Quartet-Based Learning of Shallow Latent Variables

Tao Chen and Nevin L. Zhang
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology, Hong Kong, China
{csct,lzhang}@cse.ust.hk

Abstract

Hierarchical latent class(HLC) models are tree-structured Bayesian networks where leaf nodes are observed while internal nodes are hidden. We explore the following two-stage approach for learning HLC models: One first identifies the shallow latent variables – latent variables adjacent to observed variables – and then determines the structure among the shallow and possibly some other "deep" latent variables. This paper is concerned with the first stage. In earlier work, we have shown how shallow latent variables can be correctly identified from quartet submodels if one could learn them without errors. In reality, one does make errors when learning quartet submodels. In this paper, we study the probability of such errors and propose a method that can reliably identify shallow latent variables despite of the errors.

1 Introduction

Hierarchical latent class (HLC) models (Zhang, 2004) are tree-structured Bayesian networks where variables at leaf nodes are observed and hence are called *manifest variables (nodes)*, while variables at internal nodes are hidden and hence are called *latent variables (nodes)*. HLC models were first identified by Pearl (1988) as a potentially useful class of models and were first systematically studied by Zhang (2004) as a framework to alleviate the disadvantages of LC models for clustering. As a tool for cluster analysis, HLC Models produce more meaningful clusters than latent class models and they allow multi-way clustering at the same time. As a tool for probabilistic modeling, they can model high-order interactions among observed variables and help one to reveal interesting latent structures behind data. They also facilitate unsupervised profiling.

Several algorithms for learning HLC models have been proposed. Among them, the heuristic single hill-climbing (HSHC) algorithm developed by Zhang and Kočka (2004) is currently the most efficient. HSHC has been used to successfully analyze, among others, the CoIL Challenge 2000 data set (van der Putten and van Someren, 2004), which consists of 42 manifest variables and 5,822 records, and a data set about traditional Chinese medicine (TCM), which consists of 35 manifest variables and 2,600 records.

In terms of running time, HSHC took 98 hours to analyze the aforementioned TCM data set on a top-end PC, and 121 hours to analyze the CoIL Challenge 2000 data set. It is clear that HSHC will not be able to analyze data sets with hundreds of manifest variables.

Aimed at developing algorithms more efficient than currently available, we explore a two-stage approach where one (1) identifies the shallow latent variables, i.e. latent variables adjacent to observed variables, and (2) determines the structure among those shallow, and possibly some other "deep", latent variables. This paper is concerned with the first stage.

In earlier work (Chen and Zhang, 2005), we have shown how shallow latent variables can be correctly identified from quartet submodels if one could learn them without errors. In reality, one does make errors when learning quartet-submodels. In this paper, we study the probability of such errors and propose a method that

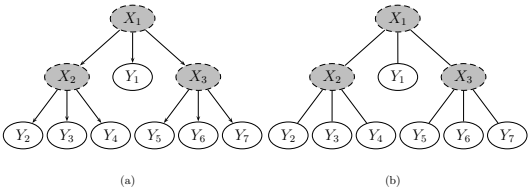

Figure 1: An example HLC model and the corresponding unrooted HLC model. The X_i's are latent nodes and the Y_j's are manifest nodes.

can reliably identify shallow latent variables despite of the errors.

2 HLC Models and Shallow Latent Variables

Figure 1 (a) shows an example HLC model. Zhang (2004) has proved that it is impossible to determine, from data, the orientation of edges in an HLC model. One can learn only *unrooted HLC models*, i.e. HLC models with all directions on the edges dropped. Figure 1 (b) shows an example unrooted HLC model. An unrooted HLC model represents a class of HLC models. Members of the class are obtained by rooting the model at various nodes. Semantically it is a Markov random field on an undirected tree. In the rest of this paper, we are concerned only with unrooted HLC models.

In this paper, we will use the term *HLC structure* to refer to the set of nodes in an HLC model and the connections among them. HLC structure is *regular* if it does not contain latent nodes of degree 2. Starting from an irregular HLC structure, we can obtain a regular structure by connecting the two neighbors of each latent node of degree 2 and then remove that node. This process is known as *regularization*. In this paper, we only consider regular HLC structures.

