of two mutually exclusive subsets Pc_i and Pd_i such that Pc_i and Pd_i hold the continuous and discrete parents of Y_i, respectively. In general, to evaluate the local score component between Y_i and its parents Pa_i, MVP first partitions the data with respect to the discrete parents Pd_i and performs least squares regression using the continuous parents Pc_i. The log likelihood and degrees of freedom for the model are calculated depending on the variable type of Y_i. BIC uses the log likelihood and degrees of freedom terms to compute the score.

A partitioning set Π_i partitions $\mathcal{D}$ with respect to the discrete parents Pd_i and contains a partition for every combination of values in the discrete parents. We define Π_i using a Cartesian product of the discrete parents Pd_i. Let $|Pd_i| = d$, then partitioning set $\Pi_i = Pd_i(1) \times Pd_i(2) \times \cdots \times Pd_i(d)$ where $Pd_i(1)$ is the set of values for the first discrete parent, $Pd_i(2)$ is the set of values for the second discrete parent, and so forth.

Let $|Pc_i| = c$, then for each partition $p \in \Pi_i$ we define a design matrix $\boldsymbol{X}_p$ with n_p observations and c variables. Additionally, we add a bias term and higher order polynomial terms for each variable in Pc_i, stopping at a maximum polynomial order specified by $g(n_p)$,

$$\boldsymbol{X}_p = \begin{bmatrix} 1 & x_{11} & \dots & x_{1c} & x_{11}^2 & \dots & x_{1c}^2 & \dots & x_{11}^{g(n_p)} & \dots & x_{1c}^{g(n_p)} \\ 1 & x_{21} & \dots & x_{2c} & x_{21}^2 & \dots & x_{2c}^2 & \dots & x_{21}^{g(n_p)} & \dots & x_{2c}^{g(n_p)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n_p1} & \dots & x_{n_pc} & x_{n_p1}^2 & \dots & x_{n_pc}^2 & \dots & x_{n_p1}^{g(n_p)} & \dots & x_{n_pc}^{g(n_p)} \end{bmatrix},$$

where x_{jk} is the jth value with respect to partition p of the kth variable in Pc_i for $j \in \{1, 2, \dots, n_p\}$ and $k \in \{1, 2, \dots, c\}$. In this paper, we report two choices for $g(n_p)$: $g(n_p) = 1$, and $g(n_p) = \lfloor \log n_p \rfloor$. We have tried other choices, such as $g(n_p) = 3$, but found the above options provide the best solutions. Define $\boldsymbol{x}_j$ and $\boldsymbol{y}_p$ as,

$$\boldsymbol{x}_{p,j} = \begin{bmatrix} 1 & x_{j1} & \dots & x_{jc} & x_{j1}^2 & \dots & x_{jc}^2 & \dots & x_{j1}^{g(n_p)} & \dots & x_{jc}^{g(n_p)} \end{bmatrix},$$

$$\boldsymbol{y}_p = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_p} \end{bmatrix},$$

where $\boldsymbol{x}_{p,j}$ is the jth observation in $\boldsymbol{X}_p$ and y_j is the jth value with respect to partition p of Y_i for $j \in \{1, 2, \ldots, n_p\}$.

We calculate the log likelihood of a variable Y_i given a set of parents Pa_i as a sum over the log likelihoods from each partition p,

$$\ell_i(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) = \sum_{p \in \Pi_i} \ell_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p, \boldsymbol{y}_p), \tag{14}$$

where $\ell_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p, \boldsymbol{y}_p)$ is defined depending on whether Y_i is discrete or continuous. Similarly, the degrees of freedom for Y_i are calculated as a sum over the parameter counts in each partition p,

$$df_i(\boldsymbol{\theta}) = \sum_{p \in \Pi_i} df_p(\boldsymbol{\theta}_p). \tag{15}$$

BIC computes the local score component using the log likelihood $\ell_i(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y})$ and degrees of freedom $df_i(\boldsymbol{\theta})$.

4.2 Modeling a continuous child

In the case where Y_i is continuous, for partition p with design matrix $\boldsymbol{X}_p$ and target vector $\boldsymbol{y}_p$, we use the maximum likelihood estimate to determine the log likelihood $\ell_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p, \boldsymbol{y}_p)$ and degrees of freedom $df_p(\boldsymbol{\theta}_p)$. By Assumption 2, for partition p, each $y_j = f_p(\boldsymbol{x}_{p,j}) + \epsilon_p$ where f_p is a continuous function defined on a bounded interval and $\epsilon_p \sim N(0, \sigma_p^2)$ is additive Gaussian noise with variance σ_p^2. By the Weierstrass Approximation Theorem, there exists a polynomial function $\hat{f}_p$ which approximates f_p such that $y_j \approx \hat{f}_p(\boldsymbol{x}_{p,j}) + \epsilon_p$. Estimating the parameters of $\hat{f}_p$ using least squares regression, we have $\hat{y}_j \sim N(\boldsymbol{X}_p\boldsymbol{\beta}_p, \sigma_p^2)$. Therefore, the log likelihood of partition p becomes,

$$\ell_p(\boldsymbol{\beta}_p, \sigma_p^2|\boldsymbol{X}_p, \boldsymbol{y}_p) = -\frac{n_p}{2}\log 2\pi - \frac{n_p}{2}\log \sigma_p^2 - \frac{(\boldsymbol{y}_p - \boldsymbol{X}_p\boldsymbol{\beta}_p)^T(\boldsymbol{y}_p - \boldsymbol{X}_p\boldsymbol{\beta}_p)}{2\sigma_p^2}, \tag{16}$$

where the maximum likelihood estimates are computed as,

$$\hat{\sigma_p}^2 = \frac{(\boldsymbol{y}_p - \boldsymbol{X}_p\hat{\boldsymbol{\beta}}_p)^T(\boldsymbol{y}_p - \boldsymbol{X}_p\hat{\boldsymbol{\beta}}_p)}{n_p}, \tag{17}$$

$$\hat{\boldsymbol{\beta}}_p = (\boldsymbol{X}_p^T\boldsymbol{X}_p)^{-1}\boldsymbol{X}_p^T\boldsymbol{y}_p. \tag{18}$$

Using the estimates in (17) and (18), the log likelihood in (16) simplifies to,

$$\ell_p(\hat{\boldsymbol{\beta}}_p, \hat{\sigma_p}^2|\boldsymbol{X}_p, \boldsymbol{y}_p) = -\frac{n_p}{2}(\log 2\pi + \log \hat{\sigma_p}^2 + 1), \tag{19}$$

which we use to calculate $\ell_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p, \boldsymbol{y}_p)$ in (14). To find the number of parameters in partition p to calculate $df_p(\boldsymbol{\theta}_p)$ for (15), we count the number of terms in $\hat{\boldsymbol{\beta}}_p$,

$$df_p(\boldsymbol{\theta}_p) = c \cdot g(n_p) + 1. \tag{20}$$

The BIC uses (14) and (15) to compute the parent–child relationship for Y_i given Pa_i.

