
Bayesian Learning of Measurement and Structural Models

Ricardo Silva

Gatsby Computational Neuroscience Unit, Alexandra House, 17 Queen Square, London WC1N 3AR, UK

RBAS@GATSBY.UCL.AC.UK

Richard Scheines

Machine Learning Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

SCHEINES@ANDREW.CMU.EDU

Abstract

We present a Bayesian search algorithm for learning the structure of latent variable models of continuous variables. We stress the importance of applying search operators designed especially for the parametric family used in our models. This is performed by searching for subsets of the observed variables whose covariance matrix can be represented as a sum of a matrix of low rank and a diagonal matrix of residuals. The resulting search procedure is relatively efficient, since the main search operator has a branch factor that grows linearly with the number of variables. The resulting models are often simpler and give a better fit than models based on generalizations of factor analysis or those derived from standard hill-climbing methods.

1. Introduction

In a large class of problems, the observed associations in our data are due to hidden variables that are common causes of our measured variables. This happens, for instance, if the observations are sensor data measuring atmospheric phenomena, medical instruments measuring biological processes, econometrical indicators measuring economical processes and so on. We are interested in modeling such type of domains. Reymont and Joreskog (1996) and Bollen (1989) provide an extensive list of examples of this class.

If one wants to model the joint distribution of such measurements, it is desirable to represent hidden (latent) variables explicitly. Families of models such as factor analysis, hidden Markov models and variants

are basic examples of *latent variable models*. Such models often can be represented as graphical models and it is of interest that any learning procedure designed to build these models from data is also capable of learning the respective graphical structure.

This paper describes a procedure for learning the structure of latent variable models with the primary purpose of performing density estimation in continuous (i.i.d.) data, since many datasets containing sensor data, economical indicators and, in general, measurements obtained through scientific instruments are continuous. We choose a class of models called *measurement/structural models* which we believe provides a good trade-off between flexibility (i.e., the model space is sufficiently large) and cost (there are practical ways of searching for models in this class). This family is explained in Section 2, motivated by limitations of current approaches for structure learning.

Section 3 is the main section of this paper, and describes the structure search algorithm along with several examples. Section 4 introduces the Bayesian score function that will be used to guide the search, while Section 5 provides experimental results. A summary and directions for future work are given in Section 6.

2. Related Work And Limitations

We identify two main approaches in latent variable model search. The first one considers the space of graphs inspired by factor analysis, where latents are parents, but never children, of observed variables, and every observed variable has at least one latent parent. Usually, this space consists of models of different number of independent latents, where each observed variable is a child of all latent variables. A particular structure is illustrated by Figure 1(a). The structure learning problem, therefore, is reduced to the problem of deciding the number of latent variables as in, e.g., Minka (2000). This is an efficient learning procedure,

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

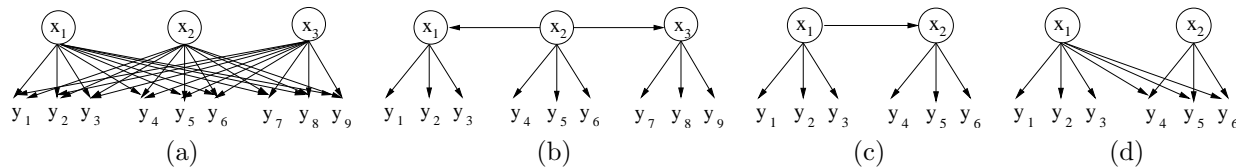


Figure 1. In this paper, x variables represent latents, and y variables represent observed variables. Many models encoded by (b) could in principle be represented by graph (a), albeit with many more parameters that makes learning hard with small samples. The same is true when using graph (d) to estimate the joint of variables given by model (c).

but does not consider that alternative models outside this space might be much simpler, as in Figure 1(b), and therefore fit the data better if sample sizes are relatively small.

An alternative is to consider full structure learning, allowing the model to assume an arbitrary directed acyclic graph (DAG) representation. As suggested by several authors, one could start with a standard hill-climbing procedure to search for a DAG without latent variables, and then introduce hidden variables as common parents of a set of observed variables that are densely connected. Another hill-climbing procedure could be started from this point, and the process iterated. An implementation of this idea is described and evaluated by Elidan et al. (2000). This is a more expensive but flexible approach.

