Latent Variable Discovery in Classification Models

Nevin L. Zhang

Department of Computer Science

Hong Kong University of Science and Technology

lzhang@cs.ust.hk

Phone: (852)-2358-7015

Thomas D. Nielsen, Finn V. Jensen

Department of Computer Science

Aalborg University, Aalborg, Denmark

{tdn, fvj}@cs.auc.dk

Phone: +45 9635 9854, +45 9635 8903

Fax: +45 9815 9889

Abstract

The naive Bayes model makes the often unrealistic assumption that the feature variables are mutually independent given the class variable. We interpret a violation of this assumption as an indication of the presence of latent variables, and we show how latent variables can be detected. Latent variable discovery is interesting, especially for medical applications, because it can lead to a better understanding of application domains. It can also improve classification accuracy and boost user confidence in classification models.

Keywords: Naive Bayes model, Bayesian networks, latent variables, learning, scientific discovery.

1 Introduction

In the past decade, there has been much research on learning Bayesian networks from data. One of the main research issues is how to learn latent variables. This topic is very interesting because latent variable detection might lead to better understanding of application domains and even to scientific discovery.

Several methods for learning latent variables from data have also been proposed. The method by [3] constructs a tree-shaped Bayesian network where the leaf nodes are observed while all other variables are latent. 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. The algorithm by [13] 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. [6] discuss how to introduce latent variables to Bayesian networks constructed for observed variables using some Bayesian network structure learning algorithms. The idea is to look for structural signatures of latent variables, and [7] gives a fast algorithm for determining the cardinality of latent variables introduced this way. To determine the conditional probabilities of the latent variables, the standard approach is to apply the EM-algorithm [4]. Alternatively, [12] proposes to use a gradient descent search to avoid the problem of slow convergence with the EM-algorithm.

This paper is concerned with latent variable discovery in the naive Bayes (NB) model. An instance of a naive Bayes model is a simple Bayesian network, which consists of a class variable and a number of feature variables. The feature variables are assumed to be conditionally independent of each other given the class variable. However, this assumption is often unrealistic in practice. In medical diagnosis, for instance, symptoms are rarely independent of each other given the disease. In this paper, we interpret the violation of the conditional independence assumption as an indication of the presence of latent variables, and we develop a method for detecting latent variables. The motivation for this work is that latent variable discovery is especially interesting in medical applications, particularly in traditional Chinese medicine which is one of the first author's

research interests.

Many methods exist for relaxing the conditional independence assumption of the naive Bayes model (see e.g. [11,14,9]). The objective behind all such methods is to improve classification performance. On the other hand, the primary goal in this paper is to discover interesting latent variables rather than to improve classification accuracy. However, the discovery of latent variables will, as a side effect, lead to better models and expectedly better classifiers.

Latent variable discovery cannot be performed without making some assumptions. In this paper we assume that data is generated from what we call hierarchical naive Bayes (HNB) models by "hiding" all variables except the class and feature variables. With this assumption, our task becomes inducing HNB models from data.

The rest of the paper is organized as follows. In Section 2 we define HNB models and deal with some theoretical issues about learning HNB models. In Section 3 we describe a hill-climbing algorithm for inducing HNB models from data. In Section 4 we present empirical results to show that our algorithm can indeed discover interesting latent variables and can result in classifiers that are better than naive Bayes classifiers. Finally, we end the paper with some concluding remarks, and in the appendix we give proofs of the theorems presented in the paper.

2 HNB models

A *hierarchical naive Bayes (HNB)* model is a Bayesian network that specifies how a class variable is related to a number of feature variables. More specifically, an HNB is a tree-shaped Bayesian network where the variable at the root is the class variable, the variables at the leaves are the feature variables, and the variables at the internal nodes are not observed and hence called *latent variables*. HNB models reduce to naive Bayes models when there are no latent variables. Fig. 1 shows an example of an HNB model. In the remainder of this paper we do not distinguish between nodes and variables. Moreover, we consider only variables with a finite numbers of states. For any such variable Z, we use Ω_Z and $|Z|$ to denote the domain space of Z and the size of the Ω_Z

respectively.

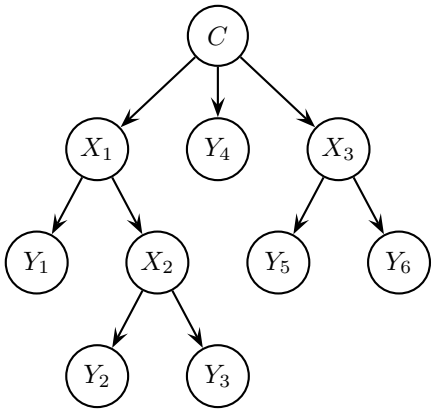

Figure 1: A hierarchical naive Bayes model. The variable at the root is the class variable, the Y_i's are the feature variables, and the X_j's are latent variables.

We use θ to refer to the collection of parameters in an HNB model M and use m to refer to what is left when the parameters are removed from M. So we usually write an HNB model as a pair $M = (m, \theta)$. Sometimes we also refer to the first component m as an HNB model, and the term *HNB model structure* is used when information about the cardinalities of latent variables are removed from m.

2.1 Parsimonious HNB models

In this paper we study learning of HNB models with the objective of discovering latent variables from data. We assume that there is a collection of identical and independently distributed (i.i.d.) samples generated by some HNB model. Each sample consists of states for the class variable and all or some of the feature variables. The task is to reconstruct the HNB model from data.

Evidently, not all HNB models can be reconstructed from data. It is hence natural to ask whether we can characterize the set of HNB models that can be reconstructed from data. To provide a partial answer to this question, we introduce the concept of parsimonious HNB models.

