The Structure of the Unobserved

Ricardo Silva
Center for Automated Learning and Discovery
rbas@cs.cmu.edu

June 7, 2002
CMU-CALD-02-102

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Graphical models such as Bayesian networks and structural equation models are widely used as a representation for probabilistic distributions and causal relations. Their representational power can be vastly increased if one includes variables that cannot be measured directly. Such "latent" variables are able to explain situations in which two measured variables are correlated but no causation exists between them, and no other observed variable is a common cause of such pair. For probabilistic modeling, latent variables often allow the representation of a probabilistic distribution with fewer parameters than in models composed only of observed variables. One useful category of latent variable graphs is the measurement/structural model in which all observed variables are measurements of some latent variable, and there are causal relationships among the unobserved variables that explain the correlation of the observed ones. In this work, we introduce and evaluate principled strategies for clustering measurements and discovering probabilistic independences among latents in order to reconstruct the causal relations of the unobserved variables.

Keywords: Causality discovery, Graphical models, Structural Equation Models

1 Introduction

Scientific models are generally limited by the information that can be measured. Important variables are often left out of the final model because they are not measurable according to the current available technology (such as the complete dynamics of proteins during gene regulation processes inside cells), or because they cannot be measured directly (such as IQ) and we have to rely on indicators that approximate such abstract concepts. Futhermore practical matters such as the curse of dimensionality may sometimes drive omission of relevant and available covariates.

The separation between observational terms and theoretical, non-observable, terms seems to be a standard practice among scientist of different domains. Many schools of philosophy of science explicitly approach these issues, from the very definition of what can be considered an observational term to how to validate a theoretical concept. For example, one key characteristic of classical positivism is requiring that all theoretical entities should be logically tied to observational terms [22].

Therefore, measured and/or measurable variables should not be the only ones considered when building a model, and not necessarily one should consider discarding variables because one does not have enough data to make a high dimensional model reliable. Actually, the addition of unmeasured quantities in practice is quite common, and researchers refer to these variables as **hidden**, or **latent** variables, and use them to sidestep some of the previously mentioned limitations [5].

Latent variables are essential in causal modeling, where they help distinguishing the contributions of a common cause from direct causation [33]. Also, a good use of latent variables usually allows the representation of probabilistic dependencies by requiring the specification of fewer parameters that would be necessary in a competing model that does not include such variables [4]. For example, by grouping many observable variables into fewer clusters tied up by latents, it is possible to reduce the dimensionality of a model and provide insights about common factors that explain the sample variance-covariance structure.

In this work, we introduce techniques to learn from data some specific classes of latent variable models. Using the formal languages of graphs and probabilities, questions such as how many latents should be introduced in our model, and how the observations should be grouped are answered under the light of some specific assumptions. Focusing in a class of graphs here called **measurement/structural models**, a family of algorithms is described with the goal of discovering equivalence classes of causal relantionships among observed and latent variables. Before introducing our approach, we provide a context by surveying standard and current work in latent variable models. Concluding this work, an extensive empirical evaluation of the proposed methods is discussed.

2 Latent variable models

Bayesian networks [26] have become a widely used knowledge representation language in artificial intelligence. Similar graphical languages to represent causality and probabilitic distributions have been used in other research fields for decades. For instance, **structural equation models** (SEMs) are well-known in econometrics, social sciences, psychology, among others. In Appendix A, we present a brief introduction in Bayesian networks and SEMs, providing some definitions such as **d-separation** and **faithfulness**. Even though automatic methods for discovery of latent variables are also present in other knowledge representation languages and learning algorithms (such as the inverse resolution method in Inductive Logic Programming [8]), we will focus only on probabilistic graphs.

One of the primary motivations for the introduction of latent variables is exemplified by this simple case introduced in [4]: in Figure 1, we have two graphical models. The node marked with H is a hidden variable, while the remaining ones are observed variables. Both graphs entail the same set of conditional independence relationships among the observed nodes. However, the leftmost graph requires many fewer parameters to represent the same probabilistic distribution. Dimensionality reduction has been the motivation for several models that introduce hidden variables, implicitly or explicitly.

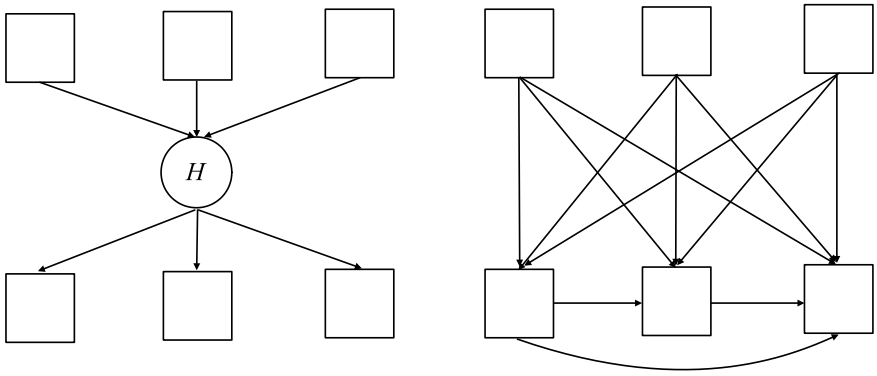

Figure 1: An example of how latent variables can help to explain the same probability distributions and d-separations relations with much simpler structures. Both graphs have the same d-separations among the observed variables (rectangular nodes). If we assume that the hidden variable is binomial, and the observed ones are multinomial with three values, the graph in the left requires only 45 parameters, while the second graph requires 708.

One classical statistical model that can be regarded as a latent graphical model is factor analysis (FA). The generative model in factor analysis assumes that a set η of latents follows a Gaussian distribution with a diagonal covariance matrix. The observed vector $\mathbf{y}$ is generated from a linear combination of the latents, with an additive error that is usually represented by independent normally distributed random variables. Algebrically, a FA model can be represented as

$$\mathbf{y} = \Lambda\eta + \epsilon \tag{1}$$

where ϵ represents the (diagonal) matrix of error terms, while Λ is the projection matrix of the latents over the subspace spanned by the observed variables minus their respective measurement errors.

Figure 2 depicts a factor analysis model as a graphical model. Following the notation introduced in Appendix A, observed nodes are represented as rectangles, while latent ones are represented as circles. Notice that the error terms are also latent variables.

Let the covariance matrix among the latents be the identity matrix $\mathbf{I}$, and let Ω be the covariance matrix of the error terms. Writing down the marginal distribution of $\mathbf{y}$ as a function of the vector of parameters $\Theta = [\Lambda, \Omega]^{\mathbf{T}}$, assuming its mean is set to zero, will result in a Gaussian probability density function with zero mean and covariance matrix given by

$$Cov(\mathbf{y}) = \Lambda\Lambda^{\mathbf{T}} + \Omega \tag{2}$$

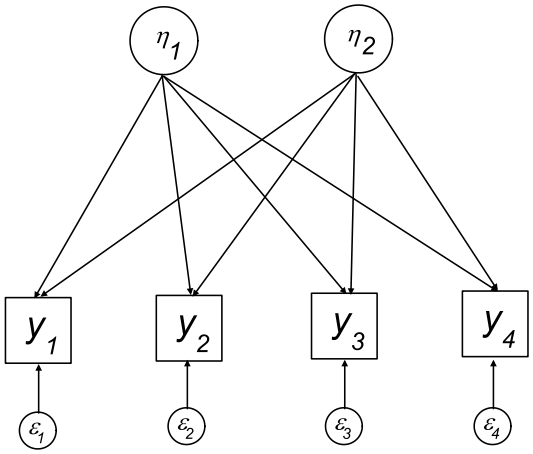

Figure 2: The graphical representation of a factor analysis model. Error terms ϵ_i are also represented as extra latent variables. Nodes representing error terms will be considered implicit and not appear in the subsequent figures.

Notice that Λ could be substituted by any matrix of the type $\Lambda\mathbf{T}$, where $\mathbf{T}$ is an orthogonal matrix. The issue of **parameter identifiability** arises often when dealing with structural equation models with latent variables: sometimes the observed marginal distribution function is not enough to uniquely identify the parameters. In FA, a standard solution is to introduce aditional constraints, such as the varimax criterion [19]. Pearl [28] discusses different definitions of parameter identifiability.

Basilevsky [2] describes several extensions of the ordinary factor analysis, including different methods for fitting and testing FA models.

2.1 Recent work

Factor analysis and related methods have gained momentum throught recent works in data mining, artificial neural networks and machine learning in general. Due to the advancement in databases technology, it is much more common today to find data sets of high dimensionality. Increase in computational power gave motivation for the design and evaluation of more complex models, beyond the simple linear and normally distributed cases of classical FA.

For instance, the closely related method of **principal component analysis** (PCA) has been the core motivation for many recent probabilistic modeling approaches. The original PCA was just a method for finding linear projections of multivariate observations that minimized the mean squared error of the projected data. PCA can be be used for dimensionality reduction because it provides a set of linear projections ordered by the amount of explained variance. Standard PCA is just an algebraic method that does not define any probabilistic model. However, one can choose a set of projections that explains some percentage of the variance of the data, and the number of principal components that correspond to a reasonable amount of the variance is usually much less than the number of variables. Extra modeling is then performed in this transformed space.

Bishop [5] derives families of latent variable models that are introduced as probabilistic principal component analysis approaches, which are basically variations of factor analysis. A Bayesian framework is used to describe these families. For instance, assume an isotropic noise model, where

ϵ follows $N(0, \sigma^2\mathbf{I})$, and the following prior over the latents variables

$$f(\eta) = (2\pi)^{-q/2} \exp\left(-\frac{1}{2}\eta^{\mathbf{T}}\eta\right) \tag{3}$$

The conditional probability for the observed vector is

$$f(\mathbf{y}|\eta) = (2\pi\sigma^2)^{-d/2} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{y} - \Lambda\eta - \mu)\right\} \tag{4}$$

where d is the dimensionality of the observed space and q is the dimensionality of the latent space. The marginal over the observed space can then be obtained by integrating out the latents:

$$\begin{aligned} f(\mathbf{y}) &= \int f(\mathbf{y}|\eta) f(\eta)\, d\eta \\ &= (2\pi)^{-d/2}|\mathbf{C}|^{-1/2} \exp\left\{-\tfrac{1}{2}(\eta - \mu)^T \mathbf{C}^{-1}(\eta - \mu)\right\} \end{aligned}$$

with covariance matrix $\mathbf{C} = \sigma^2\mathbf{I} + \Lambda\Lambda^{\mathbf{T}}$. This is the explicit Bayesian way of deriving the implied covariance matrix for this FA model. It is pointed out that the isotropic noise assumption allows an exact analytical solution for the model parameters. Bishop also discusses flexible methods that allow fitting more than multivariate Gaussian data. This is performed by introducing hierarchical mixture models. Imagine that one introduces a discrete latent variable π that assumes m values. The generative model consists in first probabilistic choosing which of the m values of this variable will be selected, and then generating the observed multivariate data point according to a π-dependent matrix, $\Lambda(\pi)$. This is a mixture of FA models (or "probabilistic principal component analysers"), and several variations of the standard optimization algorithms such as expectation-maximization (EM) can be readily applied to it [14]. Minka [23] suggests parameter priors and a Laplace approximation for computing the posterior of isotropic probabilistic PCA models. This method is a well-define criterion for performing Bayesian model selection of the number of principal components (or factors, in factor analysis parlance).