The problem of this approach is ignoring that search operators that are useful for hill-climbing learning of latent-free DAGs might not be appropriate once hidden variables are introduced. For instance, assume the structure in Figure 1(c) is the true structure in our domain, and that the respective probability model is a multivariate Gaussian. Assume also that our current candidate is the graph in Figure 1(d), and there is a choice of parameters for (d) that allows it to represent the population mean and covariance matrix of $\{y_1, y_2, \dots, y_6\}$. Even if the graph in Figure 1(d) is potentially able to represent the marginal distribution of $\{y_1, y_2, \dots, y_6\}$, with small samples sizes the simpler graphical structure of Figure (c) might provide a better estimate of the joint. However, the usual search operators that generate new candidate graphs (as in Elidan et al., 2000) consist of adding, removing or reversing an edge in the current candidate. One can verify that typically for the space of Gaussian models representable by Figure 1(d), no edge addition, removal or reversal will increase the marginal likelihood for observed variables $\{y_1, y_2, \dots, y_6\}$. The standard greedy algorithm gets trapped at this candidate. We could do better with search operators especially designed for latent variable models. For computational reasons, it is also desirable that such operators do not create a large set of candidates at each search step.

Our choice of model space is the class of *measurement/structural* models. In this class, as in factor analysis, we forbid edges directed from an observed variable into a latent variable. However, we allow edges between latent variables and between observed variables, as long as the graph is acyclic.

The name of this class comes from the econometrics and social sciences literature (Bollen, 1989), where a plausible assumption is that the dataset consists of *clusters* of strongly correlated variables. Each cluster measures a different hidden common cause (e.g., a cluster of indicators of “economical stability,” another cluster of indicators of “job satisfaction” and so on). The *structural* model is the submodel among latent variables. The *measurement* model is the submodel describing the parents of each observed variable. The relevant latents are those that are causes but not effects of our measures. For the type of applications we discussed at the beginning of this paper (e.g., applications where factor analysis is considered suitable), the assumption that latents are common causes but not effects of observations is usually plausible.

3. A Hill-climbing Algorithm

We now describe a practical algorithm for learning measurement/structural models. The probabilistic model is a *mixture of Gaussians*. The linearity of each mixture component will play an important role in this paper. The algorithm is a hill-climbing procedure to find local maxima of a score function. This function is an approximation of the posterior probability of the model. We postpone its description to Section 4, since this is not essential to understand the algorithm.

3.1. A Variable Removal Procedure

Consider the problem stated by Kano and Harada (2000): initially, an expert provides a measurement/structural latent variable model G . Suppose the given model does not fit the data (e.g., according to a significance test). The problem is to find a subset \mathbf{R} of the given observed variables such that, when we remove \mathbf{R} from G (which amounts to remove the

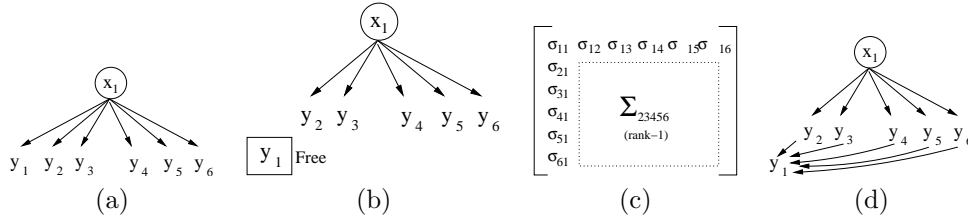


Figure 2. A way of evaluating the improvement on removing y_1 from the model given in (a) is to free the covariance between y_1 and the remaining variables, leaving the rest of the model unchanged (Figures (b), (c)). This amounts to compare the model Figure (d) against the model in Figure (a). See Example 1 in the text for details.

respective nodes and all edges into these nodes), the resulting submodel G' fits the data. The probabilistic model in this case is multivariate Gaussian.

One way of evaluating the gain of removing an observed variable y_1 , as described by Kano and Harada (2000), is to compare the original model G against a candidate model where: 1. the covariance $\sigma_{y_1 y_i}$ of y_1 and every observed variable $y_i, i \neq 1$, is *unconstrained*; 2. the marginal model for $\{y_2, y_3, \dots, y_m\}$ is *constrained* by the subgraph obtained when y_1 is removed from G .

Example 1. Suppose our model G is the model given by Figure 2(a), where we have a single latent variable x_1 and six observed variables, $\{y_1, \dots, y_6\}$. Without loss of generality, assume the mean of each variable is zero and that $\sigma_{x_1}^2$, the variance of x_1 , equals 1. Therefore, the marginal distribution of the observed variables is given by the covariance matrix of $\{y_1, \dots, y_6\}$. Since the model is multivariate Gaussian, it can be parameterized by a linear model such that

$$y_i = \lambda_i x_1 + \epsilon_i, i = 1, \dots, 6 \quad (1)$$

where ϵ_i is a zero mean Gaussian random variable. This means that the observed covariance matrix Σ is given by

$$\Sigma = \Lambda \Lambda' + \Phi \quad (2)$$

where $\Lambda' = [\lambda_1 \lambda_2 \dots \lambda_6]$ and Φ is a diagonal matrix, Φ_{ii} corresponding to the variance of ϵ_i . Therefore, Σ can be represented by the sum of a *rank-1* matrix ($\Lambda \Lambda'$) and a diagonal matrix. We say that Σ follows a *rank-1 constraint*.

