Journal of Machine Learning Research 8 (2007) 1799-1833 Submitted 09/06; Revised 5/07; Published 8/07

“Ideal Parent” Structure Learning for Continuous Variable Bayesian Networks

Gal Elidan GALEL@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305, USA

Iftach Nachman INACHMAN@CGR.HARVARD.EDU
FAS Center for Systems Biology
Harvard University
Cambridge, MA 02138, USA

Nir Friedman NIR@CS.HUJI.AC.IL
School of Computer Science and Engineering
Hebrew University
Jerusalem 91904, Israel

Editor: David Maxwell Chickering

Abstract

Bayesian networks in general, and continuous variable networks in particular, have become increasingly popular in recent years, largely due to advances in methods that facilitate automatic learning from data. Yet, despite these advances, the key task of learning the structure of such models remains a computationally intensive procedure, which limits most applications to parameter learning. This problem is even more acute when learning networks in the presence of missing values or hidden variables, a scenario that is part of many real-life problems. In this work we present a general method for speeding structure search for continuous variable networks with common parametric distributions. We efficiently evaluate the approximate merit of candidate structure modifications and apply time consuming (exact) computations only to the most promising ones, thereby achieving significant improvement in the running time of the search algorithm. Our method also naturally and efficiently facilitates the addition of useful new hidden variables into the network structure, a task that is typically considered both conceptually difficult and computationally prohibitive. We demonstrate our method on synthetic and real-life data sets, both for learning structure on fully and partially observable data, and for introducing new hidden variables during structure search.

Keywords: Bayesian networks, structure learning, continuous variables, hidden variables

1. Introduction

Probabilistic graphical models have gained wide-spread popularity in recent years with the advance of techniques for learning these models directly from data. The ability to learn allows us to overcome lack of expert knowledge about domains and adapt models to a changing environment, and can also lead to scientific discoveries. Indeed, Bayesian networks in general, and continuous variable

A preliminary version of this paper appeared in the Proceedings of the Twentieth Conference on Uncertainty in Artificial, 2004 (UAI ’04).

networks in particular, are now being used in a wide range of applications, including fault detection (e.g., U. Lerner and Koller, 2000), modeling of biological systems (e.g., Friedman et al., 2000) and medical diagnosis (e.g., Shwe et al., 1991).

A key task in learning these models from data is adapting the structure of the network based on observations. This NP-complete problem (Chickering, 1996a) is typically treated as a combinatorial optimization problem that is addressed by heuristic search procedures, such as greedy hill climbing. This procedure examines local modifications to single edges at each step, evaluates them using some score, and proceeds to apply the one that leads to the largest improvement in score, until a local maximum is reached. Even with this simple approach structure learning is computationally challenging for all but small networks due to the large number of possible modifications that can be evaluated, and the cost of evaluating each one. To make things worse, the problem is even harder in the (realistic) presence of missing values, as non-linear optimization is required to evaluate different structure modification candidates during the search. Learning is particularly problematic when we also want to allow for hidden variables and want to effectively add them during the learning process. Thus, in practice, most applications are still limited to parameter estimation.

Of particular interest to us is learning continuous variable networks, which are crucial for a wide range of real-life applications. One case that received scrutiny in the literature is learning *linear Gaussian* networks (Geiger and Heckerman, 1994; Lauritzen and Wermuth, 1989). In this case, we can use sufficient statistics to summarize the data, and a closed form equation to evaluate the score of candidate structure modifications. In general, however, we are also interested in non-linear interactions. These do not have sufficient statistics, and require applying parameter optimization to evaluate the score of candidate structures. These difficulties severely limit the applicability of standard heuristic structure search procedures to rich non-linear models.

In this work, we present a general method for speeding structure search for continuous variable networks. In contrast to innovative structure learning methods that modify the space explored by the search algorithm (e.g., Chickering, 1996b; Moore and Wong, 2003; Teyssier and Koller, 2005), our method leverages on the parametric structure of the conditional distributions in order to efficiently approximate the benefit of an *individual* structure candidate. As such, our method can be used to speed up many existing structure learning algorithms and heuristics.

The basic idea is straightforward and is inspired from the notion of *residues* in regression (McCullagh and Nelder, 1989). For each variable, we construct an *ideal parent profile* of a new hypothetical parent that would lead to the best possible prediction of that variable. Intuitively, a candidate parent of a variable is useful if it is similar to the ideal parent. Using basic principles, we derive a similarity measure for efficiently comparing a candidate parent to the ideal profile. We show that this measure approximates the improvement in score that would result from the addition of that parent to the network structure. This provides us with a fast method for scanning many potential parents and focuses more careful evaluation (exact scoring) on a smaller number of promising candidates.

The ideal parent profiles we construct during search also provide new leverage on the problem of introducing new hidden variables during structure learning. Basically, if the ideal parent profiles of several variables are sufficiently similar, and are not similar to one of their current parents, we can consider adding a new hidden variable that serves as a parent of all these variables. The ideal profile allows us to estimate the impact this new variable will have on the score, and suggest the values it takes in each instance. The method therefore provides a guided approach for introducing new variables during search and allows for contrasting them with alternative search steps in a computationally efficient manner.

We apply our method using linear Gaussian and non-linear Sigmoid Gaussian conditional probability distributions to several tasks: learning structure with complete data; learning structure with missing data; and learning structure while allowing for the automatic introduction of new hidden variables. We evaluate all tasks on both realistic synthetic experiments and real-life problems in the field of computational biology.

The rest of the paper is structured as follows: In Section 2 we provide a brief summary of continuous variable networks. In Section 3 we present the "Ideal Parent" concept as it applies to the simple case of linear Gaussian models. In Section 4 we discuss how our method is used within a structure learning algorithm. In Section 5 we show how our method can be leveraged in order to introduce new useful hidden variables during learning, and in Section 6 we discuss the computational modifications needed to address both the presence of missing values and hidden variables. In Section 7 we show how our entire framework can be generalized to the challenging case of more general non-linear distributions. In Section 8 we present a further extension to conditional probability distributions that use non-additive noise models. In Section 9 we present our experimental results for both synthetic and real-life data. We conclude with a discussion of related works and future directions in Section 10.

2. Continuous Variable Networks

Consider a finite set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of random variables. A *Bayesian network* (BN) is an annotated directed acyclic graph G that represents a joint probability distribution over $\mathcal{X}$. The nodes of the graph correspond to the random variables and are annotated with a conditional probability density (CPD) of the random variable given its parents $\mathbf{U_i}$ in the graph G. The joint distribution is the product over families (variable and its parents)

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \mathbf{U_i}).$$

The graph G represents independence properties that are assumed to hold in the underlying distribution: Each X_i is independent of its non-descendants given its parents $\mathbf{U_i}$.

Unlike the case of discrete variables, when the variable X and some or all of its parents are real valued, there is no representation that can capture all conditional densities. A common choice is the use of *linear Gaussian* conditional densities (Geiger and Heckerman, 1994; Lauritzen and Wermuth, 1989), where each variable is a linear function of its parents with Gaussian noise. When all the variables in a network have linear Gaussian conditional densities, the joint density over $\mathcal{X}$ is a multivariate Gaussian (Lauritzen and Wermuth, 1989). In many real world domains, such as in neural or gene regulation network models, the dependencies are known to be non-linear (for example, a saturation effect is expected). In these cases, we can still use Gaussian conditional densities, but now the mean of the density is expressed as a non-linear function of the parents (for example, a sigmoid).

Given a training data set $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$, where the mth instance $\mathbf{x}[m]$ assigns values to the variables in $\mathcal{X}$, the problem of learning a Bayesian network is to find a structure and parameters that maximize the likelihood of $\mathcal{D}$ given the graph, typically along with some regularization constraints. Given a data set $\mathcal{D}$ and a network structure G, we define

$$\ell(\mathcal{D} : G, \theta) = \log P(\mathcal{D} : G, \theta) = \sum_{m} \log P(\mathbf{x}[m] : G, \theta)$$

Algorithm 1: Greedy Hill-Climbing Structure Search for Bayesian Networks

```
Input  : $\mathcal{D}$    // training set
         $\mathcal{G}_0$   // initial structure
Output : A final structure G

$\mathcal{G}_{best} \leftarrow \mathcal{G}_0$
repeat
    G$\leftarrow \mathcal{G}_{best}$
    foreach Operator Add,Delete,Reverse,Replace edge in G do
        if Operator does not create a directed cycle then
            $\mathcal{G}' \leftarrow$ ApplyOperator($G$)
            if Score($\mathcal{G}'$ : $\mathcal{D}$) > Score($\mathcal{G}_{best}$ : $\mathcal{D}$) then
                $\mathcal{G}_{best} \leftarrow \mathcal{G}'$
            end
        end
    end foreach
until $\mathcal{G}_{best} == G$
return $\mathcal{G}_{best}$
```

to be the log-likelihood function, where θ are the model parameters. In estimating the *maximum likelihood* parameters of the network, we aim to find the parameters $\hat{\theta}$ that maximize this likelihood function. When the data is *complete* (all variables are observed in each instance), the log-likelihood can be rewritten as a sum of *local* likelihood functions,

$$\ell(\mathcal{D} : G, \theta) = \sum \ell_i(\mathcal{D} : \mathbf{U_i}, \theta_i)$$

where $\ell_i(\mathcal{D} : \mathbf{U_i}, \theta_i)$ is a function of the choice of $\mathbf{U_i}$ and the parameters θ_i of the corresponding CPD: it is the log-likelihood of regressing X_i on $\mathbf{U_i}$ in the data set with the particular choice of CPD. Due to this decomposition, we can find the maximum likelihood parameters of each CPD independently by maximizing the local log-likelihood function. For some CPDs, such as linear Gaussian ones, there is a closed form expression for the maximum likelihood parameters. In other cases, finding these parameters is a continuous optimization problem that is typically addressed by gradient based methods.

Learning the structure of a network is a significantly harder task. The common approach is to introduce a scoring function that balances the likelihood of the model and its complexity and then attempt to maximize this score using a heuristic search procedure that considers local changes (e.g., adding and removing edges). A commonly used score is the *Bayesian Information Criterion* (BIC) score (Schwarz, 1978)

$$BIC(\mathcal{D}, G) = \max_{\theta} \ell(\mathcal{D} : G, \theta) - \frac{\log M}{2} \mathrm{Dim}[G] \quad (1)$$

where M is the number of instances in $\mathcal{D}$, and $\mathrm{Dim}[G]$ is the number of parameters in G. The BIC score is actually an approximation to the more principled full Bayesian score, that integrates over all possible parameterizations of the CPDs. While a closed form for the Bayesian score, with a suitable prior, is known for Gaussian networks (Geiger and Heckerman, 1994), numerical computation of this score is extremely demanding for the non-linear case. Thus, we adopt the common approach and focus on the BIC approximation from here on.

A common search procedure for optimizing the score is the greedy hill-climbing procedure outlined in Algorithm 1. This procedure can be augmented with mechanisms for escaping local maxima, such as random walk perturbations upon reaching a local maxima (also known as random restarts), and using a TABU list (Glover and Laguna, 1993).

3. The “Ideal parent” Concept

Our goal is to speed up a generic structure search algorithm for a Bayesian network with continuous variables. The complexity of any such algorithm is rooted in the need to score each candidate structure change, which in turn may require non-linear parameter optimization. Thus, we want to somehow efficiently approximate the benefit of each candidate and score only the most promising of these candidates. The manner in which this helps us to discover new hidden variables will become evident in Section 5.

3.1 Basic Framework

Consider adding Z as a new parent of X whose current parents in the network are $\mathbf{U}$. Given a training data $\mathcal{D}$ of M instances, to evaluate the change in score, when using the BIC score of Eq. (1), we need to compute the change in the log-likelihood

$$\Delta_{X|\mathbf{U}}(Z) = \max_{\theta_{X|\mathbf{U},Z}} \ell_X(\mathcal{D} : \mathbf{U} \cup \{Z\}, \theta_{X|\mathbf{U},Z}) - \ell_X(\mathcal{D} : \mathbf{U}, \widehat{\theta}_{X|\mathbf{U}}) \tag{2}$$

where $\widehat{\theta}_{X|\mathbf{U}}$ are the maximum likelihood parameters of X given $\mathbf{U}$ and $\theta_{X|\mathbf{U},Z}$ are the parameters for the family where Z is an additional parent of X. The change in the BIC score is this difference combined with the change in the model complexity penalty terms. Thus, to evaluate this difference, we need to compute the maximum likelihood parameters of X given the new choice of parents. Our goal is to speed up this computation.

The basic idea of our method is straightforward. For a given variable, we want to construct a hypothetical ideal parent Y that would best predict the variable. We will then compare each existing candidate parent Z to this imaginary one using a similarity measure $C(\vec{y}, \vec{z})$ (which we describe below). Finally, we will fully score only the most promising candidates: those that are most similar to the ideal parent. Figure 1 illustrates this process. In order for this approach to be beneficial, we want the similarity score to approximate the actual change in likelihood defined in Eq. (2). Furthermore, we want to be able to compute the similarity measure in a fraction of the time it takes to fully score a candidate parent.