In an HLC model, a *shallow latent variable* (SLV) is one that is adjacent to at least one manifest variable. Two manifest variables are *siblings* if they are adjacent to the same (shallow) latent variable. For a given shallow latent variable X, all manifest variables adjacent to X constitute a *sibling cluster*. In the HLC structure shown in Figure 1 (b), there are 3 sibling clusters, namely $\{Y_1\}$, $\{Y_2, Y_3, Y_4\}$, and $\{Y_5, Y_6, Y_7\}$. They correspond to the three latent variables in the model respectively.

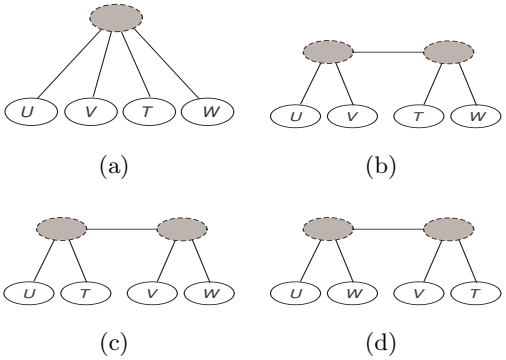

Figure 2: Four possible quartet substructures for a quartet $\mathbf{Q} = \{U, V, T, W\}$. The fork in (a) is denoted by $[UVTW]$, the dogbones in (b), (c), and (d) respectively by $[UV|TW]$, $[UT|VW]$, and $[UW|VT]$.

3 Quartet-Based SLV Discovery: The Principle

A shallow latent node is defined by its relationship with its manifest neighbors. Hence to identify the shallow latent nodes means to identify sibling clusters. To identify sibling clusters, we need to determine, for each pair of manifest variables (U, V), whether U and V are siblings. In this section, we explain how to answer this question by inspecting quartet submodels.

A *quartet* is a set of four manifest variables, e.g., $\mathbf{Q} = \{U, V, T, W\}$. The *restriction* of an HLC model structure $\mathcal{S}$ onto $\mathbf{Q}$ is obtained from $\mathcal{S}$ by deleting all the nodes and edges not in the paths between any pair of variables in $\mathbf{Q}$. Applying regularization to the resulting HLC structure, we obtain the *quartet substructure* for $\mathbf{Q}$, which we denote by $\mathcal{S}|_{\mathbf{Q}}$. As shown in Figure 2, $\mathcal{S}|_{\mathbf{Q}}$ is either the *fork* $[UVTW]$, or one of the *dogbones* $[UV|TW]$, $[UT|VW]$, and $[UW|VT]$.

Consider the HLC structure in Figure 1 (b). The quartet substrcuture for $\{Y_1, Y_2, Y_3, Y_4\}$ is the fork $[Y_1Y_2Y_3Y_4]$, while that for $\{Y_1, Y_2, Y_4, Y_5\}$ is the dogbone $[Y_1Y_5|Y_2Y_4]$, and that for $\{Y_1, Y_2, Y_5, Y_6\}$ is the dogbone $[Y_1Y_2|Y_5Y_6]$.

It is obvious that if two manifest variables U and V are siblings in the structure $\mathcal{S}$, then they must be siblings in any quartet substructure that involves both of them. Chen and Zhang

(2005) has proved the converse. So, we have

Theorem 1 *Suppose $\mathcal{S}$ is a regular HLC structure. Let (U, V) be a pair of manifest variables. Then U and V are not siblings in $\mathcal{S}$ iff there exist two other manifest variables T and W such that $\mathcal{S}|_{\{U,V,T,W\}}$ is a dogbone where U and V are not siblings.*

Theorem 1 indicates that we can determine whether U and V are siblings by examining all possible quartets involving both U and V. There are $\frac{(n-2)(n-3)}{2}$ such quartets, where n is the number of manifest variables. We next present a result that allows us to do the same by examining only $n-3$ quartets.