To evaluate how well the fit of the graph improves when removing y_1 , we essentially fix the marginal structure of y_2, \dots, y_6 (Figure 2(b)), which will impose a rank-1 constraint on the respective marginal covariance matrix, while freeing all entries $\sigma_{y_1 y_i}$ to assume arbitrary values. This is illustrated by Figure 2(c). Notice that this basically amounts to evaluating the graphical model in Figure 2(d). Call this graph

$G_{\setminus y_1}$, a notation also used later in this paper. When factorizing the observed distribution $p(y_1, \dots, y_6)$ as

$$p(y_1, \dots, y_6) = p(y_1 | y_2, \dots, y_6) \times p(y_2, \dots, y_6) \quad (3)$$

we see that $p(y_1 | y_2, \dots, y_6)$ is *unconstrained* according to $G_{\setminus y_1}$, while (the covariance matrix of) $p(y_2, \dots, y_6)$ obeys a rank-1 constraint.

The upshot is: if the model in Figure 2(a) is in fact correct, then it should score higher than the model given by Figure 2(d) since it has fewer parameters. If the model in Figure 2(a) is not correct, then it should score lower than Figure 2(d), since it imposes more constraints on the observable marginal density¹. \square

We say that a graph for a Gaussian model *entails* a rank- r constraint if, for any choice of parameter values, its observed covariance matrix Σ can be represented by the sum $\aleph + \Phi$, where \aleph is a rank r matrix and Φ is diagonal. A factor analysis/principal component analysis model of r latents entails such a constraint. We say that a graph for a *mixture* of Gaussians model entails a rank- r constraint if, for every covariance matrix Σ_i corresponding to the i th component, $\Sigma_i = \aleph_i + \Phi$, where \aleph_i is rank r . Notice that the mixture of factor analysers model with r latents (Ghahramani & Beal, 1999) is a special case of a rank- r model.

This variable elimination procedure is the main building block of our algorithm for searching for such constraints, as explained in the next section. While Kano and Harada (2000) are interested only in selecting variables from a given latent variable model, we are interested in building latent models from scratch.

3.2. A Clustering Procedure: Overview

What does the variable removal procedure buy us, and what is the role of rank constraints? In a Gaussian model without hidden variables, conditional independence constraints essentially define the model: it is a

¹The model (a) is nested within (d), i.e. marginal y_1, \dots, y_6 in (a) can be represented by (d), but not the opposite: the matrix for y_1, y_2, y_3, y_4 is rank-1 only in (a).

well-known result that Gaussian graphical models with the same conditional independence constraints are also likelihood equivalent (Spirites et al., 2000). This is not true for latent variable models and explains why search operators defined by single edge modifications are not ideal in our problem. Although there is no known full characterization of likelihood equivalence in the class of Gaussian measurement/structural models, several recent results point to the usefulness of describing latent variable models by *rank constraints*:

- for a large class of measurement models, rank constraints are all that is needed to learn the structural model given a measurement model (Spirites et al., 2000);
- rank-1 constraints can identify any subgraph of a measurement model where each latent has at least three unique measures (observed children that are not children of any other variable) (Silva et al., 2006; Silva & Scheines, 2005);
- the *cross-covariance* matrix of any two groups of random variables with rank- r can be represented by a latent variable model with r pairs of latents (Wegelin et al., 2006);
- a large number of equality constraints entailed by factor analysis models can be described by combinations of rank constraints (Drton et al., 2005);

Searching for models that entails several rank constraints seems to be a promising way of building sparse graphical models of mixtures of Gaussians. The search problem is that different *subsets* of variables are connected by different rank constraints. For instance, both graphs in Figure 1(a) and Figure 1(b) entail a rank-3 constraint for its full set of observed variables, but only the one in Figure 1(b) entails a rank-1 constraint for the subset y_1, y_2, y_3, y_4 .