3.1.1 CONDITIONAL PROBABILITY DISTRIBUTION

To make our discussion concrete, we focus on networks where we represent X as a function of its parents $\mathbf{U} = \{U_1, \ldots, U_k\}$ with a conditional probability distribution (CPD) that has the following general form:

$$X = g(\alpha_1 \mathbf{u}_1, \ldots, \alpha_k \mathbf{u}_k : \theta) + \varepsilon \tag{3}$$

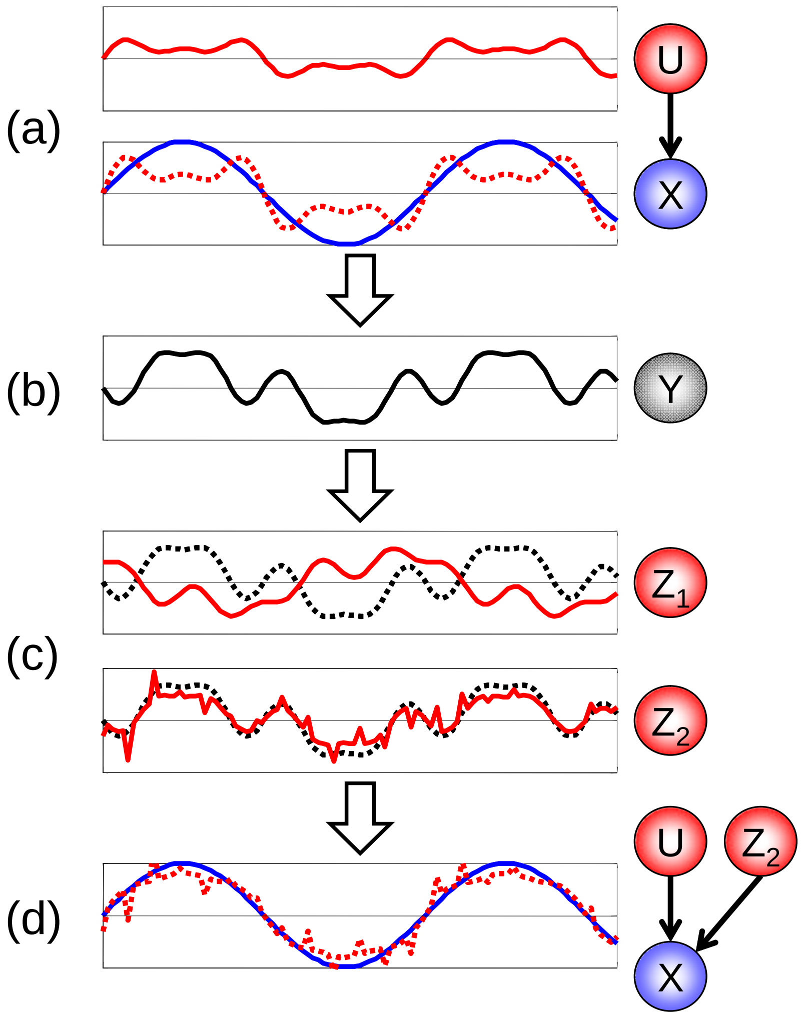


Figure 1: **The "Ideal Parent" Concept**: Illustration of the "Ideal Parent" approach for a variable with a single parent U and a linear Gaussian conditional distribution. The top panel of (a) shows the profile (assignment in all instances) of the parent. The panel below shows the profile of the child node along with the profile predicted for the child based on its parent (dotted red). (b) shows the profile of the ideal hypothetical parent that would lead to zero error in prediction of the child variable if added to the current model. In the linear Gaussian case, this profile is simply the residual of the two curves shown in (a). (c) shows the profiles of two candidate parents, compared to the profile of the ideal parent (dotted black). (d) shows the child profile along with its prediction based on the original parent *and* the new chosen parent from the candidate in (c) that was most similar to the ideal profile of (b). Note that the prediction is not perfect as the profile of the parent chosen does not, in general, match the profile of the ideal parent exactly.

where g is a *link function* that integrates the contributions of the parents with additional parameters θ, α_i that are scale parameters applied to each of the parents, and ε that is a noise random variable with zero mean. In the following discussion, we assume that ε is Gaussian with variance σ^2.

When the function g is the sum of its arguments, this CPD is the standard linear Gaussian CPD. However, we can also consider non-linear choices of g. For example,

$$g(\alpha_1 \mathbf{u}_1, \ldots, \alpha_k \mathbf{u}_k : \theta) \equiv \theta_1 \frac{1}{1 + e^{-\sum_i \alpha_i u_i}} + \theta_0 \tag{4}$$

is a sigmoid function where the response of X to its parents' values is saturated when the sum is far from zero.

3.1.2 LIKELIHOOD FUNCTION

Given the above form of CPDs, we can now write a concrete form of the log-likelihood function

$$\begin{aligned}\ell_X(\mathcal{D} : \mathbf{U}, \theta) &= -\frac{1}{2} \sum_{m=1}^{M} \left[\log(2\pi) + \log(\sigma^2) + \frac{1}{\sigma^2} (x[m] - g(\mathbf{u}[m]))^2 \right] \\ &= -\frac{1}{2} \left[M \log(2\pi) + M \log(\sigma^2) + \frac{1}{\sigma^2} \sum_m (x[m] - g(\mathbf{u}[m]))^2 \right] \end{aligned} \tag{5}$$

where, for simplicity, we absorbed each scaling factor α_j into each value of $u_j[m]$. Similarly, when the new parent Z is added with coefficient α_z, the new likelihood is

$$\ell_X(\mathcal{D} : \mathbf{U} \cup \{Z\}, \alpha_z, \theta,) = -\frac{1}{2} \left[M \log(2\pi) + M \log(\sigma_z^2) + \frac{1}{\sigma_z^2} \sum_m (x[m] - g(\mathbf{u}[m], \alpha_z z[m]))^2 \right]$$

where σ_z^2 is used to denote the variance parameter after Z is added. Consequently, the difference in likelihood of Eq. (2) takes the form of

$$\begin{aligned}\Delta_{X|\mathbf{U}}(Z) &= -\frac{M}{2} \left[\log \sigma_z^2 - \log \sigma^2 \right] \\ &\quad -\frac{1}{2} \left[\frac{1}{\sigma_z^2} \sum_m (x[m] - g(\mathbf{u}[m], \alpha_z z[m]))^2 - \frac{1}{\sigma^2} \sum_m (x[m] - g(\mathbf{u}[m]))^2 \right]. \end{aligned} \tag{6}$$

3.1.3 THE "IDEAL PARENT"

We now define the ideal parent for X

Definition 3.1: Given a data set $\mathcal{D}$, and a CPD for X given its parents $\mathbf{U}$, with a link function g and parameters θ and α, the *ideal parent* Y of X is such that for each instance m,

$$x[m] = g(\alpha_1 u_1[m], \ldots, \alpha_k u_k[m], y[m] : \theta).$$

∎

Under mild conditions, the *ideal parent profile* (i.e., value of Y in each instance) can be computed for almost any uni-modal parametric conditional distribution. The only requirement from g is that it should be invertible w.r.t. each one of the parents. Note that in this definition, we implicitly assume that $x[m]$ lies in the image of g. If this is not the case, we can substitute $x[m]$ with $x_g[m]$, the point in g's image closest to $x[m]$. This guarantees that the prediction's mode for the current set of parents and parameters is as close as possible to X.

The resulting profile for the hypothetical ideal parent Y is the optimal set of values for the $(k+1)$th parent, in the sense that it would maximize the likelihood of the child variable X. This is true since by definition, X is equal to the mode of the function of its parents defined by g. Intuitively, if we can efficiently find a candidate parent Z that is similar to the hypothetically optimal parent, we can improve the model by adding an edge from this parent to X. We are now ready to instantiate the similarity measure $C(\vec{y},\vec{z})$. Below, we demonstrate how this is done for the case of a linear Gaussian CPD. We extend the framework for non-linear CPDs in Section 7.

3.2 Linear Gaussian

Let X be a variable in the network with a set of parents $\mathbf{U}$, and a *linear Gaussian* conditional distribution. In this case, g in Eq. (3) takes the form

$$g(\alpha_1\mathbf{u}_1,\ldots,\alpha_k\mathbf{u}_k:\theta)\equiv\sum_i\alpha_i\mathbf{u}_i+\theta_0.$$

To choose promising candidate parents for X, we start by computing the ideal parent Y for X given its current set of parents. This is done by inverting the linear link function g with respect to this additional parent Y (note that we can assume, without loss of generality, that the scale parameter of this additional parent is 1). This results in

$$y[m]=x[m]-\sum_j\alpha_j u_j[m]-\theta_0.$$

We can summarize this in vector notation, by using $\vec{x}=\langle x[1],\ldots,x[M]\rangle$, and so we get

$$\vec{y}=\vec{x}-\mathcal{U}\vec{\alpha}-\theta_0$$

where $\mathcal{U}$ is the matrix of parent values on all instances, and $\vec{\alpha}$ is the vector of scale parameters.

Having computed the *ideal parent profile*, we now want to efficiently evaluate its similarity to the profile of candidate parents. Intuitively, we want the similarity measure to reflect the likelihood gain by adding Z as a parent of X. Ideally, we want to evaluate $\Delta_{X|\mathbf{U}}(Z)$ for each candidate parent Z. However, instead of re-estimating all the parameters of the CPD after adding Z as a parent, we approximate this difference by only fitting the scaling factor associated with the new parent and freezing all other parameters of the CPD (the scaling parameters of the current parents $\mathbf{U}$ and the variance parameter σ^2).

Theorem 3.2 *Suppose that X has parents $\mathbf{U}$ with a set $\vec{\alpha}$ of scaling factors. Let Y be the ideal parent as described above, and Z be some candidate parent. Then the change in the log-likelihood of X in the data, when adding Z as a parent of X, while freezing all scaling and variance parameters except the scaling factor of Z, is*

$$\begin{aligned}C_1(\vec{y},\vec{z}) &\equiv \max_{\alpha_Z}\ell_X(\mathcal{D}:\mathbf{U}\cup\{Z\},\widehat{\theta}_{X|\mathbf{U}}\cup\{\alpha_Z\})-\ell_X(\mathcal{D}:\mathbf{U},\widehat{\theta}_{X|\mathbf{U}})\\ &= \frac{1}{2\sigma^2}\frac{(\vec{y}\cdot\vec{z})^2}{\vec{z}\cdot\vec{z}}.\end{aligned}\qquad(7)$$

Proof: In the linear Gaussian case $y[m] = x[m] - g(\mathbf{u}[m])$ by definition and $g(\mathbf{u}[m], \alpha_z z[m]) = g(\mathbf{u}[m]) + \alpha_z z[m]$ so that Eq. (6) can be written as

$$\begin{aligned}
\Delta_{X|\mathbf{U}}(Z) &= -\frac{M}{2}\left[\log\sigma_z^2 - \log\sigma^2\right] - \frac{1}{2}\left[\frac{1}{\sigma_z^2}\sum_m (y[m] - \alpha_z z[m])^2 - \frac{1}{\sigma^2}\sum_m y[m]^2\right] \\
&= -\frac{M}{2}\left[\log\sigma_z^2 - \log\sigma^2\right] - \frac{1}{2}\left[\frac{1}{\sigma_z^2}\left(\vec{y}\cdot\vec{y} - 2\alpha_z\vec{z}\cdot\vec{y} + \alpha_z^2\vec{z}\cdot\vec{z}\right) - \frac{1}{\sigma^2}\vec{y}\cdot\vec{y}\right]. \qquad (8)
\end{aligned}$$

Since $\sigma_z = \sigma$ this reduces to

$$\begin{aligned}
\Delta_{X|\mathbf{U}}(Z:\alpha_z) &\equiv \ell_X(\mathcal{D}:\mathbf{U}\cup\{Z\}, \widehat{\theta}_{X|\mathbf{U}}\cup\{\alpha_Z\}) - \ell_X(\mathcal{D}:\mathbf{U}, \widehat{\theta}_{X|\mathbf{U}}) \\
&= -\frac{1}{2\sigma^2}\left(-2\alpha_z\vec{z}\cdot\vec{y} + \alpha_z^2\vec{z}\cdot\vec{z}\right). \qquad (9)
\end{aligned}$$

To optimize our only free parameter α_z, we use

$$\frac{\partial\Delta_{X|\mathbf{U}}(Z:\alpha_z)}{\partial\alpha_z} = -\frac{1}{2\sigma^2}\left(-2\vec{z}\cdot\vec{y} + 2\alpha_z\vec{z}\cdot\vec{z}\right) = 0 \quad\Rightarrow\quad \alpha_z = \frac{\vec{z}\cdot\vec{y}}{\vec{z}\cdot\vec{z}}.$$

Plugging this into Eq. (9), we get

$$\begin{aligned}
C_1(\vec{y},\vec{z}) &\equiv \max_{\alpha_z}\Delta_{X|\mathbf{U}}(Z:\alpha_z) \\
&= -\frac{1}{2\sigma^2}\left(-2\frac{\vec{z}\cdot\vec{y}}{\vec{z}\cdot\vec{z}}\vec{z}\cdot\vec{y} + \left(\frac{\vec{z}\cdot\vec{y}}{\vec{z}\cdot\vec{z}}\right)^2\vec{z}\cdot\vec{z}\right) \\
&= \frac{1}{2\sigma^2}\frac{(\vec{z}\cdot\vec{y})^2}{\vec{z}\cdot\vec{z}}.
\end{aligned}$$

The form of the similarity measure can be even further simplified

Proposition 3.3 *Let $C_1(\vec{y},\vec{z})$ be as defined above and let σ be the maximum likelihood parameter* before *Z is added as a new parent of X. Then*

$$C_1(\vec{y},\vec{z}) = \frac{M}{2}\frac{(\vec{y}\cdot\vec{z})^2}{(\vec{z}\cdot\vec{z})(\vec{y}\cdot\vec{y})} = \frac{M}{2}\cos^2\phi_{\vec{y},\vec{z}}$$

where $\phi_{\vec{y},\vec{z}}$ is the angle between the ideal parent profile vector $\vec{y}$ and the candidate parent profile vector $\vec{z}$.

Proof: To recover the maximum likelihood value of σ we differentiate the log-likelihood function as written in Eq. (5)

$$\begin{aligned}
\frac{\partial\ell_X(\mathcal{D}:\mathbf{U},\theta)}{\partial\sigma^2} &= -\frac{M}{2\sigma^2} + \frac{1}{\sigma^4}\sum_m (x[m] - g(\mathbf{u}[m]))^2 = 0 \\
&\Rightarrow\quad \sigma^2 = \frac{1}{M}\sum_m (x[m] - g(\mathbf{u}[m]))^2 = \frac{1}{M}\vec{y}\cdot\vec{y}
\end{aligned}$$

where the last equality follows from the definition of $\vec{y}$. The result follows immediately by plugging this into Theorem 3.2 and from the fact that $\cos^2\phi_{\vec{y},\vec{z}} \equiv \frac{(\vec{y}\cdot\vec{z})^2}{(\vec{z}\cdot\vec{z})(\vec{y}\cdot\vec{y})}$ ∎

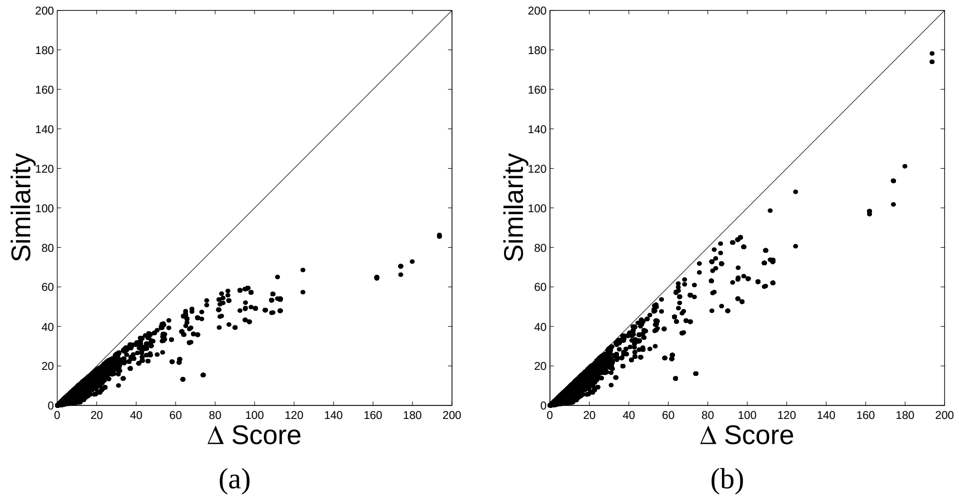


Figure 2: Demonstration of the (a) C_1 and (b) C_2 bounds for linear Gaussian CPDs. The x-axis is the true change in score as a result of an edge modification. The y-axis is the lower bound of this score. Points shown correspond to several thousand edge modifications in a run of the ideal parent method on real-life Yeast gene expressions data.

