Hierarchical Latent Class Models for Cluster Analysis

Nevin L. Zhang *
Department of Computer Science
Hong Kong University of Science and Technology
lzhang@cs.ust.hk

Abstract

Latent class models are used for cluster analysis of categorical data. Underlying such a model is the assumption that the observed variables are mutually independent given the class variable. A serious problem with the use of latent class models, known as local dependence, is that this assumption is often untrue. In this paper we propose hierarchical latent class models as a framework where the local dependence problem can be addressed in a principled manner. We develop a search-based algorithm for learning hierarchical latent class models from data. The algorithm is evaluated using both synthetic and real-world data.

Keywords: Model-based clustering, latent class models, local dependence, Bayesian networks, learning.

Introduction

Cluster analysis is the partitioning of similar objects into meaningful classes, when both the number of classes and the composition of the classes are to be determined (Kaufman and Rousseeuw 1990; Everitt 1993). In model-based clustering, it is assumed that the objects under study are generated by a mixture of probability distributions, with one component corresponding to each class. When the attributes of objects are continuous, cluster analysis is sometimes called *latent profile analysis* (Gibson 1959; Lazarsfeld and Henry 1968; Bartholomew and Knott 1999; Vermunt and Magidson 2002). When the attributes are categorical, cluster analysis is sometimes called *latent class analysis (LCA)* (Lazarsfeld and Henry 1968; Goodman 1974b; Bartholomew and Knott 1999; Uebersax 2001). There is also cluster analysis of *mixed-mode data* (Everitt 1993) where some attributes are continuous while others are categorical.

This paper is concerned with LCA, where data are assumed to be generated by a *latent class (LC)* model. An LC model consists of a class variable that represents the clusters to be identified and a number of other variables that represent attributes of objects [1]. The class variable is not observed and hence said to be *latent*. On the other hand, the attributes are observed and are called *manifest variables*.

LC models assume *local independence*, i.e. manifest variables are mutually independent in each latent class, or equivalently, given the latent variable. A serious problem with the use of LCA, known as *local dependence*, is that this assumption is often violated. If one does not deal with local dependence explicitly, one implicitly attributes it to the latent variable. This can lead to spurious latent classes and poor model fit. It can also degenerate the accuracy of classification because locally dependent manifest variables contain overlapping information (Vermunt and Magidson 2002).

The local dependence problem has attracted some attention in the LCA literature (Espeland & Handelman 1989; Garrett & Zeger 2000; Hagenaars 1988; Vermunt & Magidson 2000). Methods for detecting and modeling local dependence have been proposed. To detect local dependence, one typically compares observed and expected cross-classification frequencies for pairs of manifest variables. To model local dependence, one can join manifest variables, introduce multiple latent variables, or reformulate LC models as loglinear models and then impose constraints on them. All existing methods are preliminary proposals and suffer from a number of deficiencies (Zhang 2002).

Our work

This paper describes the first systematic approach to the problem of local dependence. We address the problem in the framework of *hierarchical latent class (HLC) models*. HLC models are Bayesian networks whose structures are rooted trees and where the leaf nodes are observed while all other nodes are latent. This class of models is chosen for two reasons. First it is significantly larger than the class of LC models and can accommodate local dependence. Second inference in an HLC model takes time linear in model size, which makes it computationally feasible to run EM.

We develop a search-based algorithm for learning HLC models from data. The algorithm systematically searches for the optimal model by hill-climbing in a space of HLC models with the guidance of a model selection criterion. When

*This work is competed while the author is on leave at Department of Computer Science, Aalborg University, Denmark.

[1] Latent class models are sometimes also referred to as Naive Bayes models. We suggest that the term "naive Bayes models" be used only in the context of classification and the term "latent class models" be used in the context of clustering.

there is no local dependence, the algorithm returns an LC model. When local dependence is present, it returns an HLC model where local dependence is appropriately modeled. It should be noted, however, that the algorithm might not work well on data generated by models that neither are HLC models nor can be closely approximated by HLC models.

The motivation for this work originates from an application in traditional Chinese medicine. In the application, model quality is of utmost importance and it is reasonable to assume abundant data and computing resources. So we take a principled (as opposite to heuristic) approach when designing our algorithm and we empirically show that the algorithm yields models of good quality. In subsequent work, we will explore ways to scale up the algorithm.

Related literature

This paper is an addition to the growing literature on hidden variable discovery in Bayesian networks (BN). Here is a brief discussion of some of this literature. Elidan *et al.* (2001) discuss how to introduce latent variables to BNs constructed for observed variables by BN structure learning algorithms. The idea is to look for structural signatures of latent variables. Elidan and Friedman (2001) give a fast algorithm for determining the cardinalities — the numbers of possible states — of latent variables introduced this way. Meila-Predoviciu (1999) studies how the so-called tree H models can be induced from data, where a tree H model is basically an LC model with each observed variable replaced by an undirected tree of observed variables. This work is based on the method of approximating joint probability distributions with dependence trees by Chow and Liu (1968).