The natural way of scoring a model with a rank- r constraint is to build a latent variable model where all observed variables share the same r latents. To score the fit of a *combination* of rank constraints is to score a model where some edges between latents and observed variables do not exist. The challenge, therefore, is to design a search procedure that tries to *efficiently* find proper subsets of variables that correspond to rank- r constraints and to combine these constraints in a globally coherent model. Finding a model that entails all rank constraints entailed by the true model that generated the data is a NP-hard problem². In practice, we will adopt a greedy heuristic procedure as follows:

²By reduction of the NP-hard problem of learning a DAG to the problem of learning a structural model with fixed measurement model as in Spirites et al. (2000).

 Algorithm K-LATENTCLUSTERING

Input: a data set \mathbf{D} of observed variables \mathbf{Y} , an integer k
 Output: a DAG

1. Let G and G_{best} be fully connected graphs with nodes \mathbf{Y} and G_{-1} an empty graph
2. Do
3. $G \leftarrow \text{INTRODUCELATENTCLUSTER}(G, G_{-1}, \mathbf{Y}, k)$
4. Do
5. Let $y_{max} \leftarrow \text{argmax}_{y_i \in \mathbf{Y}} \mathcal{F}(G_{\setminus y_i}, \mathbf{D})$
6. If $\mathcal{F}(G_{\setminus y_{max}}, \mathbf{D}) > \mathcal{F}(G, \mathbf{D})$
7. Remove y_{max} from G
8. While G is modified
9. If $\text{GRAPHIMPROVED}(G, G_{best})$
10. $G_{-1} \leftarrow G_{best}$
11. $G_{best} \leftarrow G$
12. While G_{best} is modified
13. Return G_{best}

Figure 3. Build a latent variable model where variables are “clustered” by sharing the same k latent parents.

- find first a large group of variables that share a single latent parent (rank-1 constraints);
- put these variables aside, and within the remaining variables, try to find another group where variables share a single latent parent. Iterate;
- once no such group can be found, try now the same procedure where variables now share *two* latent parents (rank-2). Keep increasing the number of latents when no model can be found;
- stop when there are too few variables remaining (more on that later);
- join all submodels found so far into a single latent variable model. Add extra connections and remaining variables as appropriate;

The greedy aspect of this approach comes when trying to find the largest subset of variables that corresponds to a rank- r constraint for a fixed r . Instead of scoring all models of r latents (by using all subsets of the available observed variables), we adopt a top-down approach using the variable removal procedure of Section 3.1. An important advantage of this operator is that it defines a space of candidates that is *linear* in the number of variables at each step. Details are discussed in the next section.

3.3. Detailed Algorithm

We will assume for now that the number of mixture components n is given, and we have a score function $\mathcal{F}(G, \mathbf{D})$ such that, given n , graph G , and dataset \mathbf{D} , returns a score for graph G . The goal is to find a local maximum for \mathcal{F} within a space that greedily selects variables corresponding to low-rank constraints.

Algorithm INTRODUCELATENTCLUSTER

Input: two graphs G, G_{-1} ; a set of observed variables \mathbf{Y} ;
 an integer k defining the cluster size
 Output: a DAG

1. Let **NodeDump** be the set of observed nodes in \mathbf{Y} that are not in G
2. Let T be the number of clusters in G
3. Add a new cluster of k latents \mathbf{X}_T to G and form a complete DAG among latents in G .
4. For all $y \in \mathbf{NodeDump}$
5. If $y \in G_{-1}$
6. Let \mathbf{X}_i be the parent set of y in G_{-1}
7. Set \mathbf{X}_{i+1} to be the parent set of y in G
8. Else
9. Set \mathbf{X}_T to be the parent set of y in G
10. Return G

Figure 4. Add a new latent to G with a new arrangement of observed nodes. The new arrangement move nodes to the next latent cluster according to their position in G_{-1} .

Algorithm K-LATENTCLUSTERING is described in Figure 3 (see also Figure 4). It searches for models with *clusters* of variables, where each cluster is a set of observed variables that share the same k latent parents. Variables in different clusters share no parents. This algorithm is used by a higher-level algorithm, introduced later in this section, that iterates through different values of k and combines the output of each call into a single graph. For now, it suffices to understand how K-LATENTCLUSTERING works.

Example 2. Suppose $k = 1$, $\mathbf{Y} = \{y_1, y_2, \dots, y_6\}$. K-LATENTCLUSTERING will start by creating a model with 1 latent, and all variables in \mathbf{Y} will be children of this latent (top of Figure 5). We will then score each graph $G_{\setminus y_i}$ (as defined in Section 3.1). If the best of all such graphs is also better than the current candidate, we update our current candidate. The rest of Figure 5 illustrates two iterations of Steps 5-7: we choose to remove y_2 at the first iteration, and y_6 in the second.