Let $M = (m, \theta)$ be an HNB model with class variable C and feature variables $Y_1, Y_2, \ldots, Y_n$. We say that the model is *parsimonious* if there does not exist another HNB model $M' = (m', \theta')$ with the same class and feature variables such that (1) M' has fewer parameters than M, and (2)

the probability distributions over the class and feature variables are the same in the two models, i.e.

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

Suppose that two HNB models represent the same probability distribution over the observed (class and feature) variables. By Occam's razor, we would always prefer the one that has fewer parameters. In other words, we would never choose non-parsimonious models over parsimonious ones. For this reason, we will restrict our attention only to parsimonious models.

2.2 Regular HNB models

For a latent variable Z in an HNB model, enumerate its neighbors (parent and children) as Z_1, Z_2, ..., Z_k. An HNB model is *regular* if for any latent variable Z,

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

and the strict inequality holds when Z has two neighbors and at least one of them is a latent node.

Theorem 1 *Parsimonious HNB models are regular.*

Theorem 2 *The set of all regular HNB models for a given set of class and feature variables is finite.*

The proofs of these two theorems can be found in the appendix. Theorem 1 tells us that, when searching for a good model for a given data set, we do not need to consider irregular HNB models. We can restrict our search to the space of regular HNB models, which is finite according to Theorem 2. In other words, we do not have to deal with an infinite search space although the models that we work with contain latent variables.

We end this section with a lemma and a related concept that will be useful later. A latent node in an HNB model has at least two neighbors. A *singly connected* latent node is one that has exactly two neighbors.

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

Proof: We see from (1) that the cardinality of a singly connect node is strictly smaller than those of its two neighbors. If two singly connected latent nodes Z_1 and Z_2 were neighbors, then we would have both $|Z_1|>|Z_2|$ and $|Z_1|<|Z_2|$. Therefore two singly connected latent nodes cannot be neighbors. Q.E.D.

Because of this lemma, we say that a HNB model structure is *regular* if no two singly connected latent nodes are neighbors. Note that the structure of a regular model does not necessarily mean that the model itself is regular. However, the structure of a regular model must be a regular structure.

3 Learning HNB models

In this section we present a hill-climbing algorithm for learning regular HNB models from data. Hill-climbing requires a scoring metric for comparing candidate models, and in this work we use the Bayesian information criterion (BIC) [15,10]. If $BIC(m|D)$ denotes the BIC score of a model m given a data set D, then:

$$BIC(m|D) = \log P(D|m, \hat{\theta}) - \frac{d}{2} \log N,$$

where $\hat{\theta}$ is the maximum likelihood estimate of the parameters in m based on data D, d is the number of free parameters in m (the dimension), and N is the size of the data set D.

Hill-climbing also requires the specification of a search space and search operators. A natural search space for our task is the set of all regular HNB models for a given set of class and feature variables. However, instead of searching this space directly, we restructure the space into two levels according to the following two subtasks, and perform the search in these two levels separately:

i. Given a model structure, find an optimal cardinality for the latent variables.

ii. 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 not obvious.

3.1 Learning cardinalities

For a given model structure, S, we obtain different HNB models, that are consistent with S, by specifying different cardinalities for the latent variables in S. Let $\mathcal{M}(S)$ be the set of all regular models that can be obtained this way. The first subtask can now be restated as finding, among all models in $\mathcal{M}(S)$, the model m that maximizes the BIC score, i.e.:

$$m = \arg \max_{m' \in \mathcal{M}(S)} BIC(m'|D). \tag{2}$$

Suppose that the class node C of an HNB model m has k (k>0) children Z_1, Z_2, ..., Z_k. For each i (i=1, 2, ..., k), let m_i be the submodel that consists of the class node C and all nodes in the subtree rooted at Z_i. If m is the HNB model shown in Fig. 1, for instance, the submodels m_i are as shown in Fig. 2. By making use of the fact that the class variable is observed in all data instances, it is easy to see that:

$$\arg \max_{m' \in \mathcal{M}(S)} BIC(m'|D) = \arg \max_{m' \in \mathcal{M}(S)} \left(\sum_{i=1}^{k} BIC(m'_i|D_i) \right), \tag{3}$$

where D_i is the projection of D onto the observed variables of submodel m_i.

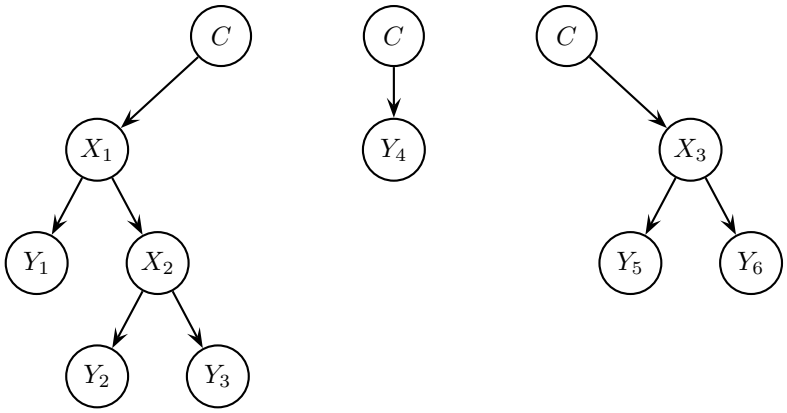

Figure 2: Decomposition of the model in Fig. 1.

Given a regular HNB model structure S, we can define sub-structures S_i (i=1, 2, ..., k) in the same way as the submodels m_i were defined from a model m. Because of Equation (3), task (2) can be divided into the following k subtasks:

$$arg\ max_{m_i \in \mathcal{M}(S_i)} BIC(m_i|D_i) \tag{4}$$

where i=1, 2, ..., k. This decomposition is important because it drastically reduces computational complexity.