4.3 Modeling a discrete child

In the case where Y_i is discrete, for partition p with design matrix $\boldsymbol{X}_p$ and target vector $\boldsymbol{y}_p$, we calculate the log likelihood $\ell_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p, \boldsymbol{y}_p)$ and degrees of freedom $df_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p, \boldsymbol{y}_p)$ using least squares regression. Suppose Y_i consists of d categories. Let $f_{p,h}$ calculate the probability of the hth category in Y_i given the values of the continuous parents Pc_i where $h \in \{1, \ldots, d\}$. By Assumption 2 and the Weierstrass Approximation Theorem, there exists a polynomial function $\hat{f}_{p,h}$ which approximates $f_{p,h}$ arbitrarily well. With this in mind, we aim to approximate each $f_{p,h}$ with the polynomial function $\boldsymbol{X}_p\hat{\boldsymbol{\beta}}_{p,h}$ where $\hat{\boldsymbol{\beta}}_{p,h}$ are the polynomial coefficients calculated from least squares regression. Our end goal is to use the approximations of each $f_{p,h}$ as components of a conditional probability mass function in order to calculate the log likelihood and degrees of freedom terms.

Define the categories of $\boldsymbol{y}_p$ such that each $y_j \in \{1, \ldots, d\}$. Expand $\boldsymbol{y}_p$ into d binary vectors where the hth vector represents the hth category. That is, the jth element from the hth vector asserts whether or not the jth observation of $\boldsymbol{y}_p$ is category h. To represent these binary vectors in matrix notation, we define $1_{\{\text{condition}\}}$ as an indicator variable which is 1 if condition is true and 0 otherwise. The hth binary vector is then defined as,

$$\mathbf{1}_{\{\boldsymbol{y}_p = h\}} = \begin{bmatrix} 1_{\{y_1 = h\}} \\ 1_{\{y_2 = h\}} \\ \vdots \\ 1_{\{y_{n_p} = h\}} \end{bmatrix}.$$

We further define $\mathbf{1}_p$ to represent an $n_p \times 1$ vector of ones. Using these binary vectors as our targets, we calculate the least squares regression estimate, which yields,

$$\hat{\boldsymbol{\beta}}_{p,h} = (\boldsymbol{X}_p^T\boldsymbol{X}_p)^{-1}\boldsymbol{X}_p^T\,\mathbf{1}_{\{\boldsymbol{y}_p = h\}}.$$

If we want to interpret the results of least squares regression $\boldsymbol{X}_p\hat{\boldsymbol{\beta}}_{p,h}$ as probabilities, we first must ensure that,

1. $\sum_{h=1}^{d} \boldsymbol{X}_p\hat{\boldsymbol{\beta}}_{p,h} = \mathbf{1}_p$
2. $\boldsymbol{x}_{p,j}\hat{\boldsymbol{\beta}}_{p,h} \geq 0, \ \forall\ j \in \{1, \ldots, n_p\}, h \in \{1, \ldots, d\}$.

We prove condition 1 is necessarily true and that condition 2 holds for an adequate maximum polynomial degree in the sample limit; see Appendix A for the proofs.

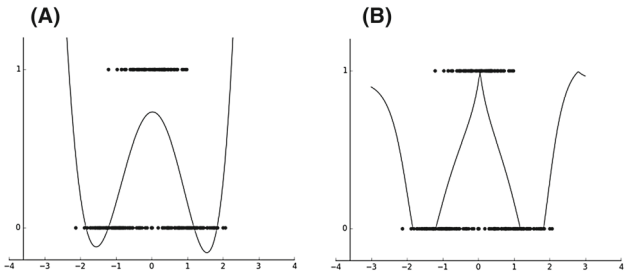

Fig. 1 **a** Shows an estimated conditional probability mass function for a particular category via least squares. **b** Shows the same estimated conditional probability mass function after it has been transformed by Algorithm 1

Unfortunately, there is no guarantee the values given by least squares regression will be strictly nonnegative for finite samples. Instead we define a procedure that maps the values calculated for each y_j to a positive value. In the procedure, we avoid setting values directly to zero in order to prevent assigning zero probability to any observed instances. Instead, we use a value which tends toward zero; we insure the minimum value of each y_j is at least $\frac{1}{n_p}$ and maintain the condition that each set of estimates sums to one. Therefore, we can treat the mapped estimates as probabilities. Algorithm 1 outlines how the procedure accomplishes this mapping. Figure 1 shows the effect of Algorithm 1 when applied to the least squares estimates of the conditional probability mass function for a particular category of a discrete variable.

Algorithm 1: Maps least squares estimates to valid probability distributions

Input: $\boldsymbol{X}_{p,j}, \hat{\boldsymbol{\beta}}_{p,h}$
Output: $probs$

1 $probs = [\,]$; // list of conditional pdfs
2 **for** $j \leftarrow 1$ **to** d **do**
// calculate least squares estimates
3 $l = [\boldsymbol{x}_{p,j}\hat{\boldsymbol{\beta}}_{p,h} | \forall h \in \{1, \ldots, d\}]$; // d-vector
// calculate minimum value
4 $m_{p,j} = \min\{\frac{1}{n_p}, l\}$;
// calculate scaling term
5 $\alpha_{p,j} = (\frac{1}{n_p} - \frac{1}{d})/(m_{p,j} - \frac{1}{d})$;
// remap estimates
6 $l = l - \frac{1}{d}$; // element-wise shift
7 $l = \alpha_{p,j} \cdot l$; // element-wise scale
8 $l = l + \frac{1}{d}$; // element-wise shift
// add remap estimates to output
9 $probs$.append(l) ;
10 **end**

In words, our procedure is as follows:

1. Shift the estimates such that they are centered about a noninformative center by subtracting $\frac{1}{d}$ (line 6).
2. Scale the estimates such that the smallest final value will be at least $\frac{1}{n_p}$ (line 7).
3. Shift the scaled estimates back to the original center by adding $\frac{1}{d}$ (line 8).