Because we have now a model with different variables than the graph G_{best} from the previous iteration, we need a way to compare these two graphs. GRAPHIMPROVED is a comparison criterion: let \mathbf{V}_{best} be the set of nodes that are in G_{best} but not in G . Add \mathbf{V}_{best} to G to create a new graph G' , make \mathbf{V}_{best} a clique in G' , and make each node in \mathbf{V}_{best} a child of the original observed nodes \mathbf{Y}_G from G . Create G'_{best} in an analogous way. GRAPHIMPROVED returns true if $\mathcal{F}(G', \mathbf{D}) \geq \mathcal{F}(G'_{best}, \mathbf{D})$. The idea is that we need to have the same variables in both graphs, but we want to score only the rank constraints entailed by G_{best} and G . Therefore $p(\mathbf{V}_{best} | \mathbf{Y}_G)$ and $p(\mathbf{V} | \mathbf{Y}_{G_{best}})$ are unconstrained. \square

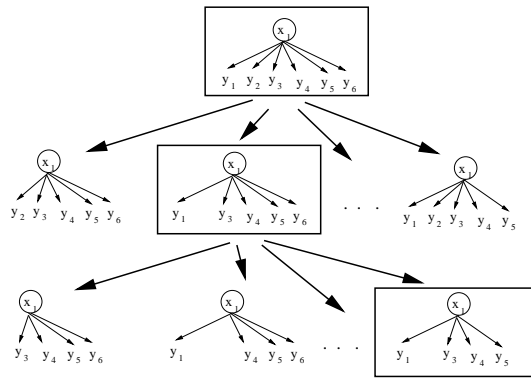


Figure 5. Example of a search tree in the K-LATENTCLUSTERING algorithm. It iteratively removes one observed variable at a time by choosing the submodel with the highest score. In the figure, each box represents the model chosen at the corresponding step.

Once we find our first cluster \mathbf{Y}_1 with a subset of observed variables sharing the same k latent parents, in the next iteration of K-LATENTCLUSTERING we try to find a second cluster \mathbf{Y}_2 that also share k latent parents, but that has none of the parents of \mathbf{Y}_1 ³. Nodes from \mathbf{Y}_1 might also need to be removed in the second iteration.

Example 3. To see why we might need to remove nodes from the first cluster, consider the following example illustrated by Figure 6. For simplicity we also use $k = 1$. Suppose the true model is the one depicted in Figure 6(a). A plausible candidate at the end of the first iteration of K-LATENTCLUSTERING is shown in Figure 6(b): since x_1 d-separates y_1, y_2, y_3, y_4, y_6 in the true model, a rank-1 constraint exists in the covariance matrix of such variables.

When the second iteration of K-LATENTCLUSTERING starts, INTRODUCELATENTCLUSTER will introduce latent t_2 as shown in Figure 6(c). Everything is on its right place, except y_6 . It is expected that the variable removal heuristic will choose y_6 to be removed, since $G_{\setminus y_6}$ can represent the actual population distribution, while no other candidate graph has this property (most entailed rank constraints that include y_6 are violated in the population). Figure 6(d) illustrates a typical candidate obtained at the end of the second iteration. In the third iteration, K-LATENTCLUSTERING starts with the exact model and, given enough data, no modifications will be judged to be necessary (the direction of the edge $t_1 \rightarrow t_2$ is not important here). In

³Notice that the cross-covariance matrix (i.e., the matrix composed of covariances of pairs in $\mathbf{Y}_1 \times \mathbf{Y}_2$) for each mixture component will be of rank k .

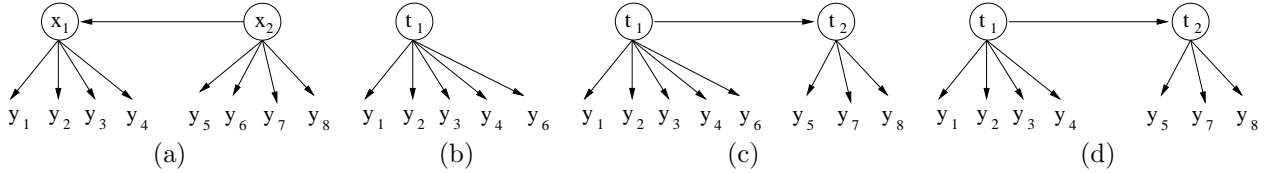


Figure 6. An illustration of different iterations of K-LATENTCLUSTERING (see Example 3 in the text). Figure (a) shows the true model, and (b) and (c) are examples of models generated at the end of the first iteration of the algorithm. Notice the arbitrary labeling t_1 for the latent in Figure (b). The remaining figures show the progression of the second iteration.