The algorithms described in Connolly (1993) and Martin and VanLehn (1994) are closely related the algorithm presented in this paper. They all aim at inducing from data a latent structure that explains correlations among observed variables. The algorithm by Martin and VanLehn (1994) builds a two-level Bayesian network where the lower level consists of observed variables while the upper level consists of latent variables. The algorithm is based on tests of association between pairs of observed variables. The algorithm by Connolly (1993) constructs exactly what we call HLC models. Mutual information is used to group variables, a latent variable is introduced for each group, and the cardinality of the latent variable is determined using a technique called conceptual clustering. In comparison with Connolly's method, our method is more principled in the sense that it determines model structure and cardinalities of latent variables using one criterion, namely (some approximation) of the marginal likelihood.

The task of learning HLC models is similar to the reconstruction of phylogenetic trees, which is a major topic in biological sequence analysis (Durbin *et al*. 1998). As a matter of fact, phylogenetic trees are special HLC models where the model structures are binary (bifurcating) trees and all the variables share the same set of possible states. However, phylogenetic trees cannot be directly used for general cluster analysis because the constraints imposed on them. And techniques for phylogenetic tree reconstruction do not necessarily cover over to HLC models. For example, the structural EM algorithm for phylogenetic tree reconstruction by Friedman *et al*. (2002) does not work for HLC models because we do not know, a priori, the number of latent variables and their cardinalities.

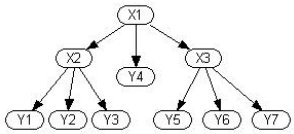

Figure 1: An example HLC model. The X_i's are latent variables and the Y_j's are manifest variables.

HLC models should not be confused with model-based hierarchical clustering (e.g. Hanson *et al*. 1991, Fraley 1998). In an LC model (or similar models with continuous manifest variables), there is only one latent variable and each state of the variable corresponds to a class. HLC models generalize LC models by allowing multiple latent variables. An HLC model contains a hierarchy of latent variables; In model-based hierarchical clustering, on the other hand, one has a hierarchy of classes. Conceptually there is only one latent variable. Classes at different levels of the hierarchy correspond to states of the variable at different levels of granularity.

Hierarchical latent class models

A *hierarchical latent class (HLC) model* is a Bayesian network where

1. The network structure is a rooted tree; and
2. The variables at the leaf nodes are observed and all the other variables are not [2].

Figure 1 shows an example of an HLC model. Following the LCA literature, we refer to the observed variables as *manifest variables* and all the other variables as *latent variables*. In this paper we do not distinguish between variables and nodes. So we sometimes speak also of *manifest nodes* and *latent nodes*. For technical convenience, we assume that there are at least two manifest variables.

We use θ to refer to the collection of parameters in an HLC model M and use m to refer to what is left when the parameters are removed from M. So we usually write an HLC model as a pair $M = (m, \theta)$. We sometimes refer to the first component m of the pair also as an HLC model. When it is necessary to distinguish between m and the pair (m, θ), we call m an *unparameterized HLC model* and the pair a *parameterized HLC model*. The term *HLC model structure* is reserved for what is left if information about cardinalities of latent variables are removed from an unparameterized model m. Model structures will be denoted by the letter S, possibly with subscripts.

[2] The concept of a variable being observed is always w.r.t some given data set. A variable is *observed* in a data set if there is at least one record that contains the state for that variable.

Parsimonious HLC models

In this paper we study the learning of HLC models. We assume that there is a collection of identical and independently distributed (i.i.d.) samples generated by some HLC model. Each sample consists of states for all or some of the manifest variables. The task is to reconstruct the HLC model from data. As will be seen later, not all HLC models can be reconstructed from data. It is hence natural to ask what models can be reconstructed. In this subsection we provide a partial answer to this question.

Consider two parameterized HLC models $M = (m, \theta)$ and $M' = (m', \theta')$ that share the same manifest variables Y_1, Y_2, ..., Y_n. We say that M and M' are *marginally equivalent* if the probability distribution over the manifest variables is the same in both models, i.e.

$$P(Y_1, \ldots, Y_n|m, \theta) = P(Y_1, \ldots, Y_n|m', \theta'). \quad (1)$$

Two marginally equivalent parameterized models are *equivalent* if they also have the same number of independent parameters. Two unparameterized HLC models m and m' are *equivalent* if for any parameterization θ of m there exists a parameterization θ' of m' such that (m, θ) and (m, θ') are equivalent and vice versa. Two HLC model structures S_1 and S_2 are *equivalent* if there are equivalent unparameterized models m_1 and m_2 whose underlying structures are S_1 and S_2 respectively.