Since we only want to perform this procedure if one of the least squares estimates is negative, we define $m_{p,j} = \min\{\frac{1}{n_p}, \boldsymbol{x}_{p,j}\hat{\boldsymbol{\beta}}_{p,h} \forall h\}$ so that $m_{p,j}$ is either the minimum estimate for y_j or $\frac{1}{n_p}$ (line 4). We calculate the scaling factor $\alpha_{p,j}$ (line 5) by noting that we want,

$$\alpha_{p,j}\left(m_{p,j} - \frac{1}{d}\right) + \frac{1}{d} = \frac{1}{n_p}.$$

Solving for $\alpha_{p,j}$ we find,

$$\alpha_j = \frac{\frac{1}{n_p} - \frac{1}{k}}{m_{p,j} - \frac{1}{k}}.$$

Note, that if $m_{p,j} = \frac{1}{n_p}$, then $\alpha_{p,j} = 1$ and we do not transform the estimates. We compute the log likelihood in the discrete case as

$$\ell_p(\hat{\boldsymbol{\theta}}_p | \boldsymbol{X}_p, \boldsymbol{y}_p) = \sum_{j=1}^{n_p} \log\left(\alpha_{p,j}\boldsymbol{x}_{p,j}\hat{\boldsymbol{\beta}}_{p,y_j} + \frac{1}{d}(1 - \alpha_{p,j})\right). \tag{21}$$

We use (21) to calculate $\ell(\boldsymbol{\theta}_p | \boldsymbol{X}_p, \boldsymbol{y}_p)$ in (14). To find the number of parameters in partition p, we count the number of terms across all $\hat{\boldsymbol{\beta}}_{p,j}$. Each $\hat{\boldsymbol{\beta}}_{p,j}$ has $c \cdot g(n_p) + 1$ parameters and j ranges over d categories. However, since Proposition 1 shows the estimated probabilities sum to one, the number of free parameters is

$$df_p(\hat{\boldsymbol{\theta}}_p) = (k-1)(c \cdot g(n_p) + 1). \tag{22}$$

As before, BIC uses (14) and (15) to compute the parent-child relationship for Y_i given Pa_i.

5 Implementation details and adaptations

In this section we consider various adaptations of the two proposed scores. In Sect. 5.1, we discuss a binomial structure prior which allows for efficient learning of large networks. In Sect. 5.2, we discuss a simplification for scoring discrete children which performs well empirically. In Sect. 5.3, we discuss how to adapt our scores into conditional independence test for constraint-based methods.

5.1 Binomial structure prior

We introduce a structure prior inspired by the binomial distribution. The idea is to give a prior distribution over the number of parents of each variable. We view the addition of each edge as an independent event that occurs with probability q. Therefore, we expect to see $q \cdot m'$ parents for any given variable where m' is the total number of possible parents. Then we have,

$$\pi(k) = (q)^k (1-q)^{m'-k},$$

where $\pi(k)$ is the prior probability that any given variable Y_i in DAG $\mathcal{G}$ has k parents. Often it is more convenient to work in log space. Thus we calculate the log prior probability as,

$$\log \pi(k) = k \log(q) + (m' - k) \log(1-q).$$

Note that since DAGs encode an ordering over the variables, the total number of possible parents is not necessarily all the variables in the data excluding the variable currently acting as a child. It is usually the case that $m' \neq m - 1$ where m is the total number of variables in the data.

In Sect. 6, we let $m' = m$ in order to calculate the binomial structure prior more efficiency.

We calculate q as $q = \frac{r}{(m-1)}$, where r represents a user-specified upper bound on the expected number of parents of any given node.

Usually, BIC assumes the prior probability of models in Eq. (1) is distributed uniformly. By using the binomial structure prior instead, we adapt BIC to further penalize networks with complex structure. There are other approaches that use a nonuniform prior for BIC, notability, the extended BIC (EBIC) [4]. (EBIC) is a similar modification to BIC which aims to address the small-n-large-P situation. In Sect. 6.2, we compare both the binomial structure prior and EBIC against the use of a uniform prior.

5.2 Multinomial scoring with continuous parents

Both scores presented in this paper reduce to multinomial scoring in the case of a discrete child with exclusively discrete parents. As an alternative, we explore the use of multinomial scoring when there are discrete children and any combination of parents. Before starting a search, we create discretized versions of each continuous variable using equal frequency binning with a predefined number of bins b. Whenever scoring a discrete child, we replace any continuous parents with the precomputed discretized versions of those variables. This allows us to quickly and efficiently perform multinomial scoring for all discrete children. We will henceforth refer to this adaptation as the discretization heuristic and report our finding when choosing $b = 3$ as a modification to CG in Sect. 6.4.

5.3 As a conditional independence test

We can readily adapt CG and MVP to produce conditional independence tests; to do so, we calculate the log likelihood and degrees of freedom as usual, but perform a likelihood ratio test instead of scoring with BIC. Suppose we wish to test $Y_0 \perp\!\!\!\perp Y_1 | Z$ where Y_0 and Y_1 are variables (nodes) and Z is a conditioning set of variables. Define ℓ_0 and df_0, respectively, as the log likelihood and degrees of freedom for Y_0 given Pa_0 where $Pa_0 = Y_1 \cup Z$. Further, define ℓ_0' and df_0', respectively, as the log likelihood and degrees of freedom for Y_0 given Pa_0' where $Pa_0' = Z$. Perform a likelihood ratio test with test statistic $2(\ell_0 - \ell_0')$ and $df_0 - df_0'$ degrees of freedom. This tests whether the model encoding $Y_0 \not\perp\!\!\!\perp Y_1 | Z$ or the model encoding $Y_0 \perp\!\!\!\perp Y_1 | Z$ fits the data better. If the scoring method used is not score equivalent, then we must also perform a likelihood ratio test with test statistic $2(\ell_1 - \ell_1')$ and $df_1 - df_1'$ degrees of freedom where Y_0 and Y_1 are swapped. In this case, we decide the variables are dependent if there is enough evidence in either test to support that hypothesis.

6 Simulation studies

To simulate mixed data, we first randomly generate a DAG $\mathcal{G}$ and designate each variable in $\mathcal{G}$ as either discrete or continuous. $\mathcal{G}$ is generated by randomly defining a causal order and adding edges between the variables. Edges are added between randomly chosen pairs of nodes such that the connections are true to the pre-specified ordering; they are continually added until the average degree of the graph reaches a user-specified amount. Variables in the network without parents are generated according to Gaussian and multinomial distributions. We create temporary discretized versions of each continuous variable using equal frequency binning with 2–5 bins uniformly chosen, for reasons described below. In causal order, we simulate the remaining variables as follows. Continuous variables are generated by partitioning on the discrete parents and randomly parameterizing the coefficients of a linear regression for each partition. Discrete variables are generated via randomly parameterized multinomial distributions of the variable being simulated, the discrete parents, and the discretized versions of the continuous parents. All temporary variables are removed after the simulation is completed. For all simulations, each variable is assigned either continuous or discrete with equal probability. Additionally, discrete variables will have a uniformly chosen number of categories between 2 and 5, inclusive.