Algorithm FULLLATENTCLUSTERING

Input: a data set \mathbf{D}

Output: a DAG

1. $i \leftarrow 0$; $\mathbf{D}_0 \leftarrow \mathbf{D}$; **Solutions** $\leftarrow \emptyset$; $k \leftarrow 1$
2. Do
3. $G_i \leftarrow \text{K-LATENTCLUSTERING}(\mathbf{D}_i, k)$
4. If G_i has some latent node
5. **Solutions** $\leftarrow \mathbf{Solutions} \cup \{G_i\}$
6. Let \mathbf{D}_{i+1} be a subset of \mathbf{D}_i containing data only for variables not in G_i
7. $i \leftarrow i + 1$
8. While **Solutions** changes
9. Increase k by 1 and repeat Steps 2-8 if the number v of remaining variables is large enough
10. Let G_{full} be the graph composed by merging all graphs in **Solutions**, where latents are fully connected as an arbitrary DAG
11. For every pair $\{G_i, G_j\} \subseteq \mathbf{Solutions}$
12. Greedily add/delete new independent latents $y_i \leftarrow x_{new} \rightarrow y_j$ to G_{full} , ($y_i \in G_i, y_j \in G_j$), to maximize $\mathcal{F}(G_{full}, \mathbf{D})$
13. Return G_{full}

Figure 8. Merge the solutions of multiple K-LATENTCLUSTERING calls.

general, a node might traverse all clusters in a complete run of the algorithm. \square

This example is also useful to understand algorithm INTRODUCELATENTCLUSTER. This algorithm checks the position of each removed node (those in that are in \mathbf{Y} but not in G), and moves each removed node to its “next” cluster, following the order by which such clusters were created (obtained from G_{-1}). Notice that if $T = 0$, this will create the initial cluster, and all nodes will be added to LC_0 . Notice that all latents are connected in an arbitrary graph. We will deal with learning the structural model at the end.

We are now ready to introduce the full latent clustering algorithm, as described in Figure 8. The initial stages just call K-LATENTCLUSTERING repeatedly, while increasing k when necessary. This first stage (Steps 1-9) ends when k gets large enough⁴. Notice that not all

⁴When there are more edges in the graph than entries in the sample covariance matrix.

observed variables might be included at this stage.

In the remaining of the algorithm, we merge all solutions resulting from a call to K-LATENTCLUSTERING into a single graph. Since each solution was derived individually, there might be correlations between variables in different graphs that are not accounted by the latent variables. We account for such residual correlations by greedily introducing “ancillary” latents: these are latents that are disconnected from all other variables, except a pair of observed variables y_i, y_j . Such a latent affects only the covariance $\sigma_{y_i y_j}$ (on each mixture component)⁵.

Example 4. Figure 7 provides an example on how K-LATENTCLUSTERING is used in the full clustering procedure. Different number of latents per cluster can in principle be found by increasing k as needed. \square

Finally, we finish the search procedure by inserting the possibly remaining observed nodes. The final search consists of a standard hill-climbing search for edges among latents, and for edges into the newly added nodes. We call the full algorithm composed by FULLLATENTCLUSTERING and the final greedy search the RANKSEARCH algorithm, evaluated in Section 5.

4. A Bayesian Score Function

As stated before, our probability model is a mixture of Gaussians. The model can therefore be represented by a linear parameterization conditioned on a discrete variable representing the mixture component. Since we are interested in scoring graphs using a posterior distribution, we need priors for such parameters. We adopt the parameterization and priors for mixture of factor analysers given by Ghahramani and Beal (1999) and a uniform prior for graphs. The only difference is that we allow edges between observed variables and between latents. Such edges can use the same type of priors and hyperparameters. We also use the same type of variational approximation and hyperparam-

⁵This is not strictly precise. It might also constrain the variances $\sigma_{y_i}^2$ and $\sigma_{y_j}^2$ if the residual covariance between this pair is strong enough, for instance.

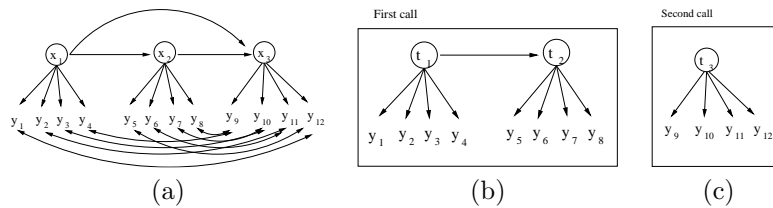


Figure 7. An illustration of the interaction between FULLLATENTCLUSTERING and K-LATENTCLUSTERING. Suppose the true model is the one in Figure (a). Each bi-directed edge represents a single latent that is independent of all other latents in the model. In (b), we have a typical output of the first call of K-LATENTCLUSTERING ($k = 1$) from FULLLATENTCLUSTERING. Notice that the third cluster is not included with this model, because K-LATENTCLUSTERING at this stage searches for clusters that are separated by single latents. This third cluster, however, can be generated individually by a second call to K-LATENTCLUSTERING, as shown in (c). It is intuitive that if we want to merge these two graphs, we have to search for the residual covariances that are due to the original “bi-directed” edges.

ter fitting of Ghahramani and Beal (1999). The only difference is that we fully factorize the marginal variational distribution of the “ancillary latents,” since they might grow to large numbers.