A parameterized HLC model M is *parsimonious* if there does not exist another model M' that is marginally equivalent to M and that has fewer independent parameters than M. An unparameterized HLC model m is *parsimonious* if there exists a parameterization θ of m such that (m, θ) is parsimonious.

Let M be a parameterized HLC model and D be a set of i.i.d. samples generated by M. If M is not parsimonious, then there must exist another HLC model whose penalized loglikelihood score given D (Green 1998, Lanterman 2001) is greater than that of M. This means that, if one uses penalized loglikelihood for model selection, one would prefer this other parsimonious models over the non-parsimonious model M. The following theorem states that, to some extent, the opposite is also true, i.e. one would prefer M to other models if M is parsimonious.

Theorem 1 *Let M and M' be two parameterized HLC models with the same manifest variables. Let D be a set of i.i.d. samples generated from M.*

1. *If M and M' are not marginally equivalent, then the loglikelihood $l(M|D)$ of M is strictly greater than the loglikelihood $l(M'|D)$ of M' when the sample size is large enough.*
2. *If M is parsimonious and is not equivalent to M', then the penalized loglikelihood of M is strictly larger than that of M' when the sample size is large enough.*

The proofs of all theorems and lemmas in this paper can be found in Zhang (2002).

Model equivalence

In this subsection we give an operational characterization of model equivalence. Let X_1 be the root of a parameterized HLC model M_1. Suppose X_2 is a child of X_1 and it is a latent node (see Figure 2). Define another HLC model M_2 by reversing the arrow $X_1 \rightarrow X_2$ and, while leaving the values for all other parameters unchanged, defining for $P_{M_2}(X_2)$ and $P_{M_2}(X_1|X_2)$ as follows:

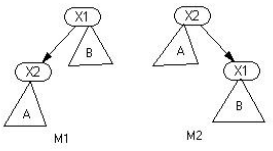

Figure 2: The operation of root walking.

$$P_{M_2}(X_2) = \sum_{X_1} P_{M_1}(X_1, X_2)$$

$$P_{M_2}(X_1|X_2) = \begin{cases} \frac{P_{M_1}(X_1, X_2)}{P_{M_2}(X_2)} & \text{if } P_{M_2}(X_2) > 0 \\ \frac{1}{|X_1|} & \text{otherwise,} \end{cases}$$

where $P_{M_1}(X_1, X_2) = P_{M_1}(X_1) P_{M_1}(X_2|X_1)$. We use the term *root walking* to refer to the process of obtaining M_2 from M_1. In the process, the root has walked from X_1 to X_2.

Theorem 2 [3] *Let M_1 and M_2 be two parameterized HLC models. If M_2 is obtained from M_1 by one or more steps of root walking, then M_1 and M_2 are equivalent.*

The two HLC models shown in Figure 3 are equivalent to the model in Figure 1. The model on the left is obtained by letting the root of the original model walk from X_1 to X_2, while the model on the right is obtained by letting the root walk from X_1 to X_3.

In general, the root of an HLC model can walk to any latent node. This implies the root node cannot be determined from data [4]. A question about the suitability of HLC models for cluster analysis naturally arises. We take the position that the root node can be determined from the objective in clustering and domain knowledge. Moreover we view the presence of multiple latent variables an advantage because it enables one to cluster data in multiple ways. Note that multiple clusterings due to multiple latent variables are very different from multiple clusterings in hierarchical clustering. In the latter case, a clustering at a lower level of the hierarchy is a refinement of a clustering at a higher level. The same relationship does not exist in the former case.

The inability of determining the root node from data also have some technical consequences. We can never induce HLC models from data. Instead we obtain what might be called unrooted HLC models. An *unrooted HLC model* is an

[3] A similar but different theorem was proved by Chickering (1996) for Bayesian networks with no latent variables. In Chickering (1996), model equivalence implies equal number of parameters. Here equal number of parameters is part of the definition of model equivalence.

[4] In the case of phylogenetic trees, this is a well-known fact (Durbin *et al*. 1998).

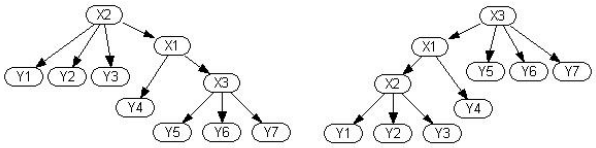

Figure 3: HLC models that are equivalent to the one in Figure 1.

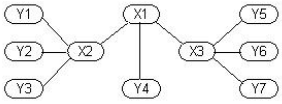

Figure 4: The unrooted HLC model that corresponds to the HLC model in Figure 1.