We solve each instance of subtask (4) by searching in the space $\mathcal{M}(S_i)$. We start with the model where the cardinalities of all latent variables are minimum. In most cases, the minimum possible cardinality for a latent variable is 2. Due to inequality (1), however, there is an exception. Suppose a latent variable Z has only two neighbors Z_1 and Z_2, which are latent variables themselves. Then Z_1 and Z_2 must have at least three states. 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. Among the new models, those which violate the regularity constraint are discarded. We then evaluate each of the new models and pick the best one to seed the next search step. To evaluate a model, we need to estimate its parameters. We use the EM algorithm for this task [4].

3.2 Learning model structures

The search space for the subtask of finding an optimal model structure consists of all regular HNB model structures for a given set of class and feature variables. To search this space, we start with the simplest HNB model structure, namely the naive Bayes model structure. At each step, we modify the current model to construct a number of new model structures. The new structures are then evaluated and the best structure is selected to seed the next search step. To evaluate a model structure, we need to estimate the cardinalities of its latent variables. This issue was addressed in the previous subsection.

We use three operators to modify the current model structure, namely parent-introduction, parent-alteration, and node-deletion. To motivate the *parent-introduction* operator, consider the feature variables Y_1 and Y_2 in model m_1 of Fig. 3. They are assumed to be conditionally independent given the class variable C. If this is not the case, then there is evidence to believe that, as shown in m_2, there should be a latent variable X_1 that comes between C and the two variables Y_1 and Y_2. Naturally, the model m_2 should be examined as a potentially better model than m_1.

The operation of constructing m_2 from m_1 is parent-introduction; a new parent X_1 is introduced for Y_1 and Y_2.

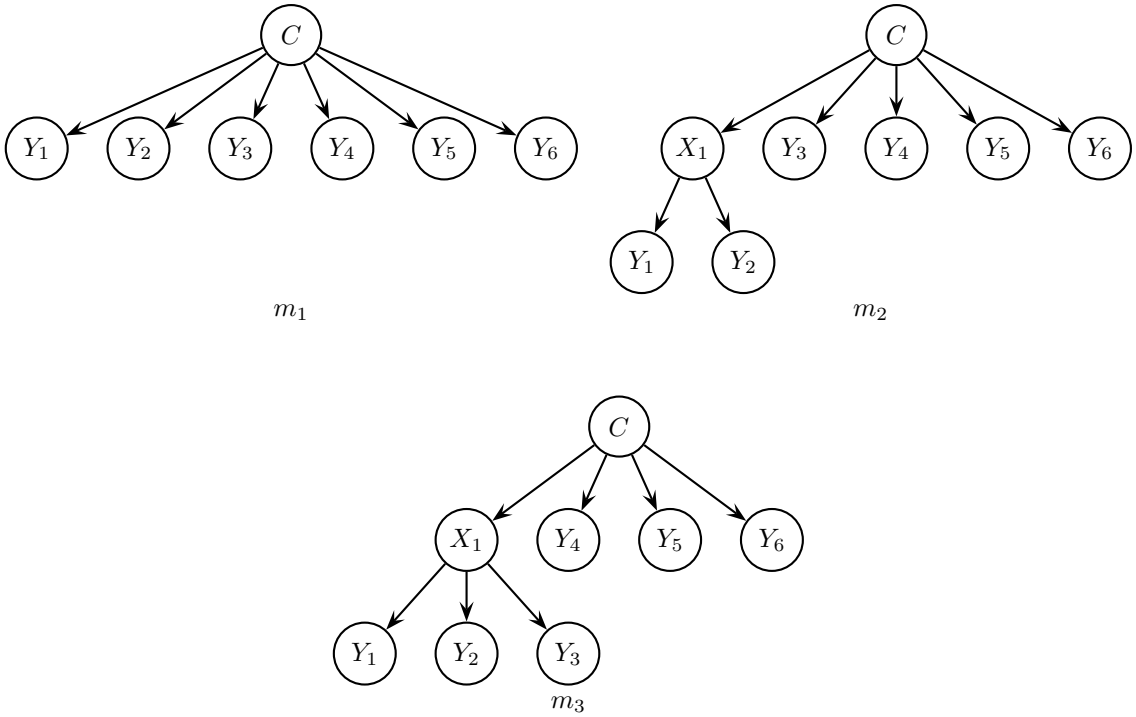

Figure 3: Motivation for structural search operators.

In general, we can introduce a new parent for any two children Z_1 and Z_2 of some node Z. Doing so can hopefully lead to a better model when Z_1 and Z_2 are not independent given Z. However, there is one restriction. We cannot apply the parent-introduction operator if Z has a parent that is latent and singly connected. In this case, the operator would create two neighboring singly connected latent nodes, namely Z and its parent.

One question arises: where should we apply the parent-introduction operation? This question is answered by the search process. For each pair of children of each non-leaf node, a new model is constructed by introducing a new parent for the pair. The list of new models created this way, and by the other operators, are then evaluated one by one. The best one is picked to seed the next search step.

For the sake of computational efficiency, we do not consider introducing new parents for groups

of three or more nodes. This restriction on the parent-introduction operator results in the difficulty of not being able to reach m_3 from either m_1 or m_2. To overcome this problem, we introduce a second operator, namely *parent-alteration*; in m_2, if we change the parent of Y_3 to X_1, we reach m_3.

In general we can change the parent of a node to its grand parent or any of its siblings that are non-leaf nodes. Parent-alteration is not permitted in cases where it yields irregular model structures. To be more specific, let Z_1 be a child of another latent node Z. One cannot alter the parent of Z_1 if Z has only one child because doing so would make the latent node Z a leaf node. Moreover, suppose that Z has only two children, namely Z_1 and Z_2. Then we cannot change the parent of Z_1 to the parent of Z if Z_2 is a singly connected latent node. Similarly we cannot change the parent of Z_1 to Z_2 if the parent of Z is singly connected. In both cases, parent-alteration would create two neighboring singly connected latent nodes.