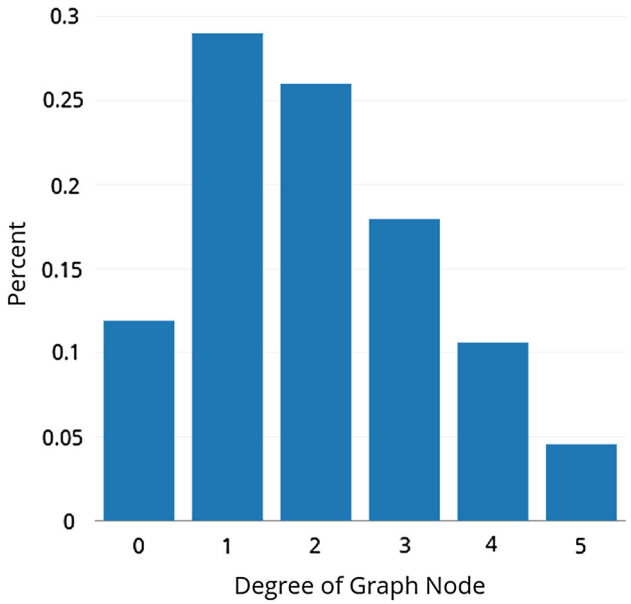

Fig. 2 Distribution of node degrees in graphs of average degree 2

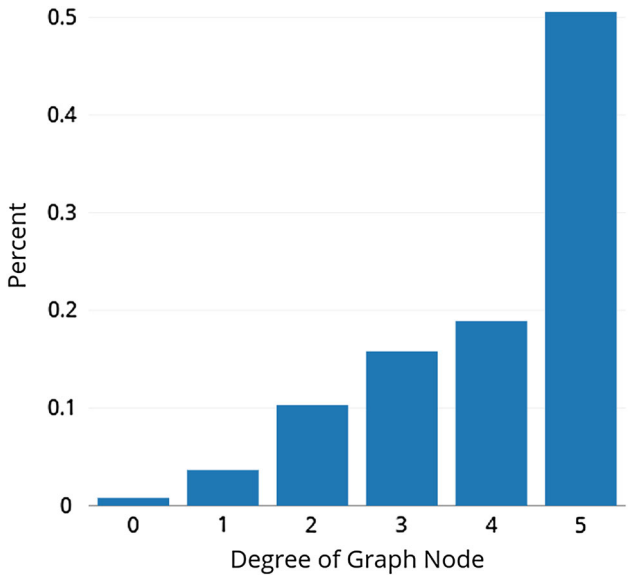

Fig. 3 Distribution of node degrees in graphs of average degree 4

In order to prevent the number of multinomial cells for discrete variables from getting too large, we bound the maximum degree of any node in the generated graph to 5. In our experiments, we tested on graphs of average degree 2 and 4. Figures 2 and 3 show the distribution of the node degrees for different settings of average degree. All simulations and comparison took place within the Tetrad system's algorithm comparison tool [20]. Appendix B contains details about how the data were simulated and the parameters used.

We compare CG with and without the discretization heuristic and MVP with $g(n_p) = 1$, $g(n_p) = \lfloor \log(n_p) \rfloor$ using the following performance measures.

AP - adjacency precision: the ratio of correctly predicted adjacent to all predicted adjacent

AR - adjacency recall: the ratio of correctly predicted adjacent to all true adjacent

AHP - arrowhead precision: the ratio of correctly predicted arrowheads to all predicted arrowheads

AHR - arrowhead recall: the ratio of correctly predicted arrowheads to all true arrowheads (in found adjacencies)

T - elapsed time (seconds)

All results are averaged over 10 randomly simulated graphs and were run on a laptop with an Intel(R) Core I7 @ 3.1 GHz with 16 GB of memory. The results in Tables 1–5 use the same simulated dataset and can be directly compared to each other. The results in Tables 6 and 7 each required a different set of simulation parameters and thus use different simulated datasets. Prior to running tests on any algorithm, all continuous data were standardized to have mean 0 and standard deviation 1. As a search algorithm, we use fGES [18], an optimized version of GES [5]. In general, algorithms in the GES family perform a two phased search. Starting from a completely disconnected graph, the first phase of the search algorithm greedily adds edges until there is no addition that can improve the score. The second phase then removes edges in the same greedy fashion until no more removals can improve the score. At that point, the current graph will be returned.

6.1 The conditional Gaussian approximation

We empirically evaluated the choice of approximating a mixture of Gaussians with a single Gaussian for CG (Assumption 3) in Table 1. We denote the use of a single Gaussian as Approx and the use of the correct mixture calculation as Exact. Originally the results did not appear comparable as the approximate method output a much denser graph than the exact method. In the results shown, we use the binomial structure prior proposed in Sect. 5.1 and achieve comparable results. We see that the approximation performs better in term of precision and comparably in term of recall when compared to the exact method. In the comparisons, we simulate graphs of average degree 2 and 4 with 200 and 1000 samples and 100 measured variables using fGES. Results are given with the binomial structure prior adjustment set to 1.

6.2 Binomial structure prior

We tested the usefulness of the binomial structure prior by simulating 200 and 1000 samples from graphs of average degree 2 and 4 with 100 measured variables using fGES. We compare our scoring functions with and without the binomial structure prior. Additionally, we compare against extended

Table 1 Compares the approximate method to the exact method for CG on graphs of average degree 2 and 4 with 100 measured variables

Sample size		200					1000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	Exact	0.56	**0.56**	0.37	**0.31**	1.03	0.79	0.79	0.65	**0.55**	2.94
	Approx	**0.82**	0.53	**0.75**	0.19	**0.31**	**0.91**	**0.81**	**0.85**	0.49	**0.59**
Avg Deg 4	Exact	0.59	**0.39**	0.44	**0.26**	0.73	0.84	**0.64**	0.73	**0.51**	3.73
	Approx	**0.82**	0.36	**0.69**	0.23	**0.17**	**0.92**	0.62	**0.84**	**0.51**	**0.99**

Sample size is varied to be 200 or 1000 and the binomial structure prior is used with the expected number of parents set to 1
The best results in each group are shown in bold, and all reported results are averaged over 10 repetitions

Table 2 Compares the use of different priors for CG, MVP 1, and MVP log n on graphs of average degree 2 with 100 measured variables

Sample Size		200					1,000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
CG	Uniform	0.54	**0.59**	0.40	**0.38**	0.29	0.76	**0.83**	0.66	**0.59**	0.63
	EBIC	**0.85**	0.45	**0.80**	0.12	0.26	**0.93**	0.78	**0.88**	0.43	**0.53**
	Binomial	0.82	0.53	0.75	0.19	**0.18**	0.91	0.81	0.85	0.49	0.55
MVP 1	Uniform	0.36	**0.57**	0.24	**0.35**	9.32	0.70	**0.81**	0.56	**0.57**	6.43
	EBIC	**0.84**	0.39	**0.69**	0.17	**1.35**	**0.85**	0.71	**0.74**	0.46	**3.53**
	Binomial	0.53	0.53	0.35	0.31	1.53	0.83	0.77	0.70	0.52	3.93
MVP log n	Uniform	0.37	**0.55**	0.23	**0.31**	7.51	0.77	**0.79**	0.60	**0.50**	14.47
	EBIC	**0.84**	0.31	**0.65**	0.09	**2.50**	**0.87**	0.65	**0.73**	0.37	**7.25**
	Binomial	0.52	0.51	0.33	0.28	2.51	0.84	0.76	0.68	0.47	8.54

Sample size is varied to be 200 or 1000. The best results in each group are shown in bold, and all reported results are averaged over 10 repetitions

Table 3 Compares the use of different priors for CG, MVP 1, and MVP log n on graphs of average degree 4 with 100 measured variables