HLC model with all directions on the edges dropped. Figure 4 shows the unrooted HLC model that corresponds to the HLC model in Figure 1. An unrooted HLC model represents a class of HLC models; members of the class are obtained by rooting the model at various nodes and by directing the edges away from the root. Semantically it is a Markov random field on an undirected tree. The leaf nodes are observed while the interior nodes are latent. The concepts of marginal equivalence, equivalence, and parsimony can be defined for unrooted HLC models in the same way as for rooted models.

From now on when we speak of HLC models we always mean unrooted HLC models unless it is explicitly stated otherwise.

Regular HLC models

In this subsection we first introduce the concept of regular HLC models and show that all parsimonious models must regular. We then show that the set of unparameterized regular HLC models for a given set of manifest variables is finite. This provides a search space for the learning algorithm to be developed in the next section.

For any variable X, use Ω_X and $|X|$ to denote its domain and cardinality respectively. For a latent variable Z in an HLC model, enumerate its neighbors as $X_1, X_2, \ldots, X_k$. An HLC model is *regular* if for any latent variable Z,

$$|Z| \leq \frac{\prod_{i=1}^{k} |X_i|}{\max_{i=1}^{k} |X_i|}, \tag{2}$$

and when Z has only two neighbors,

$$|Z| \leq \frac{|X_1||X_2|}{|X_1| + |X_2| - 1}. \tag{3}$$

Note that this definition applies to parameterized as well as unparameterized models. The first condition was suggested by Kocka and Zhang (2002).

Theorem 3 *Irregular HLC models are not parsimonious*

In the rest of this section, we give three lemmas and one theorem that exhibit several interesting properties of regular HLC models.

A latent node in an HLC model has at least two neighbors. A *singly connected* latent node is one that has exactly two neighbors.

Lemma 1 *In a regular HLC model, no two singly connected latent nodes can be neighbors.*

This lemma inspires the following two definitions. We say that an HLC model structure is *regular* if no two singly connected latent nodes are neighbors. If there are no singly connected latent nodes at all, we say that the model structure is *strictly regular*.

Lemma 2 *Let S be an HLC model structure with n manifest variables. If S is regular, then there are fewer than $3n$ latent nodes. If S is strictly regular, then there are fewer than n latent nodes.*

Lemma 3 *There are fewer than 2^{3n^2} different regular HLC model structures for a given set of n manifest nodes.*

Theorem 4 *The set of all regular unparameterized HLC models for a given set of manifest variables is finite.*

Searching for optimal models

In this section we present a hill-climbing algorithm for learning HLC models. Hill-climbing requires a scoring metric for comparing candidate models. In this work we experiment with four existing scoring metrics, namely AIC (Akaike 1974), BIC (Schwarz 1978), the Cheeseman-Stutz (CS) score (Cheeseman and Stutz 1995), and the holdout logarithmic score (LS) (Cowell *et al*. 1999) .

Hill-climbing also requires the specification of a search space and search operators. According to Theorem 3, a natural search space for our task is the set of all regular (unparameterized) HLC models for the set of manifest variables that appear in data. By Theorem 4, we know that this space is finite.

Instead of searching this space directly, we structure the space into two levels according to the following two subtasks and we search those two levels separately:

1. Given a model structure, find optimal cardinalities for the latent variables.
2. Find an optimal model structure.

This search space restructuring is motivated by the fact that natural search operators exist for each of the two levels, while operators for the flat space are less obvious.

Estimating cardinalities of latent variables

The search space for the first subtask consists of all the regular models with the given model structure. To hill-climb in this space we start with the model where the cardinalities of all the latent variables are the minimum. In most cases, the minimum cardinality for a latent variable is 2. For a latent variable next to a singly connected latent node, however, the minimum possible cardinality is 4 because of the inequality (3). At each step, we modify the current model to get a number of new models. The operator for modifying a model is to increase the cardinality of a latent variable by one. Irregular new models are discarded. We then evaluate each of the new

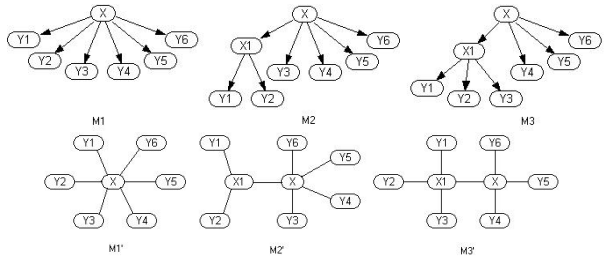

Figure 5: Illustration of structural search operators.

models and picks the best one to seed the next search step. To evaluate a model, one needs to estimate its parameters. We use the EM algorithm for this task.