Let (U, V) be a pair of manifest variables and T be a third one. We use $\mathcal{Q}_{UV|T}$ to denote the following collection of quartets:

$$\mathcal{Q}_{UV|T} := \{\{U, V, T, W\} | W \in \mathbf{Y} \backslash \{U, V, T\}\},$$

where $\mathbf{Y}$ is the set of all manifest variables. T appears in every quartet in $\mathcal{Q}_{UV|T}$ and thus called a *standing member* of $\mathcal{Q}_{UV|T}$. It is clear that $\mathcal{Q}_{UV|T}$ consists of $n-3$ quartets. Chen and Zhang (2005) has also proved the following:

Theorem 2 *Suppose $\mathcal{S}$ is a regular HLC structure. Let (U, V) be a pair of manifest variables and T be a third one. U and V are not siblings in $\mathcal{S}$ iff there exists a quartet $\mathbf{Q} \in \mathcal{Q}_{UV|T}$ such that $\mathcal{S}|_{\mathbf{Q}}$ is a dogbone where U and V are not siblings.*

In learning tasks, we do not know the structure of the generative model structure $\mathcal{S}$. Where do we obtain the quartet substructures? The answer is to learn them from data. Let $\mathcal{M}$ be an HLC model with a regular structure $\mathcal{S}$. Suppose that $\mathbf{D}$ is a collection of i.i.d samples drawn from $\mathcal{M}$. Each record in $\mathbf{D}$ contains values for the manifest variables, but not for the latent variables. Let $\texttt{QSL}(\mathbf{D}, \mathbf{Q})$ be a routine that takes data $\mathbf{D}$ and a quartet $\mathbf{Q}$ as inputs, and returns an HLC structure on the quartet $\mathbf{Q}$. One can use the HSHC algorithm to implement QSL, and one can first project the data $\mathbf{D}$ onto the quartet $\mathbf{Q}$ when learning the substructure for $\mathbf{Q}$.

Suppose QSL is error-free, i.e. $\texttt{QSL}(\mathbf{D}, \mathbf{Q}) = \mathcal{S}|_{\mathbf{Q}}$ for any quartet $\mathbf{Q}$. By Theorem 2, we can determine whether two manifest variables U and V are siblings (in the generative model) as follow:

- Pick a third manifest variable T.
- For each $\mathbf{Q} \in \mathcal{Q}_{UV|T}$, call $\texttt{QSL}(\mathbf{D}, \mathbf{Q})$.
- If U and V are not siblings in one of the resulting substructures, then conclude that they are not siblings (in the generative model).
- If U and V are siblings in all the resulting substructures, then conclude that they are siblings (in the generative model).

We can run the above procedure on each pair of manifest variables to determine whether they are siblings. Afterwards, we can summarize all the results using a *sibling graph*. The sibling graph is an undirected graph over the manifest variables where two variables U and V are adjacent iff they are determined as siblings.

If QSL is error-free, then each connected component of the sibling graph should be completely connected and correspond to one latent variable. For example, if the structure of the generative model is as Figure 3 (a), then the sibling graph that we obtain will be as Figure 3 (b). There are four completely connected components, namely $\{Y_1, Y_2, Y_3\}$, $\{Y_4, Y_5, Y_6\}$, $\{Y_7, Y_8, Y_9\}$, $\{Y_{10}, Y_{11}, Y_{12}\}$, which respectively correspond to the four latent variables in the generative structure.

4 Probability of Learning Quartet Submodels Correctly

In the previous section, we assumed that QSL is error-free. In reality, one does make mistakes when learning quartet submodels. We have empirically studied the probability of such errors.

For our experiments, QSL was implemented using the HSHC algorithm. For model selection, we tried each of the scoring functions, namely BIC (Schwarz, 1978), BICe (Kočka and Zhang, 2002), AIC (Akaike, 1974), and the Cheeseman-Stutz(CS) score (Cheeseman and Stutz, 1995).

We randomly generated around 20,000 quartet models. About half of them are fork-structured, while the rest are dogbone-structured. The cardinalities of the variables

range from 2 to 5. From each of the models, we sampled data sets of size 500, 1,000, 2,500, 5,000. The QSL was then used to analyze the data sets. In all the experiments, QSL produced either forks or dogbones. Consequently, there are only three classes of errors:

F2D: The generative model was a fork, and QSL produced a dogbone.

D2F: The generative model was a dogbone, and QSL produced a fork.

D2D: The generative model was a dogbone, and QSL produced a different dogbone.