Sample Size		200					1000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
CG	Uniform	0.66	**0.41**	0.52	**0.31**	0.23	0.87	**0.66**	0.79	**0.57**	1.09
	EBIC	**0.85**	0.30	**0.70**	0.16	**0.15**	**0.94**	0.57	**0.86**	0.43	0.92
	Binomial	0.82	0.36	0.69	0.23	0.16	0.92	0.62	0.84	0.51	**0.85**
MVP 1	Uniform	0.45	**0.42**	0.34	**0.30**	16.85	0.85	**0.66**	0.77	**0.56**	7.24
	EBIC	**0.84**	0.27	**0.69**	0.16	**0.98**	**0.92**	0.54	**0.84**	0.43	**4.29**
	Binomial	0.53	0.36	0.39	0.25	6.24	0.90	0.61	0.83	0.51	4.90
MVP log n	Uniform	0.44	**0.37**	0.30	**0.23**	20.70	0.89	**0.62**	0.78	**0.49**	16.74
	EBIC	**0.85**	0.18	**0.64**	0.08	**2.06**	**0.94**	0.46	**0.84**	0.34	**9.25**
	Binomial	0.52	0.33	0.36	0.20	6.50	0.93	0.58	**0.84**	0.46	11.55

Sample size is varied to be 200 or 1000
The best results in each group are shown in bold, and all reported results are averaged over 10 repetitions

Table 4 Compares the use of CG, CGd, MVP 1, and MVP log n in the constraint-based paradigm with α set to 0.001 on graphs of average degree 2 and 4, respectively, with 100 measured variables

Sample Size		200					1000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	CG	0.91	0.40	0.96	0.04	0.57	0.93	0.68	0.93	0.24	1.26
	CGd	0.92	**0.42**	**0.98**	**0.05**	**0.46**	**0.95**	**0.69**	**0.94**	**0.26**	**1.15**
	MVP 1	0.77	0.27	0.67	0.01	1.37	0.88	0.60	0.67	0.15	6.63
	MVP log n	**0.97**	0.29	0.75	0.01	2.03	0.92	0.67	0.68	0.23	16.82
Avg Deg 4	CG	0.93	**0.26**	**0.93**	**0.05**	**0.44**	0.94	**0.50**	0.93	**0.24**	**4.47**
	CGd	0.93	**0.26**	0.89	**0.05**	**0.44**	**0.95**	**0.50**	**0.95**	**0.24**	8.55
	MVP 1	0.81	0.14	0.43	0.01	1.28	0.92	0.41	0.65	0.16	11.21
	MVP log n	**0.98**	0.15	0.56	0.01	2.01	0.93	0.48	0.60	0.21	48.73

Sample size is varied to be 200 or 1000. The best results in each group are shown in bold, and all reported results are averaged over 10 repetitions

Table 5 Compares the use of CG, CGd, MVP 1, LR 1, MVP $\log n$, LR $\log n$, and MN using linear data from graphs of average degree 2 and 4, respectively, with 100 measured variables

Sample Size		200					1000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	CG	0.82	**0.53**	0.75	0.19	0.29	0.91	**0.81**	0.85	0.49	0.56
	CGd	0.90	0.45	0.82	0.17	0.19	0.95	0.77	**0.93**	0.47	0.53
	MVP 1	0.53	**0.53**	0.35	0.31	1.92	0.83	0.77	0.70	**0.52**	4.01
	LR 1	0.53	**0.53**	0.36	**0.32**	18.55	0.84	0.78	0.71	0.51	52.34
	MVP $\log n$	0.52	0.51	0.33	0.28	2.63	0.84	0.76	0.68	0.47	8.55
	LR $\log n$	0.53	0.51	0.34	0.28	34.86	0.87	0.78	0.71	0.49	165.53
	MN	**0.93**	0.41	**0.85**	0.07	**0.09**	**0.97**	0.72	0.90	0.36	**0.48**
Avg Deg 4	CG	0.82	**0.36**	0.69	0.23	0.16	0.92	0.62	0.84	0.51	0.90
	CGd	0.92	0.32	0.80	0.18	0.15	0.96	0.58	**0.91**	0.48	0.73
	MVP 1	0.53	**0.36**	0.39	**0.25**	8.82	0.90	0.61	0.83	0.51	5.20
	LR 1	0.53	**0.36**	0.39	**0.25**	27.22	0.91	**0.63**	0.83	**0.52**	62.08
	MVP $\log n$	0.53	0.33	0.36	0.20	6.25	0.93	0.58	0.84	0.46	12.00
	LR $\log n$	0.53	0.33	0.36	0.20	45.97	0.93	0.59	0.84	0.47	215.93
	MN	**0.93**	0.26	**0.86**	0.07	**0.06**	**0.98**	0.51	0.84	0.36	**0.19**

Sample size is varied to be 200 or 1000 and the binomial structure prior is used with the expected number of parents set to 1. The best results in each group are shown in bold, and all reported results are averaged over 10 repetitions

Table 6 Compares the use of CG, CGd, MVP 1, LR 1, MVP $\log n$, LR $\log n$, and MN using nonlinear data from graphs of average degree 2 and 4, respectively, with 100 measured variables

Sample Size		200					1000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	CG	0.53	0.54	0.35	0.33	0.47	0.58	0.67	0.45	**0.45**	1.91
	CGd	0.82	0.52	0.61	0.23	0.23	0.83	0.68	0.67	0.39	1.03
	MVP 1	0.67	**0.55**	0.45	**0.32**	0.82	0.76	**0.69**	0.58	0.41	3.37
	LR 1	0.67	**0.55**	0.45	0.31	10.93	0.76	0.68	0.58	0.41	46.03
	MVP $\log n$	0.75	0.54	0.51	0.29	1.42	0.87	0.67	0.71	0.42	6.90
	LR $\log n$	0.75	0.54	0.51	0.29	22.07	0.87	0.67	0.71	0.42	158.09
	MN	**0.95**	0.49	**0.96**	0.05	**0.11**	**0.96**	0.65	**0.83**	0.31	**0.24**
Avg Deg 4	CG	0.48	0.34	0.34	**0.24**	0.57	0.70	0.51	0.60	0.41	1.38
	CGd	0.81	0.32	0.64	0.19	0.33	0.86	0.51	0.77	0.39	1.00
	MVP 1	0.71	**0.35**	0.53	0.23	1.03	0.83	0.52	0.73	0.41	3.82
	LR 1	0.72	**0.35**	0.54	0.23	12.50	0.82	0.52	0.72	0.40	48.62
	MVP $\log n$	0.81	0.33	0.63	0.23	1.88	0.93	0.52	**0.84**	0.41	8.29
	LR $\log n$	0.81	0.34	0.63	0.23	35.04	0.93	**0.53**	**0.84**	**0.42**	191.39
	MN	**0.95**	0.27	**0.85**	0.05	**0.06**	**0.95**	0.44	0.75	0.29	**0.20**

Sample size is varied to be 200 or 1,000 and the binomial structure prior is used with the expected number of parents set to 1. The best results in each group are shown in bold, and all reported results are averaged over 10 repetitions

BIC (EBIC). In these experiments, the binomial structure prior is set to 1 and EBIC's gamma parameter is set to 0.5 upon suggestion of the authors [4]. In Tables 2 and 3 report findings when the average degrees of the graphs are 2 and 4, respectively.