Search for optimal model structures

The search space for the subtask of finding an optimal model structure consists of all the regular HLC model structures for the given manifest variables. To search this space, we start with the simplest HLC model structure, namely the LC model structure (viewed as an unrooted HLC model structure). At each step, we modify the current structure to construct a number of new structures. The new structures are then evaluated and the best structure is selected as the starting point for the next step. To evaluate a model structure, one needs to estimate the cardinalities of its latent variables. This issue is addressed in subtask 1.

We use three search operators to modify model structures, namely node-introduction, node-elimination, and neighbor-relocation.

Node introduction To motivate the node-introduction operator, we need to go back to rooted models. Consider the rooted HLC model M_1 shown in Figure 5. Suppose variables Y_1 and Y_2 are locally dependent. A nature way to model this local dependence is to introduce a new parent for Y_1 and Y_2, as shown in M_2.

When translated to unrooted model structures, the new parent introduction operator becomes the *node-introduction* operator. Let X be a latent node in an unrooted model structure. Suppose X has more than two neighbors. Then for any two neighbors of X, say Z_1 and Z_2, we can introduce a new latent node Z to separate X from Z_1 and Z_2. Afterwards, X is no longer connected to Z_1 and Z_2. Instead X is connected to Z and Z is connected to Z_1 and Z_2. To see an example, consider the model structure M_1' in Figure 5. Introducing a new latent node X_1 to separate X from Y_1 and Y_2 results in the model structure M_2'.

In the case of rooted model structures, we do not consider introducing new parents for groups of three or more nodes for the sake of computational efficiency. This constraint implies that the model M_3 in Figure 5 cannot be reached from M_1 in one step. In the case of unrooted model structures, we do not allow the introduction of a new node to separate a latent node from three or more of its neighbors. This implies that we cannot reach M_3' from in Figure 5 cannot M_1' in one step.

Node-introduction is not allowed when it results in irregular model structures. This means that we cannot introduce a new node to separate a latent node X from two of its neighbors if it has only one other neighbor and that neighbor is a singly connected latent node. Moreover, we cannot introduce a new node to separate a singly connected latent node from its two neighbors. [5]

Node elimination The opposite of node-introduction is *node-elimination*. We notice that a newly introduced node has exactly three neighbors. Consequently we allow a latent node be eliminated only when it has three neighbors. Of course, node elimination cannot be applied if there is only one latent node.

Neighbor relocation In Figure 5, we cannot reach M_3' from either M_1' or M_2' using node-introduction and node-elimination. To overcome this difficult, we introduce the third search operator, namely *neighbor-relocation*. Suppose a latent node X has a neighbor Z that is also a latent node. Then we can relocate any of the other neighbors Z' of X to Z, which means to disconnect Z' from X and reconnect it to Z. To see an example, consider the model structure M_2' in Figure 5. If we relocate the neighbor Y_3 of X to X_1, we reach M_3'.

For the sake of computational efficiency, we do not allow neighbor relocation between two non-neighboring latent nodes. In Figure 4, for example, we cannot relocate neighbors of X_2 to X_3 and vice versa. Moreover neighbor relocation is not allowed when it results in irregular model structures. To be more specific, suppose X is a latent node that has a latent node neighbor Z. We cannot relocate another neighbor Z' of X to Z if X has only three neighbors and the third neighbor is a singly connected latent node. The relocation is not allowed, of course, if X has only two neighbors. Finally note that the effects of any particular neighbor relocation can always be undone by another application of the operator. [6]

Theorem 5 *Consider the collection of regular HLC model structures for a given set of manifest variables. One can go between any two structures in the collection using node-introduction, node-elimination, and neighbor-relocation.*

Empirical Results on Synthetic Data

Our algorithm for learning HLC models has been evaluated on both synthetic and real-world data. This section reports the results on synthetic data. The synthetic data were generated using the HLC model structure in Figure 1. The car-

[5] Node-introduction is similar to an operator that PROMTL, a system for inferring phylogenetic trees, uses to search for optimal tree topologies via star decomposition (Kishino *et al.* 1990). The former is slightly less constrained than the latter in that it is allowed to create singly connected nodes as by-products.

[6] Neighbor relocation is related to but significantly different than an operator called branch swapping that PAUP, a system for inferring phylogenetic trees, uses to search for optimal tree topologies (Swofford 1998). The latter includes what are called nearest neighbor interchange; subtree pruning and regrafting; and tree bisection/reconnection .