The statistics are shown in Table 5. To understand the meaning of the numbers, consider the number 0.83% at the upper-left corner of the top table. It means that, when the sample size was 500, QSL returned dogbones in 0.83% percent, or 83, of the 10,011 cases where the generative models were forks. In all the other cases, QSL returned the correct fork structure.

It is clear from the tables that: The probability of D2F errors is large; the probability of F2D errors is small; and the probability of D2D errors is very small, especially when BIC or BICe are used for model selection. Also note that the probability of D2F decreases with sample size, but that of F2D errors do not. In the next section, we will use those observations when designing an algorithm for identifying shallow latent variables.

In terms of comparison among the scoring functions, BIC and BICe are clearly preferred over the other two as far as F2D and D2D errors are concerned. It is interesting to observe that BICe, although proposed as an improvement to BIC, is not as good as BIC when it comes to learning quartet models. For the rest of this paper, we use BIC.

5 Quartet-Based SLV Discovery: An Algorithm

The quartet-based approach to SLV discovery consists of three steps: (1) learn quartet submodels, (2) determine sibling relationships among manifest variables and hence obtain a sibling graph, and (3) introduce SLVs based on the sibling graph. Three questions ensue:

Table 1: Percentage of times that QSL produced the wrong quartet structure. The table on the top is for the fork-structured generative models, while the table at the bottom is for the dogbone-structured generative models.

	500(F2D)	1000(F2D)	2500(F2D)	5000(F2D)
BIC	0.83%	1.87%	3.70%	3.93%
BICe	1.42%	3.16%	6.58%	7.86%
AIC	12.8%	10.2%	6.81%	4.85%
CS	6.20%	5.05%	6.19%	6.41%
Total=10011				

	500		1000		2500		5000	
	D2F	D2D	D2F	D2D	D2F	D2D	D2F	D2D
BIC	95.3%	0.00%	87.0%	0.00%	68.2%	0.00%	51.5%	0.00%
BICe	95.0%	0.05%	86.8%	0.00%	67.8%	0.04%	50.6%	0.04%
AIC	71.2%	2.71%	60.5%	1.58%	46.7%	0.54%	39.0%	0.38%
CS	88.5%	1.93%	83.4%	0.74%	67.0%	0.30%	51.5%	0.13%
Total=10023								

i) Which quartets should we use in Step 1?
ii) How do we determine sibling relationships in Step 2 based on results from Step 1?
iii) How do we introduce SLVs in Step 3 based on the sibling graph constructed in Step 2?

Our answer to the second question is simple: two manifest variables are regarded as non-siblings if they are not siblings in one of the quartet submodels. In the next two subsections, we discuss the other two questions.

5.1 SLV Introduction

As seen in Section 3, when QSL is error-free, the sibling graph one obtains has a nice property: every connected component is a fully connected subgraph. In this case, the rule for SLV introduction is obvious:

> Introduce one latent variable for each connected component.

When QSL is error-prone, the sibling graph one obtains no longer has the aforementioned property. Suppose data are sampled from a model with the structure shown in Figure 3 (a). Then what one obtains might be the graphs (c), (d), or (e) instead of (b).

Nonetheless, we still use the above SLV introduction rule for the general case. We choose it for its simplicity, and for the lack of better alternatives. This choice also endows the SLV discovery algorithm being developed with error tolerance abilities.