Independent component analysis (ICA) [16] is another generative model where latents are independent and common causes of the observed variables. The key difference between standard FA and standard ICA is the fact that while FA assumes that the latents follow a normal distribution, ICA is more flexible and actually assumes that η is *not* multivariate normal. While FA needs an aditional constraint statement in order to keep the coefficient matrix Λ identifiable, ICA sidesteps this problem by assuming that at most one of the latent components is normal. This condition is enough to guarantee parameter identifiability. Two limitations of standard ICA are evident: the number of latents equals the number of observed variables, and there are no noise terms added to the model. Observational terms are deterministic functions of their latent causes. This is not a major issue for many ICA applications, since this method was created mostly for signal processing tasks where probability estimations are not crucial.

However, given that there are many variants within ICA and other methods, these differences may sound artificial and it is not surprising that a variety of frameworks with the goal of unifying these methodologies have been recently developed. Attias [1] presents a very general algorithm that can handle the case where the number of latents is different from the number of observed variables, noise covariance is estimated and the distributions among the latents can be made increasingly complex by adding more basis distribution in a mixture model that is not necessarily Gaussian. His **independent factor analysis** method reduces to standard FA when latents are normally distributed, and is equivalent to PCA in the zero-noise limit. A variational approximation is introduced to make implementations practical when the number of latents is high.

The methods described above have in common the fact that latents are independent causes of the observed variables. Under an algorithmic point of view, these procedures can be interpreted as projection pursuit strategies [20]: the search for a projection that maximizes some "interesting" criterion such as an orthogonal projection that maximizes the variance of its components. More generally, latent graphical models may have correlated latent variables, and observed variables may be causes of latents also. A variety of algorithms for learning Bayesian networks with latents exists, each with its own advantages and disadvantages.

Spirtes, Glymour and Scheines [33] introduced a general method for discovering hidden variables using conditional independence statements among the observed ones as an input. For some parametric families such as normal and multinomial distributions, there are known tests of statistical significance of conditional independences in samples. However, since one is performing multiple tests, this increase the chance of erroneously accepting the outcome of certain tests due to the multiplicity effect [18]. This algorithm, **Fast Causal Inference** (FCI), still presents a variety of important advantages: it does not require any prior knowledge about the number and position of latent variables; output is represented by a special kind of graph (i.e., not a DAG) that encodes causal information with an elegant notation that allows easy interpretation [1]; even though FCI is exponential in the number of observed variables, in average case it can be very efficient assuming that the true graph that generated the sample is relatively sparse.

A complementary approach is Bayesian learning [15] [7]. True Bayesian learning is unfeasible for all but very small networks: making predictions using a complete Bayesian approach consists in averaging the prediction of every possible network among the observed nodes, weighted by their posterior probability given the sample, which is super-exponential in the number of such variables. Instead, in many cases just one network (or a small set of networks) is used for prediction. This network is usually found by a maximum a posteriori approach: given a score measuring (or approximating) the posterior probability of a network, the ideal solution is return the network with the highest score. Still, searching for this network is unfeasible due to the large search space, and heuristics are necessary. This kind of Bayesian-inspired heuristic search is usually called **score-based search**.

Even with results such as [13], still there is no agreement about which is the best way to score Bayesian networks with an undetermined number of latent variables. Friedman and Koller [12] present recent results in heuristics for Bayesian model averaging for structure discovery in Bayesian networks. Friedman's **Bayesian Structural EM Algorithm** [11] is a generalization of EM algorithms for model fitting. During the maximization step, instead of searching for the parameter values that maximize the score of the current model given expected values for the unmeasured variables, Structural EM searches for the best model among a set of possible models with their respective best parameters. Friedman introduces a series of approximations in order to make this procedure computationally feasible. As in regular EM, this algorithm is guaranteed to converge. However, unlike FCI, the number of hidden variables should be fixed a priori or chosen by a grid search over a pre-specified interval, making a final comparison among the best models of each step.

Elidan *et al.* [10] describe heuristics for introducing latent variables. The idea is finding dense subnetworks (i.e., subgroups that are almost cliques) and add latent variables into the paths connecting those variables. The goal is minimizing the number of parameters that are necessary to enconde the underlying probability function, much in the spirit of Figure 1. Elidan and Friedman [9] also used it estimate the cardinality of categorical hidden variables. This method for latent discovery, however, is just a heuristic and does not entail correct discovery of causal relationships

[1] The output of FCI may be not much informative, depending on the constraints that are used as an input.

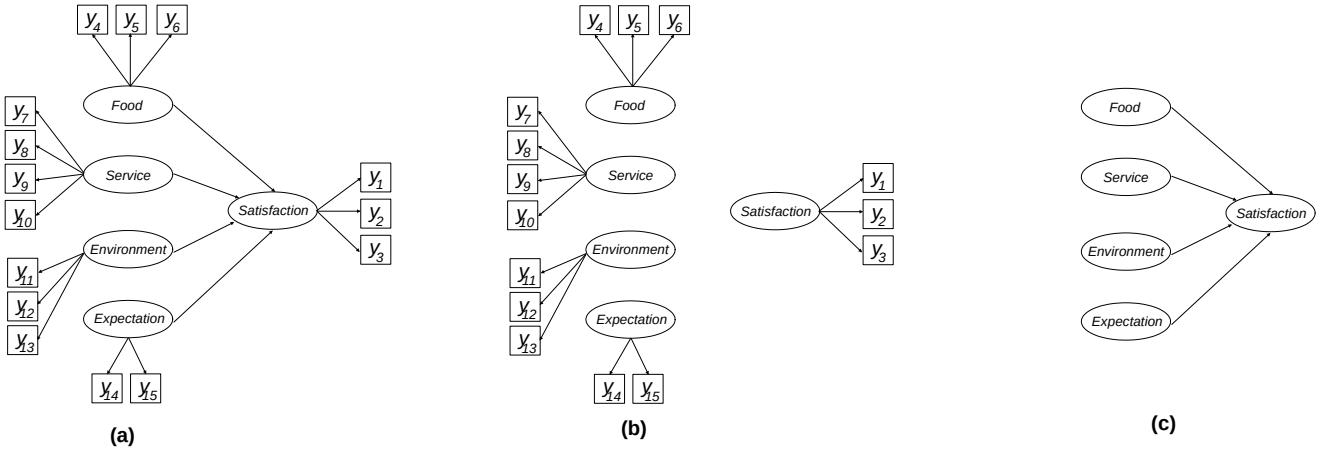

Figure 3: An example of measurement/structural model. Part (a) depicts the full model, while (b) is the measurement model and (c) the structural model.

under explicit assumptions, as performed, for instance, by FCI. Its power relies in finding models with good fit and fewer parameters, adequate for probabilistic modeling, but not necessarily searching in a search space that deduces the correct conditional independence statements in the limit of infinite sample size.

Having correct conditional independences expressed in a Bayesian network or related graphical models is essential for causal inference. On the other hand, constraint-satisfaction methods such as FCI are more brittle with respect to smaller sample sizes, and that is also a very important issue. Ultimately, the method of choice will depend in the availability of data and in the interest of discovering causal relationships or good models for prediction without manipulation.

In the next section, we describe in full detail a specific class of latent variable model. The next two sections will describe methods for reconstructing such models out of samples.

3 Measurement and structural models

The general acyclic Bayesian network is a powerful method for representing uncertainty and causal relationships. However, it may be computationally intensive to learn from data and performing inference. A solution to this problem is performing numerical approximations and using heuristics. As another alternative, or as a complement for approximations, one could focus in specialized graphs such as factor analysis variations.

Consider a generalization of FA where the factors are causally related and each factor is a cause of only a subset of the available measures. In econometrics, psychometrics, sociology and other domains, it is common that measured variables are **indicators** of unmeasured, latent concepts. They are measures of only one or a small handful of the latents. In artificial intelligence, clustering observed variables according to underlying abstract concepts may render very large models tractable. Figure 3a depicts one example of this kind of model, extracted from [17]. Five latents are used: *Satisfaction*, *Food Quality*, *Service Quality*, *Dining environment*, and *Confirmation of expectations*. Measures are made throught questionnaires given to students that use their university dining services. Among the measures, we have the student's own indication of satisfaction, nutritive quality of the food, student's opinion about the friendlyness and efficience of the service, student's opinion of cleanness and pleasantness of the environment, and their prior expectations.

	Description
η	latent variables
$\mathbf{y}$	indicators
$\mathbf{B}$	matrix of latent loadings
Λ	matrix of indicator loadings
ζ	error terms for latents
ϵ	error terms for indicators
Φ	covariance matrix of ζ
Ω	covariance matrix of ϵ

Table 1: Notation for the parameters used in the measurement/structure model.

We say that the subgraph formed by the latents and their measures is the **measurement model**, while the subgraph among the latents and the respective edges is the **structural model**. A group formed by one latent and its indicators can also be called a **cluster**. Figures 3b and 3c depicts this categorization according to our example.

We argue that measurement/structural models provide better insight about data generating processes than factor analysis and its variants. In general applications, there is no reason for assuming that factors are independent. This has been acknowledge even by research in ICA [16], which is widely used for signal processing applications where in many cases the independence assumption makes sense. While useful for dimensionality reduction, methods such as factor analysis and principal components are non-trivial to interpret outside the realm of textbooks. An approach that does not overconstraint the latents, while not necessarily tackling the more complex problem of representing general acyclic networks may represent a good compromise. One natural application is modeling large survey questionnaires, where many of the questions are intended to be indicators of more abstract terms such as "education", "political preferences", "economical stability", to name a few.

Our main interest in this work is unveiling the structure of the unobserved variables, discovering equivalence classes of causal relationships among the latents relying in non-experimental data. We will restrict our approach to linear models with additive errors following normal distributions, in the vein of standard factor analysis. Although not as flexible as more general nonlinear methods, this will provide a basis for future research in causal discovery algorithms for measurement/structural models. Linear Gaussian models still have a variety of applications, as exemplified in the comprehensive review by Roweis and Ghahramani [30].