Thus, there is an intuitive geometric interpretation to the measure $C_1(\vec{y},\vec{z})$: we prefer a profile $\vec{z}$ that is similar to the ideal parent profile $\vec{y}$, regardless of its norm. It can easily be shown that $\vec{z} = c\vec{y}$ (for any constant c) maximizes this similarity measure. We retain the less intuitive form of $C_1(\vec{y},\vec{z})$ in Theorem 3.2 for compatibility with later developments.

Note that, by definition, $C_1(\vec{y},\vec{z})$ is a *lower bound* on $\Delta_{X|\mathbf{U}}(Z)$, the improvement on the log-likelihood by adding Z as a parent of X: When we add the parent we optimize all the parameters, and so we expect to attain a likelihood as high, or higher than, the one we attain by freezing some of the parameters. This is illustrated in Figure 2(a) that plots the true likelihood improvement vs. C_1 for several thousand edge modifications taken from an experiment using real life Yeast gene expression data (see Section 9).

We can get a better lower bound by optimizing additional parameters. In particular, after adding a new parent, the errors in predictions change, and so we can readjust the variance term. As it turns out, we can perform this readjustment in closed form.

Theorem 3.4 *Suppose that X has parents $\mathbf{U}$ with a set $\vec{\alpha}$ of scaling factors. Let Y be the ideal parent as described above, and Z be some candidate parent. Then the change in the log-likelihood of X in the data, when adding Z as a parent of X, while freezing all other parameters except the scaling factor of Z and the variance of X, is*

$$\begin{aligned} C_2(\vec{y},\vec{z}) &\equiv \max_{\alpha_Z,\sigma_Z} \ell_X(\mathcal{D} : \mathbf{U} \cup \{Z\}, \widehat{\theta}_{X|\mathbf{U}} \cup \{\alpha_Z, \sigma_Z\}) - \ell_X(\mathcal{D} : \mathbf{U}, \widehat{\theta}_{X|\mathbf{U}}) \\ &= -\frac{M}{2} \log \sin^2 \phi_{\vec{y},\vec{z}} \end{aligned}$$

where $\phi_{\vec{y},\vec{z}}$ is the angle between $\vec{y}$ and $\vec{z}$.

Proof: To optimize σ_z we again consider Eq. (8) and set

$$\frac{\partial \Delta_{X|\mathbf{U}}(Z)}{\partial \sigma_z} = -\frac{M}{\sigma_z} + \frac{1}{\sigma_z^3}\left[\vec{y}\cdot\vec{y} - 2\alpha_z \vec{z}\cdot\vec{y} + \alpha_z^2 \vec{z}\cdot\vec{z}\right] = 0.$$

Solving for σ_z and plugging the maximum likelihood parameter α_z from the development of $C_1(\vec{y},\vec{z})$ (which does not depend on σ_z), we get

$$\sigma_z^2 = \frac{1}{M}\left[\vec{y}\cdot\vec{y} - 2\alpha_z \vec{z}\cdot\vec{y} + \alpha_z^2 \vec{z}\cdot\vec{z}\right] = \frac{1}{M}\left[\vec{y}\cdot\vec{y} - \frac{(\vec{z}\cdot\vec{y})^2}{\vec{z}\cdot\vec{z}}\right].$$

As in the case of Proposition 3.3 where $\sigma = \frac{1}{M}\vec{y}\cdot\vec{y}$, the variance term σ_z^2 "absorbs" the sum of squared errors when optimized. Thus, the second term in Eq. (8) becomes zero and we can write

$$\begin{aligned}
C_2(\vec{y},\vec{z}) &= -\frac{M}{2}\left[\log(\sigma_z^2) - \log(\sigma^2)\right] \\
&= \frac{M}{2}\log\left(\frac{\vec{y}\cdot\vec{y}}{\vec{y}\cdot\vec{y} - \frac{(\vec{z}\cdot\vec{y})^2}{\vec{z}\cdot\vec{z}}}\right) = \frac{M}{2}\log\left(\frac{1}{1-\frac{(\vec{z}\cdot\vec{y})^2}{(\vec{z}\cdot\vec{z})(\vec{y}\cdot\vec{y})}}\right) = \frac{M}{2}\log\left(\frac{1}{1-\cos^2\phi_{\vec{y},\vec{z}}}\right) \\
&= -\frac{M}{2}\log\sin^2\phi_{\vec{y},\vec{z}}.
\end{aligned}$$

∎

It is important to note that both C_1 and C_2 are monotonic functions of $\frac{(\vec{y}\cdot\vec{z})^2}{\vec{z}\cdot\vec{z}}$, and so they consistently rank candidate parents of the same variable. However, when we compare changes that involve different ideal parents, such as adding a parent to X_1 compared to adding a parent to X_2, the ranking by these two measures might differ. Due to the choice of parameters we freeze in each of these measures, we have

$$C_1(\vec{y},\vec{z}) \leq C_2(\vec{y},\vec{z}) \leq \Delta_{X|\mathbf{U}}(Z)$$

and so C_2 can provide better guidance to some search algorithms. Indeed, Figure 2(b) clearly shows that C_2 is a tighter bound than C_1, particularly for promising candidates.

4. Ideal Parents in Search

The technical developments of the previous section show that we can approximate the score of candidate parents for X by comparing them to the ideal parent Y using the similarity measure. Is this approximate evaluation useful?

When performing a local heuristic search such as the one illustrated in Algorithm 1, at each iteration we have a current candidate structure and we consider some operations on that structure. These operations might include edge addition, edge replacement, edge reversal and edge deletion. We can readily use the ideal profiles and similarity measures developed to speed up two of these: edge addition and edge replacement. In a network with N nodes, there are in the order of $O(N^2)$ possible edge additions, $O(E\cdot N)$ edge replacement where E is the number of edges in the model, and only $O(E)$ edge deletions and reversals. Thus our method can be used to speed up the bulk of edge modifications considered by a typical search algorithm.

When considering adding an edge $Z \rightarrow X$, we use the ideal parent profile for X and compute its similarity to Z. We repeat this for every candidate parent for X. We then compute the full score only for the K most similar candidates, and insert them (and the associated change in score) into a queue of potential operations. In a similar way, we can use the ideal parent profile for considering edge replacement for X. Suppose that $\mathbf{U_i} \in \mathbf{U}$ is a parent of X. We can define the ideal profile for replacing U while freezing all other parameters of the CPD of X.

Definition 4.1: Given a data set $\mathcal{D}$, and a CPD for X given its parents $\mathbf{U}$, with a link function g, parameters θ and α, the *replace ideal parent* Y of X and $\mathbf{U_i} \in \mathbf{U}$ is such that for each instance m,

$$x[m] = g(\alpha_1 u_1[m], \ldots, \alpha_{i-1}u_{i-1}, \alpha_{i+1}u_{i+1}, \ldots, \alpha_k u_k[m], y[m] : \theta).$$

▮

The rest of the developments of the previous section remain the same. For each current parent of X we compute a separate ideal profile that corresponds to the replacement of that parent with a new one. We then use the same policy as above for examining the replacement of each one of the parents. In particular, we freeze the scale parameters computed with $\mathbf{U}$ as the parent set of X, take out the parameter corresponding to $\mathbf{U_i}$, and use the C_1 or the C_2 measures to rank candidate replacements for $\mathbf{U_i}$.

For both operations, we can trade off between the accuracy of our evaluations and the speed of the search, by changing K, the number of candidate changes per family for which we compute a full score. Using $K = 1$, we only score the best candidate according to the ideal parent method ranking, thus achieving the largest speedup, However, since our ranking only approximates the true score difference, this strategy might miss good candidates. Using higher values of K brings us closer to the standard search algorithm both in terms of candidate selection quality but also in terms of computation time.

In the experiments in Section 9, we integrated the changes described above into a greedy hill climbing heuristic search procedure. This procedure also examines candidate structure changes that remove an edge and reverse an edge, which we evaluate in the standard way. The greedy hill climbing procedure applies the best available move at each iteration (among those that were chosen for full evaluation) as in Algorithm 1. Importantly, the ideal parent method is independent of the specifics of the search procedure and simply pre-selects promising candidates for the search algorithm to consider. Algorithm 2 outlines a generalization of the basic greedy structure search algorithm of Algorithm 1 to include a candidate ranking/selection algorithm such as our "Ideal Parent" method.

5. Adding New Hidden Variables

Somewhat unexpectedly, the "Ideal Parent" method also offers a natural solution to the difficult challenge of detecting new hidden variables. Specifically, the ideal parent profiles provide a straightforward way to find when and where to add hidden variables to the domain in continuous variable networks. The intuition is fairly simple: if the ideal parents of several variables are similar to each other, then we know that a similar input is predictive of all of them. Moreover, if we do not find a variable in the network that is close to these ideal parents, then we can consider adding a new hidden variable that will serve as their combined input, and, in addition, have an informed initial estimate of its profile. Figure 3 illustrates this idea.

To introduce a new hidden variable, we would like to require that it be beneficial for several children at once. The difference in log-likelihood due to adding a new parent with profile $\vec{z}$ is the

Algorithm 2: Greedy Hill-Climbing Structure Search with Candidate Ranking/Selection

```
Input  : $\mathcal{D}$   // training set
         $\mathcal{G}_0$   // initial structure
         CE  // candidate evaluation method such as our “Ideal Parent”
         K   // number of candidates to evaluate
Output : A final structure G

$\mathcal{G}_{best} \leftarrow \mathcal{G}_0$
repeat
    G $\leftarrow \mathcal{G}_{best}$
    L $\leftarrow \emptyset$   // initialize list of modifications to evaluate
    // for each family, choose the top ’add’ and ’replace’ candidates for evaluation
    foreach $X_i$ node in G do
        Q $\leftarrow \emptyset$   // initialize family specific queue
        foreach Add,Replace parent of $X_i$ in G do
            score $\leftarrow$ CE.Score(Operator)
            Q $\leftarrow$ (Operator,score)
        end foreach
        foreach top K Operators in Q do
            L $\leftarrow$ (Operator)
        end foreach
    end foreach
    // add all delete and reverse operations
    foreach Delete,Reverse edge in G do
        L $\leftarrow$ (Operator)
    end foreach
    // process all candidate operations chosen for evaluation
    foreach Operator in L do
        if Operator does not create a directed cycle then
            $\mathcal{G}' \leftarrow$ ApplyOperator($G$)
            if Score($\mathcal{G}'$ : $\mathcal{D}$) > Score($\mathcal{G}_{best}$ : $\mathcal{D}$) then
                $\mathcal{G}_{best} \leftarrow \mathcal{G}'$
            end
        end
    end foreach
until $\mathcal{G}_{best}$ == $G$
return $\mathcal{G}_{best}$
```

sum of differences between the log-likelihoods of families it is involved in:

$$\Delta_{X_1,\ldots,X_L}(Z) = \sum_{i}^{L} \Delta_{X_i|\mathbf{U_i}}(Z)$$

where we assume, without loss of generality, that the members of the cluster (children set of the candidate hidden variable) are $X_1,\ldots,X_L$. To score the network with Z as a new hidden variable, we

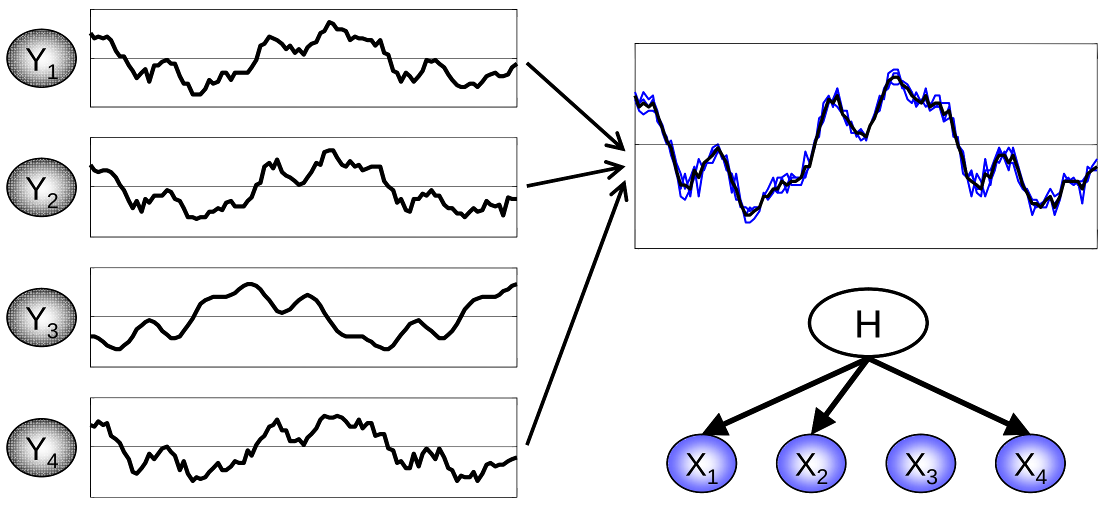


Figure 3: Illustration of how the ideal parent profiles can be used to suggest new hidden variables. Shown on the left are the ideal parent profiles $Y_1 \ldots Y_4$ of the variables $X_1 \ldots X_4$, respectively. These correspond to the residual information of these variables that is not explained by the current model. As can be seen, the first, second and fourth variables have similar ideal profiles. These profiles are averaged, resulting in a candidate hidden parent profile of these three variables (top right). Assuming that there is no variable in the network with a similar profile, our method will propose adding this hidden variable to the network as shown on the bottom right. Note that the average ideal profile of these variables provides an informed starting point for the EM algorithm.