It is interesting to notice that this approach can be trivially extended to a non-parametric Bayesian setting using Dirichlet process mixtures (and the truncated variational approximation of Blei and Jordan (2004)). We did not perform experiments with such model due to its high computational cost.

5. Experiments

It is important to know how well the variable removal heuristic performs on finding correct measurement/structural graphs. This can be done by simulations, which are not included in this paper due to lack of space but described by Silva (2005). This heuristic largely succeeds on rebuilding a variety of structures.

To evaluate RANKSEARCH (RS) on the task of density estimation, we compare its performance against two standard algorithms: mixture of factor analysers (FA) (Ghahramani & Beal, 1999) and FINDHIDDEN (FH) (Elidan et al., 2000). FH is designed to exploit conditional independencies among observed variables, which is not the case in RS and FA. If the domain contains many of such independencies, then FH should be more appropriate. We stress that RS was never intended to be used on this type of problem, but on those where factor analysis seems appropriate (see Spirtes et al., 2000, for examples of applications of each type)⁶.

⁶Some implementation details: both RS and FH use STRUCTURAL EM (Elidan et al., 2000) to speed up search. Both use the same score function and number of mixture components, chosen a priori by the output of the mixture of factor analysers algorithm. That is, all three algorithms use the same number of mixture components. Moreover, due to its high computational cost on our datasets, our implementation of FINDHIDDEN only expands the semiclique

Eight datasets are used. Two datasets were simulated from a sparse latent variable model of 2 mixture components and 5 latents, where each latent has exactly 5 observed children, and each observed variable has a single parent. Latents are fully connected as an arbitrary DAG. Datasets SYNTH200 and SYNTH500 were generated by sampling 200 and 500 cases from this model, respectively. The remaining datasets are⁷:

- arrhythmia (arr): 452 instances / 16 variables
- ionosphere (iono): 351 / 33
- heart images (spectf): 349 / 44
- water treatment plant (water): 380 / 38
- waveform generator (wave): 5000 / 21
- breast cancer (wdbc): 561 / 25

Table 1 shows the results. We use 5-fold cross-validation, and report the results for each partition.

By analysing the synthetic datasets SYNTH200 and SYNTH500, where the assumption of a sparse measurement structure holds, it seems clear the RS algorithm is able to give a better fit with smaller datasets, although as expected all algorithms will do well with larger datasets. In the SPECTF dataset, it is clear that RS tends to do better than FA, although in this particular cross-validation setting we missed statistical significance (by a sign test) due to one anomalous case. The WAVE dataset is also synthetic, and data is abundant: all three algorithms perform approximately the same, with RS identifying a single latent separating most observed variables. Differences in IONO are more extreme arguably due to truncations in the data.

The dataset WATER is used to illustrate an interest—that has the best initial score (details in Elidan et al., 2000)

⁷All datasets are from the UCI Repository (Blake & Merz, 1998). Non-ordinal discrete variables and instances with missing values were removed. For the ARRHYTHMIA dataset, we used only the first 20 continuous variables.

Table 1. Evaluation of the average test log-likelihood of the outcomes of three algorithms. For each structure output by a corresponding search algorithm, we fit its parameters with the training set using maximum likelihood estimation. Each line is the result of a single split in a 5-fold cross-validation. The entry R – M is the difference between RS and mixture of factor analysers. The entry R – F is the difference between RS and FINDHIDDEN. The table also provides the respective averages (avg). A star (*) indicates positive differences that are statistically significant at a level of 0.05 using a sign test.