Adapting from Bollen [6] [2], we will use a similar notation to represent the linear equations that parameterize our models, as summarized in Table 1. Each latent variable is a function of other latents that are direct causes (parents in the graph), plus an error term.

$$\eta_i = \beta_1 \eta_{i1} + \beta_2 \eta_{i2} + ... + \beta_P \eta_{iP} + \zeta_i \tag{5}$$

In matrix notation,

$$\eta = \mathbf{B}\eta + \zeta \tag{6}$$

where the diagonal of $\mathbf{B}$ is zero. Similarly,

$$\mathbf{y} = \Lambda\eta + \epsilon \tag{7}$$

[2] For simplicity, here we do not make the distinction between *exogenous* and *endogenous* variables, since in our model this distinction is not necessary

We assume here that all measures are centered at their means, and that all measure errors ϵ_i are uncorrelated with all latents η_i. The implied covariance matrix is then given by

$$\begin{aligned}
\Sigma(\Theta) &= E\left(\Lambda\eta + \epsilon\right)\left(\Lambda\eta + \epsilon\right)^T \\
&= \Lambda E\left(\eta\eta^{\mathbf{T}}\right)\Lambda^{\mathbf{T}} + E\left(\epsilon\epsilon^{\mathbf{T}}\right) \\
&= \Lambda E\left[\left(\mathbf{I}-\mathbf{B}\right)^{-1}\zeta\right]\left[\left(\mathbf{I}-\mathbf{B}\right)^{-1}\zeta\right]^T\Lambda^T + \Omega \\
&= \Lambda\left(\mathbf{I}-\mathbf{B}\right)^{-\mathbf{1}}\Phi\left(\mathbf{I}-\mathbf{B}\right)^{-\mathbf{T}}\Lambda^{\mathbf{T}} + \Omega \qquad (8)
\end{aligned}$$

The key issue in developing methods to learn this kind of graphical model is deciding if the method to be used should try to search for the model as a whole, or first estabilishing the measurement model and then the structural model. We favour the latter approach, since it is less computationally expensive and makes possible the use of constraint-satisfaction Bayesian network learning algorithms which are provably correct in the limit of infinite sample size. Before describing which kinds of constraints can be used for this task, we will define a special kind of measurement/structural model.

Let $\mathbf{L}$ represent the set of latents in our model. Let L_i represent some arbitrary latent, and $\mathbf{M}(L_i)$ the set of all indicators of L_i and $\mathbf{M}$ the set of all indicators in the model. Let $\mathbf{C}$ be another set of latent variables that are common causes of $\mathbf{L} \cup \mathbf{M}$ that are not included in the model. With the definitions of **trek**, **d-separation** and **faithfulness** as given in Appendix A, we have:

Definition 1 *We say that the graph of a measuremente/structural model is* **impure** *if one of the following situations hold:*

- *if there is a trek from some L_i to some measure $M \in \mathbf{M}(L_j)$, $i \neq j$ that does not contain L_j or any other observed variable, we say that M is* **latent-measured impure***;*
- *if there is a trek between two measures $M_1, M_2 \in \mathbf{M}(L_i)$ that does not contain any $L \in \mathbf{L}$, we say that M_1 and M_2 are* **intra-construct impure***;*
- *if there is a trek between a pair of measures $M_1 \in L_i$ and $M_2 \in L_j$, $i \neq j$, which does not contain any $L \in \mathbf{L}$, we say that M_1 and M_2 are* **cross-construct impure***;*
- *if there is some $C \in \mathbf{C}$ such that such that C is a common cause of L_i and some $M \in \mathbf{M}(L_i)$, we say that M is a* **common cause impure**.

All these situations are illustrated in Figure 4.

The following definitions are key concepts used in the algorithms introduced in this report:

Definition 2 *A measurement model is* **almost pure** *if the only kind of impurities among the measured variables are common cause impurities.*

Definition 3 *An* **almost pure latent variable graph** *is a graph faithful to some almost pure measurement model.*

The reason why we want almost pure measurement models is because it highly simplifies the discovery of causal relationships among the latents. Almost pure models have a very important property: for any two variables $M_1 \in \mathbf{M}(L_1), M_2 \in \mathbf{M}(L_2)$ and some set $\mathbf{Q} \subset \mathbf{L}$ such that $\mathbf{Q} \cap \{L_1, L_2\} = \emptyset$, M_1 is d-separated from M_2 given $\mathbf{Q}$ if and only if L_1 is d-separated from L_2 given

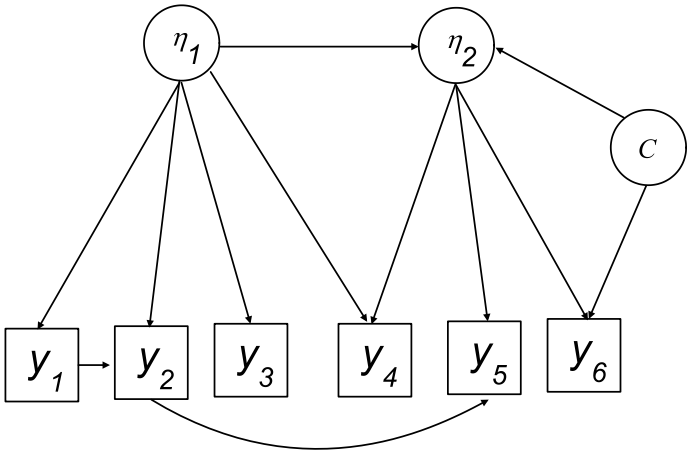

Figure 4: Illustrations of the four different kinds of impurities defined here. The edge from y_1 to y_2 define an intra-construct impure. The removal of either one of these nodes will remove this impurity. The edge from η_1 to y_4 is a latent-measured impure. The edge from y_2 to y_5 defines an cross-construct impurity. Finally, the common cause C between η_2 and y_6 is a common cause impurity, which does not affect the algorithms proposed in this work.

Q. M_1 and M_2 are also d-separated given any set containing L_1 and L_2. For example, Spirtes, Glymour and Scheines [33] describe the `MIMBuild` method. This algorithm uses what are called **tetrad constraints** to be able to detect zero and first order conditional independencies. Tetrad constraints are some constraints among sets of four elements of the correlation matrix that hold only if the true underlying graph is an almost pure latent variable graph.

It is very important to remember that testing conditional independence among the observed variables do not tell us much about the structural model. For almost pure models, they are able to tell us that some latents are uncorrelated and therefore can help in eliminating some of the edges. However, they would not tell us if two latents are d-separated given a third one.

The tetrad constraints, on the other hand, use the parameters of the model, assuming that the model is linear with additive errors. There is a one-to-one correspondence between two latents that are d-separated given a third, and a specific set of tetrad constraints. These constraints are also used for **purifying** a model, i.e., selecting a subset of the indicators such that those selected do not introduce impurities in the final graph [3]. However, this algorithm suffers from three main shortcomings:

- it requires a given measurement model. The model does not need to be complete, though. If a variable is a measure of two latents (i.e., a latent measure impurity), it can appear as an indicator of any of each. However, the number of latents must be specified, and no measures that are not clustered together in the true graph can appear in the same cluster;

- due to the number of tetrad constraints that have to be tested for each test, it is not uncommon that at least one of them may fail. Some heuristics are usually introduced to improve the empirical performance of this method;

[3] After estimating which should be the structural model, one is able to plug back the removed measures depending on the background knowledge available, or estabilishing automatic methods that complement the graph by adding impurities in their correct clusters with correct causal relationships with respect to the other nodes.

- it is able to detect only zero-order and first-order d-separations;

Due to these limitations, we now introduce new algorithms for performing purification of linear models without requiring any background knowledge about the measurement model, followed by learning of structural models given their respective measurement models. The requirements for this approach are:

- a sample of independent identically distributed measurements over the observed variables;
- the faithfulness assumption;
- the measurement assumption, i.e., the true model is an identifiable measurement/structural model, with an identifiable subgraph that is an almost pure latent variable graph with all latents of the complete true graph;
- the parametric form of our model, i.e., the true model is a structural equation model;

Even requiring such assumptions, our method has the following important features:

- no assumptions about the number of latents;
- no assumptions about the latent structure, unlike, for example, factor analysis;
- no extra assumptions about the relations between observed and latent variables, unlike the methods discussed in [33].

The overall method consists in three main components:

- **a clustering component**, by which variables are grouped into clusters. Each clusters should contain measurements of a single latent variable, and there will be as many clusters as latents in the true graph;
- **a purification component**, which will filter those variables in the clustering that are not pure measures. The outputs of the clustering and purification components combined will give an almost pure measurement model;
- **a structure discovery components**, which takes as an input an almost pure measurement model and discovers the causal structure of the unobserved variables;

For practical reasons, the clustering and purification components are interleaved. The complete algorithm is described in Section 4. The structure discovery algorithm is introduced in Section 5.

4 Purification of measurement models

We now extend the procedures introduced by Spirtes *et al.* [33]. We describe a method that returns clusters of measurements assuming that they are causally related accordingly to an almost pure measurement model. Also, this method should select only those measurements that do not introduce impurities, in the sense described in the previous section.

function Purify	
input	an almost pure latent variable graph G; a probability distribution $D_{\mathbf{M}}$;
output	an almost pure latent variable graph

Let $\mathbf{M}$ be the set of observed variables of G
Freely correlate the latents in G, i.e., keep a bidirected edge between each pair of latents
$G' \leftarrow G$
$best\_score \leftarrow ScoreMM(G', D_{\mathbf{M}})$
Discards $\leftarrow \emptyset$
repeat
  $best_m \leftarrow argmin_{m \in \mathbf{M}} ScoreMM(G' - m, D_{\mathbf{M}-m})$
  $current\_score \leftarrow ScoreMM(G' - best_m, D_{\mathbf{M}-best_m})$
  if $current\_score < best\_score$
    $\mathbf{M} \leftarrow \mathbf{M} - best_m$
    $G' \leftarrow G' - best_m$
    $best\_score \leftarrow current\_score$
    Discards $\leftarrow$ **Discards** $\cup\, best_m$
  end if
until $MMScore(G', D_M) = 0$
return **Discards**

Table 2: The Purify algorithm for choosing a subset of a graph G that is faithful to the marginal of some distribution D over a subset of the observed variables of G.

4.1 An algorithm for measurement clustering

Let G be an almost pure latent variable graph with a set of latent factors $\mathbf{L}$ and observed variables $\mathbf{M}$. Let D_M be a probability distribution over $\mathbf{M}$ that is faithful to some almost pure latent variable graph, and Θ_M the vector of parameters that describe D_M.

Define $MMScore(G, D_M)$ as the number of constraints among elements of Θ_M that are not entailed by G, or vice-versa. The algorithm Purify described in Table 2 takes as an input an almost pure latent variable graph G and a probability distribution $D_{\mathbf{M}}$ over the observed variables of G. It then performs a quadratic search to generate candidate graphs and return a set of observed variables that should be removed in order to decrease this score.