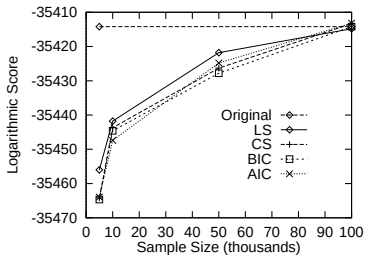

Figure 6: Logarithmic scores of learned models on testing data.

dinalities of all variables were set at 3. The model was randomly parameterized. Four training sets with 5,000, 10,000, 50,000, and 100,000 records were sampled. A testing set of 5,000 records was also sampled. Each sample record consists of states for all the manifest variables.

We ran our learning algorithm on each of the four training sets, once for each of the four scoring metrics BIC, AIC, CS, and LS. There are hence a total number of 16 settings. For the LS scoring metric, 25% of the training data was set aside and used as validation data. Candidate models are compared using their logarithmic scores on the validation data. The EM termination threshold was set at 0.01 during model selection and at 0.0001 when estimating parameters for the final model. Irrespective of the threshold, EM was allowed to run no more than 200 iterations on any given model. For local maxima avoidance, we used the Chickering and Heckerman (1997) variant of the multiple-restart approach.

The logarithmic scores of the learned models on the testing data are shown in Figure 6. The scores are grouped into four curves according to the four scoring metrics. The score of the original model is also shown for the sake of comparison. We see that the scores of the learned models are quite close to that of the original model in the relative sense. This indicates that those models are as good as the original model when it comes to predicting the testing set. We also see that scores do not vary significantly across the scoring metrics.

The structures of the learned models do depend heavily on the scoring metrics. With either BIC or CS, our algorithm obtained, from the 50k and 100k training sets, models whose structures (unrooted trees) correspond precisely to the original structure. In this sense we say that the correct structure was recovered. From the 5k and 10k training sets, it obtained models with structures that differ from the original structure by only one or two search operations. With AIC and LS, our algorithm also recovered the correct structure from the 50k and 100k training sets. But the structures it produced for the 5k and 10k differ significantly from the original structures. [7]

Although the correct model structure was recovered from the 50k and 100k training sets regardless of the scoring metrics used, different scoring metrics gave different estimates for the cardinalities of the latent variables. As can be seen from Table 1, BIC and CS have the tendency to underestimate while AIC and LS have the tendency to overestimate. Overall, BIC and CS seem to give better estimates.

[7] Structures and other details of the learned models are given in Zhang (2002).

	50k				100k			
	BIC	CS	LS	AIC	BIC	CS	LS	AIC
X_1	2	2	2	2	2	2	2	2
X_2	3	3	3	3	3	3	3	4
X_3	2	2	4	4	3	3	4	4

Table 1: Cardinalities of latent variables in learned models. In the original model all variables have 3 states.

In a second experiment, we parameterized the original model also in random fashion except that we ensured each conditional probability distribution have one component with mass no smaller than 0.6. Everything else was the same as in the first experiment. The performance of our algorithm, with BIC or CS, was better here than in the first experiment. It recovered the correct model structure from all four training sets. Cardinalities of X_2 and X_3 were estimated correctly in all cases. The cardinality of X_3 was estimated correctly in the 100k training set, while it was underestimated by 1 in all other training sets.

When AIC and LS was used, however, the algorithm performed worse in the second experiment than in the first one. It was not able to recover the correct model structure even from the 50k and 100k training sets.

Empirical Results on Real-World Data

We have also evaluated our algorithm on four real-world data sets taken from the LCA literature. The first data set is the Hannover Rheumatoid Arthritis data. It involves five binary manifest variables and consists of 7,162 records. Kohlmann and Formann (1997) conclude that the best model for this data set is a four class LC model. Using scoring metrics BIC, CS, and AIC, our algorithm discovered exactly the same model. When LS was used, however, it computed a very different model which does not fit the data well.

The second data set is known as the Coleman Data (Coleman 1964). It involves four binary manifest variables named A, B, C, and D. There are 3,398 records. This data set has been previously analyzed by Goodman (1974) and Hagenaars (1988). Goodman started with a 2-class LC model and found that it does not fit the data well ($L = 249.50$, $df = 6$, $p < 0.001$). He went on to consider the loglinear model that is represented by the path diagram M1 in Figure 7. In the model, both X_1 and X_2 are binary variables. This model fits data well ($L = 1.27$, $df = 4$, $p = 0.87$). Hagenaars examined several possible models and reached the conclusion that the loglinear model M2, where X is a binary variable, best explains the data. This model fits the data very well ($L = 1.43$, $df = 5$, $p = 0.92$).