Set	RS	R - M	R - F	RS	R - M	R - F	RS	R - M	R - F	RS	R - M	R - F
	arrythmia			iono			syn200			syn500		
1	-18.76	1.14	0.56	-34.65	4.84	9.54	-18.50	0.16	0.97	-13.63	-0.01	0.07
2	-22.50	1.35	1.47	-25.60	6.06	11.58	-17.99	0.32	0.24	-11.09	-0.08	0.12
3	-18.32	1.29	0.32	-28.30	7.05	11.53	-18.70	0.46	1.10	-11.70	0.08	0.26
4	-23.61	1.99	1.37	-32.90	4.25	6.73	-16.24	0.59	1.36	-10.67	0.02	0.22
5	-22.97	1.03	1.17	-32.87	7.72	9.89	-18.67	0.34	0.67	-11.89	0.06	0.11
avg	-21.23	1.36*	0.98*	-30.86	5.98*	9.85*	-18.02	0.37*	0.87*	-11.80	0.01	0.16*
	spectf			water			wave			wdbc		
1	-47.60	1.48	2.33	-30.91	6.33	5.1	-24.11	-0.06	0.80	-15.78	-0.81	-0.62
2	-45.76	4.72	4.66	-29.69	2.48	4.36	-23.97	-0.05	-0.61	-19.87	-1.72	-0.52
3	-47.93	-0.01	0.21	-40.76	7.57	-1.74	-23.87	-0.1	0.96	-17.73	-1.45	-1.2
4	-43.42	2.31	4.64	-42.57	4.97	-2.77	-24.1	-0.09	0.97	-15.76	0.28	-0.06
5	-41.52	3.01	5.13	-44.4	8.08	-9.63	-24.24	-0.05	-0.04	-17.11	-2.06	0.08
avg	-45.25	2.30	3.39*	-37.67	5.89*	-0.94	-24.06	-0.07	0.42	-17.25	-1.15	-0.46

ing phenomenon: RS and FA do not work well with a dataset which has several discrete ordinal variables, being very unstable. Another interesting phenomenon was the outcome of the WDBC dataset. RS actually performed worse and seemed to underfit the data, greedily choosing clusters with small number of latents. FA chooses a relatively large number of latents, approximately 12 per iteration, i.e., almost 1 latent per 2 observed variables. However, if we start FULLLATENTCLUSTERING with 2 latents per clusters ($k = 2$) instead of 1, then RS is better than FA in 3 out of 5 partitions (4 out of 5 compared to FH). This suggests other possible variations of RANKSEARCH.

6. Conclusion

To the best of our knowledge, RANKSEARCH is the first structure learning algorithm for graphical models explicitly designed to use constraints that are relevant to continuous latent variable models. The algorithm can certainly be extended in a variety of ways. A natural extension, for instance, is to allow K-LATENTCLUSTERING to also re-insert nodes that are removed, which is actually suggested by Kano and Harada (2000). In this paper, we tried to keep the algorithm as simple as possible to evaluate how it performs with a minimum set of search operators. As future work, we plan to adapt it to ordinal data and different types of parametric and nonparametric models. If latents are connected by non-linear functions, but observed variables are still linear functions of their parents, rank constraints are still of singular importance on structure learning (Silva & Scheines, 2005).

References

- Blake, C. L., & Merz, C. J. (1998). UCI repository, <http://www.ics.uci.edu/~mllearn/mlrepository.html>.
- Blei, D., & Jordan, M. (2004). Variational methods for the Dirichlet process. *21st ICML*.
- Bollen, K. (1989). *Structural Equation Models with Latent Variables*. John Wiley & Sons.
- Drton, M., Sturmfels, B., & Sullivant, S. (2005). Algebraic factor analysis: tetrads, pentads and beyond. *arxiv.org*.
- Elidan, G., Lotner, N., Friedman, N., & Koller, D. (2000). Discovering hidden variables. *NIPS*, 13.
- Ghahramani, Z., & Beal, M. (1999). Variational inference for Bayesian mixture of factor analysers. *NIPS*, 12.
- Kano, Y., & Harada, A. (2000). Stepwise variable selection in factor analysis. *Psychometrika*, 65.
- Minka, T. (2000). Automatic choice of dimensionality for PCA. *NIPS*, 13.
- Reyment, R., & Joreskog, K. (1996). *Applied Factor Analysis in the Natural Sciences*. Cambridge Press.
- Silva, R. (2005). Automatic discovery of latent variable models. *PhD Thesis, Machine Learning Dpt., CMU*.
- Silva, R., & Scheines, R. (2005). New d-separation results for learning continuous latent variable models. *ICML*.
- Silva, R., Scheines, R., Glymour, C., & Spirtes, P. (2006). Learning the structure of linear latent variable models. *JMLR*, 7, 191–246.
- Spirtes, P., Glymour, C., & Scheines, R. (2000). *Causation, Prediction and Search*. Cambridge Press.
- Wegelin, J., Packer, A., & Richardson, T. (2006). Latent models for cross-covariance. *J. of Multivar. Analysis*, 97.