The full algorithm should return an almost pure latent variable graph that is faithful to some marginal of a given distribution, where the only given information is the set of observed variables and their joint distribution. This requires the introduction of a step to iteratively create new clusters as they are needed, and a search for correct cluster assignments. This procedure, called here Washdown and which includes Purify as a substep, is described in Table 3.

The main idea of this approach is to iteratively split the observed variables into two groups: those that should be kept in their respective groups, and those that should be removed. That is the role of Purify. After the split is performed, the discarded variables are moved to new candidate clusters, where new clusters are created if necessary. At the end, every variable will be in a single cluster. For example, suppose we have the true model given by Figure 5. Figure 6 presents some steps performed by the Purify/Washdown algorithm.

function	`Washdown`
input	`a set` $\mathbf{M}$ `of random variables; a probability distribution` $D_{\mathbf{M}}$ `over` O`;`
output	`an almost pure latent variable graph;`

$N \leftarrow 1$
`Let` G `be the graph composed by` $\{L_0, \mathbf{M}\}$`, where` L_0 `is a latent parent of every element in` $\mathbf{M}$`, and there are no other parental relations in` G
repeat
  Discards $\leftarrow Purify(G, D_{\mathbf{M}})$
  if Discards $\neq \emptyset$
    if `all measures of a latent were discarded, remove then from` **Discards** `and from` G
    `Add latent` L_N `to` G
    `Add a bidirected edge between each other latent in` G `and` L_N
    for `all` $X \in$ **Discards**
      for `all` $i < N$`, if` L_i `is the parent of` X`, set` L_{i+1} `as the new parent of` X
    end for
    $N \leftarrow N + 1$
  end if
until Discards $= \emptyset$
`return` G

Table 3: The `Washdown` algorithm regroups sets of observed variables into different clusters by using `Purify` to identify which variables should not be in their current groups. The misplaced variables are then slid through all latents that may be required.

4.2 Practical implementation

The score defined in the previous section requires knowledge of the population distribution function, as well as a well-defined procedure to identify all possible constraints that are mismatched between the graph and the underlying probability distribution. For example, Spirtes *et al.* [33] use counts of tetrad constraints that are violated. As suggested before, with finite sample sizes, this requires some heuristics to weight how the contribution of these constraints should be summed up. As a simplified solution, we suggest using a probabilistic score, such as the χ^2 score of the model, determined by the likelihood of the model evaluated at its maximum likelihood estimates and the number of degrees of freedom. Other variations such as BIC could be used in principle. The heuristic is choosing a score that, as the sample size increases, penalizes graphs that introduce undesired constraints (by not fitting the data as well as a model that does not have these constraints) and graphs that do not introduce constraints that should be there (by having fewer degrees of freedom than a model that does not allow this unconstrained relations between parameters of the distribution). Notice that the ideal *MMScore* always achieve zero at each cycle of `Purify`. The suggested stopping criterion for `Purify` is testing if the model has a p-value greater than a given threshold. Unlike the forward or backward search for loglinear models, for instance, this method is not comparing nested models and should be interpreted as a heuristic.

The `Washdown` algorithm was also introduced without referring to any specific distribution. The question of parameter identifiability should also be addressed carefully. Althought there is no known tractable algorithm for deciding if the parameters of an arbitrary structural equation model are all identifiable, it is known that for linear almost pure latent variable graph with at least three indicators per latent is identifiable. Because of that, we add the following assumption:

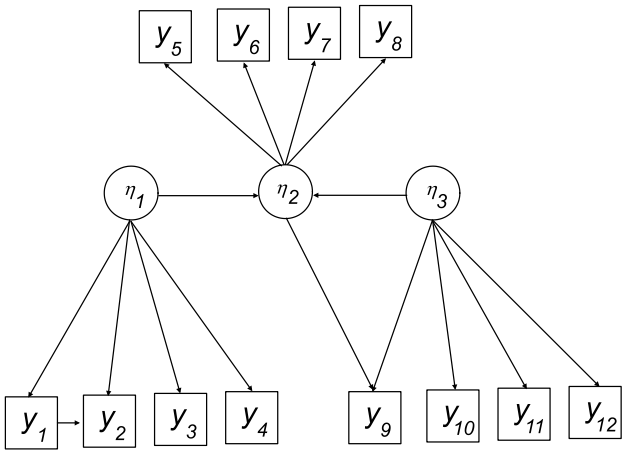

Figure 5: An example of a measurement model with impurities.

Assumption 1 *The data that is used as an input for* `Washdown` *was generated by a measurement/structural model that contains a linear almost pure latent variable subgraph with at least three indicators per latent.*

When implementing `Washdown` with finite samples, one has to take in account that during the process clusters of fewer than three variables may be formed. A straightforward solution is simply eliminating those clusters along with their measures from the next steps of the algorithm. This also means that by the end, it is possible that an empty graph may be generated. The modifications in `Purify` should be:

- if the node that must be discarded is part of a cluster of three variables, and there are other clusters in the graph, remove the node, adding it to the list of discarded nodes. Remove the whole cluster, but do **not** add those variables to the list of discards;
- if the node that must be discarded is part of a cluster of three variables, and there is no other cluster in the graph, add it to the discards list and halt;

`Washdown` should also be adapted:

- if, after moving nodes to the next cluster, there are clusters with less than three variables, eliminate them definitely of the graph;

Notice that modification should substitute the step in the original algorithm where a cluster whose variables all were eliminated implies that all of them should be removed from the final model. In the original algorithm, it works as a way to eliminate the impurities. In the practical implementation, it is also a way to not get too far astray from the right track if incorrect statistical decisions were taken before and the model gets too fragmented in small (and incorrect) clusters. Notice that if one cluster is eliminated due to the new rule of `Purify` discussed above, it will be eliminated definitely in the next step if no variable from the immediate cluster is eliminated. Althought is may seem we are making a premature elimination, this heuristic works very well in practice according to the experiments performed in Section 6.

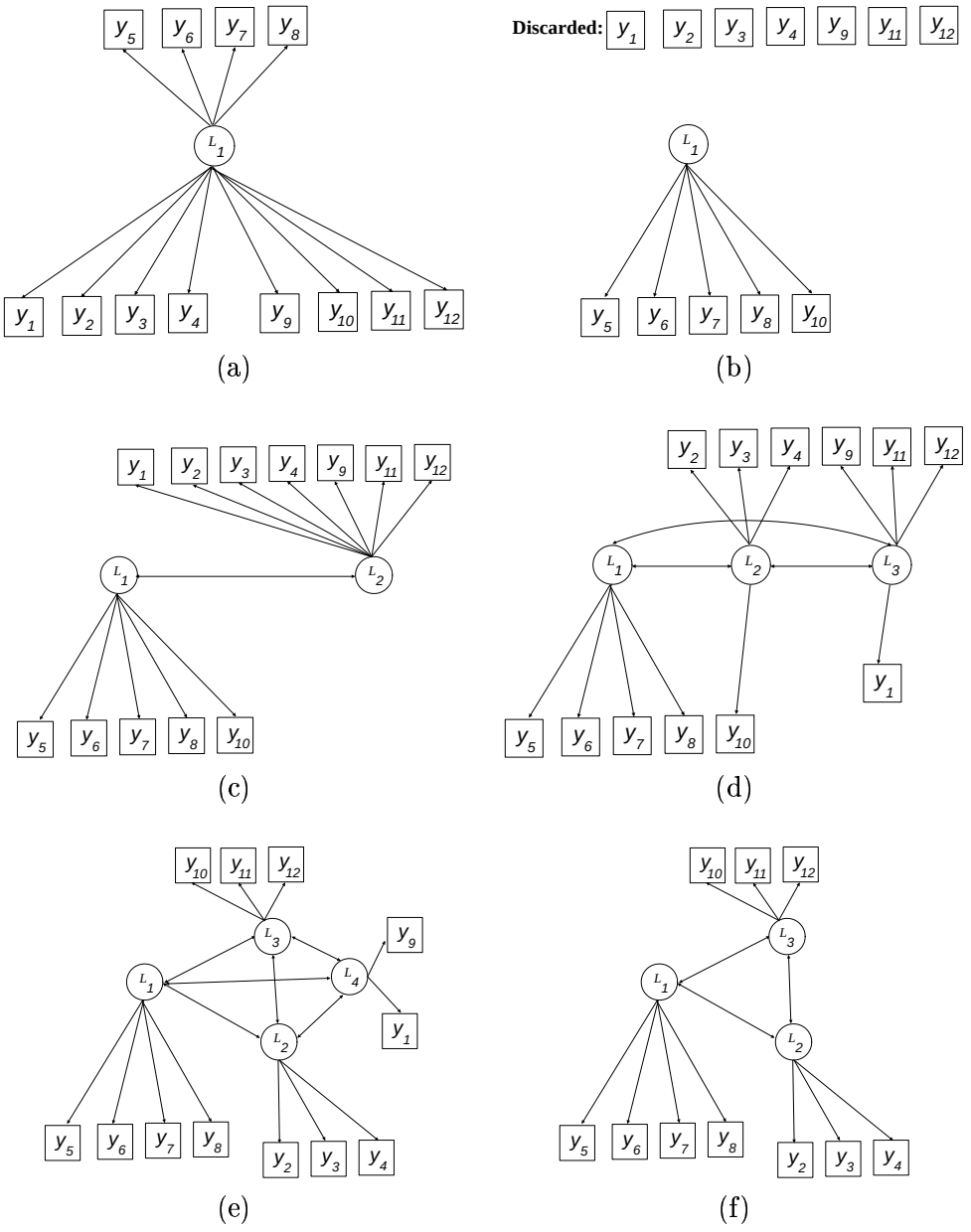

Figure 6: In (a), we have the beginning of the first call of `Purify`, where we start with only one latent and all variables are clustered together. In (b), the end of the first call to `Purify`. It is expected that variables from a different cluster will initially appear in the wrong cluster, even if the true distribution is known. This happens in the linear case because there are no forbidden or necessary constraints in the parameters for representing this marginal. In (c), the search state at the end of the first cycle of `Washdown`. In (d), the search state at the end of the second cycle of `Washdown`. In (e), the end of the third cycle of `Washdown`. In (f), the final model after a whole clustering (the last one) has been removed.

When fitting the model in order to get maximum likelihood estimates, one can use gradient-based method to perform the numerical optimization. Given the implied covariance matrix of

Equation 8, it is straightforward to compute the gradient of the likelihood function [4] with respect to the parameters $\Lambda, \mathbf{B}, \Phi$ and Ω. Details are provided in Appendix B. Since the algorithm has a $O(qd^2)$ cost, where q is the number of latents and d is the number of measures, speed-up heuristics are important. One particularly valuable heuristic for choosing the best measure to drop at each iteration of `Purify` is as follows:

- before dropping any measure, perform maximum likelihood estimation of the model. Try multiple starting points for a better fit;
- at every step where one of the measures is dropped, start the respective parameters of the graph according to the estimates found in the complete graph. Do not run with multiple starting points;

The convergence of the fitting procedure for each tentative graph is much faster than what would happen if we started from a random point. Bad local maxima seems to be less likely also. This heuristic performed well in practice in the experiments described in Section 6.