also need to deal with the difference in the complexity penalty term, and the likelihood of Z as a root variable. These terms, however, can be readily evaluated. The difficulty is in finding the profile $\vec{z}$ that maximizes $\Delta_{X_1,\ldots,X_L}(Z)$. Using the C_1 ideal parent approximation, we can lower bound this improvement by

$$\sum_i^L C_1(\vec{y}_i,\vec{z}) \equiv \sum_i^L \frac{1}{2\sigma_i^2}\frac{(\vec{z}\cdot\vec{y}_i)^2}{\vec{z}\cdot\vec{z}} \leq \Delta_{X_1,\ldots,X_L}(Z) \tag{10}$$

and so we want to find $\vec{z}^*$ that maximizes this bound. We will then use this optimized bound as our approximate cluster score. That is we want to find

$$\vec{z}^* = \arg\max_{\vec{z}} \sum_i \frac{1}{2\sigma_i^2}\frac{(\vec{z}\cdot\vec{y}_i)^2}{\vec{z}\cdot\vec{z}} \equiv \arg\max_{\vec{z}} \frac{\vec{z}^T \mathcal{Y}\mathcal{Y}^T \vec{z}}{\vec{z}^T\vec{z}} \tag{11}$$

where $\mathcal{Y}$ is the matrix whose columns are y_i/σ_i. Note that the vector $\vec{z}^*$ must lie in the *column span* of $\mathcal{Y}$ since any component orthogonal to this span increases the denominator of the right hand term but leaves the numerator unchanged, and therefore does not obtain a maximum. We can therefore express the solution as:

$$\vec{z}^* = \sum_i v_i \frac{y_i}{\sigma_i} = \mathcal{Y}\vec{v} \tag{12}$$

where $\vec{v}$ is a vector of coefficients. Furthermore, the objective in Eq. (11) is known as the *Rayleigh quotient* of the matrix $\mathcal{Y}\mathcal{Y}^T$ and the vector $\vec{z}$. The optimum of this quotient is achieved when $\vec{z}$ equals the eigenvector of $\mathcal{Y}\mathcal{Y}^T$ corresponding to its largest eigenvalue (Wilkinson, 1965). Thus, to solve for $\vec{z^*}$ we want to solve the following eigenvector problem

$$(\mathcal{Y}\mathcal{Y}^T)\vec{z^*} = \lambda\vec{z^*}. \tag{13}$$

Note that the dimension of $\mathcal{Y}\mathcal{Y}^T$ is M (the number of instances), so that, in practice, this problem cannot be solved directly. However, by plugging Eq. (12) into Eq. (13), multiplying on the right by $\mathcal{Y}$, and defining $A \equiv \mathcal{Y}^T\mathcal{Y}$, we get a reduced generalized eigenvector problem [1]

$$AA\vec{v} = \lambda A\vec{v}.$$

Although this problem can now be solved directly, it can be further simplified by noting that A is only singular if the residue of observations of two or more variables are linearly dependent along *all* of the training instances. In practice, for continuous variables, A is indeed non-singular, and we can multiply both sides A^{-1} and end up with a simple eigenvalue problem:

$$A\vec{v} = \lambda\vec{v}$$

which is numerically simpler and easy to solve as the dimension of A is L, the number of variables in the cluster, which is typically relatively small. Once we find the L dimensional eigenvector $\vec{v}^*$ with the largest eigenvalue λ^*, we can express with it the desired parent profile $\vec{z^*}$.

We can get a better bound of $\Delta_{X_1,\ldots,X_L}(Z)$ if we use C_2 similarity rather than C_1. Unfortunately, optimizing the profile $\vec{z}$ with respect to this similarity measure is a harder problem that we do not have a closed solution for. Since the goal of the cluster identification is to provide a good starting point for the following iterations that will eventually adapt the structure, we use the closed form solution for Eq. (11). Note that once we optimize the profile z using the above derivation, we can still use the C_2 similarity score to provide a better bound on the quality of this profile as a new parent for $X_1, \ldots, X_L$.

Now that we can approximate the benefit of adding a new hidden parent to a cluster of variables, we still need to consider different clusters to find the most beneficial one. As the number of clusters is exponential, we adapt a heuristic *agglomerative clustering* approach (e.g., Duda and Hart, 1973) to explore different clusters. We start with each variable as an individual cluster and repeatedly merge the two clusters that lead to the best expected improvement (or the least decrease) in the BIC score. This procedure potentially involves $O(N^3)$ merges, where N is the number of possible variables. We save much of the computations by pre-computing the matrix $\mathcal{Y}^T\mathcal{Y}$ only once, and then using the relevant sub-matrix in each merge. In practice, the time spent in this step is insignificant in the overall search procedure.

6. Learning with Missing Values

In real-life domains, it is often the case that the data is incomplete and some of the observations are missing. Furthermore, once we add a hidden variable to the network structure, we have to copy with missing values in subsequent structure search even if the original training data was complete.

[1] In the *Generalized Eigenvector Problem*, we want to find eigenpairs $(\lambda, \vec{v})$ so that $B\vec{v} = \lambda A\vec{v}$ holds.

To deal with this problem, we use an Expectation Maximization approach (Dempster et al., 1977) and its application to network structure learning (Friedman, 1997). At each step in the search we have a current network that provides an estimate of the distribution that generated the data, and use it to compute a distribution over possible completions of the data. Instead of maximizing the BIC score, we attempt to maximize the expected BIC score

$$\boldsymbol{E}_Q[BIC(\mathcal{D},G) \mid \mathcal{D}_o] = \int Q(\mathcal{D}_h \mid \mathcal{D}_o) BIC(\mathcal{D},G) d\mathcal{D}_h$$

where $\mathcal{D}_o$ is the observed data, $\mathcal{D}_h$ is the unobserved data, and Q is the distribution represented by the current network. As the BIC score is a sum over local terms, we can use linearity of expectations to rewrite this objective as a sum of expectations, each over the scope of a single CPD. This implies that when learning with missing values, we need to use the current network to compute the posterior distribution over the values of variables in each CPD we consider. Using these posterior distributions we can estimate the expectation of each local score, and use them in standard structure search. Once the search algorithm converges, we use the new network for computing expectations and reiterate until convergence (see Friedman, 1997).

How can we combine the ideal parent method into this structural EM search? Since we do not necessarily observe X and all of its parents, the definition of the ideal parent cannot be applied directly. Instead, we define the ideal parent to be the profile that will match the expectations given Q. That is, we choose $y[m]$ so that

$$\boldsymbol{E}_Q[x[m] \mid \mathcal{D}_o] = \boldsymbol{E}_Q[g(\alpha_1 u_1[m], \ldots, \alpha_k u_k[m], y[m] : \theta) \mid \mathcal{D}_o].$$

In the case of linear CPDs, this implies that

$$\vec{y} = \boldsymbol{E}_Q[\vec{x} \mid \mathcal{D}_o] - \boldsymbol{E}_Q[\mathcal{U} \mid \mathcal{D}_o]\vec{\alpha}.$$

Once we define the ideal parent, we can use it to approximate changes in the expected BIC score (given Q). For the case of a linear Gaussian, we get terms that are similar to C_1 and C_2 of Theorem 3.2 and Theorem 3.4, respectively. The only change is that we apply the similarity measure on the expected value of $\vec{z}$ for each candidate parent Z. This is in contrast to exact evaluation of $\boldsymbol{E}_Q\left[\Delta_{X|\mathbf{U}}(Z) \mid \mathcal{D}_o\right]$, which requires the computation of the expected sufficient statistics of $\mathbf{U}$, X, and Z. To facilitate efficient computation, we adopt an approximate variational *mean-field* form (e.g., Jordan et al., 1998; Murphy and Weiss, 1999) for the posterior. This approximation is used both for the ideal parent method and the standard greedy approach used in Section 9. This results in computations that require only the first and second moments for each instance $z[m]$, and thus can be easily obtained from Q.

Finally, we note that the structural EM iterations are still guaranteed to converge to a local maximum. In fact, this does *not* depend on the fact that C_1 and C_2 are lower bounds of the true change to the score, since these measures are only used to pre-select promising candidates which are scored before actually being considered by the search algorithm. Indeed, the ideal parent method is a modular structure candidate selection algorithm and can be used as a black-box by any search algorithm.

7. Non-linear CPDs

We now address the important challenge of non-linear CPDs. In the class of CPDs we are considering, this non-linearity is mediated by the link function g, which we assume here to be invertible.

Examples of such functions include the sigmoid function shown in Eq. (4) and hyperbolic functions that are suitable for modeling gene transcription regulation (Nachman et al., 2004), among many others. When we learn with non-linear CPDs, parameter estimation is harder. To evaluate a potential parent P for X we have to perform non-linear optimization (e.g., conjugate gradient) of all of the α coefficients of all parents as well as other parameters of g. In this case, a fast approximation can boost the computational cost of the search significantly.

As in the case of linear CPDs, we compute the ideal parent profile $\vec{y}$ by inverting g. (We assume that the inversion of g can be performed in time that is proportional to the calculation of g itself as is the case for the CPDs considered above.) Suppose we are considering the addition of a parent to X in addition to its current parents $\mathbf{U}$, and that we have computed the value of the ideal parent $y[m]$ for each sample m by inversion of g. Now consider a particular candidate parent Z whose value at the mth instance is $Z[m]$. How will the difference between the ideal value and the value of Z reflect in the prediction of X for this instance?

As we have seen for the linear case in Section 3, the difference $z[m] - y[m]$ translated through g to a prediction error. In the non-linear case, the effect of the difference on predicting X depends on other factors, such as the values of the other parents. To see this, consider again the sigmoid function g of Eq. (4). If the sum of the arguments to g is close to 0, then g locally behaves like a sum of its arguments. On the other hand, if the sum is far from 0, the function is in one of the saturated regions, and big differences in the input almost do not change the prediction. This complicates our computations and does not allow the development of similarity measures as in Theorem 3.2 and Theorem 3.4 directly.

We circumvent this problem by approximating g with a linear function around the value of the ideal parent profile. We use a first-order Taylor expansion of g around the value of $\vec{y}$ and write

$$g(\vec{\mathbf{u}}, \vec{z}) \approx g(\vec{\mathbf{u}}, \vec{y}) + (\alpha_z \vec{z} - \vec{y}) \cdot \left. \frac{\partial g(\vec{\mathbf{u}}, \vec{z})}{\partial \alpha_z \vec{z}} \right|_{\alpha_z \vec{z} = \vec{y}} .$$

As a result, the "penalty" for a distance between $\vec{z}$ and $\vec{y}$ depends on the gradient of g at the particular value of $\vec{y}$, given the value of the other parents. In instances where the derivative is small, larger deviations between $y[m]$ and $z[m]$ have little impact on the likelihood of $x[m]$, and in instances where the derivative is large, the same deviations may lead to worse likelihood.

To understand the effect of this approximation in more detail we consider a simple example with a sigmoid Gaussian CPD as defined in Eq. (4), where X has no parents in the current network and Z is a candidate new parent. Figure 4(a) shows the sigmoid function (dotted) and its linear approximation at $Y = 0$ (solid) for an instance where $X = 0.5$. The computation of $Y = \log\left(\frac{1}{0.5} - 1\right) = 0$ by inversion of g is illustrated by the dashed lines. (b) is the same for a different sample where $X = 0.85$. In (c),(d) we can see the effect of the approximation for these two different samples on our evaluation of the likelihood function. For a given probability value, the likelihood function is more sensitive to changes in the value of Z around Y when $X = 0.5$ when compared to the instance $X = 0.85$. This can be seen more clearly in (e) where equi-potential contours are plotted for the sum of the approximate log-likelihood of these two instances. To recover the setup where our sensitivity to Z does *not* depend on the specific instance as in the linear case, we consider a skewed version of $Z \cdot \partial g / \partial y$ rather than Z directly. The result is shown in Figure 4(f). We can generalize the example above to develop a similarity measure for the general non-linear case:

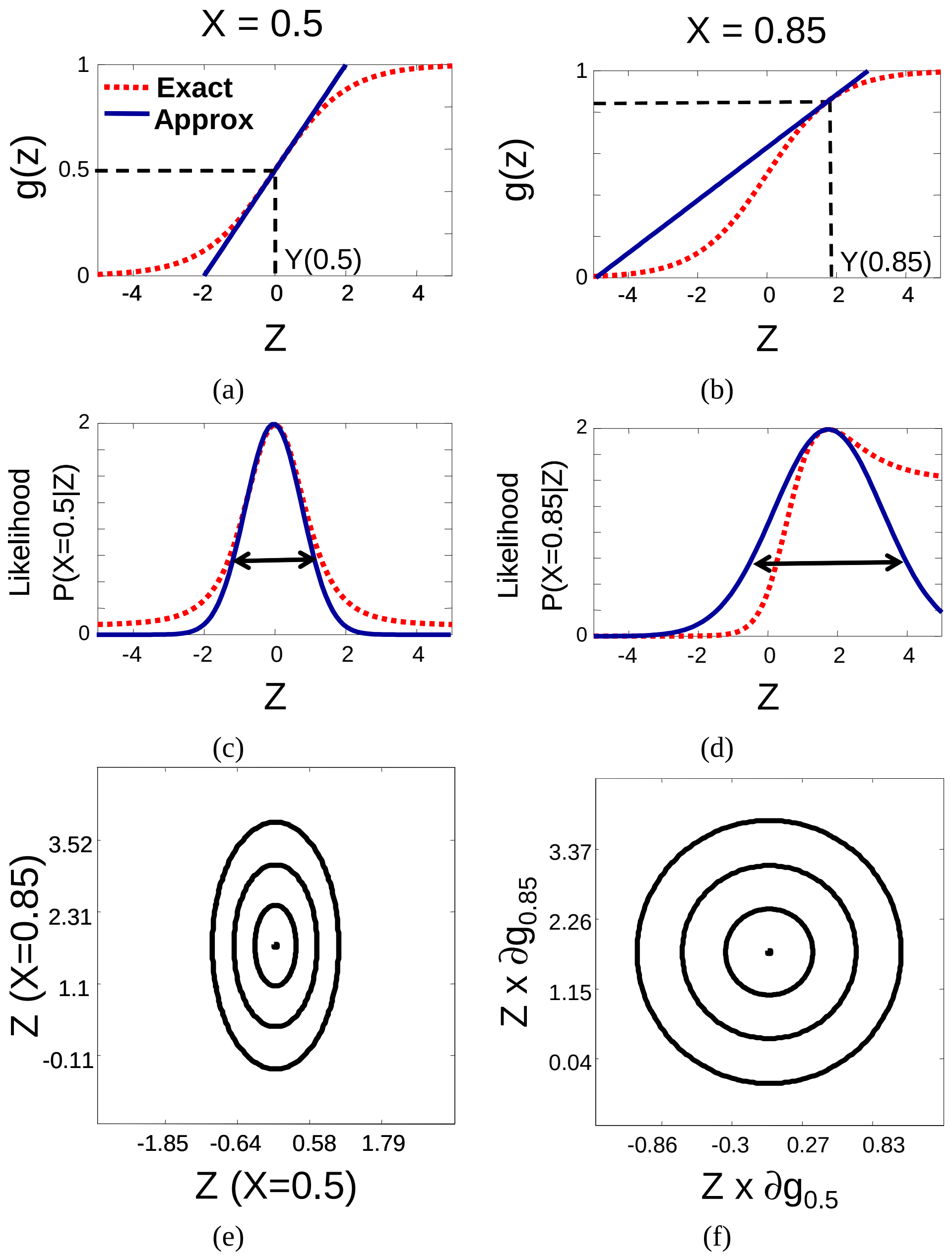


Figure 4: A simple example of the effect of the linear approximation for a sigmoid CPD where X has no parents in the current network and Z is considered as a new candidate parent. Two samples (a) and (b) show the function $g(y_1,\ldots,y_k : \theta) \equiv \theta_1 \frac{1}{1+e^{-\sum_i y_i}} + \theta_0$ for two instances where $X = 0.5$ and $X = 0.85$, respectively, along with their linear approximation at the ideal parent value Y of X. (c) and (d) show the corresponding likelihood function and its approximation. (e) shows the equi-potential contours of the sum of the log-likelihood of the two instances as a function of the value of Z in each of these instances. (f) is the same as (e) when the axes are skewed using the gradient of g with respect to the value of Y.

Theorem 7.1 *Suppose that X has parents $\mathbf{U}$ with a set $\vec{\alpha}$ of scaling factors. Let Y be the ideal parent as described above, and Z be some candidate parent. Then the change in log-likelihood of X in the data, when adding Z as a parent of X, while freezing all other parameters, is approximately*

$$C_1(\vec{y} \circ g'(\vec{y}), \vec{z} \circ g'(\vec{y})) - \frac{1}{2\sigma^2}(k_1 - k_2) \tag{14}$$

where $g'(\vec{y})$ is the vector whose mth component is $\partial g(\vec{\alpha}\mathbf{u}, y)/\partial y \mid_{\mathbf{u}[m], y[m]}$, and $\circ$ denotes component-wise product. Similarly, if we also optimize the variance, then the change in log-likelihood is approximately

$$C_2(\vec{y} \circ g'(\vec{y}), \vec{z} \circ g'(\vec{y})) - \frac{M}{2} \log \frac{k_1}{k_2}.$$

In both cases,

$$k_1 = (\vec{y} \circ g'(\vec{y})) \cdot (\vec{y} \circ g'(\vec{y}))\ ;\ k_2 = (\vec{x} - g(\vec{\mathbf{u}})) \cdot (\vec{x} - g(\vec{\mathbf{u}}))$$

do not depend on $\vec{z}$.

Thus, we can use exactly the same measures as before, except that we "distort" the geometry with the weight vector $g'(y)$ that determines the importance of different instances. To approximate the likelihood difference, we also add the correction term which is a function of k_1 and k_2. This correction is not necessary when comparing two candidates for the same family, but is required for comparing candidates from different families, or when adding hidden variable. Note that unlike the linear case, and as a result of the linear approximation of g, our theorem now involves an approximation of the difference in likelihood.

Proof: Using the general form of the Taylor linear approximation for a non-linear link function g, Eq. (6) can be written as

$$\begin{aligned}
&\Delta_{X|\mathbf{U}}(Z) \\
&\approx \quad -\frac{M}{2}\log\frac{\sigma_z^2}{\sigma^2} - \frac{1}{2}\left[\frac{1}{\sigma_z^2}\left[\vec{x} - g(\vec{\mathbf{u}}, \vec{y}) - (\alpha_z\vec{z} - \vec{y}) \circ g'\right]^2 - \frac{1}{\sigma^2}\left[\vec{x} - g(\vec{\mathbf{u}})\right]^2\right] \\
&= \quad -\frac{M}{2}\log\frac{\sigma_z^2}{\sigma^2} - \frac{1}{2\sigma_z^2}\left[\alpha_z^2(\vec{z} \circ g')^2 - 2\alpha_z(\vec{z} \circ g') \cdot (\vec{y} \circ g') + (\vec{y} \circ g')^2\right] + \frac{1}{2\sigma^2}\left[\vec{x} - g(\vec{\mathbf{u}})\right]^2 \\
&= \quad -\frac{M}{2}\log\frac{\sigma_z^2}{\sigma^2} - \frac{1}{2\sigma_z^2}\left[\alpha_z^2\vec{z}_\star \cdot \vec{z}_\star - 2\alpha_z\vec{z}_\star \cdot \vec{y}_\star + \vec{y}_\star \cdot \vec{y}_\star\right] + \frac{1}{2\sigma^2}\left[\vec{x} - g(\mathbf{u})\right]^2
\end{aligned} \tag{15}$$

where we use the fact that $\vec{x} - g(\vec{\mathbf{u}}, \vec{y}) = 0$ by construction of $\vec{y}$, and we denote for clarity $\vec{y}_\star \equiv \vec{y} \circ g'$ and $\vec{z}_\star \equiv \vec{z} \circ g'$. To optimize α_z we use

$$\frac{\partial \Delta_{X|\mathbf{U}}(Z)}{\partial \alpha_z} \approx -\frac{1}{2\sigma}\left[2\alpha_z\vec{z}_\star \cdot \vec{z}_\star - 2\vec{z}_\star \cdot \vec{y}_\star\right] \qquad \Rightarrow \qquad \alpha_z = \frac{\vec{z}_\star \cdot \vec{y}_\star}{\vec{z}_\star \cdot \vec{z}_\star}.$$

Plugging this into Eq. (15) we get

$$\begin{aligned}
\Delta_{X|\mathbf{U}}(Z) \quad &\approx \quad \frac{1}{2\sigma^2}\frac{(\vec{z}_\star \cdot \vec{y}_\star)^2}{\vec{z}_\star \cdot \vec{z}_\star} - \frac{1}{2\sigma^2}\vec{y}_\star \cdot \vec{y}_\star + \frac{1}{2\sigma^2}\left[\vec{x} - g(\vec{\mathbf{u}})\right]^2 \\
&= \quad C_1(\vec{y}_\star, \vec{z}_\star) - \frac{1}{2\sigma^2}(k_1 - k_2)
\end{aligned}$$

which proves Eq. (14). When we also optimize that variance, as noted before, the variance terms absorbs the sum of squared errors, so that

$$\sigma_z = \frac{1}{M}\left[\vec{y}_\star \cdot \vec{y}_\star - \frac{(\vec{z}_\star \cdot \vec{y}_\star)^2}{\vec{z}_\star \cdot \vec{z}_\star}\right].$$

Plugging this into Eq. (15) results in

$$\begin{aligned}
\Delta_{X|\mathbf{U}}(Z) &\approx -\frac{M}{2}\log\frac{\sigma^2}{\sigma_z^2} \\
&= \frac{M}{2}\log\frac{[\vec{x}-g(\mathbf{u})]^2}{\vec{y}_\star \cdot \vec{y}_\star - \frac{(\vec{z}_\star \cdot \vec{y}_\star)^2}{\vec{z}_\star \cdot \vec{z}_\star}} = \frac{M}{2}\log\frac{[\vec{x}-g(\mathbf{u})]^2}{\vec{y}_\star \cdot \vec{y}_\star\left[1-\frac{(\vec{z}_\star \cdot \vec{y}_\star)^2}{\vec{z}_\star \cdot \vec{z}_\star \vec{y}_\star \cdot \vec{y}_\star}\right]} \\
&= \frac{M}{2}\log\frac{1}{1-\frac{(\vec{z}_\star \cdot \vec{y}_\star)^2}{\vec{z}_\star \cdot \vec{z}_\star \vec{y}_\star \cdot \vec{y}_\star}} + \frac{M}{2}\log[\vec{x}-g(\mathbf{u})]^2 - \frac{M}{2}\log(\vec{y}_\star \cdot \vec{y}_\star) \\
&= C_2(\vec{y}_\star, \vec{z}_\star) - \frac{M}{2}\log\frac{k_1}{k_2}.
\end{aligned}$$

■

As in the linear case, the above theorem allows us to efficiently evaluate promising candidates for the *add edge* step in the structure search. The *replace edge* step can also be approximated with minor modifications. As before, the significant gain in speed is that we only perform a few parameter optimizations (that are expected to be costly as the number of parents grows), rather than $O(N)$ such optimizations for each variable.

Adding a new hidden variable with non-linear CPDs introduces further complications. We want to use, similar to the case of a linear model, the structure score of Eq. (10) with the distorted C_1 measure. Optimizing this measure has no closed form solution in this case and we need to resort to an iterative procedure or an alternative approximation. We use an approximation where the correction terms of Eq. (14) are omitted so that a form that is similar to the linear Gaussian case is used, with the "distorted" geometry of $\vec{y}$. Having made this approximation, the rest of the details are the same as in the linear Gaussian case.

8. Other Noise Models

So far, we only considered conditional probability distributions of the form of Eq. (3) where the uncertainty is modeled using an additive Gaussian noise term. In some cases, such as when modeling biological processes related to regulation, using a multiplicative noise model may be more appropriate, as most noise sources in these domains are of multiplicative nature (Nachman et al., 2004). We can model such a noise process using CPDs of the form

$$X = g(\alpha_1\mathbf{u}_1, \ldots, \alpha_k\mathbf{u}_k : \theta)(1+\varepsilon) \tag{16}$$

where, as in Eq. (3), ε is a noise random variable with zero mean. Another popular choice for modeling multiplicative noise is the log-normal form:

$$\log(X) = \log(g(\alpha_1\mathbf{u}_1, \ldots, \alpha_k\mathbf{u}_k : \theta)) + \varepsilon$$

where the log of the random variable is distributed normally. In this section we present a formulation that generalizes the concepts introduced so far to these more general scenarios. We present explicit derivations for the multiplicative noise CPD of Eq. (16) in Section 8.3.

8.1 General Framework

To cope with CPDs that use a multiplicative noise model, we first formalize the general form of a CPD we consider. We then generalize the concept of the ideal parent to accommodate this general form of distributions and state the approximation to the likelihood we make based on this new definition. We will then show that our generalized ideal parent definition leads, as before, to a natural similarity measure that includes our previous results as a special case.

Concretely, we consider conditional density distributions of the following general form

$$P(X \mid \mathbf{U}) = q(X : g(\alpha_1 u_1[m], \ldots, \alpha_k u_k[m] : \theta), \phi)$$

where g is the link function with parameters θ as before, and q is the "noise" distribution with parameters ϕ (e.g., variance parameters). In the additive case of Eq. (3) we have $q = \mathcal{N}(X; g, \sigma^2)$. In the multiplicative case of Eq. (16) we have $q = \mathcal{N}(X; g, (g\sigma)^2)$.

We now revisit our idea of the ideal parent. Recall that our definition of the ideal parent profile $\vec{y}$ was motivated by the goal of maximizing the likelihood of the child variable profile $\vec{x}$. However, unlike the case of additive noise, in general and in the case of the multiplicative noise model, g is not necessarily the mode of q. To accommodate this, we generalize our definition of an ideal parent:

Definition 8.1: Let $\mathcal{D}$ be a data set and let $P(X \mid \mathbf{U}) = q(X : g(\mathbf{U} : \theta), \phi)$ be a CPD for X given its parents $\mathbf{U}$ with parameters θ, α and ϕ, where both q and g are twice differentiable and g is invertible with respect to each one of the parents $\mathbf{U}$. The *ideal parent* Y of X is such that for each instance m,

$$\left. \frac{\partial q(x[m]; g, \phi)}{\partial g} \right|_{g = g(\alpha_1 u_1[m], \ldots, \alpha_k u_k[m], y[m] : \theta)} = 0. \quad (17)$$

∎

That is, $\vec{y}$ is the vector that makes $g(\mathbf{u}, \vec{y})$ maximize the likelihood of the child variable at each instance. Since $\frac{\partial q}{\partial z} = \left. \frac{\partial q}{\partial g} \frac{\partial g}{\partial z} \right|_{z=y} = 0$, this definition also means that the ideal parent maximizes the likelihood w.r.t. the values of a new parent. The above definition is quite general and allows for a wide range of link functions and uni-modal noise models. We note that in the case where the distribution is a simple Gaussian with any choice of g, this definition coincides with Definition 3.1. As an example of a conditional form that does not fall into our framework, $g = \sin(\sum_i \alpha_i \mathbf{u}_i)$ is not only not invertible but also allows for infinitely many "ideal" parents. As we show below, this more complex definition is useful as it will allow us to efficiently evaluate candidate parents for the general CPDs we consider in this section.

The above new definition of the ideal parent motivates us to use a different approximation than the one used in the case of non-linear CPDs with additive noise. Specifically, instead of simply approximating g, we now approximate the likelihood directly around $\vec{y}$, using a second order approximation:

$$\log P(\vec{x} \mid \mathbf{u}, \alpha_z z) \approx \log P(\vec{x} \mid \mathbf{u}, \vec{y}) + (\alpha_z \vec{z} - \vec{y}) \cdot \nabla_{\alpha_z \vec{z}} \log P(\vec{x} \mid \mathbf{u}, \vec{z})\big|_{\alpha_z \vec{z} = \vec{y}} + \frac{1}{2} (\alpha_z \vec{z} - \vec{y})^T H (\alpha_z \vec{z} - \vec{y}) \quad (18)$$

where H is the Hessian matrix of $\log P(\vec{x} \mid \mathbf{u}, \vec{z})$ at the point $\alpha_z \vec{z} = \vec{y}$.