The opposite of parent-introduction is *node-deletion*. For example, deleting X_1 in m_2 gives us m_1. Just as we have placed some restrictions on parent-introduction, we have some restrictions on node-deletion as well. We can delete a latent node only if it has no more than two children.

Theorem 3 *Suppose there are two or more feature variables. Then starting from the naive Bayes model structure, we can reach any regular HNB model structure using parent-introduction and parent-alteration. Moreover, starting from any regular HNB model structure we can reach any other regular HNB model structure using parent-introduction, parent-alteration, and node-deletion.*

4 Empirical results

In the previous section we have described an algorithm for learning regular HNB models from data. In this section we report on experiments designed to answer the following two questions:

1. Can our algorithm discover interesting latent variables?

2. Can our algorithm yield better classifiers than the naive Bayes classifier.

In this section, when we speak about HNB models we always mean HNB models produced by our algorithm.

4.1 Experiments with synthetic data sampled from HNB models

Three experiments were conducted with synthetic data sampled from HNB models. The generative models in all the experiments are based on the structure in Fig. 1. In Experiments 1 and 2, the structure was used without any modification. In Experiment 3, the structure was modified in such a way that each of the three latent nodes has one more child representing an additional feature variable. All the variables in all the models have three states.

In Experiment 1, model parameters were generated in a purely random fashion. In Experiment 2, model parameters were also randomly generated except that we ensured that $\max_y P(Y{=}y|X) \geq 0.8$ for each node Y and its parent X. In Experiment 3, we generated, for each node Y and its parent X, a bijection δ from the domain of X to that of Y and ensured that $P(Y{=}\delta(x)|X{=}x) \geq 0.8$ for any state x of X. Clearly the strength of correlation between the observed variables and latent variables increases as we move from Experiment 1 to Experiment 3. The strength of correlation among observed variables also increases as we move from Experiment 1 to Experiment 3.

In each experiment, five training sets with 1,000, 5,000, 10,000, 50,000, and 100,000 records were sampled from the generative model. Our algorithm was then used to learn a regular HNB model from each of the training sets. In these and all other experiments, 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 a variant of the multiple-restart approach (see [2]). The number of starting points was set at 64.

We compare the learned models with the original generative models in terms of model structures, cardinalities of latent variables, and marginal distributions over the class and feature variables.

In all cases, our algorithm recovered the structures of the original models precisely. In other

words, it correctly detected the latent variables.

Cardinalities of the latent variables in the learned HNB models are shown in Table 1. These results match our intuition quite well. In Experiment 1, the correlation between the observed and the latent variables is weak. Hence there is not much information in the training data about the latent variables . This explains why there are more underestimates in Experiment 1 than in the other two experiments. In Experiment 3, the correlation between the observed and the latent variables is very strong. It is therefore no surprise that the cardinalities of the latent variables were estimated correctly even in the case of 1,000 training samples.

Ref to Tab. 1

	Exp 1	Exp 2	Exp 3
1k	[2, 2, 2]	[3, 2, 3]	[3, 3, 3]
5k	[2, 3, 2]	[3, 2, 3]	[3, 3, 3]
10k	[2, 3, 2]	[3, 2, 3]	[3, 3, 3]
50k	[2, 3, 3]	[3, 2, 3]	[3, 3, 3]
100k	[2, 3, 3]	[3, 3, 3]	[3, 3, 3]

Table 1: Estimated cardinalities of latent variables.

The Kullback-Leibler (KL) divergences between the distributions over the class and feature variables in the original models and those in the learned models are shown in Fig. 4. We see that the divergences for the naive Bayes models are large. This indicates that the naive Bayes models are poor approximations of the original models. On the other hand, the divergences for the HNB models are very close to zero, especially in cases with large training sets. As far as the distribution over the class and feature variables are concerned, the learned HNB models and the original models are almost identical. The divergences are larger in Experiment 3 than in the other two experiments because there are three more feature variables.

The experiments were continued to determine whether our algorithm improves classification accuracy. Naive Bayes models were constructed based on the same training sets as those we used to build HNB models. In each of the experiments, a test set of 5,000 samples was generated. The HNB and naive Bayes models were then used to classify each of the test samples and the results were compared with the true class memberships.

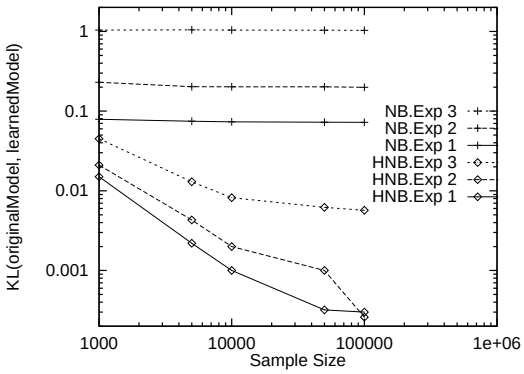

Figure 4: The Kullback-Leibler divergences between the distributions over the observed variables in the original models and those in the learned models.

In Experiment 1, HNB models are slightly better than naive Bayes models. But the differences are not significant. Statistics about the other two experiments are shown in Fig. 5, where the X-coordinates of data points are sometimes slightly shifted so that the error bars do not overlap. We see that the HNB models are significantly more accurate than the naive Bayes models, especially in Experiment 3.