5 Discovering the causal structure of latent variables

The second component of our method for learning measurement/structural models is a procedure to identify the structural model given almost pure measures. A constraint-satisfaction approach will be described in this section.

The idea behind the generalized `MIMBuild` algorithm consists in using a hypothesis test to decide when two latent variables are d-separated given a set of other latents. Unlike in the approach based in tetrad constraints, described in Section 3, this set can contain more than one latent. When variables are observed, there are well-known tests of independencies for particular distributions such as Gaussians and multinomials. But even for the Gaussian case in which we are interested, testing independence between unobserved variables is non-trivial. The following theorem is due to Spirtes [32]:

Theorem 1 *Let G be an almost pure linear measurement/structural model with Gaussian errors. In G, L_1 and L_2 are two latents and $\mathbf{Q}$ a set of latents such that $\mathbf{Q} \cap \{L_1, L_2\} = \emptyset$. Define $\mathbf{Z}$ as a set containing one measure of L_1, one measure of L_2 and n measures from $\mathbf{Q}$. If L_1 is not d-separated from L_2 given $\mathbf{Q}$, then the Lesbesgue measure of the set of parameters of G for which the rank of the correlation matrix of $\mathbf{Z}$ is less than or equal to n is zero.*

That means it is highly unlikely that the correlation matrix of $\mathbf{Z}$ has a rank greater than n if L_1 is independent of L_2 given $\mathbf{Q}$. With finite samples, it is necessary to use a statistical test in order to detect the rank of this matrix. One way of performing this is throught the construction of factor models.

Definition 4 *A* **n factor model** *of a set of variables $\mathbf{V}$ is a directed acyclic graph with n latent variables $\mathbf{Q}$ in which each member of $\mathbf{V}$ is a child of each member of $\mathbf{Q}$, and each pair of variables in $\mathbf{Q}$ is connected by an edge.*

[4] We are not taking in account problems that may happen due to the shape of the likelihood function, with unexpected behavior of optimization algorithms at the edge of the parameter space [13].

The rank of a correlation matrix is less than or equal to n if and only if there is a n factor model of the variables in the correlation matrix. To test the existence of this factor model, one alternative is creating a graph that includes a complete acyclic graph among nodes in $\mathbf{Q}$ (the direction of the edges is not important, as long as the graph is acyclic), L_1 is a child of every element in $\mathbf{Q}$, L_2 is a child of every element in $\mathbf{Q}$, and the respective measures of each latent are included in their respective clusters. Figure 7 illustrates this construct.

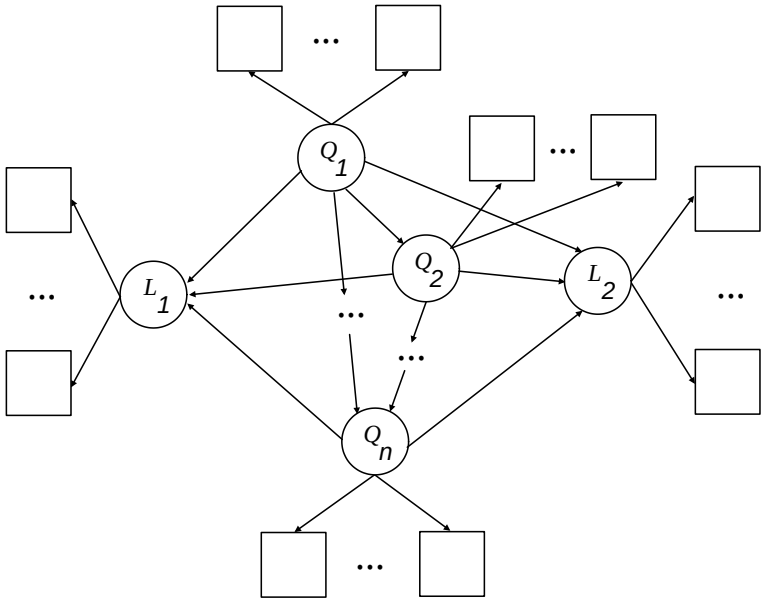

Figure 7: A n factor model over $\mathbf{Q}$ and $\mathbf{L}$.

To test for the existence for an edge between L_1 and L_2, one can fit the data to this factor model and accept the d-separation if its p-value is greater than some threshold. Another alternative is making a likelihood ratio test by comparing two nested models: one which includes an edge from L_1 to L_2 (or vice-versa), and another without it. We are testing if the label of this edge should be zero. The difference of their χ^2 scores is a χ^2 variable with one degree of freedom. In our empirical evaluation discussed in the next section, we favor the former approach, because it is more computationally efficient and performed very close to the latter one in preliminary experiments. It is also less susceptible to bad local maxima problem, since it is just comparing one model with a threshold, instead of the difference of two models (where bad local maxima in one of them may prevent the difference from being insignificant). By Theorem 1, the existence of a n factor model implies d-separation, and it is relatively straigthforward to see that d-separation of two latents implies that all independencies entailed by the respective n factor model exist in the true underlying distribution.

Now that we have a test for d-separation among latents, we can plug it into some constraint-satisfaction causal discovery algorithm, such as the `PC` and `FCI` algorithms. Table 4 describes the complete `MIMBuild` algorithm using as a basis the `PC` algorithm for discovering the structure of the unobserved variables.

The correctness of `PC` algorithm is proved in [33]. It assumes there are no extra latents among the already predefined variables. However, one could trivially adapt `MIMBuild` to make use of the `FCI` algorithm in order to detect latents among the latents explicitly included in the measurement model In both algorithms and their variations, the output is not a directed acyclic graph, but a

function	MIMBuild
input	an almost pure measurement model graph G; a sample covariance matrix $\mathbf{S}$ over the observed variables in G;
output	an almost pure latent variable graph with the same measurement model of G;

Form a complete undirect graph among the latents in G
for all pair of latents $\{X, Y\}$ in G, set **SepSet**$(X, Y) \leftarrow 0$ and **SepSet**$(Y, X) \leftarrow 0$
$n \leftarrow 0$
repeat
  repeat
    select an ordered pair of latents L_1 and L_2 that are adjacent in G such that **Adjacencies**$(G, L_1)\backslash L_2$ has cardinality greater than or equal to n, and a subset $\mathbf{Q}$ from **Adjacencies**$(G, L_1)\backslash L_2$ of cardinality n. If the factor model of $\{L_1, L_2, \mathbf{Q}\}$ passes a statistical test, remove the edge between L_1 and L_2 and record $\mathbf{Q}$ in **SepSet**(L_1, L_2) and **SepSet**(L_2, L_1).
  until all ordered pairs of adjacent variables (L_1, L_2) such that **Adjacencies**$(G, L_1)\backslash L_2$ has cardinality greater than or equal to n and all subsets $\mathbf{Q}$ of **Adjacencies**$(G, L_1)\backslash L_2$ of cardinality n have been tested for d-separation.
  $n \leftarrow n + 1$
until for each ordered pair of adjacent variables (L_1, L_2), **Adjacencies**$(G, L_1)\backslash L_2$ is of cardinality less than n
for each triple of latents (L_1, L_2, L_3) such that $\{L_1, L_2\}$ are adjacent in G, $\{L_2, L_3\}$ are adjacent in G and $\{L_1, L_3\}$ are *not* adjacent in G, orient $L_1 - L_2 - L3$ as $L_1 \rightarrow L_2 \leftarrow L_3$ if and only if L_2 is not in **SepSet**(L_1, L_3).
repeat
  if $L_1 \rightarrow L_2$, L_2 and L_3 are adjacent, L_1 and L_3 are not adjacent, and there is no arrowhead at L_2, then orient $L_2 - L_3$ as $L_2 \rightarrow L_3$.
  if there is a directed path from L_1 to L_2, and an edge between L_1 and L_2, then orient $L_1 - L_2$ as $L_1 \rightarrow L_2$.
  if there are the edges $L_X \rightarrow L_1$, $L_1 \leftarrow L_Y$, $L_2 - L_X$, $L_2 - L_1$ and $L_2 - L_Y$, then orient $L_2 - L_1$ as $L_2 \rightarrow L_1$.
until no more edges can be oriented.
return G

Table 4: The MIMBuild algorithm as a variation of the PC algorithm to causal discovery among unbserved variables.

Performance of Washdown			
	3 latents	**4 latents**	**5 latents**
500 cases			
missing latents	10.0 ± 19.0	17.5 ± 16.4	18.0 ± 17.0
missing indicators	10.3 ± 19.5	11.5 ± 12.7	11.0 ± 9.7
misplaced indicators	10.3 ± 17.6	15.1 ± 15.2	14.2 ± 14.5
1000 cases			
missing latents	11.2 ± 16.3	7.5 ± 11.8	18.0 ± 18.2
missing indicators	5.0 ± 10.0	7.8 ± 8.5	14.6 ± 20.5
misplaced indicators	9.0 ± 14.0	5.2 ± 8.5	9.5 ± 10.3
5000 cases			
missing latents	3.3 ± 10.2	7.5 ± 14.3	4.0 ± 8.2
missing indicators	5.7 ± 9.7	7.5 ± 15.8	7.4 ± 9.0
misplaced indicators	1.7 ± 5.6	4.8 ± 14.6	1.8 ± 5.1

Table 5: Results obtained for `Washdown`, in percentage, when true models are pure. The average degree in each case is 2. Numbers indicate average and standard deviation over 20 trials.

graph where some of the edges are not simple arrows. For example, for the `PC` algorithm, some of the edges remain unoriented, indicating that conditional independence statements based on nonexperimental data are not enough to decide which should be the causal direction. The resulting graph is called a **pattern**.

6 Evaluation

We now perform empirical evaluation of the procedures introduced in the previous sections. The data sets we used in this preliminary study are synthetic data sets generated using the Tetrad IV program [5]. Let the average degree of a node be the number of neighbors a node has in the graph. Given a number of latents, a number of indicators per latent, an average degree for each latent node and a sample size, each synthetic model was generated by first iterating through all pairs of latent nodes and adding an directed edge [6] if a random number generated uniformly in the interval $[0, 1]$ was less than $\frac{average\ degree}{number\ of\ latents - 1}$.