8.2 Evaluating the benefit of a Candidate Parent

With the generalized definition of an ideal parent of Eq. (17) and the approximation chosen for the likelihood function in Eq. (18) we can approximately evaluate the benefit of a candidate parent:

Theorem 8.2 *Suppose that X has parents $\mathbf{U}$ with a set $\vec{\alpha}$ of scaling factors. Let Y be the ideal parent as defined in Eq. (17), and Z be some candidate parent. Then the change in log-likelihood of X in the data, when adding Z as a parent of X, while freezing all other parameters except the scaling factor of Z, is approximately*

$$\begin{aligned} C_1(\vec{y},\vec{z}) &\approx \log P(\vec{x} \mid \mathbf{u},\vec{y}) - \max_{\alpha_Z} \frac{1}{2} K(\alpha_z\vec{z}-\vec{y},\alpha_z\vec{z}-\vec{y}) - \log P(\vec{x} \mid \mathbf{u}) \\ &= \log P(\vec{x} \mid \mathbf{u},\vec{y}) - \frac{1}{2}K(\vec{y},\vec{y}) + \frac{1}{2}\frac{(K(\vec{y},\vec{z}))^2}{K(\vec{z},\vec{z})} - \log P(\vec{x} \mid \mathbf{u}) \end{aligned} \tag{19}$$

where $K(.,.)$ is an inner product of two vectors defined as:

$$K(\vec{a},\vec{b}) = \sum_m a[m]b[m] \frac{-1}{q_m}\frac{\partial^2 q_m}{\partial {g_m}^2}\left(g'_m\right)^2$$

and

$$\begin{aligned} g_m &= g(\mathbf{u}[m],y[m]:\theta) \\ q_m &= q(x[m]:g_m,\phi) \\ g'_m &= \frac{\partial g(\mathbf{u},y:\theta)}{\partial y}\Big|_{\mathbf{u}[m],y[m]} . \end{aligned}$$

Before proving this result, we first consider its elements and how they relate to our previous results of Theorem 3.2 and Theorem 7.1. The inner product K captures the deformation for the general case: The factor $(g'_m)^2$ weighs each vector by the gradient of g, as explained in Section 7. The new factor $\frac{-1}{q_m}\frac{\partial^2 q_m}{\partial {g_m}^2}$ measures the sensitivity of q_m to changes in g_m for each instance. This factor is always positive as a maximum point of q_m is involved. Note that in the Gaussian noise models we considered in the previous sections, this term is constant: $\frac{1}{\sigma^2}$. In non-Gaussian models, this sensitivity can vary between instances.

It is easy to see that the generalized definition of C_1 coincides with our previous results. As in the linear Gaussian case, the (approximate) difference in likelihood $C_1(\vec{y},\vec{z})$ is expressed as a function of some distance between the new parent $\alpha_z\vec{z}$ and the ideal parent $\vec{y}$. This distance is then deformed by a sample dependent weight similarly to the non-linear case discussed in Section 7. In the case of a linear Gaussian CPD, we have $g'_m = 1$, and so $K(\vec{a},\vec{b}) = \frac{1}{\sigma^2}\vec{a}\cdot\vec{b}$. All terms which do not depend on $\vec{z}$ cancel out in this case, resulting in our original definition for C_1 in Eq. (7). For the non-linear Gaussian with additive noise, we have $K(\vec{a},\vec{b}) = \frac{1}{\sigma^2}(\vec{a}\circ g'(y))\cdot(\vec{b}\circ g'(y))$, and the form of Eq. (14) is recovered.

Importantly, we note that our new formulation is applicable to a wide range of link functions and uni-modal noise models (with the minimal restrictions detailed above). The difference between different choices simply manifest as difference in the form of the derivatives that appear in the kernel function K, and in the additional $\log P$ terms in Eq. (19). Finally, we note that we cannot derive a similarly general expression for C_2, since it requires optimizing both σ and α_z, and the solution to this problem depends on the form of the distribution q.

For completeness, we now prove the result of Theorem 8.2.

Proof: The first term in the second order approximation of Eq. (18) vanishes since, by our definition, $\frac{\partial q}{\partial z}|_{\alpha_z \vec{z}=\vec{y}} = 0$, which implies also $\frac{\partial \log(q)}{\partial z}|_{\alpha_z \vec{z}=\vec{y}} = 0$. Using the chain rule, we derive the expression for the Hessian:

$$\begin{aligned} H_{m,n} &= \left.\frac{\partial^2 \log P(\vec{x} \mid \mathbf{u}, \vec{z})}{\partial \alpha_z z[m] \partial \alpha_z z[n]}\right|_{\alpha_z \vec{z}=\vec{y}} \\ &= \delta_{mn} \frac{1}{q_m^2} \left(-\left(\frac{\partial q_m}{\partial g_m} \frac{\partial g_m}{\partial y[m]} \right)^2 + q_m \left\{ \frac{\partial^2 q_m}{\partial {g_m}^2} \left(\frac{\partial g_m}{\partial y[m]} \right)^2 + \frac{\partial q_m}{\partial g_m} \frac{\partial^2 g_m}{\partial y[m]^2} \right\} \right). \end{aligned}$$

The Hessian matrix is always diagonal, since each term in the log-likelihood involves $y[m]$ that corresponds to a single sample m. After eliminating terms involving $\frac{\partial q}{\partial g}$, the diagonal elements of the Hessian simplify to:

$$H_{m,m} = \frac{1}{q_m} \frac{\partial^2 q_m}{\partial {g_m}^2} \left(g'_m \right)^2$$

where $g'_m \equiv \frac{\partial g_m}{\partial y[m]}$. With this simplification of the Hessian, the approximation of the log-likelihood can be written as

$$\log P(\vec{x} \mid \mathbf{u}, \alpha_z \vec{z}) \approx \log P(\vec{x} \mid \mathbf{u}, \vec{y}) + \frac{1}{2} \sum_m \frac{(\alpha_z z[m] - y[m])^2}{q_m} \frac{\partial^2 q_m}{\partial {g_m}^2} \left(g'_m \right)^2. \tag{20}$$

The difference in the log-likelihood with and without a new parent z can now be immediately retrieved and equals to the second term of the right hand side of Eq. (20). Denoting this difference by $C_1(\vec{y}, \vec{z})$ and replacing α_z with its maximum likelihood estimator $\frac{K(\vec{y},\vec{z})}{K(\vec{z},\vec{z})}$, we get the desired result.
∎

8.3 Multiplicative Noise CPD

We now complete the detailed derivation of the general framework we presented in the previous section for the case of the multiplicative noise conditional density of Eq. (16). Written explicitly, the CPD has the following form:

$$q(x : g, \sigma^2) = \frac{1}{\sqrt{2\Pi}\sigma |g|} \exp\left(-\frac{1}{2\sigma^2} \left(\frac{x}{g} - 1 \right)^2 \right).$$

To avoid singularity, we will restrict the values of g to be positive. The partial derivatives of q_m are:

$$\begin{aligned} \frac{\partial q_m}{\partial g_m} &= \left[-\frac{1}{g_m} + \frac{1}{\sigma^2} \left(\frac{x}{g_m} - 1 \right) \frac{x}{g_m^2} \right] q_m \\ \frac{\partial^2 q_m}{\partial {g_m}^2} &= \left[-\frac{1}{g_m} + \frac{1}{\sigma^2} \left(\frac{x}{g_m} - 1 \right) \frac{x}{g_m^2} \right]^2 q_m + \left[\frac{1}{g_m^2} + \frac{1}{\sigma^2} \left(\frac{x}{g_m} - 1 \right) \frac{-2x}{g_m^3} - \frac{1}{\sigma^2} \frac{x^2}{g_m^4} \right] q_m. \end{aligned}$$

By the definition of $\vec{y}$ the first derivative is zero so that

$$-\frac{1}{g_m} + \frac{1}{\sigma^2} \left(\frac{x}{g_m} - 1 \right) \frac{x}{g_m^2} = 0 \tag{21}$$

which is equivalent to requiring that the following holds:

$$g(\alpha_1 u_1[m], \ldots, \alpha_k u_k[m], \vec{y}[m] : \theta) = x[m] \frac{-1+\sqrt{1+4\sigma^2}}{2\sigma^2}. \tag{22}$$

Note that the negative solution is discarded due to the constraint $g > 0$. Also note that the link function in this case is in fact, as can be expected, a scaled version of $x[m]$. We can now extract $y[m]$ as before by simply inverting g_m.

The terms of the second derivative can now also be simplified:

$$\begin{aligned}
\frac{\partial^2 q}{\partial g^2} &= \left[\frac{1}{g^2} - \frac{1}{\sigma^2}\left(\frac{x}{g} - 1\right)\frac{2x}{g^3} - \frac{1}{\sigma^2}\frac{x^2}{g^4}\right] q \\
&= -\frac{1}{g^2}\left[1 + \frac{1}{\sigma^2}\frac{x^2}{g^2}\right] q \\
&= -\frac{1}{g^2} k_\sigma q
\end{aligned}$$

where the second and third equalities result from substituting Eq. (21) and Eq. (22), respectively, and k_σ is a positive constant function of σ. We can now express K in a dot product compact form

$$K(\vec{a}, \vec{b}) = k_\sigma \left(\vec{a} \circ \frac{g'(y)}{g(y)}\right) \cdot \left(\vec{b} \circ \frac{g'(y)}{g(y)}\right)$$

where $\frac{g'(y)}{g(y)}$ is the vector whose mth component is $\frac{g'_m}{g_m}$. Note that this instance specific weight is similar to the one we used for the non-linear additive Gaussian case of Theorem 7.1. In this more general setting, each instance m is additionally scaled by g_m. This has an intuitive explanation in the case of the multiplicative conditional density: the noise level is expected to go up with g and so all samples are rescaled to the same noise level.

For completeness, we write the additional $\log P$ terms in the expression of Eq. (19) for C_1 in the case of the multiplicative conditional density:

$$\begin{aligned}
\log P(\vec{x} \mid \mathbf{u}, \vec{y}) - \log P(\vec{x} \mid \mathbf{u}) &= -\sum \log(\sigma' g(\mathbf{u}[m], y[m])) + \sum \log(\sigma g(\mathbf{u}[m])) - \\
&\quad \frac{1}{2\sigma'^2}\sum\left(\frac{x[m]}{g(\mathbf{u}[m], y[m])} - 1\right)^2 + \frac{1}{2\sigma^2}\sum\left(\frac{x[m]}{g(\mathbf{u}[m])} - 1\right)^2 \\
&= -M \log \frac{-1+\sqrt{1+4\sigma'^2}}{2\sigma'} - \sum \log x[m] + \sum \log \sigma g(\mathbf{u}[m]) - \\
&\quad \frac{M}{2\sigma'^2}\left(\frac{2\sigma'^2}{-1+\sqrt{1+4\sigma'^2}} - 1\right)^2 + \frac{1}{2\sigma^2}\sum\left(\frac{x[m]}{g(\mathbf{u}[m])} - 1\right)^2
\end{aligned}$$

where σ' denotes the new variance parameter.

9. Experiments

We now examine the impact of the ideal parent method in two settings. In the first setting, we use this method for pruning the number of potential moves that are evaluated by greedy hill climbing structure search. We use this learning procedure to learn the structure over the observed or partially observed variables. In the second setting, we use the ideal parent method as a way of introducing new hidden variables, and also as a guide to reduce the number of evaluations when learning structure that involves hidden variables and observed ones, using a Structural EM search procedure.

9.1 Structure learning with Known Variables

In the first setting, we applied standard greedy hill climbing search (Greedy) and greedy hill climbing supplemented by the ideal parent method as discussed in Section 4 (Ideal). In using the ideal parent method, we used the C_2 similarity measure described in Section 3 to rank candidate edge additions and replacements, and then applied full scoring only to the top K ranking candidates per variable.

We first want to evaluate the impact of the approximation we make on the quality of the model learned. To do so, we start with a synthetic experiment where we know the true underlying network structure. In this setting we can evaluate the magnitude of the performance cost that is the result of the approximation we use. (We examine the speedup gain of our method on more interesting real-life examples below.) To make the synthetic experiment realistic, for the generating distribution we used a network learned from real data (see below) with 44 variables. From this network we can generate data sets of different sizes and apply our method with different values of K. Figure 5 compares the ideal parent method and the standard greedy procedure for linear Gaussian CPDs (left column) and sigmoid CPDs (right column). Using $K = 5$ is, as we expect, closer to the performance of the standard greedy method both in terms of training set [(a),(e)] and test set [(b),(f)] performance than $K = 2$. For linear Gaussian CPDs test performance is essentially the same for both methods. Using sigmoid CPDs we can see a slight advantage for the standard greedy method. When considering the percent of true edges recovered [(c),(g)], as before, the standard method shows some advantage over the ideal method with $K = 5$. However, by looking at the total number of edges learned [(d),(h)], we can see that the standard greedy method achieves this by using close to 50% more edges than the original structure for sigmoid CPDs. Thus, a relatively small advantage in performance comes at a high complexity price (and as we demonstrate below, at a significant speed cost).

We now examine the effect of the method on learning from real-life data sets. We base our data sets on a study that measures the expression of the baker's yeast genes in 173 experiments (Gasch et al., 2000). In this study, researchers measured the expression of 6152 yeast genes in its response to changes in the environmental conditions, resulting in a matrix of 173×6152 measurements. In the following, for practical reasons, we use two sets of genes. The first set consists of 639 genes that participate in general metabolic processes (Met), and the second is a subset of the first with 354 genes which are specific to amino acid metabolism (AA). We choose these sets since part of the response of the yeast to changes in its environment is in altering the activity levels of different parts of its metabolism. For some of the experiments below, we focused on subsets of genes for which there are no missing values, consisting of 89 and 44 genes, respectively. On these data sets we can consider two tasks. In the first, we treat genes as variables and experiments as instances. The learned networks indicate possible regulatory or functional connections between genes (Friedman et al., 2000). A complementary task is to treat the 173 experiments as variables (Cond). In this case the network encodes relationships between different conditions.

In Table 1 we summarize differences between the Greedy search and the Ideal search with K set to 2 and 5, for the linear Gaussian CPDs as well as sigmoid CPDs. Since the C_2 similarity is only a lower bound of the BIC score difference, we expect the candidate ranking of the two to be different.