Ref to Fig. 5

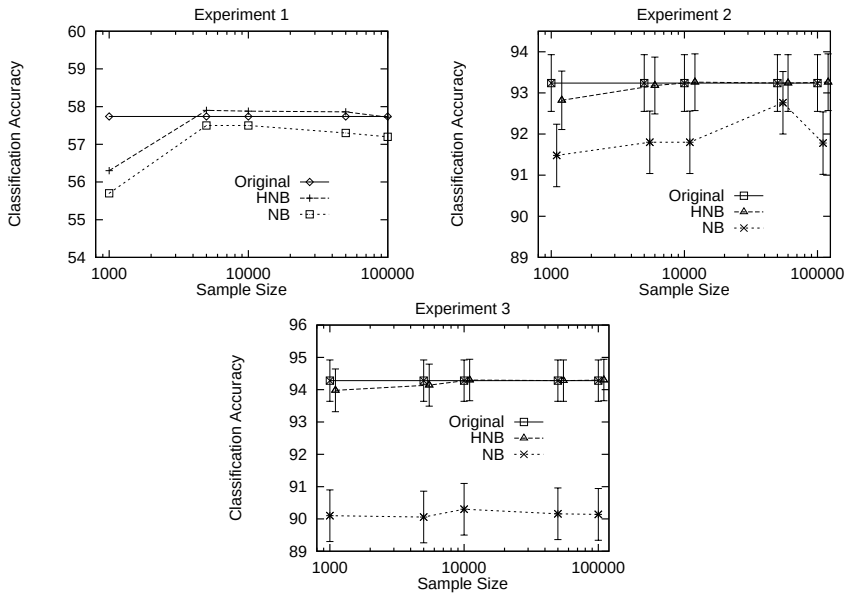

Figure 5: Comparison of HNB models and naive Bayes models on HNB data.

Note that the performance differences between HNB and naive Bayes models correlate nicely

with the strength of correlation among observed variables. In Experiment 1, the correlation is the weakest, and the HNB and naive Bayes models have virtually the same classification accuracy. In Experiment 3, on the other hand, the correlation is the strongest, and the performance differences between HNB and naive Bayes models are also the greatest.

We also evaluated the classification performance of a learned model in the following way. For each of the $5,000$ records in the test set, we computed the posterior distribution of the class variable given values of the feature variables. This was done in the generative model and in the learned models, resulting in two distributions. The KL-divergence between the distribution in the generative model and that in the learned model was calculated. We used the average of such KL-divergences over all the test data as a measurement of the classification performance of the learned model. The measurements for all the learned models are shown in Figure 6. In this experiment, we in fact look at the conditional likelihood. Although we have used the joint likelihood for scoring (plus a penalty term), the resulting models have a very nice performance with respect to conditional likelihood

Ref to Fig. 6

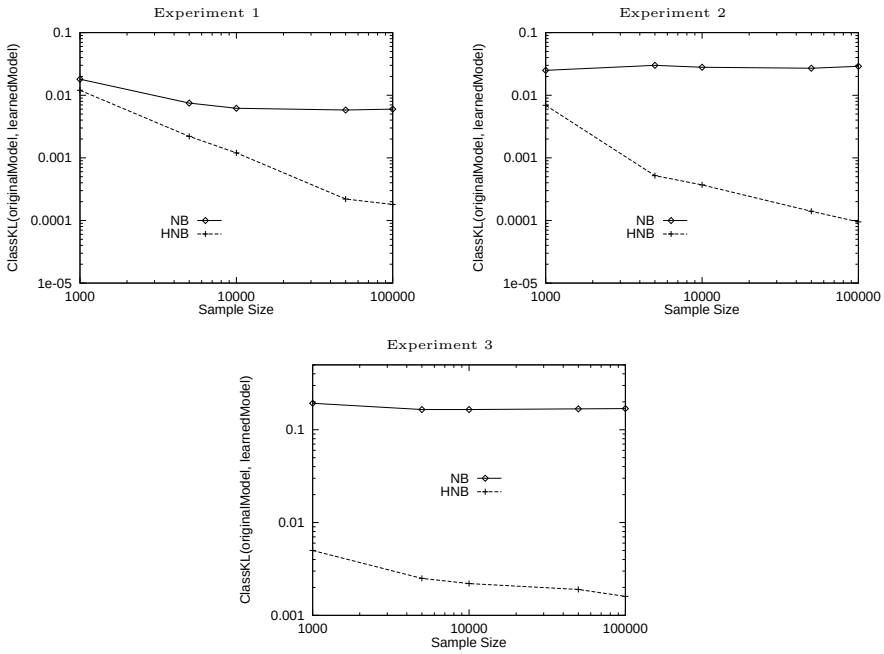

Figure 6: The average KL-divergences between the posterior distribution of the class variable in the generative model and the class variable in the learned models.

4.2 Experiments with UCI data

4.2.1 Results on Monk data sets

We next report on results for the three well-known Monk's problems Monk-1, Monk-2, and Monk-3 [16].We obtained the data files from the UCI machine learning repository [1].All the problems involve one binary class variable and six features variables $a_1 \ldots, a_6$ that have 2 to 4 possible states. Each problem has a data set of 432 records. In [16], the data sets were used to compare various machine learning algorithms. Between 30% and 40% of the records were used for training and all records were used in testing.

The target concepts embedded in the three data sets are `[`a_1=a_2 `or` a_5=1], `[exactly 2 of` $\{a_1$=1, a_2=1, a_3=1, a_4=1, a_5=1, a_6=1$\}$`]`, and `[(`a_5=3 `and` a_4=1`) or (`$a_5 \neq 4$ `and` $a_2 \neq 3$`)]` respectively. Clearly, correlation among observed variables is an important issue in those data sets. This makes them a good testbed for methods that deal with correlation among observed variables . We used them to compare naive Bayes models and HNB models via 10-fold cross-validation.

In Monk-1, the model structure of the HNB model is shown in Fig. 7. It matches nicely with the structure of the concept `[`a_1=a_2 `or` a_5=1`]`. In other words, meaningful latent variables are detected. Moreover, the classification accuracy of the HNB model is 100 ± 0.0, while that of the naive Bayes model is 75 ± 4.1 (percentage) (see Fig. 8).