While we set the binomial structure prior's parameter to 1 for the experiments presented in this paper, it is important to note that this parameter can be chosen to be any value greater than 0. By varying the expected number parents, we can influence how sparse or dense the output graph will be. The choice of a low value results in a relatively sparse graph and a high value in a denser one.

From Table 2 and 3, for both the binomial structure prior and EBIC, we see boosts in precision with a reduction in recall. Additionally, we see vast reductions in the computation times. In general, EBIC seems to work better with small sample sizes. This makes sense, since EBIC is aimed at the small-n-large-P situation. However, for 1000 samples, we

Table 7 Compares the use of CG, CGd, MVP 1, MVP log n, and MN using linear data from graphs of average degree 2 and 4, respectively, with 500 measured variables

Sample Size		200					1000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	CG	0.67	**0.51**	0.48	**0.28**	2.11	0.88	**0.77**	0.81	0.50	**7.28**
	CGd	0.86	0.44	0.75	0.19	**1.74**	0.94	0.71	**0.89**	0.46	8.42
	MVP 1	0.40	0.49	0.24	0.27	56.83	0.81	0.73	0.67	**0.51**	44.68
	MVP log n	0.40	0.46	0.23	0.24	81.20	0.84	0.72	0.68	0.46	96.91
	MN	**0.93**	0.39	**0.84**	0.07	1.79	**0.97**	0.67	**0.89**	0.36	14.74
Avg Deg 4	CG	0.75	**0.35**	0.59	**0.25**	2.21	0.91	**0.61**	0.84	**0.51**	**9.70**
	CGd	0.88	0.31	**0.78**	0.19	2.22	0.95	0.58	**0.90**	0.48	16.59
	MN	**0.93**	0.26	0.77	0.07	**1.32**	**0.98**	0.51	0.86	0.37	12.43

Sample size is varied to be 200 or 1000 and the binomial structure prior is used with the expected number of parents set to 1. Omitted scores failed to return a result. The best results in each group are shown in bold, and all reported results are averaged over 10 repetitions

find the binomial structure prior performs relatively well. We use the binomial structure prior for the remainder of our score based experiments.

6.3 Conditional independence tests

We tested the usefulness of the CG and MVP scores as conditional independence tests by simulating, 200 and 1000 samples from graphs of average degree 2 and 4 with 100 measured variables. As a search algorithm, we used CPC Stable [19], which is a modified version of PC [26] that treats ambiguous triples as noncolliders. For independence testing, we set the significance level $\alpha = 0.001$. Here we also use the discretization heuristic with $b = 3$ for CG, denoted CGd, however we do not use a structure prior since we are no longer scoring a full Bayesian network in this paradigm. We did not include results for a version of MVP which uses the discretization heuristic because it had little effect. The results are shown in Table 4.

In general, we find that our methods perform better as scores, but still perform reasonably well as conditional independence tests. This is promising for use in algorithms, such as FCI, that model the possibility of latent confounding [26].

6.4 Tests against baseline scores

We used two simple baseline scores as a point of comparison for our methods. The first, which we denote MN, uses multinomial scoring for all cases. In order to do so, we essentially extend the discretization heuristic to the continuous child case so that we are always scoring with a multinomial. The second, which we denote as LR, uses partitioned linear regression in the continuous child case and partitioned logistic regression in the discrete child case. In our experiments, we applied Lib Linear [7], a widely used and efficient toolkit for logistic regression which uses truncated Newton optimization [9]. In a recent paper, Zaidi et al. [27] note that among the many optimization methods that have been evaluated, the truncated Newton method has been shown to converge the fastest, which provides support that Lib Linear is a competitive, state-of-the-art method to apply in our evaluation, as a baseline point of comparison. As with MVP, the appended term on LR denotes the maximum polynomial degree of the regressors.

We compared CG, CGd, MVP 1, LR 1, MVP log n, LR log n, and MN by simulating 200 and 1000 samples from graphs of average degree 2 and 4 with 100 measured variables. As a search algorithm, we again used fGES. Here we also use the discretization heuristic with $b = 3$ for CGd and the binomial structure prior set to 1 for all scores. Additionally, boldface text highlights the best performing score for each statistic for each group in the table. The results are shown in Tables 5 and 6. For the results in Table 6, we extended our method for simulating data. Since MVP is designed to handle nonlinearity, while CG is not, we modified the continuous child phase of data generation to allow for nonlinearities. To do so, we additionally generate second, and third order polynomial terms. However, because of the nature of these nonlinear functions, the values of the data often become unmanageably large. To correct for this issue, we resample a variable with square-root and cube-root relationships if the values are too large. Appendix B contains details about how the data were simulated and the parameters used.

Table 5 shows the results when using linearly generated data and 100 variables. As a general pattern, MN had better precision than the CG methods which had better precision than the MVP and LR methods. For recall, just the opposite pattern tended to occur. In terms of timing, in general, MN was faster than the CG methods, which were faster than the MVP methods, which were considerably faster than LR.

Table 6 shows the results when using nonlinearly generated data and 100 variables. MN tended to have a higher precision than the MVP and LR methods, which often had

higher precision than the CG methods. The relatively good performance of MN is surprising; although multinomial distributions can represent nonlinear relationships, the process of discretizing continuous variables loses information; the manner in which we generated the data (see the beginning of Sect. 6) when there is a discrete child and continuous parents may play a role in producing this result. The relatively better precision performance of the MVP methods compared to CG methods is not surprising, given that MVP can model nonlinear relationships and CG cannot. In terms of recall, MVP and CG performed comparably, while both performed better than MN. The relative timing results in Table 6 are similar to those in Table 5.

In Tables 5 and 6, there is almost no difference in precision and recall performance between MVP and LR. This result is understandable, since MVP is using an approximation to logistic regression in the case of a discrete child with continuous parents and performing all other cases identically. However, MVP is often tenfold or more faster than LR.

Table 7 shows the results of assessing the scalability of the methods. We simulated linear data on 500 variables. For average degree 4, no MVP results are shown because our machine ran out of memory while searching. Also, LR is not included at all in Table 7, because LR (as implemented) cannot scale to networks of this size due to time complexity. Table 7 shows that the CG methods had similar precision to MN, which generally had better precision than MVP. For the results shown, the recall of the MVP and CG methods were similar, which were generally better than the recall for MN. MN and the CG methods had similar timing results, which were faster than those of MVP.

In Table 7, we see that the CG and MVP methods are capable of scaling to graphs containing 500 measured variables, albeit sparse ones. CG was able to scale to a slightly denser graph of 500 variables. In general, we see the same performance on these larger networks as before on the networks of 100 measured variables. Additionally, for the smaller sample size of 200, MN performed comparably to CDd, but with a slightly higher precision and lower recall.