Using scoring metric BIC, CS, and AIC, our algorithm found the model M3, where both X_1 and X_2 are binary variables. It's obvious that M3 is equivalent to M1 and hence fit data equally well. Our algorithm does not examine model M2 because it is not an HLC model. Fortunately, M2 is

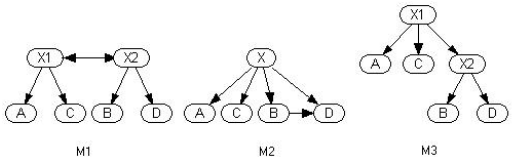

Figure 7: Models for the Coleman data.

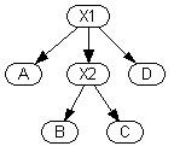

Figure 8: Model for the HIV data.

quite close to M3. Using LS, our algorithm found a model that is the same as M3 except the number of states of X_1 is 3. This model does not fit data well ($L = 1.27$, $df = 0$, $p = 0.0$).

The third data set is known as the HIV data (Alvord *et al.* 1988). It also involves four binary manifest variables named A, B, C, and D. There are 428 records. Alvord *et al.* (1988) reasoned that there should be two latent classes, corresponding to the presence and absence of the HIV virus. However, the two-class LC model does not fit data well ($L = 16.23$, $df = 6$, $p = 0.01$). This indicates the presence of local dependence.

The performance of our algorithm on this data set is similar to that on the Coleman data set. Using BIC, CS, and AIC, it found the model in Figure 8, where both latent variables are binary variables. The model is the same as one of the equivalent models Uebersax (2000) reached using some heuristic techniques. The model fit data well ($L = 3.056$, $df = 4, p = 0.548$). With LS score, our algorithm produced the same model structure. However the cardinalities of both latent variables are overestimated by 2. The model fits data poorly.

The final data set we used is the housing building data (Hagenaars 1988). It involves four binary manifest variables and has 283 records. Hagenaars derived several loglinear models that fit the data well. None of those models are close to any HLC models, indicating that the data probably was not generated by an HLC model or a model that can be approximated closely by an HLC model. As such we expect our algorithm to perform poorly. This turns out to be the case. All the models produced by our algorithm fit data poorly.

Concluding Remarks

We have introduced a new class of models for cluster analysis, namely HLC models. HLC models are significantly more general than LC models and can accommodate local dependence. At the same time, the simplicity in their structures makes computation feasible. A search-based algorithm has been developed for learning HLC models from data. Both synthetic and real-world data have been used to evaluate the performance of the algorithm with four different scoring metrics, namely AIC, BIC, CS, and LS. The results indicate that the algorithm works well with the BIC and CS scores.

The focus of this paper has been on developing a principled search-based method for learning HLC models. Not much consideration was given to computational complexity. While not prohibitive, our algorithm is quite expensive computationally. To reduce complexity one can employ various heuristics to construct initial models for search that hopefully are close to the optimal. One can also replace hill-climbing that generates and evaluates a large number of candidate models at each step with heuristic search that inspects aspects of the current model that need improvement and constructs a model for the next step heuristically.

Acknowledgement

Research was partially supported Hong Kong Research Grants Council under grant HKUST6093/99E.

I thank Tomas Kocka for insightful discussions on parsimonious and regular HLC models. I am also grateful to Finn V. Jensen, Thomas Nielsen, Kristian G. Olesen, Olav Bangso, Jose Pena, Jiri Vomlel, and Marta Vomlelova for valuable feedbacks on earlier versions of this paper.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Autom. Contr.*, 19, 716-723.

Alvord, W. G., Drummond, J. E., Arthur. L.O., Biggar, R. J., Goedert, J. J., Levine, P. H., Murphy, E. L. Jr, Weiss, S. H., Blattner, W. A. (1988). A method for predicting individual HIV infection status in the absence of clinical information. *AIDS Res Hum Retroviruses*, 4(4):295-304.

Bartholomew, D. J. and Knott, M. (1999). *Latent variable models and factor analysis*, 2nd edition. Kendall's Library of Statistics 7. London: Arnold.

Bohrnstedt, G. W. and Knoke D. (1994). *Statistics for social data analysis (3rd Edition)*. F. E. Peacock Publishers Inc., Itasca, Illinois.

Cheeseman, P. and Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. In Fayyad, U., Piatesky-Shaoiro, G., Smyth, P., and Uthurusamy, R. (eds.), *Advancesin Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA.

Chickering, D. M. and Heckerman, D. (1997). Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning* 29(2-3): 181-212.

Chow, C. K. and Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, IT-14(3): 462-467.

Coleman, J. S. (1964). *Introduction to Mathematical Sociology*. London: Free Press.

Connolly, D. (1993). Constructing hidden variables in Bayesian networks via conceptual learning. *ICML-93*, 65-72.

Cover, T. M., Thomas, J. A. (1991). *Elements of Information Theory*, John Wiley & Sons.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic networks and expert systems*, Springer.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. B*, 39:1–38.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press.