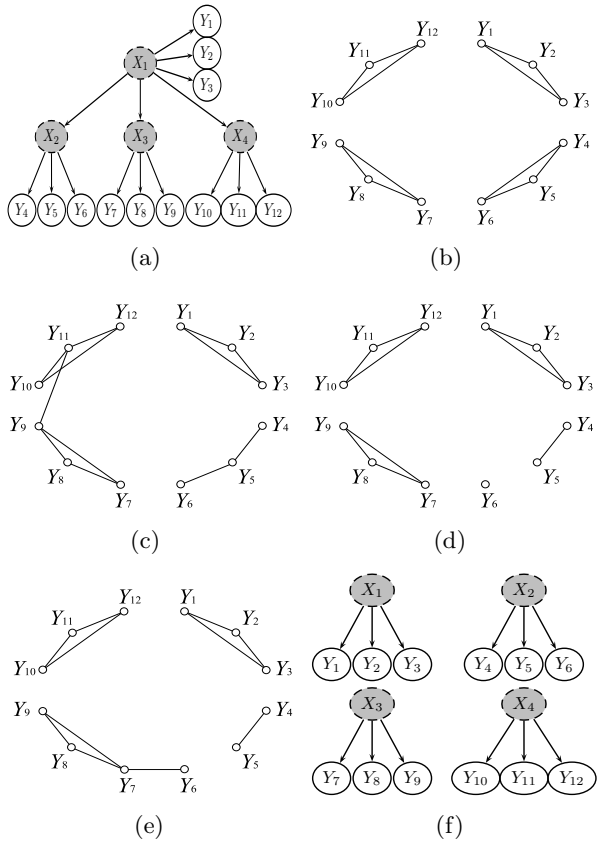

Figure 3: Generative model (a), sibling graphs (b, c, d, e), and shallow latent variables (f).

There are three types of mistakes that one can make when introducing SLVs, namely *latent omission, latent commission*, and *misclustering*. In the example shown in Figure 3, if we introduce SLVs based on the sibling graph (c), then we will introduce three latent variables. Two of them correspond respectively to the latent variables X_1 and X_2 in the generative model, while the third corresponds to a "merge" of X_3 and X_4. So, one latent variable is *omitted*.

If we introduce SLVs based on the sibling graph (d), then we will introduce five latent variables. Three of them correspond respectively to X_1, X_3, and X_4, while the other two are both related to X_2. So, one latent variable is *commissioned*.

If we introduce SLVs based on the sibling graph (e), then we will introduce four latent variables. Two of them correspond respectively to X_1 and X_4. The other two are related to X_2 and X_3, but there is not clear correspondence. This is a case of *misclustering*.

We next turn to Question 1. There, the most important concern is how to minimize errors.

5.2 Quartet Selection

To determine whether two manifest variables U and V are siblings, we can consider all quartets in $\mathcal{Q}_{UV|T}$, i.e. all the quartets with a third variable T as a standing member. This selection of quartets will be referred to as the *parsimonious selection*. There are only $n-3$ quartets in the selection.

When `QSL` were error-free, one can use $\mathcal{Q}_{UV|T}$ to correctly determine whether U and V are siblings. As an example, suppose data are sampled from a model with the structure shown in Figure 3 (a), and we want to determine whether Y_9 and Y_{11} are siblings based on data. Further suppose Y_4 is picked as the standing member. If `QSL` is error-free, we get 4 dogbones where Y_9 and Y_{11} are not sibling, namely $[Y_9Y_7|Y_{11}Y_4]$, $[Y_9Y_8|Y_{11}Y_4]$, $[Y_9Y_4|Y_{11}Y_{10}]$, $[Y_9Y_4|Y_{11}Y_{12}]$, and hence conclude that Y_9 and Y_{11} are not siblings.

In reality, `QSL` does make mistakes. According to Section 4, the probability of D2F errors is quite high. There is therefore good chance that, instead of the aforementioned 4 dogbones, we get 4 forks. In that case, Y_9 and Y_{11} will be regarded as siblings, resulting in a *fake edge* in the sibling graph.

$\mathcal{Q}_{UV|T}$ represents one extreme when it comes to quartet selection. The other extreme is to use all the quartets that involve both U and V. This selection of quartets will be referred to as the *generous selection*. Suppose U and V are non-siblings in the generative model. This generous choice will reduce the effects of D2F errors. As a matter of fact, there are now many more quartets, when compared with the case of parsimonious quartet selection, for which the true structures are dogbones with U and V being non-siblings. If we learn one of those structures correctly, we will be able to correctly identify U and V as non-siblings.

The generous selection also comes with a drawback. In our running example, consider the task of determining whether Y_1 and Y_2 are siblings based on data. There are 36 quartets

that involve Y_1 and Y_2 and for which the true structures are forks. Those include $[Y_1Y_2Y_4Y_7]$, $[Y_1Y_2Y_4Y_{10}]$, and so on. According to the Section 4, the probability for QSL to make an F2D mistake on any one of those quartets is small. But there are 36 of them. There is good chance for QSL to make an F2D mistake on one of them. If the structure QSL learned for $[Y_1Y_2Y_4Y_7]$, for instance, turns out to be the dogbone $[Y_1Y_4|Y_2Y_7]$, then Y_1 and Y_2 will be regarded as non-siblings, and hence the edge between Y_1 and Y_2 will be *missing* from the sibling graph.