7 Conclusions

This paper introduces two novel scoring methods for learning Bayesian networks in the presence of both continuous and discrete variables. One of the methods scales to networks of 500 variables or more on a laptop. We introduce a structure prior for learning large networks and find that using a structure prior with BIC generally leads to relatively good network discovery performance, while requiring considerably less computation time. We showed how the CG and MVP scoring methods are readily adapted as conditional independence tests for constraint-based methods to support future use in algorithms such as FCI.

The MVP and LR methods had precision and recall results that were almost identical; however, MVP was considerably faster than LR. Such a speed difference is particularly important when performing Bayesian network learning, where the scoring method must be applied thousands of times in the course of learning a network. Using a different implementation of LR might affect the magnitude of this speed difference, but for the reasons we give in Sect. 4.3, we would not expect it to lead to LR becoming faster than MVP.

The fully discrete approach, MN, performed surprisingly well in our experiments in terms of precision and speed, although recall was often lower, and sometimes much lower, than that of CG and MVP.

The results of the experiments reported here support using CG when recall is a priority and the relationships are linear. If the relationships are likely to be nonlinear and recall remains a priority, then we suggest using MVP when there 100 or fewer variables and using CG when there are 500 variables or more. If precision is a priority, then our results support using MN.

All algorithms and simulation reported here were implemented in the Tetrad system [22], and the code is available in the Tetrad repository on GitHub.[1]

There are several directions for future work. First, we would like to apply the methods to real datasets for which knowledge of the causal relationships is available. Second, we would like to expand the CG and MVP methods to model ordinal discrete variables. Although the nominal discrete variables that these methods currently model can represent ordinal variables, we would expect the methods to have greater power when they take advantage of knowledge about particular discrete variables being ordinal versus nominal. Third, we would like to further explore how to adaptively discretize variables in the MN method in order to improve its recall, while not substantially reducing its precision. Fourth, we would like to investigate alternative basis functions to polynomials for the MVP method.

Acknowledgements We thank Clark Glymour, Peter Spirtes, Takis Benos, Dimitrios Manatakis, and Vineet Raghu for helpful discussions about the topics in this paper. We also thank the reviewers for their helpful comments.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

[1] https://github.com/cmu-phil/tetrad.

Appendix A

Proposition 1 *The Conditional Gaussian Score is score equivalent.*

Proof Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be directed acyclic graphs with Conditional Gaussian scores $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Further, let $\mathcal{G}_1 \neq \mathcal{G}_2$, but $\mathcal{G}_1$ and $\mathcal{G}_2$ in the same Markov equivalence class. □

Remove all shared local components between $\mathcal{S}_1$ and $\mathcal{S}_2$ and the corresponding edges in $\mathcal{G}_1$ and $\mathcal{G}_2$. Call the newly pruned scores $\mathcal{S}_1'$ and $\mathcal{S}_2'$ and the newly pruned graphs $\mathcal{G}_1'$ and $\mathcal{G}_2'$, respectively. Note that we have removed all initial unshielded colliders common to both $\mathcal{G}_1$ and $\mathcal{G}_2$. Additionally, it follows from Meek's rules that $\mathcal{G}_1'$ and $\mathcal{G}_2'$ must contain no unshielded colliders since any component which could have become an unshielded collider is necessarily shared between graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ and thus pruned [14].

Since $\mathcal{G}_1'$ and $\mathcal{G}_2'$ are acyclic graphs without any unshielded colliders, we can represent them both as a join tree of cliques. Further, they share the same skeleton because they come from the same Markov equivalence class, and thus, they can be represented by the same join tree of cliques.

It follows from Pearl, Probabilistic Reasoning in Intelligent Systems 3.2.4, Theorem 8, that the distribution encoded by $\mathcal{G}_1'$ and $\mathcal{G}_2'$ can be written as a product of the distributions of the cliques of $\mathcal{G}_1'$ and $\mathcal{G}_2'$ divided by a product of the distributions of their intersections [16]. Therefore, when we calculate $\mathcal{S}_1'$ and $\mathcal{S}_2'$, we can use the same ratio of joint distributions to obtain the log likelihood and degrees of freedom terms. Hence $\mathcal{S}_1' = \mathcal{S}_2'$ and therefore, after adding back the shared local components, we have $\mathcal{S}_1 = \mathcal{S}_2$.

Proposition 2 *If the approximating polynomial from Weierstrass Approximation Theorem for the data generating function is a polynomial of degree d, and the maximum-degree polynomial used by MVP is at least d, then the MVP score will be consistent in the large sample limit.*

Proof Let $\boldsymbol{p}$ be the approximating polynomial(s) from Weierstrass Approximation Theorem [11]. Assuming the least squares estimate(s) contains the same polynomials degrees as $\boldsymbol{p}$, the least squares estimate will converge to $\boldsymbol{p}$ as the number of samples $n \to \infty$ [1]. Therefore, by Weierstrass Approximation Theorem, the least squares estimate(s) will converge to the true data generating function(s). Accordingly, the log likelihood term of MVP will be maximal for any model that has either been correctly specified or over specified, where by over over specified we are referring to a model containing all the true parameters and more. □

However for any over specified model, the parameter penalty will necessarily be larger and hence the MVP score in that case will be lower for the over specified model.

Additionally, in the case of an underspecified model, note that the parameter penalty term is of order $O(\log n)$ while the likelihood term is of order $O(n)$. This means that in the large sample limit, when comparing an underspecified model to any model containing the correctly specified model, the MVP score will be lower for the under specified model since the log likelihood is not maximal while in the other case it is.

Therefore, the MVP score is consistent.

Proposition 3 *The approximated values from least squares regression sum to one.*

Proof Let the terms in the below equation be defined according to Sect. 4.3.

$$\begin{aligned}\sum_{h=1}^{d} \boldsymbol{X}_p \hat{\boldsymbol{\beta}}_h &= \boldsymbol{X}_p (\boldsymbol{X}_p^T \boldsymbol{X}_p)^{-1} \boldsymbol{X}_p^T \sum_{h=1}^{d} \mathbf{1}_{\{y_p = h\}} \\ &= \boldsymbol{X}_p (\boldsymbol{X}_p^T \boldsymbol{X}_p)^{-1} \boldsymbol{X}_p^T \mathbf{1}_p \\ &= \mathbf{1}_p\end{aligned}$$

For the last step, we use that $\mathbf{1}_p$ is in the column space of $\boldsymbol{X}_p$ and is thus projected to itself. □

Proposition 4 *If the approximating polynomial from Weierstrass Approximation Theorem for the data generating function is a polynomial of degree d, and the maximum-degree polynomial used by MVP is at least d, then the least squares approximations for probability mass functions will be strictly nonnegative in the large sample limit.*

Proof Let f be a generating component of a conditional probability mass function. Since the Weierstrass Approximation Theorem is satisfied by the assumptions of MVP, there must exists a polynomial p such that for every $\epsilon > 0$ and all $x \in [a, b]$, we have $|f(x) - p(x)| < \epsilon$ [11]. □

For $x \in [a, b]$ where $p(x) \geq f(x)$, $p(x)$ is trivially nonnegative since $f(x) > 0$.