Values for the cofficients are then uniformly sampled from the interval $[-1.5, -0.5] \cup [0.5, 1.5]$. Error variances are sampled from the interval $[0.01, 1]$. The motivation for choosing such intervals is generating artificial models where the causal effects are not too big or too small. After the full parameterized model is set, independent samples are pseudoramdonly generated. The pseudorandom number generator used in the following experiments was the one used in the Java 1.2 virtual machine. We run the `PC` algorithm in each generated graph using the known d-separation relations of the latents in order to obtain patterns of equivalence classes that are then compared to the graphs estimated by `MIMBuild`. To perform maximum likelihood estimation, we used the library of numerical optimization methods available at `http://www.stat.uni-muenchen.de/ strimmer/pal-project/`. In order to guarantee parameter identifiability, one indicator of each latent has its respective latent edge fixed to 1.

[5] Available at `http://www.phil.cmu.edu/tetrad`.

[6] We define an arbitrary order among the latents, such that an edge is directed only from the node in the lowest position in this order to the node if the highest position. This avoids circularity.

The first batch of experiments evaluates `Washdown` as a method to find correct clusters when the true model is already a pure measurement model. We used three diferent sample sizes (500, 1000, 5000 cases), five indicators per latent and an average degree of 2. Three different results were evaluated for each case:

- **average proportion of missing latents**, the proportion of latents that do not appear in the estimated measurement model but are present in the true graphs;
- **average proportion of missing measurements**, as evaluated with respect to all indicators in the true graph irrespective of the cluster they belong to;
- **average proportion of misplaced measurements**, the proportion of measures that end up in the wrong cluster;

To perform the comparison, we should indicate which latent found in the estimation corresponds to which of the original latents. The straightforward way is making the match according to the original cluster of the majority of the indicators in a given estimated cluster: for example, suppose we have an estimated latent L_E. If, for instance, 70% of the measures in L_E are measures of the true latent L_2, we label L_E as L_2 in the estimated graph and calculate the statistics of comparison as described above. In case of a tie (i.e., when there are two or classes with the same proportion and they represent the majority), one solution is considering each possible instantiation at a time (not considering invalid instances, such as two estimated latents labeled as the same true latent) and average over those numbers. Results are summarized in Table 5.

With `MIMBuild`, we performed a similar series of experiments using randomly generated networks with up to 7 latents, 5 indicators each. The average degree for latents was 2 for networks up to 5 latents. For networks with 6 and 7 latents, the average degre used was 3. We are using the `PC` algorithm as the underlying causal discovery algorithm. Comparisons are made with respect to the pattern among latents that are generated by the `PC` algorithm when information about d-separation statements among the latents are given [7]. The criteria for evaluating the outcome of `MIMBuild` are as follows:

- **average proportion of edge comissions**, which is the number of edges that are in the estimated graph but not in the true graph, divided by the possible number of edges that could be added with error. This latter number is given by $q(q-1)/2 - |E|$, where q is the number of latents and $|E|$ is the number of edges among the latents in the true graph;
- **average proportion of edge omissions**, the ratio of the number of edges in the true graph that are not in the estimated graph, divided by $|E|$;
- **average proportion of orientation comissions**, the proportion of edges that are oriented in the estimated graph but which are not oriented in the pattern, with respect to the number of unoriented edges in the pattern;
- **average proportion of orientation omissions**, which is the number of edges that are unoriented in the estimated graph but are oriented in the true pattern, divided by the number of oriented edges in the pattern;

[7] We used the `PC` algorithm instead of `FCI` because it generates more simple outputs that can be more easily compared. For real-world applications, `FCI` should be the choice unless there is a strong believe that there are not other latents among the latents of the structural model.

Performance of `MIMBuild`					
	3 latents	**4 latents**	**5 latents**	**6 latents**	**7 latents**
100 cases					
edge comission	0.0 ± 0.0	9.1 ± 25.1	–	–	–
edge omission	26.7 ± 23.2	40.1 ± 19.7	–	–	–
orientation comission	30.0 ± 47.0	29.5 ± 33.4	–	–	–
orientation omission	0.0 ± 0.0	6.25 ± 6.0	–	–	–
500 cases					
edge comission	0.0 ± 0	3.8 ± 12.2	7.0 ± 16.0	–	–
edge omission	23.4 ± 21.9	24.6 ± 25.1	32.9 ± 22.3	–	–
orientation comission	10.0 ± 30.0	17.5 ± 33.5	26.9 ± 31.7	–	–
orientation omission	0.0 ± 0.0	9.2 ± 22.6	15.2 ± 21.4	–	–
1000 cases					
edge comission	0.0 ± 0.0	8.8 ± 24.4	6.2 ± 13.8	6.5 ± 13.3	9.6 ± 9.5
edge omission	15.0 ± 20.1	25.7 ± 24.2	18.6 ± 16.0	32.2 ± 14.3	34.7 ± 14.2
orientation comission	15.0 ± 36.6	24.6 ± 31.6	24.9 ± 32.2	37.1 ± 24.9	33.9 ± 29.2
orientation omission	0.0 ± 0.0	7.9 ± 19.5	4.6 ± 11.3	10.0 ± 16.6	12.7 ± 11.9
5000 cases					
edge comission	–	–	4.7 ± 10.6	7.6 ± 12.6	7.0 ± 8.4
edge omission	–	–	10.3 ± 15.4	15.9 ± 14.8	19.3 ± 14.6
orientation comission	–	–	19.2 ± 28.6	27.8 ± 21.4	27.1 ± 21.9
orientation omission	–	–	11.3 ± 24.5	12.4 ± 12.8	15.1 ± 16.1
10000 cases					
edge comission	–	–	–	5.8 ± 11.6	10.8 ± 12.5
edge omission	–	–	–	17.4 ± 17.6	17.0 ± 13.1
orientation comission	–	–	–	36.4 ± 23.2	25.6 ± 19.8
orientation omission	–	–	–	8.4 ± 11.1	15.1 ± 15.8

Table 6: Results obtained for `MIMBuild` for different numbers of latents, in percentage. Numbers indicate average and standard deviation over 20 trials.

Performance of hill-climbing heuristic	
	5 latents
500 cases	
edge comission	23.6 ± 29.0
edge omission	22.4 ± 15.0
orientation comission	32.5 ± 40.8
orientation omission	19.5 ± 34.6
1000 cases	
edge comission	30.1 ± 36.1
edge omission	8.7 ± 14.7
orientation comission	16.0 ± 24.8
orientation omission	20.7 ± 28.4
5000 cases	
edge comission	24.8 ± 26.4
edge omission	7.5 ± 11.2
orientation comission	7.3 ± 16.1
orientation omission	18.6 ± 26.7

Table 7: Results obtained for discovery of structural model by a hill-climbing score-based heuristic.

Results are presented in Table 6, with averages and standard deviations taken over 20 trials for each experiment. A p-value of 0.05 was used as the threshold to decide when an independence test should be accepted. For comparison, we also performed some experiments with a variation of heuristic Bayesian learning, a hill-climbing algorithm that starts with no edges among the latents. At each stage, we test the model that mostly decreases the χ^2 score of the current one. The candidate models are generated from the current one by adding one directed edge between a given pair of nodes that are not directly connected. The stop criterion is the significance of the model: as soon as the model reaches a p-value greater than 0.05, we stop the search and output the model. Table 7 shows the results obtained.

The hill-climbing heuristic seems to converge faster to a smaller edge omission error than `MIMBuild`. However, in practice its computational cost is much higher than `MIMBuild`, and edge comission error is significantly higher. Other variations of hill-climbing score-based algorithms can be used (using different score functions, search operators such as in genetic algorithms, etc.) Future work with extra experimental results should follow.

We also performed an evaluation of the model depicted in Figure 8. In this situation, we have some impurities. The results obtained were:

- **average proportion of missing latents:** $22.0\% \pm 23.3\%$;
- **average proportion of impurities:** $6.6\% \pm 13.7\%$;
- **average proportion of missing measurements:** $28.9\% \pm 22.3\%$;
- **average proportion of misplaced measurements:** $7.5\% \pm 10.6\%$;

where the average proportion of impurities is the proportion of impure measurements (up to three in our example) that appear in the final estimated model, averaged over 20 trials. It may be important

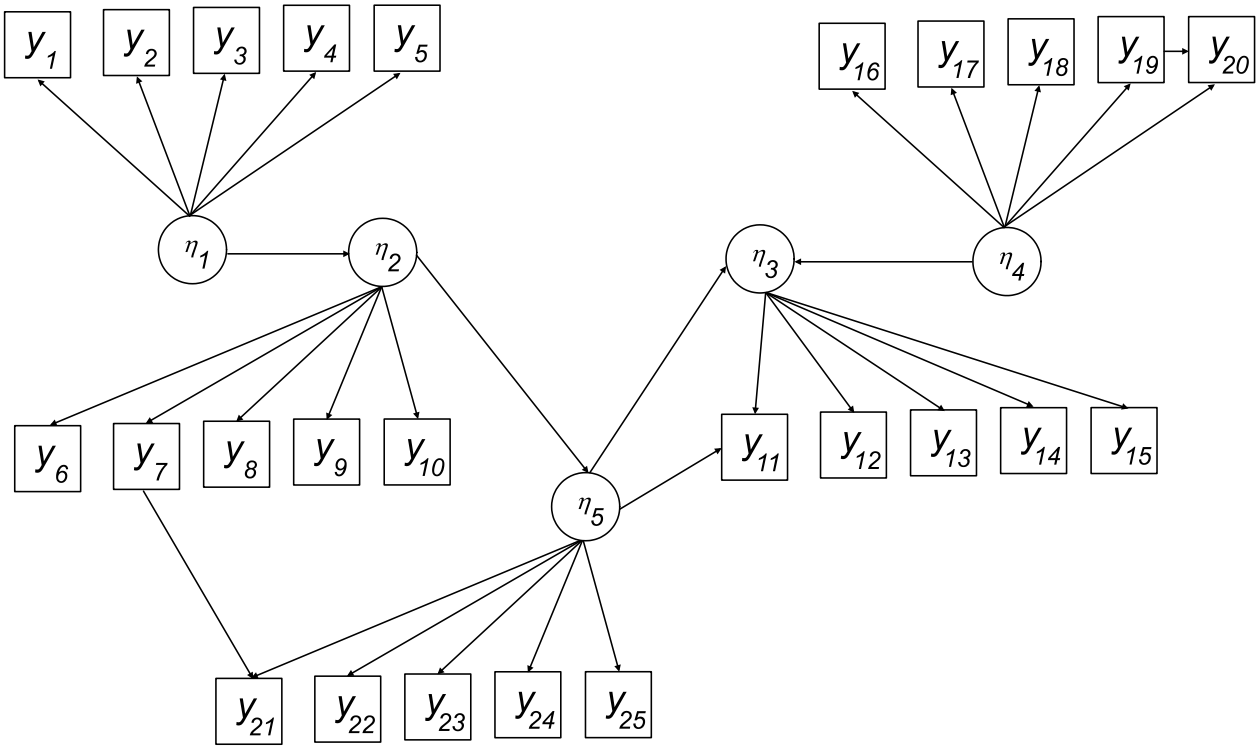

Figure 8: An example of measurement/structural model with three impurities defined by the edges $y_7 \to y_{21}$, $\eta_5 \to y_{11}$ and $y_{19} \to y_{20}$.

to mention that in this case at every cycle of `Purify` (after the node that best increases the score is selected) we fit the parameters with multiple starting points (5, in this case). In the experiments we performed with pure models only, we try multiple starting point only once, at the very first iteration of `Purify`. Trying multiple points at every major cycle of this algorithm considerably improved the results [8] while providing a good trade-off that does not increase the computational cost too much.