Those discussions point to the middle ground between parsimonious and generous quartet selection. One natural way to explore this middle ground is to use several standing members instead of one.

5.3 The Algorithm

Figure 4 shows an algorithm for discovering shallow latent variables, namely DiscoverSLVs. The algorithm first calls a subroutine ConstructSGraph to construct a sibling graph, and then finds all the connected components of the graph. It is understood that one latent variable is introduced for each of the connected components.

The subroutine ConstructSGraph starts from the complete graph. For each pair of manifest variables U and V, it considers all the quartets that involve U, V, and one of the m standing members T_i $(i = 1, 2, \ldots, m)$. QSL is called to learn a submodel structure for each of the quartets. If U and V are not siblings in one of the quartet substructures, the edge between U and V is deleted.

DiscoverSLVs has error tolerance mechanism naturally built in. This is mainly because it regards connected components in sibling graph as sibling clusters. Let $\mathbf{C}$ be a sibling cluster in the generative model. The vertices in $\mathbf{C}$ will be placed in one cluster by DiscoverSLVs if they are in the same connected component in the sibling graph G produced by ConstructSGraph. It is not required for variables in $\mathbf{C}$ to be pairwise connected in G. Therefore, a few mistakes by ConstructSGraph when determining sibling relationships are tolerated. When the set $\mathbf{C}$ is not very small, it takes a few or more mistakes in the right combination to break up $\mathbf{C}$.

Algorithm DiscoverSLVs($\mathbf{D}, m$):

1. $G \leftarrow$ ConstructSGraph($\mathbf{D}, m$).
2. **return**
   the list of the connected components of G.

Algorithm ConstructSGraph($\mathbf{D}, m$):

1. $G \leftarrow$ complete graph over manifest nodes $\mathbf{Y}$.
2. **for** each edge (U, V) of G,
3. pick $\{T_1, \cdots, T_m\} \subseteq \mathbf{Y} \backslash \{U, V\}$
4. **for** each $\mathbf{Q} \in \cup_{i=1}^{m} \mathcal{Q}_{UV|T_i}$,
5. **if** QSL($\mathbf{D}, \mathbf{Q}$)=$[U * |V *]$
6. delete edge (U, V) from G, break.
7. **endFor**.
8. **endFor**.
9. **return** G.

Figure 4: An algorithm for learning SLVs.

6 Empirical Evaluation

We have carried out simulation experiments to evaluate the ability of DiscoverSLVs in discovering latent variables. This section describes the setup of the experiments and reports our findings.

The generative models in the experiments share the same structure. The structure consists of 39 manifest variables and 13 latent variables. The latent variables form a complete 3-ary tree of height two. Each latent variable in the structure is connected to 3 manifest variables. Hence all latent variables are shallow. The cardinalities of all variables were set at 3.

We created 10 generative models from the structure by randomly assigning parameter values. From each of the 10 generative models, we sampled 5 data sets of 500, 1,000, 2,500, 5,000 and 10,000 records. DiscoverSLVs was run on each of the data sets three times, with the number of standing members m set at 1, 3 and 5 respectively. The algorithms were implemented in Java and all experiments were run on a Pentium 4 PC with a clock rate of 3.2 GHz.

The performance statistics are summarized in Table 2. They consist of errors at three different

Table 2: Performance statistics of our SLV discovery algorithm.