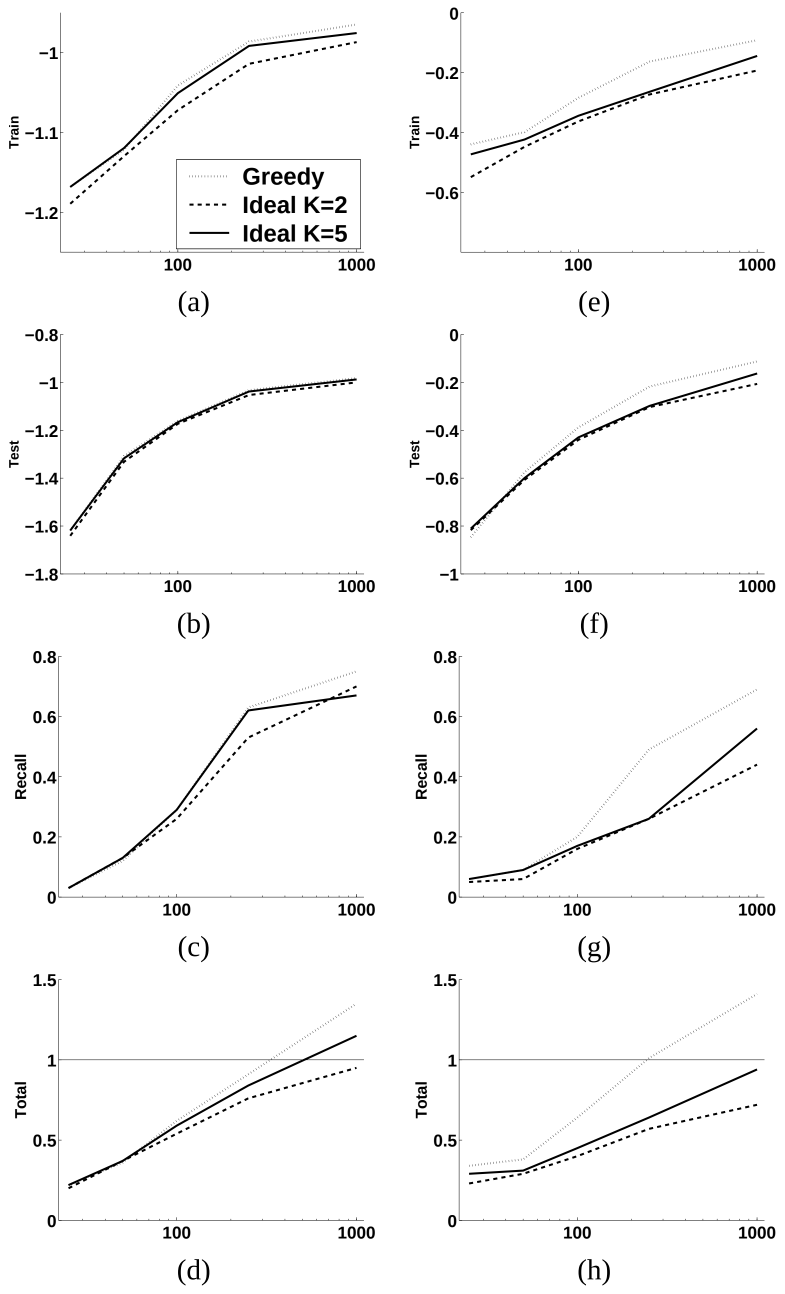


Figure 5: Evaluation of Ideal search on synthetic data generated from a real-life like network with 44 variables. We compare Ideal search with $K = 2$ (dashed) and $K = 5$ (solid), against the standard Greedy procedure (dotted). The figures show, as a function of the number of instances (x-axis), for linear Gaussian CPDs: (a) average training log-likelihood per instance per variable; (b) same for test; (c) fraction of true edges obtained in learned structure; (d) total number of edges learned as fraction of true number of edges; (e)-(h) same for sigmoid CPDs.

Data set	vars	inst	Greedy train	Greedy test	Ideal $K=2$ vs Greedy Δtrain	Δtest	edge	move	ev	sp	Ideal $K=5$ vs Greedy Δtrain	Δtest	edge	move	ev	sp
Linear Gaussian with complete data																
AA	44	173	-0.90	-1.07	-0.024	0.006	87.1	96.5	3.6	2	-0.008	0.007	94.9	96.5	9.3	2
AA Cond	173	44	-0.59	-1.56	-0.038	0.082	92.2	92.6	1.2	2	-0.009	0.029	96.9	98.2	2.9	2
Met	89	173	-0.79	-1.00	-0.033	-0.024	88.7	91.5	1.6	3	-0.013	-0.016	94.5	96.9	4.4	2
Met Cond	173	89	-0.59	-1.06	-0.035	-0.015	91.3	98.0	1.0	2	-0.007	-0.023	98.9	98.5	2.4	2
Linear Gaussian with missing values																
AA	354	173	-0.13	-0.50	-0.101	-0.034	81.3	95.2	0.4	5	-0.048	-0.022	90.7	96.0	0.9	5
AA Cond	173	354	-0.20	-0.38	-0.066	-0.037	74.7	87.5	0.4	14	-0.033	-0.021	86.3	101.1	1.6	11
Sigmoid with complete data																
AA	44	173	0.03	-0.12	-0.132	-0.065	49.7	59.4	2.0	38	-0.103	-0.046	60.4	77.6	6.1	18
AA Cond	173	44	-0.12	-0.81	-0.218	0.122	62.3	76.7	1.0	36	-0.150	0.103	73.7	79.4	2.3	21
Met	89	173	0.12	-0.08	-0.192	-0.084	47.9	58.3	0.9	65	-0.158	-0.059	56.6	69.8	2.6	29
Met Cond	173	89	0.22	-0.17	-0.207	-0.030	60.5	69.5	0.8	53	-0.156	-0.042	69.8	77.7	2.2	29

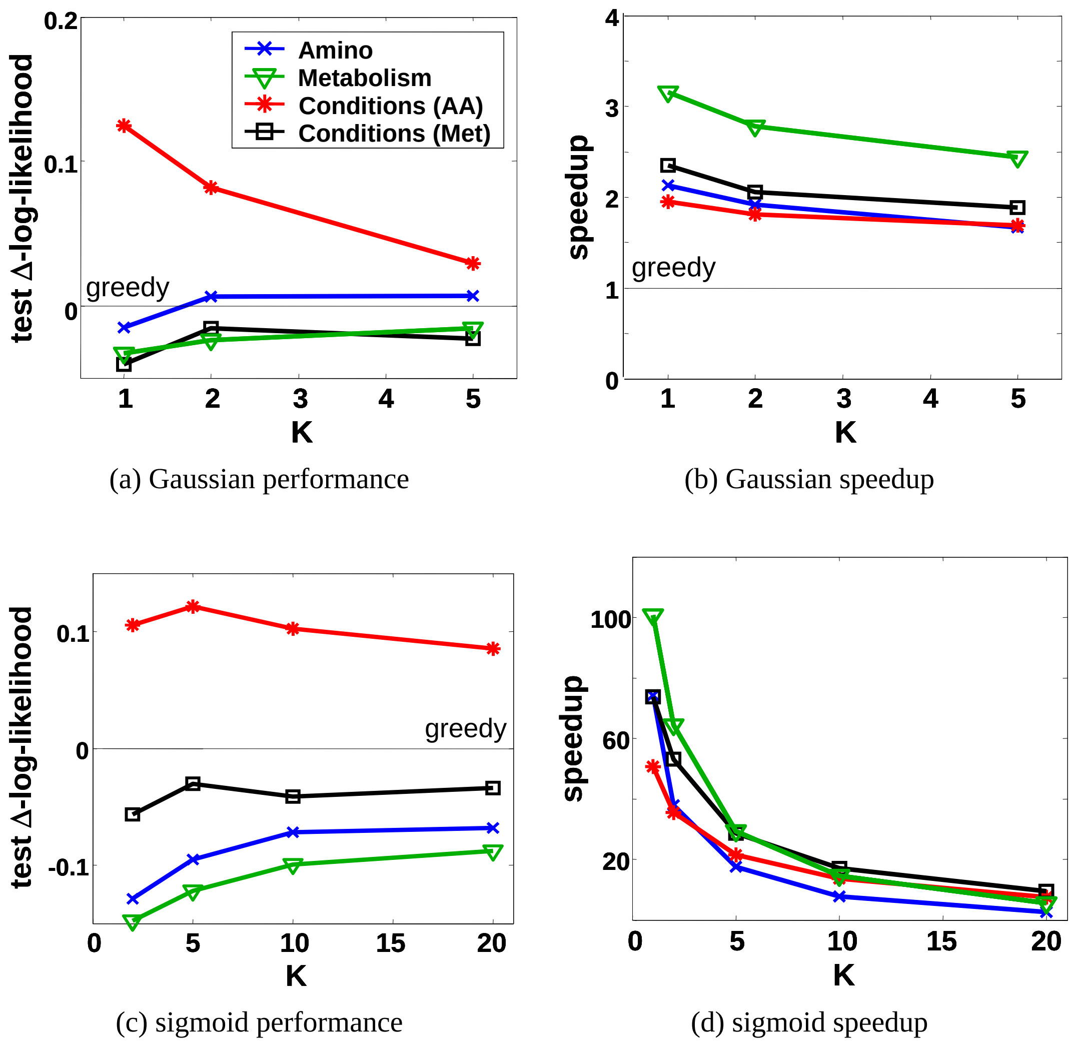
Table 1: Performance comparison of the Ideal parent search with $K=2$, $K=5$ and Greedy on real data sets. *vars* - number of variables in the data set; *inst* - the number of instances in the data set; *train* - average training set log-likelihood per instance per variable; *test* - same for test set; Δ*train* - average difference in training set log-likelihood per instance per variable; Δ*test* - same for test set; *edges* - percent of edges learned by Ideal with respect to those learned by Greedy; *moves* - percent of structure modifications taken during the search; *ev* - percent of moves evaluated; *sp* - speedup of Ideal over greedy method. All numbers are averages over 5 fold cross validation sets.

As most of the difference comes from freezing some of the parameters, a possible outcome is that the Ideal search is less prone to over-fitting. Indeed, as we see, though the training set log-likelihood in most cases is lower for Ideal search, the test set performance is only marginally different than that of the standard greedy method, and often surpasses it.

Of particular interest is the tradeoff between accuracy and speed when using the ideal parent method. In Figure 6 we examine this tradeoff in four of the data sets described above using linear Gaussian and sigmoid CPDs. For both types of CPDs, the performance of the ideal parent method approaches that of Greedy as K is increased. As we can expect, in both types of CPDs the ideal parent method is faster even for $K=5$. However, the effect on total run time is much more pronounced when learning networks with non-linear CPDs. In this case, most of the computation is spent in optimizing the parameters for scoring candidates. Indeed, careful examination of the number of structural moves taken and the number of moves evaluated in Table 1, shows that the dramatic speedup is mostly a result of the reduction in the number of candidates evaluated. Importantly, this speedup in non-linear networks makes previously "intractable" real-life learning problems (like gene regulation network inference) more accessible.

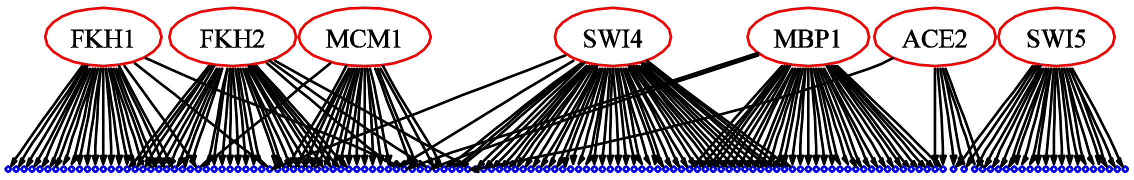
9.2 Learning Hidden Variables

In the second experimental setting, we examine the ability of our algorithm to learn structures that involve hidden variables and introduce new ones during the search. In this setting, we focus on *two layered networks* where the first layer consists of hidden variables, all of which are assumed to be roots, and the second layer consists of observed variables. Each of the observed variables is a leaf and can depend on one or more hidden variables. Learning such networks involves introducing

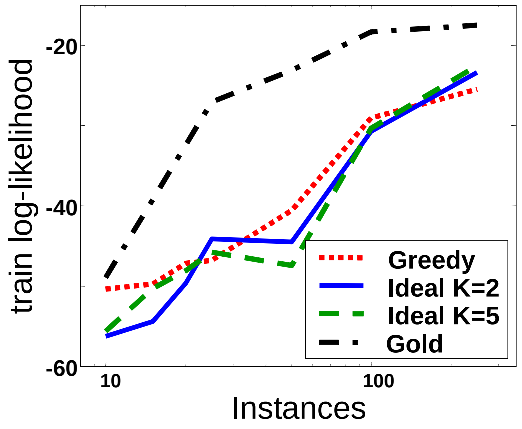


(a) Gaussian performance (b) Gaussian speedup (c) sigmoid performance (d) sigmoid speedup

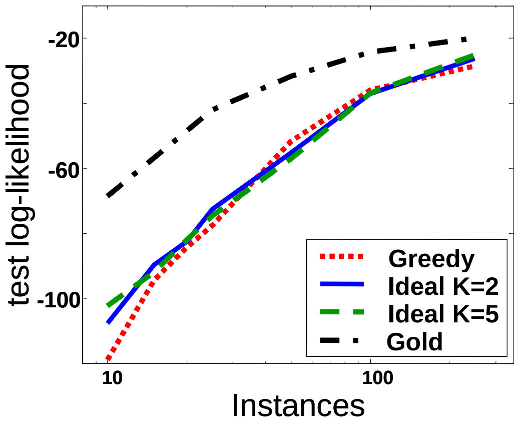
Figure 6: Evaluation of Ideal search on real-life data using 5-fold cross validation. (a) average difference in log-likelihood per instance on test data when learning with linear Gaussian CPDs relative to the Greedy baseline (y-axis) vs. the number of ideal candidates for each family K (x-axis). (b) Relative speedup over Greedy (y-axis) against K (x-axis). (c),(d) same for sigmoid CPDs.