The problem of unveleiling the structural model seems to require large sample sizes. Notice that there is a trade-off between the number of comission and omission errors, and so it is not strange that sometimes one of these measure does not change much with increased sample size. Also, using more indicators per latent should make the estimation more reliable [33]. Although the types of problems tackled by `Washdown/MIMBuild` are of interest for social sciences also, the number of required samples may limit its usability in those domains, since sample sizes of less than one thousand cases are common. However, it is valuable to remember that one way to increase the feasibility of data-intensive models and algorithms is provide prior knowledge in the form of partial measurement and structural models that will require less statistical decisions by this algorithm.

7 Conclusion

It should be clear that the role of latent variables in probabilistic and causal modeling is larger than it may seem at a first glance. The methods here described provide tools for building mod-

[8] In a previous experiment with 20,000 samples and no multiple starting points, we got 35.0% of missing latents.

els where latent variables are naturally integrated. The advantages of our method for inducing measurement/structural models out of observational data can be summarized as follows:

- besides linearity and faithfulness, no other major assumptions are made about the structure of the model, i.e., one does not assume independence of the latents, or time order, or any other restrictive requirements about the structure of the unobsorved variables;
- measurements/structural models have a much more meaningful interpretation than FA & similar models;
- it naturally allows the discovery of hierarchical models by means of applying the `FCI` algorithm along with `MIMBuild`, which can identify situations where there are latents among the latents of each cluster;
- no true competing approaches exist;
- it is relatively computationally efficient when compared with pure score-based search;

On the other side, one should use these tools with care, since sampling variability will introduce errors in the generated estimations with almost certainty. One reason to explain the slow increase in the accuracy of `MIMBuild` as a function of the sample size is due to the multiplicity effect that happens when multiple hypothesis tests are performed in a sequence [18]. No joint distributions for these tests are known, and common correction methods for multiple hypothesis tests, such as the Bonferroni correction, are too conservative. In the future, we may study how procedures such as controlling for false discovery rates [3] could be modified for constraint-satisfaction algorithms, or how resampling based techniques could estimate adjusted p-values that take in account this scenario [34]. The approaches here introduced are still valuable at least as exploratory data analysis tools.

Heuristic score-based methods are well know for being robust when the task is finding graphical models which fit well the data [15]. However, this does not necessarily mean that the resulting structure will reflect the independencies constraints of the true model due to the greedy nature of those approximations. Those methods do not provide any guarantees or bounds of how close it will get of finding those independencies that allow us to make causal inference based in simple axioms [33], [27]. Improving robustness is still a major issue for constraint-satisfaction approaches, and ideas inspired by score-based search may be explored in order to fulfill those necessities.

For future work, the following extensions are also planned:

- experiments with empirical (i.e., non artificial) data sets, followed by interpretation and evaluation by domain experts;
- extended comparison with similar approaches. Even though there are no algorithms especially designed for the discovery of measurement/structural models, some related approaches could be compared as a way to give new insights about the power of `Washdown/Purify/MIMBuild`. For example, one could use the model selection approach of [23] to choose the number of latents and some standard variable-clustering method, followed by iterative purification methods in order to rebuild the measurement model. Approaches such as the ones described in [10] could be adapted for the structural model discovery task, providing more state-of-the-art score-based methods that could be compared with `MIMBuild`;
- evaluation by prediction of causal effects. In this framework, graphs discovered by our algorithms could be evaluated not by comparing the differences in omission/comission of

edges/directions, but in how to predict the effect of manipulations. This is similar to criteria such as reporting least-squares errors, but with manipulated graphs instead of regular prediction models (see [33] for more information in prediction in manipulated models);

- treatment of other families of probability models such as multinomial;

Ultimately, latent variable modeling may be essential for more ambitious tasks. In their textbook of artificial intelligence [31], Russell and Norvig make the following comment (p. 641):

> Some of the deepest revolutions in science come from the invention of new predicates and functions - for example, Galileo's invention of acceleration or Joule's invention of thermal energy. Once these terms are available, the discovery of new laws becomes (relatively) easy. The difficult part lies in realizing that some new entity, with a specific relationship to existing entities, will allow an entire body of observations to be explained with a much simpler and more elegant theory than previously existed.

Models of artificial intelligence and automated knowledge discovery need to scale-up for larger domains. Encoding causal knowledge seems to be an important requirement to perform this next step [9] and the role of latent variable models should only increase as we try to get closer and closer to this goal.

Acknowledgements

This work was a result of discussions with Peter Spirtes, Clark Glymour and Richard Scheines. I would like also to thank Joe Ramsey for all the support in understanding and modying the TETRAD IV software in order to perform my experiments.

References

[1] Hagai Attias. Independent factor analysis. *Neural Computation*, 11, 1999.

[2] Alexander Basilevsky. *Statistical Factor Analysis and Related Methods.* John Wiley & Sons, 1994.

[3] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57:289–300, 1995.

[4] John Binder, Daphne Koller, Stuart Russell, and Kenji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.

[5] Christopher Bishop. Latent variable models. In Michael Jordan, editor, *Learning in Graphical Models*, pages 371–403. MIT Press, 1998.

[6] Kennet Bollen. *Structural Equations with Latent Variables.* John Wiley & Sons, 1989.

[7] Greg Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[9] For instance, see the example on page 442 of Russell and Norvig as an illustration of how causal representation may highly simplify a model, and also allows inference about the effects of manipulation without requiring extra rules

[8] Sašo Džeroski and Nada Lavrač. An introduction to inductive logic programming. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 48–73. Springer, 2001.

[9] Gal Elidan and Nir Friedman. Learning the dimensionality of hidden variables. In *Proceedings of the Seventeenth Conference in Uncertainty in Artificial Intelligence (UAI-2001)*, pages 144–151, San Francisco, CA, 2001.

[10] Gal Elidan, Noam Lotner, Nir Friedman, and Daphne Koller. Discovering hidden variables: a structure-based approach. In *Neural Information Processing Systems 13*, 2000.

[11] Nir Friedman. The Bayesian structural EM algorithm. In *Proceedings of the Forteenth Conference on Uncertainty in Artificial Intelligence*, pages 129–138, 1998.

[12] Nir Friedman and Daphne Koller. Being Bayesian about network structure: a Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, page to appear, 2002.

[13] Dan Geiger, David Heckerman, and Chris Meek. Asymptotic model selection for directed networks with hidden variables. Technical Report MSR-TR-96-07, Microsoft Research, 1996.

[14] Zoubin Ghahramani and Geoffrey Hinton. The EM algorithm for mixtures of factor analyzers. Technical report, Department of Computer Science, University of Toronto, 1996.

[15] David Heckerman. A Tutorial on Learning Bayesian Networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

[16] Aapo Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[17] Kamel Jedidi and Asim Ansari. Bayesian structural equation models for multilevel data. In George Marcoulides and Randall Schumacker, editors, *New Developments and Techniques in Structural Equation Modeling*, pages 129–158. Lawrence Erlbaum Associates, 2001.

[18] David Jensen and Paul Cohen. Multiple comparison in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.

[19] Richard Johnson and Dean Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 1998.

[20] M. C. Jones and Robin Sibson. What is projection pursuit? *Journal of the Royal Statistical Society, series A*, 150(1):1–36, 1987.

[21] Karl Jöreskog. A general method for estimating a linear structural equation system. In Arthur Goldberger and Otis Duncan, editors, *Structural Equation Models in the Social Sciences*, pages 85-112. Seminar Press, 1973.

[22] Robert Klee. *Introduction to the Philosophy of Science: Cutting Nature at Its Seams*. Oxford University Press, 1997.

[23] Thomas Minka. Automatic choice of dimensionality for PCA. In *Neural Information Processing Systems 13*, pages 598–604, 2000.

[24] Kevin Murphy. A brief introduction to graphical models and Bayesian networks. URL: `http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html`, 2001.

[25] Luis Ortiz and Lesli Kaelbling. Notes on methods based on maximum-likelihood estimation for learning the parameters of the mixture of Gaussian model. Technical Report CS-99-03, Computer Science Department, Brown University, 1999.

[26] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[27] Judea Pearl. *Causality: Models, Reasoning and Inference.* Cambridge University Press, 2000.

[28] Judea Pearl. Parameter identification: A new perspective. Technical report, UCLA Cognitive Systems Laboratory, 2001.

[29] Sam Roweis. Notes on matrix identities, June 1999. `http://www.cs.toronto.edu/~roweis/notes.html`.

[30] Sam Roweis and Zoubin Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345, 1999.

[31] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[32] Peter Spirtes. Discovering causal relations among latents variables in directed acyclic graphical models. Technical report, Department of Philosophy, Carnegie Mellon University, 1996.

[33] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction and Search.* MIT Press, 2 edition, 2000.

[34] Peter Westfall. *Resampling-based Multiple Testing: Examples and Methods for P-value Adjust.* Wiley Interscience, 1992.

Appendix

A Graphical models and structural equation models

In this section we will provide only a very brief overview of some of the most common graphical models. For more details, the reader is invited to look elsewhere [26], [24]. We say that a **graphical model** is a representation of probabilistic and causal relations using graphs as the language of choice. **Nodes** represent random variables, while the **edges** that relate these nodes indicate some kind of probabilistic and/or causal relation. The absence of a edge between two nodes indicate some kind of independence, and these independencies are a key feature of graphical models that allows sparse encoding of probability distributions, often requiring many less parameters than those that should be specified in a joint probability density over the same variables. Unfortunately, not all probability distributions can be represented in a given graphical model, but they are usually flexible enough to accomodate a large variety of useful distributions. Also, constrains imposed by these models are useful for testing models and defining conditions where causality can be inferred.

Bayesian networks are directed, usually acyclic, graphs where each node has an associated function that defines its conditional probability distribution/density given its parents. This is a consequence of the **Markov condition** assumed in the definition of a Bayesian network: a node is probabilistically independent of its nonparental non-descendent nodes given its parents. There is a well-defined criterion for deciding when a pair of nodes is probabilistically independent given

another (possibly empty) set of nodes: the so-called **d-separation** rule. For example, let a **trek** be an undirected path between two nodes in a graph such that there is no node C in this path where the two edges adjacents to C point to it. If there is a trek between two nodes, we say they are not d-separated given an empty set of nodes.