	QSL-level			Edge-level		SLV-level		
m	D2F	D2D	F2D	ME	FE	LO	LCO	MC
Sample size = 500								
1	77.0%(4.5%)	0.06%(0.05%)	0.40%(0.13%)	1.0(1.1)	88.2(9.2)	11(3.0)	0(0)	0(0)
3	71.8%(4.0%)	0.05%(0.03%)	0.36%(0.14%)	2.9(1.3)	28.0(8.3)	9.3(3.1)	0(0)	0.1(0.3)
5	68.8%(3.0%)	0.05%(0.02%)	0.30%(0.14%)	3.3(1.6)	14.7(6.0)	5.1(1.9)	0.2(0.4)	0(0)
Sample size = 1000								
1	65.3%(6.2%)	0.01%(0.03%)	0.17%(0.14%)	0.3(0.6)	55.3(15.7)	11.2(0.7)	0(0)	0(0)
3	58.3%(2.7%)	0.02%(0.02%)	0.10%(0.07%)	0.7(0.9)	10.1(6.5)	4.8(2.9)	0(0)	0(0)
5	56.6%(3.2%)	0.01%(0.01%)	0.10%(0.08%)	1.1(0.7)	6.2(3.8)	2.3(1.3)	0.1(0.3)	0(0)
Sample size = 2500								
1	50.1%(2.3%)	0.00%(0.00%)	0.04%(0.07%)	0.2(0.4)	20.4(8.9)	8.6(2.3)	0(0)	0(0)
3	38.0%(4.6%)	0.00%(0.00%)	0.02%(0.04)%	0.2(0.4)	3.1(3.7)	1.4(1.4)	0(0)	0(0)
5	37.3%(3.8%)	0.00%(0.01%)	0.07%(0.08)%	1.1(1.4)	1.3(2.5)	1.5(0.9)	0.1(0.3)	0(0)
Sample size = 5000								
1	32.6%(5.5%)	0.00%(0.00%)	0.03%(0.09%)	0(0)	7.7(6.0)	3.9(2.4)	0(0)	0(0)
3	25.2%(4.4%)	0.01%(0.01%)	0.04%(0.06%)	0.3(0.5)	1.5(2.7)	0.5(0.7)	0(0)	0(0)
5	25.7%(3.9%)	0.00%(0.01%)	0.10%(0.11%)	0.9(0.9)	0.9(2.7)	0.1(0.3)	0(0)	0(0)
Sample size = 10000								
1	21.8%(5.9%)	0.00%(0.00%)	0.02%(0.07%)	0(0)	2.0(3.3)	1.2(1.8)	0(0)	0(0)
3	17.5%(3.9%)	0.00%(0.00%)	0.05%(0.06%)	0.2(0.4)	0.4(1.2)	0.1(0.3)	0(0)	0(0)
5	17.4%(4.1%)	0.00%(0.01%)	0.03%(0.04%)	0.6(0.8)	0.1(0.3)	0.1(0.3)	0(0)	0(0)

levels of the algorithm: the errors made when learning quartet substructures (QSL-level), the errors made when determining sibling relationships between manifest variables (edge-level), and the errors made when introducing SLVs (SLV-level). Each number in the table is an average over the 10 generative models. The corresponding standard deviations are given in parentheses.

QSL-level errors: We see that the probabilities of the QSL-level errors are significantly smaller than those reported in Section 4. This is because we deal with strong dependency models here, while the numbers in Section 4 are about general models. This indicates that strong dependency assumption does make learning easier. On the other hand, the trends remain the same: the probability of D2F errors is large, that of F2D errors is small, and that of D2D errors are very small. Moreover, the probability of D2F errors decreases with sample size.

Edge-level errors: For the edge-level, the numbers of missing edges (ME) and the number of fake edges (FE) are reported. We see that the number of missing edges is always small, and in general it increases with the number of standing members m. This is expected since the larger m is, the more quartets one examines, and hence the more likely one makes F2D errors.

The number of fake edges is large when the sample size is small and m is small. In general, it decreases with sample size and m. It dropped to 1.3 when for the case of sample size 2,500 and $m = 5$. This is also expected. As m increases, the number of quartets examined also increases. For two manifest variables U and V that are not siblings in the generative model, the probability of obtaining a dogbone (despite D2F errors) where U and V are not siblings also increases. The number of fake edges decreases with m because as m increases, the probability of D2F errors decreases.

SLV-level errors: We now turn to SLV-level errors. Because there were not many missing edges, true sibling clusters of the generative models were almost never broken up. There are only five exceptions. The first exception happened for one generative model in the case of sample size 500 and $m = 3$. In that case, a manifest variable from one true cluster was placed into another, resulting in one misclustering error (MC).

The other four exceptions happened for the following combinations of sample size and m: (500, 5), (1000, 5), (2500, 5). In those cases, one true sibling cluster was broken into two clusters, resulting in four latent commission errors (LCO).