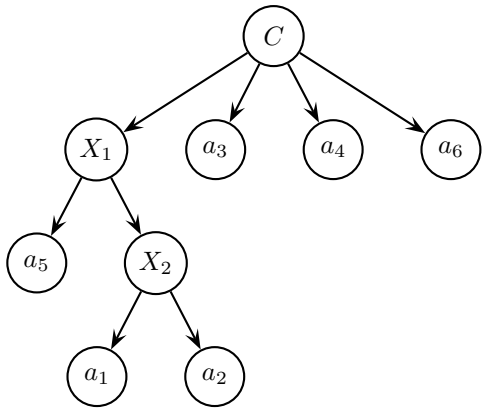

Figure 7: Structures of HNB models constructed for Monk-1. They match the target concepts nicely, indicating that meaningful latent variables are discovered.

The HNB models found for Monk-2 and Monk-3 are naive Bayes models. A closer inspection reveals that our algorithm restricted the search to only the neighborhood of naive Bayes models. More complex models were not examined at all. This suggests lack of data. To get an idea about what would happen if there were more data, we increased record counts by 3, 9, and 27 times respectively [1] and repeated the experiments with the enlarged data sets.

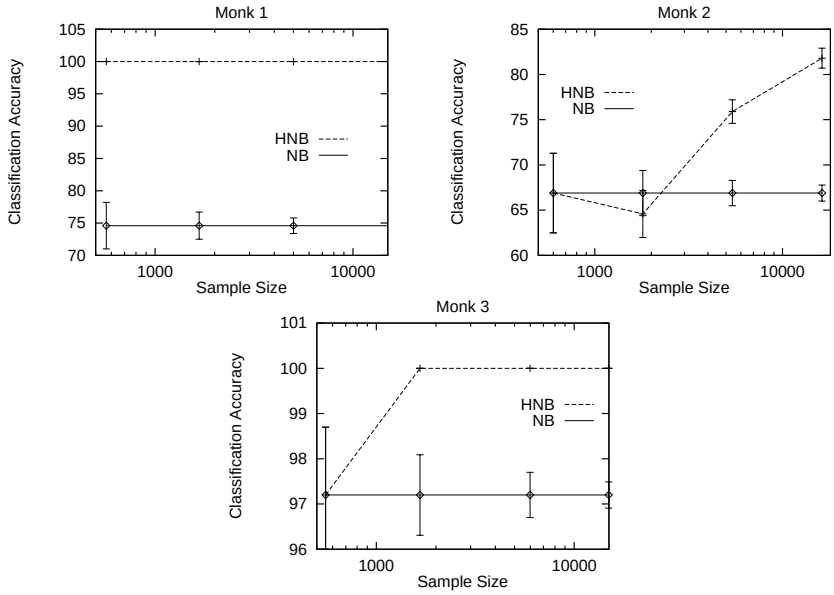

Figure 8: Classification accuracy of HNB and naive Bayes models on the Monk data sets.

After the sample size was increased by 3 or more times, the model structure for Monk-3 is as shown in Fig. 9. It matches well with the structure of the target concept `[(`a_5=3 `and` a_4=1) `or (`$a_5 \neq 4$ `and` $a_2 \neq 3$`)]`. So, meaningful latent variables are detected. The ideal HNB model for Monk-2 would be one with one latent variable that is the child of the class variable and the parent of all the feature variables. This latent variable would have a large number of states. Consequently, the model would compare favorably to other models under BIC only when there is huge amount of data. In our experiments, we did not find this model. Instead we found models with two or three latent variables when the sample size was increase by three or more times.

[1] Increasing record counts in a data set by n times means to duplicate each record $n-1$ times and add the new records to the data sets. In 10-fold cross-validation, a data set is partitioned into 10 subsets. There are two different ways record counts can be increased in relation to cross-validation. One increases record counts in the original data set and then partition the enlarged data set into 10 subsets. Another increases record counts in the subsets directly.

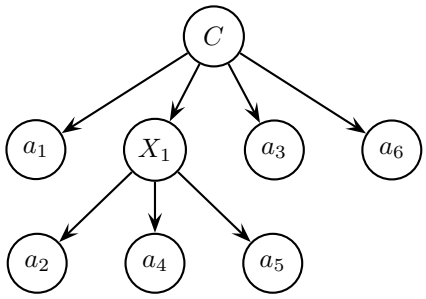

Figure 9: Structures of HNB models constructed for Monk-3. They match the target concepts nicely, indicating that meaningful latent variables are discovered.

The results on classification accuracy are summarized in Fig. 8. We see that as the sample size increases, significant differences between HNB models and naive Bayes models begin to emerge. In Monk-2, the classification accuracy of HNB models decreased before it surpassed naive Bayes models. This illustrates the fact that better density estimation does not necessarily imply better classification. In Monk-3, HNB models achieved perfect classification when the sample size was increased by 3 or more times. On the other hand, the accuracy of naive Bayes models does not change at all although the confidence interval shrinks as sample size grows.

4.2.2 Results on other UCI data sets

In addition to the Monk's problems, two other data sets from the UCI repository were used in our experiments, namely the Wisconsin Breast Cancer data set and the Pima Indians Diabetes data set. Those data sets were chosen because they consist of more than 500 records and do not have fewer than 10 feature variables. We refer the reader to the UCI repository for detailed information about the data sets. Feature variables in the diabetes data set take real values. They were discretized using the recursive minimal entropy partitioning method by [5]. Evaluation was conducted via 10-fold cross validation.

We are not experts to tell whether the latent variables in the HNB models are meaningful in the domains of the three data sets. However, aspects of the models certainly seem reasonable. For

We chose the second approach so that at least some records in the validation sets do not appear in the training sets.

the breast cancer data set, features "uniformity-of-cell-size" and "uniformity-of-cell-shape" always share the same (latent) parent. Intuitively, this latent variable might be interpreted s "uniformity-of-cell-morphology". For the diabetes data set, there is always a latent node that is the parent of both "age" and "number-of-pregnancies". This is reasonable because the two feature variables are intuitively correlated given the class variable. Since arcs are not allowed between feature variables, a latent variable was introduced to capture the correlation. In summary, interesting latent variables were detected in both data sets.