The exact direction of the arrows is not important in many situations when the goal is only encoding a probability distribution, but Bayesian networks may also have causal interpretations where the directions do matter in the general case. A set of simple axioms entails how to determine causal effects (or detect when they cannot be determined). See [33] and [27] for details. The crucial difference between the causal and probabilistic-only interpretations is that a causal network allows one to compute the effect of an external intervention in the system, instead of allowing only the computation of conditional probabilities.

One important assumption that is made when one is willing to adopt a causality calculus is the **faithfulness** assumption: it basically says that two nodes are d-separated in the graph if and only if they are probabilistically independent (conditioned in the same set of variables). While it is possible to prove that unfaithful distributions have zero Lesbegue measure, making them unlikely to happen, in practice unfaithful empirical distributions can appear due to sampling variability and small causal effects in the underlying true graph. In this work, we are always assuming faithfulness when speaking about causal networks.

One important specific type of Bayesian network is the **Structural Equation Model** (SEM). SEMs have been widespread in econometrics and social sciences literature long before Bayesian networks became the representation of choice for uncertainty in artificial intelligence. See [6] as a introduction to SEMs. In this kind of graphical model, the relationship between one node and its parents is defined as a linear function, usually with an additive Gaussian distributed error term.

The following example provides an illustration of the inner properties of SEMs. Suppose we have a multivariate Gaussian distribution over $\mathbf{Y} = \{y_1, y_2, y_3\}$. Suppose also that we know that these variables are correlated due to a single common cause η that is unmeasured [10]:

$$\begin{aligned} y_1 &= \lambda_1 \eta + \epsilon_1 \\ y_2 &= \lambda_2 \eta + \epsilon_2 \\ y_3 &= \lambda_3 \eta + \epsilon_3 \end{aligned}$$

A graphical model that represents those causal and probabilistic dependencies is shown in Figure 9. In principle, the parameters could be determined by solving a system of linear equations: the graph entails covariances among the observed that are a function of the parameters. With that, we can find the **implied covariance matrix** of the model. Assuming that the variables in our example are centered in their mean, we have:

$$\begin{aligned} Cov(y_1, y_2) &= E(\lambda_1 \eta + \epsilon_1)(\lambda_2 \eta + \epsilon_2) \\ &= \lambda_1 \lambda_2 E\eta\eta + E\epsilon_1\epsilon_2 \\ &= \lambda_1 \lambda_2 \end{aligned}$$

since we assumed that $Var(\eta) = 1$ and, by faithfulness, one can easily verify that ϵ_1 is d-separated of ϵ_2, which implies that $Cov(\epsilon_1, \epsilon_2) = 0$. Analogously, denoting by ω_i the variance of ϵ_i,

$$\begin{aligned} Var(y_1) &= E(\lambda_1 \eta + \epsilon_1)(\lambda_1 \eta + \epsilon_1) \\ &= \lambda_1^2 + \omega_1 \end{aligned}$$

[10] Unfortunately, the standard notation in SEMs is using the signal of equality, =, to relate child and parents, hindering the causal interpretation

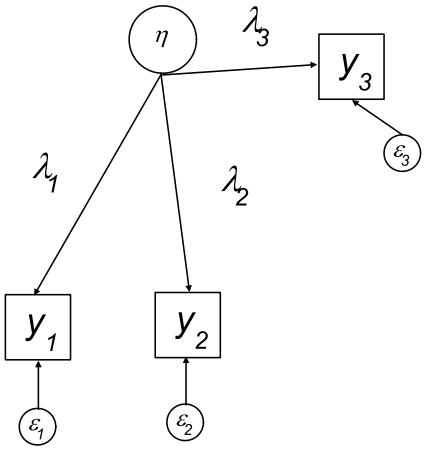

Figure 9: A simple SEM with one unobserved and three observed variables.

The implied covariance matrix among the observed variables will be:

$$\Sigma(\Theta) = \begin{bmatrix} \lambda_1^2 + \omega_1 & & \\ \lambda_1\lambda_2 & \lambda_2^2 + \omega_2 & \\ \lambda_1\lambda_3 & \lambda_2\lambda_3 & \lambda_3^2 + \omega_3 \end{bmatrix} \tag{9}$$

where $\Theta = \{\lambda_1, \lambda_2, \lambda_3, \omega_1, \omega_2, \omega_3\}$.

One can then try to identify the parameters by solving the following system of linear equations:

$$\begin{aligned} \lambda_1^2 + \omega_1 &= \sigma_{11} \\ \lambda_1\lambda_2 &= \sigma_{12} \\ \lambda_2^2 + \omega_2 &= \sigma_{22} \\ \lambda_1\lambda_3 &= \sigma_{31} \\ \lambda_2\lambda_3 &= \sigma_{32} \\ \lambda_3^2 + \omega_3 &= \sigma_{33} \end{aligned}$$

This system, however, is not **identifiable**, because each λ parameter can be positive or negative, and yet yield the same observed covariance matrix. In this case, it is not a failure of the representation: it is actually quite intuitive that this situation should happen, since the scale of the latent was not completely determined. The latent is not measured, and therefore its scale should be defined with respect to some standard, such as the scale of the measurements. One way is fixing its variance to one (setting its magnitude) and fixing the sign of one of the λ coefficients. Another simple way is letting its variance be a free parameter, and set one of the λ coefficients to 1, or some other constant.

However, this is not enough. Since we do not know the true covariance matrix, it has to be estimated from the sample covariance matrix. Due to sampling variability or to the own approximative nature of the model, this system will mostly likely have no solution. One has to define a criterion for a "good" solution, and a common way is writing down the likelihood of the sample as a function of Θ and then performing maximum likelihood estimation of those parameters. Bollen [6] gives an overview of diferent criteria for parameter estimation, as well as issues on identifiability.

B Gradients for fitting pure measurement/structural models

Jöreskog [21] introduced an early work in fitting procedures for maximum likelihood estimation of specific structural equation models. Similarly, we derive here the gradients of the parameters of the measurement/structural model used in this work. Given that the implied covariance matrix is

$$\Sigma(\Theta) = \Lambda \left(\mathbf{I} - \mathbf{B}\right)^{-\mathbf{1}} \Phi \left(\mathbf{I} - \mathbf{B}\right)^{-\mathbf{T}} \Lambda^{\mathbf{T}} + \Omega \tag{10}$$

and the loglikelihood function is

$$L(\Theta) = \sum_{i=1}^{N} -\frac{1}{2}|\Sigma| - \frac{1}{2}(\mathbf{x_i} - \mu)^T \Sigma^{-1} (\mathbf{x_i} - \mu) \tag{11}$$

where we implicity define $\Sigma = \Sigma(\Theta)$, we have [25]

$$\frac{\partial L(\Theta)}{\partial \Sigma} = -\frac{N}{2}\left[\Sigma^{-\mathbf{1}} - \Sigma^{-\mathbf{1}}\mathbf{S}\Sigma^{-\mathbf{1}}\right] \tag{12}$$

where $\mathbf{S}$ is the sample covariance matrix and N is the sample size.

For a given matrix $\mathbf{A}$, let a_{ij} be the element in row i, column j. Let $\mathbf{J}_{ij}$ be a matrix where $j_{ij} = 1$, and all other elements are zero. By defining matrix differentation as

$$\left(\frac{\partial \mathbf{A}}{\partial x}\right)_{ij} = \frac{\partial a_{ij}}{\partial x} \tag{13}$$

and using the properties of matrix differentiation given in [2] and [29], we have the following results:

$$\frac{\partial \Sigma(\Theta)}{\partial \beta_{ij}} = \Lambda(\mathbf{I} - \mathbf{B})^{-\mathbf{1}}\Phi(\mathbf{I} - \mathbf{B})^{-\mathbf{T}}\mathbf{J}_{ij}^{\mathbf{T}}(\mathbf{I} - \mathbf{B})^{-\mathbf{T}}\Lambda^{\mathbf{T}} + \Lambda(\mathbf{I} - \mathbf{B})^{-\mathbf{1}}\mathbf{J}_{ij}(\mathbf{I} - \mathbf{B})^{-\mathbf{1}}\Phi(\mathbf{I} - \mathbf{B})^{-\mathbf{T}}\Lambda^{\mathbf{T}} \tag{14}$$

$$\frac{\partial \Sigma(\Theta)}{\partial \phi_{ij}} = \Lambda(\mathbf{I} - \mathbf{B})^{-\mathbf{1}}\mathbf{J}_{ij}(\mathbf{I} - \mathbf{B})^{-\mathbf{T}}\Lambda^{\mathbf{T}} \tag{15}$$

$$\frac{\partial \Sigma(\Theta)}{\partial \omega_{ij}} = \mathbf{J}_{ij} \tag{16}$$

There is a simple way to compute $\partial\Sigma(\Theta)/\partial\Lambda$. Let $\eta_{(i)}$ and $\eta_{(j)}$ be the latent parents of the two measures y_i and y_j, respectively. Also, let λ_i be the loading of $\eta_{(i)}$ in y_i, and define λ_j similarly. We have that $\sigma_{ij}(\Theta) = \lambda_i\lambda_j Covar(\eta_{(i)}, \eta_{(j)})$. Therefore,

$$\frac{\partial \sigma_{ij}}{\partial \lambda_i} = \frac{\partial \sigma_{ji}}{\partial \lambda_i} = \lambda_j Covar(\eta_{(i)}, \eta_{(j)}) \tag{17}$$

and $\partial\Sigma(\Theta)/\partial\lambda_i$ will be zero in all other entries. This illustrates that the computation of these derivatives can be speeded up if one explores the fact that those gradients are very sparse matrices. For instance, $\partial\sigma_{pq}(\Theta)/\partial\beta_{ij}$ is just $\left\{\Lambda(\mathbf{I} - \mathbf{B})^{-\mathbf{1}}\Phi\right\}_{pj}\left\{(\mathbf{I} - \mathbf{B})^{\mathbf{T}}\Lambda^{\mathbf{T}}\right\}_{iq} + \left\{\Lambda(\mathbf{I} - \mathbf{B})^{-\mathbf{1}}\Phi\right\}_{qj}\left\{(\mathbf{I} - \mathbf{B})^{\mathbf{T}}\Lambda^{\mathbf{T}}\right\}_{ip}$.

After computing the derivatives of $\Sigma(\Theta)$ with respect to the parameters Θ, the final gradient will be given by

$$\frac{\partial L(\Theta)}{\partial \theta_i} = \sum_{x=1}^{d}\sum_{y=1}^{d} \frac{\partial L(\Theta)}{\partial \sigma_{xy}} \frac{\partial \sigma_{xy}}{\partial \theta_i} \tag{18}$$

where d is the number of observed variables.