(a)



(b)



(c)

Figure 7: Evaluation of performance in two-layer network experiments. (a) Gold structure with 141 which was curated by a biological expert and used to generate synthetic data; (b) average log-likelihood per instance on *training* data (y-axis) for Greedy , Ideal search with $K = 2$ and Ideal search with $K = 5$, when learning with linear Gaussian CPDs against the number of training samples (x-axis); (c) Same for *test* set.

different hidden variables, and determining for each observed variable which hidden variables it depends on.

As in the case of standard structure learning, we first want to evaluate the impact of our approximation on learning. To test this, we used a network topology that is curated (Nachman et al., 2004) from biological literature for the regulation of cell-cycle genes in yeast. This network involves 7 hidden variables and 141 observed variables. We learned the parameters for the network from a cell cycle gene expression data set (Spellman et al., 1998). From the learned network we then sampled data sets of varying sizes, and tried to recreate the regulation structure using either greedy search or ideal parent search. In both search procedures we introduce hidden variables in a gradual manner. We start with a network where a single hidden variable is connected as the only parent to all observed variables. After parameter optimization, we introduce another hidden variable - either as a parent of all observed variables (in greedy search), or to members of the highest scoring cluster (in ideal parent search, as explained in Section 5). We then let the structure search modify edges (subject to the two-layer constraints) until no beneficial moves are found, at which point we introduce

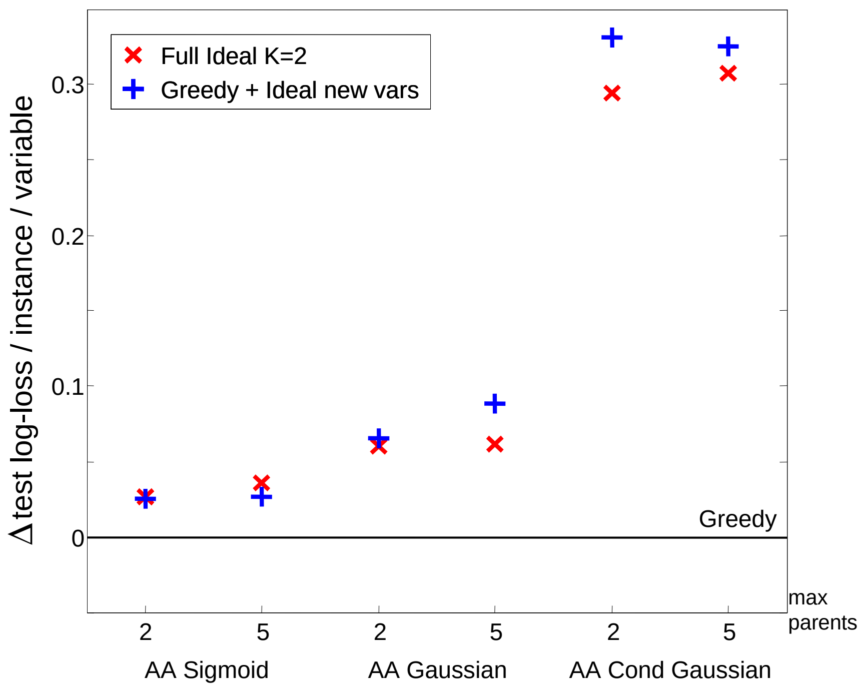


Figure 8: Structure learning of bipartite networks where the parents are new hidden variables and the children are the observed variables. The different data sets of the baker's Yeast include: AA with 44 variables for both Gaussian and sigmoid Gaussian CPDs; AA Cond with 173 variables and Gaussian CPDs. For each data set a structure with up to 2 or 5 parents was considered. Shown is the test log-likelihood per instance per variable relative to the baseline of the standard greedy structure learning algorithm.

another hidden variable, and so on. The search terminates when it is no longer beneficial to add a new variable.

Figure 7 shows the performance of the ideal parent search and the standard greedy procedure as a function of the number of instances, for linear Gaussian CPDs. As can be seen, although there are some differences in training set likelihood, the performance on test data is essentially the same. Thus, as in the case of the yeast experiments considered above, there was no degradation of performance due to the approximation made by our method.

We then considered the application of the algorithms to real-life data sets. Figure 8 shows the test set results for several of the data sets of the baker's yeast (Gasch et al., 2000) described above, for both Gaussian and sigmoid Gaussian CPDs. The full ideal parent method (red 'x') with $K = 2$ and the ideal method for adding new hidden variables is consistently better than the baseline greedy procedure. To demonstrate that the improvement is in large part due to the guided method for adding hidden variables we also ran the baseline greedy procedure for structure changes augmented with the ideal method for adding new hidden variables (blue '+'). As can be seen, the performance of this method is typically slightly better than the full ideal method, since it does not approximate the structural adaptation stage. In this setup, the only difference from the greedy baseline is the way that new hidden variables are introduced. Thus, these results support our hypothesis that the ideal method is able to introduce effective new hidden variables, that are preferable to a hidden variables that are naively introduced into the network structure.

The superiority of the sigmoid Gaussian over the Gaussian model for the AA data set (in the order of 1 bit per instance per variable) motivates us to pursue learning of models with non-linear CPDs. We could not compare the different methods for the larger data sets as the greedy method was several orders of magnitudes slower than our ideal parent method and did not complete runs given several days of CPU time (in the linear Gaussian case the ideal parent method was roughly 5 times faster than the standard greedy approach). We believe that the ability of the ideal method to avoid over-fitting will only increase its strength in these more challenging cases.

We also considered the application of our algorithm to the real-life cell-cycle gene expression data described in the previous section with linear Gaussian CPDs. Although this data set contains only 17 samples, it is of high interest from a biological perspective to try and infer from it as much as possible on the structure of regulation. We performed leave-one-out cross validation and compared the ideal parent method with $K = 2$ and $K = 5$ to the standard greedy method. To help avoid over-fitting, we limited the number of hidden parents for each observed variable to 2. In terms of training log-likelihood per instance per variable, the greedy method is better than the ideal method by 0.4 and 0.42 bits per instance, for $K = 5$ and $K = 2$, respectively. However, its test log-likelihood performance is significantly worse as a result of high over-fitting of two particular instances, and is worse by 0.72 bits per instance than the ideal method with $K = 5$ and by 0.88 bits per instance than the ideal method with $K = 2$. As we have demonstrated in the synthetic example above, the ability of the ideal method to avoid over-fitting via a guided search, does not come at the price of diminished performance when data is more plentiful. When the observed variables were allowed to have up to 5 parents, all methods demonstrated over-fitting, which for Greedy was far more severe.

10. Discussion and Future Work

In this work we set out to learn the structure of Bayesian networks with continuous variables. Our contribution is twofold: First, we showed how to speed up structure search, particularly for non-linear conditional probability distributions. This speedup is essential as it makes structure learning feasible in many interesting real life problems. Second, we presented a principled way of introducing new hidden variables into the network structure. We used the concept of an ideal parent for both of these tasks and demonstrated its benefits on both synthetic and real-life biological domains. In particular, we showed that our method is able to effectively learn networks with hidden variables that improve generalization performance. In addition, it allowed us to cope with domains where the greedy method proved too time consuming.

Several works in recent years have tried to address the complexities involved in structure learning using different approaches. To name a few examples, Chickering (1996b) suggests searching the smaller space of Bayesian network equivalence classes. Moore and Wong (2003) suggest innovative global search operators that completely sever and reinsert a variable into the network structure. They take advantage of the fact that the set of children can be computed efficiently and use a branch and bound technique for computing the parent set. Koivisto and Sood (2004) were the first to show how the problem of exact structure learning can be made less than super-exponential by conditioning on the ordering of variables and the use of dynamic programming. Singh and Moore (2005) propose a different dynamic programming approach for learning the exact structure of Bayesian networks by considering an alternative recursive formulation. They compare the complexity of their approach to that of Koivisto and Sood (2004) under different settings. Silander and Myllym (2006) build on the same order based idea and propose a somewhat simpler algorithm that recursively builds the network structure from the "sinks" of the optimal structure toward the roots. Teyssier and Koller (2005) perform an intelligent order-based search that is not guaranteed to find the optimal structure but significantly reduces the running time of the search procedure, and finds high scoring structures in practice.

In contrast to these approaches that focus on the search strategy, our "Ideal Parent" approach leverages on the parametric structure of the conditional distributions. This allows us to get a fast approximation of the contribution of a search operator. In here, we applied this approach in conjunction with a greedy search algorithm. However, it can also be supplemented to many other search procedures as a way of dramatically reducing the number of candidate moves that are carefully evaluated.

Two works are of particular interest and relevance to ours. Della Pietra et al. (1997) suggest an efficient method for incrementally inducing features of Markov random fields. To efficiently approximate the merit of candidate feature, they evaluate the improvement in likelihood when the only parameter that can change is the one associated with the new feature. Thus, all other parameters of the model are held fixed during the evaluation. For binary features, they find a closed-form solution for the improvement. For more general features, they use non-linear optimization to perform the evaluation. The idea of freezing some parameters in order to facilitate approximate but efficient computations is also the basis for our development of the approximate score. The context of continuous Bayesian networks, as well as the details of the likelihood functions involved in computations, however, are quite different.

Another connection is to the "Sparse Candidate" procedure of Friedman et al. (1999), which limits the number of candidate parents considered by the search procedure. While sharing the motivation of our work, their pre-pruning of candidates does not take advantage of the form of the conditional distribution nor does it try to approximate the benefit of a candidate directly. Instead, they used statistical signals as a surrogate for the benefit of a candidate parent. Thus, these methods are in fact orthogonal and it would be intriguing to see if the "Ideal Parent" method can help the "Sparse Candidate" method during the pruning stage.

The parametric form of CPDs we examined here are specific instances of *generalized linear models* (GLMs) (McCullagh and Nelder, 1989). This class of CPDs uses a function g that is applied to the sum of its arguments, called the *link function* in the GLM literature. However, we can also consider more complex functions, as long as they are well defined for any desired number of parents. For example, in Nachman et al. (2004) models based on chemical reaction models are considered, where the function g does not have a GLM form. An example of a two variable function of this type is:

$$g(y_1, y_2 : \theta) = \theta \frac{y_1 y_2}{(1+y_1)(1+y_2)}.$$

We also note that GLM literature deals extensively with different forms of noise. While we mainly focus here on the case of additive Gaussian noise, and briefly addressed other noise models, the ideas we propose here can be extended to many of these noise distributions.

Few works touched on the issue of when and how to add a hidden variable in the network structure (e.g., Elidan et al., 2001; Elidan and Friedman, 2003; Martin and VanLehn, 1995; Zhang, 2004). Only some of these methods are potentially applicable to continuous variable networks, and none have been adapted to this context. To our knowledge, this is the first method to address this issue in a general context of continuous variable networks.

Many challenges remain. First, instead of scoring the top K candidate parents of each variable, we could evaluate only the K most promising candidates over *all* possible structure modifications. In doing so we could make use of the superiority of the $C2$ measure over the $C1$ measure, and further improve the speed of our method, possibly by another order of magnitude. Second, the “Ideal Parent” method can be combined as a plug-in for candidate selection with other innovative search procedures. Third, we want to adapt our method for additional and more complex conditional probability distributions (e.g., Nachman et al., 2004), and extend it to multi-modal distributions. Fourth, we want to improve the approximation for adding new hidden variables in the non-linear case. Finally, it might be possible to leverage on the connection to Generalized Linear Models for handling more elaborate noise models.

Acknowledgments

We thank Shai Shwartz and the anonymous reviewers for comments on earlier versions of this manuscript. This work was supported, in part, by grants from the Israeli Ministry of Science and US-Israel Bi-national Foundation. I. Nachman and G. Elidan were also supported by the Horowitz fellowship. N. Friedman was also supported in part by the Bauer Center for Genomics Research, Harvard University.

References

D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, New York, 1996a.

D. M. Chickering. Learning equivalence classes of Bayesian network structures. In E. Horvitz and F. Jensen, editors, *Proc. Twelfth Conference on Uncertainty in Artificial Intelligence (UAI ’96)*, pages 150–157, San Francisco, 1996b. Morgan Kaufmann.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39:1–39, 1977.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

G. Elidan and N. Friedman. The information bottleneck EM algorithm. In C. Meek and U. Kjærulff, editors, *Proc. Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI ’03)*, pages 200–208, San Francisco, 2003. Morgan Kaufmann.

G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: A structure-based approach. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 479–485, Cambridge, Mass., 2001. MIT Press.

N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In D. Fisher, editor, *Proc. Fourteenth International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, San Francisco, 1997.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Computational Biology*, 7:601–620, 2000.

N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive data sets: The 'sparse candidate" algorithm. In K. Laskey and H. Prade, editors, *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, page 206–215, San Francisco, 1999.

A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression program in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11:4241–4257, 2000.

D. Geiger and D. Heckerman. Learning Gaussian networks. In R. López de Mantarás and D. Poole, editors, *Proc. Tenth Conference on Uncertainty in Artificial Intelligence (UAI '94)*, pages 235–243, San Francisco, 1994. Morgan Kaufmann.

F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational approximations methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.

J. Martin and K. VanLehn. Discrete factor analysis: Learning hidden variables in Bayesian networks. Technical report, Department of Computer Science, University of Pittsburgh, 1995.

P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman & Hall, London, 1989.

A. Moore and W. Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, pages 552–559, Menlo Park, California, 2003.

K. Murphy and Y. Weiss. Loopy belief propagation for approximate inference: An empirical study. In K. Laskey and H. Prade, editors, *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, page 467–475, San Francisco, 1999. Morgan Kaufmann.

I. Nachman, A. Regev, and N. Friedman. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20(Suppl 1):S1248–1256, 2004.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–55, 1991.

T. Silander and P. Myllym. A simple approach for finding the globally optimal Bayesian network structure. In Dechter and Richardson, editors, *Proc. Twenty Second Conference on Uncertainty in Artificial Intelligence (UAI '06)*, San Francisco, 2006. Morgan Kaufmann.

A. Singh and A. Moore. Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, 2005.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12): 3273–97, 1998.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In F. Bacchus and T. Jaakkola, editors, *Proc. Twenty First Conference on Uncertainty in Artificial Intelligence (UAI '05)*, pages 584–590, San Francisco, 2005. Morgan Kaufmann.

R. Parr U. Lerner and D. Koller. Bayesian fault detection and diagnosis in dynamic systems. In *Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI)*, pages 531–537, 2000.

J. Wilkinson. *The Algebric Eigenvalue Problem*. Claderon Press, Oxford, 1965.

N.L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.