The classification accuracy statistics are shown in Table 2. We see that the statistics are roughly the same for HNB models and naive Bayes models. Ref to Tab. 2

	Breast Cancer	Diabetes
HNB	95.85 ± 1.48	79.17 ± 2.87
NB	96.28 ± 1.40	78.38 ± 2.91

Table 2: Classification accuracy on UCI data sets.

5 Concluding remarks

In this paper we have introduced HNB models as a framework for detecting latent variables in naive Bayes models. A hill-climbing algorithm for inducing HNB models from data has been developed. Empirical studies indicate that the algorithm is able to reveal interesting latent structures that are not explicit in data. This is helpful in gaining new insights about application domains and boosting user confidence in the induced models. Moreover, the HNB models produced by the algorithm often have significantly higher classification accuracy than naive Bayes models.

A major drawback of the algorithm, as it currently stands, is its high complexity. At each step of the search it generates a set of new candidate models based on the current model, estimates parameters for each of the candidate models using EM, scores them, and pick the best one to seed the next step of search. This process is computationally expensive primarily because of the reliance on the EM algorithm. A way to overcome this difficulty might be to use structural EM [8].

Acknowledgments

We thank Tomas Kocka for insightful discussions and the anonymous reviewers for useful comments. The first author's work on this paper was partially supported by Hong Kong Research Grants Council under grant HKUST6093/99E.

Appendix: Proofs

Proof of Theorem 1: Suppose M is an irregular HNB model. We show that M cannot be parsimonious.

Because M is irregular, there must exist a latent node Z such that inequality (1) is not true for Z and its neighbors Z_1, Z_2, ..., Z_k. Without loss of generality, let Z_1 be the parent of Z. Then Z_2, ..., Z_k are the children of Z.

There are three possible cases. The first case is when k=2. In this case, the strict-inequality version of (1) is not true. In other words, $|Z| \geq min\{|Z_1|, |Z_2|\}$. Construct another model M' by (1) removing Z from M and (2) setting $P(Z_2|Z_1) = \sum_Z P(Z|Z_1)P(Z_2|Z)$. Then M' represents the same distribution over the observed variables as M and it has fewer parameters than M. Hence M is not parsimonious.

The second case is when k>2 and $|Z_1| = \max_{i=1}^{k} |Z_i|$. In this case, we have $|Z| > \prod_{i=2}^{k} |Z_i|$. Let M' be the same as M except the variable Z is replaced by a new variable Z'. The domain of Z' is $\prod_{i=2}^{k} \Omega_{Z_i}$. A state of Z' can be written as $<z'_2, \ldots, z'_k>$, where z'_i is a state of Z_i. Set the conditional probability $P(Z'|Z_1)$ as follows:

$$P(Z'=<z'_2, \ldots, z'_k>|Z_1 = z_1) = P(Z_2=z'_2, \ldots, Z_k = z'_k|Z_1 = z_1),$$

where $P(Z_2, \ldots, Z_k|Z_1) = \sum_Z P(Z|Z_1) \prod_{i=2}^{k} P(Z_i|Z)$. For each $i = 2, \ldots, k$, set the conditional probability $P(Z_i|Z')$ as follows:

$$P(Z_i=z_i|Z'=<z'_2, \ldots, z'_k>) = \begin{cases} 1 & \text{if } z_i = z'_i \\ 0 & \text{if } z_i \neq z'_i. \end{cases}$$

Then M' represents the same distribution over the observed variables as M and it has fewer parameters than M. Hence M is not parsimonious.

The third possible case is when k>2 and $|Z_j| = \max_{i=1}^{k} |Z_i|$ for some j, $1 < j \leq k$. Without loss of generality, assume j=k. In this case, we have $|Z| > \prod_{i=1}^{k-1} |Z_i|$. Let M' be the same as M except the variable Z is replaced by a new variable Z'. The domain of Z' is $\prod_{i=1}^{k-1} \Omega_{Z_i}$. A state of Z' can be written as $<z'_1, \ldots, z'_{k-1}>$, where z'_i is a state of Z_i. Set the conditional probability $P(Z'|Z_1)$ as follows:

$$P(Z'=<z'_1, \ldots, z'_{k-1}>|Z_1 = z_1) = \begin{cases} P(Z_2=z'_2, \ldots, Z_k = z'_{k-1}|Z_1 = z_1) & \text{if } z_1 = z'_1 \\ 0 & \text{if } z_1 \neq z'_1. \end{cases}$$

where $P(Z_2, \ldots, Z_{k-1}|Z_1) = \sum_Z P(Z|Z_1) \prod_{i=1}^{k-1} P(Z_i|Z)$. For each $i = 2, \ldots, k-1$, set the conditional probability $P(Z_i|Z')$ as follows:

$$P(Z_i=z_i|Z'=<z'_1, \ldots, z'_{k-1}>) = \begin{cases} 1 & \text{if } z_i = z'_i \\ 0 & \text{if } z_i \neq z'_i. \end{cases}$$

Finally, set the conditional probability $P(Z_k|Z')$ as follows:

$$P(Z_k = Z_k|Z'=<z'_1, \ldots, z'_{k-1}>) = \frac{P(Z_2=z'_2, \ldots, Z_k = z'_{k-1}, Z_k = z'_k|Z_1 = z_1)}{P(Z_2=z'_2, \ldots, Z_k = z'_{k-1}|Z_1 = z_1)},$$

where $P(Z_2, \ldots, Z_{k-1}, Z_k|Z_1) = \sum_Z P(Z|Z_1) \prod_{i=1}^{k} P(Z_i|Z)$. Then M' represents the same distribution over the observed variables as M and it has fewer parameters than M. Hence M is not parsimonious

In summary, M is not parsimonious in any of the possible cases. Therefore, the theorem must be true.Q.E.D

Before proving Theorem 2, we need to prove two more lemmas, which are interesting in their own right. An HNB model structure is *strictly regular* if there are no singly connected latent nodes.