Fake edges cause clusters to merge and hence lead to latent omission errors. In our experiments, the true clusters were almost never broken up. Hence a good way to measure latent omission errors is to use the total number of shallow latent variables, i.e. 13, minus the number of clusters returned by `DiscoverSLVs`. We call this the number of LO errors. In Table 2,we

see that the number of LO errors decreases with sample size and the number of standing members m. It dropped to 1.4 when for the case of sample size 2,500 and $m = 3$. When the sample size was increased to 10,000 and m set to 3 or 5, LO errors occurred only once for one of the 10 generative models.

Running time: The running times of `DiscoverSLVs` are summarized in the following table (in hours). For the sake of comparison, we also include the times HSHC took attempting to reconstruct the generative models based on the same data as used by `DiscoverSLVs`. We see that `DiscoverSLVs` took only a small fraction of the time that HSHC took, especially in the cases with large samples. This indicates that the two-stage approach that we are exploring can result algorithms significantly more efficient than HSHC.

RunningTime(hrs)	500	1000	2500	5000	10000
m=1	1.05	1.06	0.92	0.89	0.84
m=3	1.57	1.52	1.49	1.79	1.91
m=5	1.85	2.07	2.17	2.72	3.02
HSHC	4.65	8.69	23.7	43.0	118.6

The numbers of quartets examined by `DiscoverSLVs` are summarized in the following table. We see that `DiscoverSLVs` examined only a small fraction of all the C_{39}^4=82251 possible quartets. We also see that doubling m does not imply doubling the number of quartets examined, nor doubling the running time. Moreover, the number of quartets examined decreases with sample size. This is because the probability of D2F errors decrease with sample size.

	500	1000	2500	5000	10000
m=1	5552	4191	2861	2131	1816
m=3	8326	5939	4659	4292	4112
m=5	9801	8178	6776	6536	6496
Total: 82251					

7 Related Work

Linear latent variable graphs (LLVGs) are a special class of structural equation models. Variables in such models are continuous. Some are observed, while others are latent. Sliva *et al.* (2003) has studied the problem of identifying SLVs in LLVGs. Their approach is based on the Tetrad constraints (Spirtes *et al.*, 2000), and its complexity is $O(n^6)$.

Phylogenetic trees (PT) (St. John *et al.*, 2003) can be viewed as a special class of HLC models. The quartet-based method for PT reconstruction first learn submodels for all C_n^4 possible quartets and then use them to build the overall tree (St. John *et al.*, 2003).

8 Conclusions

We have empirically studied the probability of making errors when learning quartet HLC models and have observed some interesting regularities. In particular, we observed the probability of D2F errors is high and decreases with sample size, while the probability of F2D errors is low. Based on those observations, we have developed an algorithm for discovering shallow latent variables in HLC models reliably and efficiently.

Acknowledgements

Research on this work was supported by Hong Kong Grants Council Grant #622105.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automotic Control,*.

Cheeseman, P. and Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. In *Advances in knowledge discovery and data mining.*

Chen, T. and Zhang, N.L. (2006). Quartet-Based Learning of HLC Models: Discovery of Shallow Latent Variables. In *AIMath-06.*

Kočka, T. and Zhang, N.L. (2002). Dimension correction for hierarchical latent class models. In *UAI-02.*

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.*

Schwarz, G. (1978). Estimating the dimension of a model. *Annuals of Statistics.*

Silva, R., Scheines, R., Glymour, C. and Spirtes, P. (2003). Learning measurement models for unobserved variables. In *UAI-03.*

Spirtes, P., Glymour, C. and Scheines, R. (2000). *Causation, Prediction and Search.*

St. John, K., Warnow, T., Moret, B.M.E. and Vawter, L. (2003) Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining. *Journal of Algorithms.*

van der Putten, P. and van Someren, M. (2004). A Bias-Variance Analysis of a Real World Learning Problem: The CoIL Challenge 2000. *Machine Learning.*

Zhang, N.L. (2004). Hierarchical latent class models for cluster analysis. *JMLR.*

Zhang, N.L. and Kočka, T. (2004). Efficient learning of hierarchical latent class models. In *ICTAI-04.*