For $x \in [a, b]$ where $p(x) < f(x)$, let $m = f(x)$ and choose $\epsilon = \frac{m}{2}$. Then,

$$\begin{aligned}|f(x) - p(x)| &< \epsilon \\ f(x) - p(x) &< \epsilon \\ p(x) &> f(x) - \epsilon \\ p(x) &> m - \frac{m}{2} \\ p(x) &> 0\end{aligned}$$

since $m > 0$.

Assuming the least squares estimate(s) contains the same polynomials degrees as p, the least squares estimate will converge to p as the number of samples $n \to \infty$ [1]. Thus, as the number of samples $n \to \infty$, the least squares approximations are strictly nonnegative.

Appendix B

In this appendix, we detail the parameters used for simulation of the data. Each parameter will be followed by the values we used in simulation and a short description. We split the parameters into 3 groups: general parameters used across all simulations, parameters specific to linear simulation, and parameters specific to nonlinear simulation.

General Parameters

numRuns: *10* - number of runs
numMeasures: *100, 500* - number of measured variables
avgDegree: *2, 4* - average degree of graph
sampleSize: *200, 1000* - sample size
minCategories: *2* - minimum number of categories
maxCategories: *5* - maximum number of categories
percentDiscrete: *50* - percentage of discrete variables (0 - 100) for mixed data
differentGraphs: *true* - true if a different graph should be used for each run
maxDegree: *5* - maximum degree of the graph
maxIndegree: *5* - maximum indegree of graph
maxOutdegree: *5* - maximum outdegree of graph
coefSymmetric: *true* - true if negative coefficient values should be considered

Linear Parameters

varLow: *1* - low end of variance range
varHigh: *3* - high end of variance range
coefLow: *0.05* - low end of coefficient range
coefHigh: *1.5* - high end of coefficient range
meanLow: *-1* - low end of mean range
meanHigh: *1* - high end of mean range

Nonlinear Parameters

dirichlet: *0.5* - alpha parameter for Dirichlet to draw multinomials
interceptLow: *1* - low end of intercept range
interceptHigh: *2* - high end of intercept range
linearLow: *1.0* - low end of linear coefficient range
linearHigh: *2.0* - high end of linear coefficient range
quadraticLow: *0.5* - low end quadratic coefficient range
quadraticHigh: *1.0* - high end of quadratic coefficient range
cubicLow: *0.2* - low end of cubic coefficient range
cubicHigh: *0.3* - high end of cubic coefficient range
varLow: *0.5* - low end of variance range
varHigh: *0.5* - high end of variance range

References

1. Anderson, T., Taylor, J.B.: Strong consistency of least squares estimates in normal linear regression. The Ann. Stat., pp. 788–790 (1976)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Berlin (2006)
3. Bøttcher, S.G.: Learning bayesian networks with mixed variables. Ph.D. thesis, Aalborg University (2004)
4. Chen, J., Chen, Z.: Extended bic for small-n-large-p sparse glm. Statistica Sinica pp. 555–574 (2012)
5. Chickering, D.M.: Optimal structure identification with greedy search. J. Mach. Learn. Res. **3**, 507–554 (2002)
6. Daly, R., Shen, Q., Aitken, S.: Review: learning bayesian networks: approaches and issues. The Knowl. Eng. Rev. **26**(2), 99–157 (2011)
7. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
8. Heckerman, D., Geiger, D.: Learning bayesian networks: a unification for discrete and gaussian domains. In: Proceedings of Conference on Uncertainty in Artificial Intelligence, pp. 274–284. Morgan Kaufmann Publishers Inc. (1995)
9. Hsia, C.Y., Zhu, Y., Lin, C.J.: A study on trust region update rules in newton methods for large-scale linear classification. In: Asian Conference on Machine Learning, pp. 33–48 (2017)
10. Huang, T., Peng, H., Zhang, K.: Model selection for gaussian mixture models. Statistica Sinica **27**(1), 147–169 (2017)
11. Jeffreys, H., Jeffreys, B.: Weierstrasss theorem on approximation by polynomials. Methods of Mathematical Physics pp. 446–448 (1988)
12. Peters, J., Janzing, D., Schölkopf, B.: Elements of Causal Inference: Foundations and Learning Algorithms. The MIT Press, Cambridge (2017)
13. McGeachie, M.J., Chang, H.H., Weiss, S.T.: Cgbayesnets: conditional gaussian bayesian network learning and inference with mixed discrete and continuous data. PLoS Comput. Biol. **10**(6), e1003676 (2014)
14. Meek, C.: Complete orientation rules for patterns (1995)
15. Monti, S., Cooper, G.F.: A multivariate discretization method for learning bayesian networks from mixed data. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence, pp. 404–413. Morgan Kaufmann Publishers Inc. (1998)
16. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, Burlington (1988)
17. Raftery, A.E.: Bayesian model selection in social research. Sociol. Methodol., pp. 111–163 (1995)
18. Ramsey, J., Glymour, M., Sanchez-Romero, R., Glymour, C.: A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. Int. J. Data Sci. Anal., pp. 1–9 (2016)
19. Ramsey, J., Zhang, J., Spirtes, P.: Adjacency-faithfulness and conservative causal inference. In: Proceedings of Conference on Uncertainty in Artificial Intelligence, pp. 401–408. AUAI Press, Arlington, Virginia (2006)
20. Ramsey, J.D., Malinsky, D.: Comparing the performance of graphical structure learning algorithms with tetrad. arXiv preprint arXiv:1607.08110 (2016)
21. Romero, V., Rumí, R., Salmerón, A.: Learning hybrid bayesian networks using mixtures of truncated exponentials. Int. J. Approx. Reason. **42**(1–2), 54–68 (2006)
22. Scheines, R., Spirtes, P., Glymour, C., Meek, C., Richardson, T.: The tetrad project: constraint based aids to causal model specification. Multivar. Behav. Res. **33**(1), 65–117 (1998)

23. Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)
24. Sedgewick, A.J., Shi, I., Donovan, R.M., Benos, P.V.: Learning mixed graphical models with separate sparsity parameters and stability-based model selection. BMC Bioinform. **17**(Suppl 5), 175 (2016)
25. Sokolova, E., Groot, P., Claassen, T., Heskes, T.: Causal discovery from databases with discrete and continuous variables. In: European Workshop on Probabilistic Graphical Models, pp. 442–457. Springer (2014)
26. Spirtes, P., Glymour, C.N., Scheines, R.: Causation, Prediction, and Search. MIT Press, Cambridge (2000)
27. Zaidi, N.A., Webb, G.I.: A fast trust-region newton method for softmax logistic regression. In: Proceedings of the 2017 SIAM International Conference on Data Mining, pp. 705–713. SIAM (2017)