Eaton, W. W., Dryman, A., Sorenson, A., and McCutcheon, A. (1989). DSM-III Major depressive disorder in the community: A latent class analysis of data from the NIMH epidemiologic catchment area programme. *British Journal of Psychiatry*, 155, 48-54.

Elidan, G., Lotner, N., Friedman, N. and Koller, D. (2000). Discovering hidden variables: A structure-based approach. *NIPS-01*.

Elidan, G. and N. Friedman (2001). Learning the dimensionality of hidden variables. *UAI-01*.

Espeland, M. A. and Handelman, S. L. (1989). Using latent class models to characterize and assess relative error in discrete measurements. *Biometrics*, 45, 587-599.

Everitt, B. S. (1993). *Cluster Analysis*. London: Edward Arnold.

Fraley, C. (1998). Algorithms for model-based Gaussian hierarchical clustering. *SIAM J. Sci. Comput.*, 20 (1), 270-281.

Friedman, N., Ninio, M., Pe'er, I., and Pupko, T. (2002). A structural EM algorithm for phylogenetic inference. *Journal of Computational Biology*, to appear.

Garrett, E. S. and Zeger, S. L. (2000). Latent class model diagnosis. *Biometrics*, 56, 1055-1067.

Geiger, D., Heckerman, D., King, H., and Meek, C. (1998). Stratified exponential families: Graphical models and model selection. Technical Report MSR-TR-98-31, Microsoft Research.

Gibson, W. A. (1959). Three multivariate models: Factor analysis, latent structure analysis, and latent profile analysis. *Psychometrika*, 24: 229-252.

Goodman, L. A. (1974a). The analysis of systems of qualitative variables when some of the variables are unobservable. Part I-A Modified latent structure approach. *American Journal of Sociology*, 7(5), 1179-1259.

Green, P. (1998). Penalized likelihood. In *Encyclopedia of Statistical Sciences*, Update Volume 2. John Wiley & Sons.

Goodman, L. A. (1974b). Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61, 215-231.

Hagenaars, J. A. (1988). Latent structure models with direct effects between indicators: local dependence models. *Sociological Methods and Research*, 16, 379-405.

Hanson, R., Stutz, J., and Cheeseman, P. (1991). Bayesian classification with correlation and inheritance. In *Proc. 12th. International Joint Conference on A.I. (IJCAI-91)*, 2, 692-698, Morgan Kaufmann.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding groups in data: An introduction to cluster analysis.* New York: John Wiley and Sons, Inc..

Kishino, H., Miyata, T., and Hasegawa, M. (1990). Maximum likelihood inference of protein phylogeny and the origin of the chloroplasts. J. Mol. Evol. 31, 151-160.

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

Kohlmann, T., and Formann, A. K. (1997). Using latent class models to analyze response patterns in epidemiologic mail surveys. Rost, J. and Langeheine, R. (eds.). *Applications of latent trait and latent class models in the social sciences*. Muenster: Waxman Verlag.

Lanterman, A. D. (2001). Schwarz, Wallace, and Rissanen: Intertwining themes in theories of model order estimation. *International Statistical Review*, 69(2), 185-212.

Lazarsfeld, P. F., and Henry, N.W. (1968). *Latent structure analysis*. Boston: Houghton Mifflin.

Martin, J. and VanLehn, K. (1994). Discrete factor analysis: learning hidden variables in Bayesian networks. Technical Report LRGC_ONR-94-1, Department of Computer Science, University of Pittsburgh.

Meila-Predoviciu, M. (1999). *Learning with mixtures of trees*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6(2), 461-464.

Swofford, D. L. (1998). *PAUP* 4.0 - Phylogenetic Analysis Using Parsimony (*and Other Methods)*.. Sinauer Assoc., Sunderland, MA.

Uebersax, J. (2000). A practical guide to local dependence in latent class models. http://ourworld.compuserve.com/homepages/jsuebersax/condep.htm.

Vermunt, J.K. and Magidson, J. (2000). *Latent GOLD User's Guide*. Belmont, Mass.: Statistical Innovations, Inc..

Vermunt, J.K. and Magidson, J. (2002). Latent class cluster analysis. in Hagenaars, J. A. and McCutcheon A. L. (eds.), *Advances in latent class analysis*. Cambridge University Press.

Wasmus, A., Kindel, P., Mattussek, S. and Raspe, H. H. (1989). Activity and severity of rheumatoid arthritis in Hannover/FRG and in one regional referral center. *Scandinavian Journal of Rheumatology*, Suppl. 79, 33-44.

Zhang, N. L. (2002). Hierarchical latent class models for cluster analysis. Technical Report HKUST-CS02-02, Department of Computer Science, Hong Kong University of Science & Technology, Hong Kong, China.