Lemma 2 *Let S be an HNB model structure with n feature variables. If S is regular, then there are no more than $3n$ latent nodes. If S is strictly regular, then there are no more than n latent nodes.*

Proof: We prove the second part first. Let h be the number of latent nodes. Then the total number of nodes is $n+h+1$. Hence the number of edges is $n+h$.

On the other hand, each feature node appears in exactly one edge and, because of strict regularity, each latent node appears in at least three edges. Because each edge involves exactly two variables, there are at least $(n+3h)/2$ edges. Hence $n+h \geq (n+3h)/2$. Solving this inequality yields $h \leq n$.

To prove the first part, let m be the total number of nodes in a regular structure. The child of a singly connected latent node is either a feature node or another latent node that is not singly connected. Moreover, the children for different singly connected nodes are different. So if we eliminate all the singly connected latent nodes, the resulting structure will have at least $m/2$ nodes. The resulting structure is strictly regular. Hence $m/2<2n$. This implies that $m<4n$. Since there are n feature nodes, the number of latent is no more than $3n$. Q.E.D

Lemma 3 *There are no more than 2^{3n^2} different regular HNB model structures for a given set of n feature nodes.*

Proof: Let $\mathcal{P}$ be the power set of the set of feature nodes and let $\mathcal{V}$ be the collection of vectors that consists of $3n$ elements of $\mathcal{P}$. Duplicates are allowed in any given vector. Since the cardinality of $\mathcal{P}$ is 2^n, the cardinality of $\mathcal{V}$ is $(2^n)^{3n}=2^{3n^2}$.

Let $\mathcal{S}$ be the set of all regular HNB model structures for the given feature nodes. Define a mapping from $\mathcal{S}$ to $\mathcal{V}$ as follows: For any given model structure in $\mathcal{S}$, first arrange all the latent nodes into a vector according to the depth-first traversal order. According to Lemma 2, the length of the vector cannot exceed $3n$. Then replace each latent node with the subset of manifest nodes in its subtree. Finally, add copies of the empty set to the end so that the length of the vector is $3n$. It is not difficult to see that the mapping is bijective. Therefore the cardinality of $\mathcal{S}$ cannot exceed that of $\mathcal{V}$, which is 2^{3n^2}. The lemma is proved. Q.E.D.

Proof of Theorem 2: According to Lemma 3, the number of regular HNB model structures is finite. It is clear from (1) that the number of regular HNB models for a given model structure

must also be finite. The theorem is therefore proved. Q.E.D

Proof of Theorem 3: Note that the effects of one application of parent-alteration can be undone using another appropriate application of parent-alteration. The effects of parent-introdution can be undone using node-deletion. Hence the second part of the theorem follows readily from the first part.

We prove the first part by induction on the number of latent variables. When the number is zero, the statement is trivially true. Suppose the statement is true when there are h latent variables. Consider the case of h+1 latent variables. Use S to denote the model structure under discussion. Pick one latent node and apply the parent-alteration operation to its children repeatedly so that it has no more than two children aferwards. Then apply the node-deletion operation to remove the node. Denote the resulting model structrue by S'. By the induction hypothesis, we can reach S' from the naive Bayes model structure. Because the way S' is constructed, we can reach S from S' using parent-introduction and parent-alteration. The theorem is therefore proved. Q.E.D

References

[1] C. L. Blake and C. J. Merz, UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, (1998).

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

[3] D. Connolly, Constructing hidden variables in Bayesian networks via conceptual learning, In Proc. 10th Int. Conf. on Machine Learning (ICML-93) (1993) 65-72.

[4] A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B, 39 (1977) 1–38.

[5] J. Dougherty, R. Kohavi and M. Sahami, Supervised and unsupervised discretization of continuous features, In Proc. 12th Int. Conf. on Machine Learning (ICML-95) (1995) 194-202.

[6] G. Elidan, N. Lotner, N. Friedman and D. Koller, Discovering hidden variables: A structure-based approach, Advances in Neural Information Processing Systems 13 (NIPS-00) (2000) 479-485.

[7] G. Elidan and N. Friedman, Learning the dimensionality of hidden variables, In Proc. 17th Conf. on Uncertainty in Artificial Intelligence (UAI-01) (2001) 144-151.

[8] N. Friedman, Learning belief networks in the presence of missing values and hidden variables, In Proc. of 14th Int. Conf. on Machine Learning (ICML-97) (1997) 125-133.

[9] N. Friedman, D. Geiger and M. Goldszmidt, Bayesian networks classifiers, Machine Learning 29 (1997) 131-163.

[10] D. Heckerman, A tutorial on learning with Bayesian networks, Technical Report MSR-TR-95-06, Microsoft Research, March, 1995 (revised November, 1996).

[11] I. Kononenko, Semi-naive Bayesian classifier, In Proc. 6th European Working Session on Learning, Berlin: Springer-Verlag 1991 206-219.

[12] C. Kwoh and D. F. Gillies, Using hidden nodes in Bayesian networks, Artificial Intelligence 88 (1996) 1-38.

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

[14] M. J. Pazzani, Searching for dependencies in Bayesian classifiers, In Proc. 5th Int. Workshop on AI and Statistics, Ft. Lauderdale, FL (1995).

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

[16] S. B. Thrun and 20 others, The MONK's Problems: A Performance Comparison of Different Learning Algorithms, CMU-CS-91-197, Carnegie Mellon University (1991).