Case-Based Bayesian Network Classifiers

Eugene Santos Jr. and Ahmed Hussein
Department of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269-3155
eugene@engr.uconn.edu and ahussein@engr.uconn.edu

Abstract

We propose a new approach for learning Bayesian classifiers from data. Although it relies on traditional Bayesian network (BN) learning algorithms, the effectiveness of our approach lies in its ability to organize and structure the data in such a way that allows us to represent the domain knowledge more accurately than possible in traditional BNs. We use clustering to partition the data into meaningful patterns, where each pattern is characterized and discriminated from other patterns by an index. These patterns decompose the domain knowledge into different components with each component defined by the context found in its index. Each component can then be represented by a local BN. We argue that this representation is more expressive than traditional BNs in that it can represent domain dependency assertions more precisely and relevantly. Our empirical evaluations show that using our proposed approach to learning classifiers results in improved classification accuracy.

Introduction

Learning accurate classifiers from data continues to be an active research area. Many algorithms have been developed for learning classifiers of different functional representation such as decision trees, neural networks, and Bayesian networks (Han and Kamber 2001).

Bayesian network (BN) classifiers (Cheng and Greiner, 2001) have gained more attention from machine learning and data mining researchers since the discovery of the first BN classifier known as naive-Bayes (Langley, Iba and Thompson 1992). This classifier is merely a very simple BN with a strong assumption of independence among its variables *given* the classification variable C, though it has surprisingly shown a competitive performance (i.e., classification accuracy) with state-of-the-art non-Bayesian classifiers such as C4.5 (Quinlan 1993).

The encouraging performance of naive classifiers has motivated researchers to build other BN classifiers that relax the naive classifier's strong independency assumption. The Tree Augmented naive-Bayes (TAN) (Cheng and Greiner 1999; Friedman, Geiger and Goldszmidt 1997) approximates the interactions between attributes by using a tree structure. TAN has been shown to outperform naive-Bayes classifier.

Consequently, and with the advances in developing algorithms that learn BNs from data, many researchers have explored unrestricted BN classifiers. These classifiers are learned based on recent learning algorithms that can learn multiply connected BNs. Among these classifiers is the Bayesian network Augmented naive-Bayes (BAN) (Cheng and Greiner 1999; Friedman, Geiger and Goldszmidt 1997) which extends TAN by allowing the attributes to form an arbitrary graph, rather than a tree, and the General Bayesian Network classifiers (GBN) (Cheng and Greiner 1999; Friedman, Geiger and Goldszmidt 1997) which treats the classification node as an ordinary node and identifies a relevant attribute subset around the classification node defined by its Markov blanket. These two classifiers have been built and examined based on two different BN learning algorithms. Friedman et al. (1997) used the MDL score algorithm (Lam and Bacchus 1994) while Cheng et al. (1999) used the CBL learning algorithm (Cheng, Bell and Liu 1997). In both studies, the empirical evaluation of the two classifiers showed that these classifiers perform better than naive and in many cases outperform TAN. Friedman et al. (1997), however, have mentioned that GBN classifiers learned via the score-based learning algorithms may result in relatively poor classification accuracy since a good score function does not necessarily lead to good classification accuracy. Cheng et al. (1999) have shown that GBN classifiers built based on non-scoring learning algorithms (i.e., CI-test algorithms), do not suffer from this problem and that these algorithms can effectively learn unrestricted BN classifiers.

The above earlier work suggests that unrestricted BN classifiers can capture the relationships among the domain attributes better, therefore, leading to more accurate classifiers. This fact motivates us to raise the question of whether improving the capability of BNs as a tool for representing dependency assertions can further improve their performance as classifiers. In this paper, we introduce a new type of Bayesian classifier called "*Case-Based Bayesain Network (CBBN)*" classifiers. Although, this type of Bayesian classifier is learned from data using traditional BN learning algorithms, we will show that our learning methodology organizes the data in such a way that allows more precise and more relevant representation of the domain dependency re-

lationships. In particular, we introduce the concept of "*case-dependent relationships*" and show that while traditional BNs are not suitable to represent this type of knowledge, our CBBNs can capture and encode them, hence improving classification accuracy.

CBBN Methodology

Our approach to learning classifiers employs a clustering technique to discover meaningful patterns in the training data set represented by different clusters of data. Each cluster is then characterized and discriminated from other clusters by a unique assignment to its most relevant and descriptive attributes. This assignment is called an index. As we shall see, these indices provide a natural attribute selection for the CBBN classifiers. Intuitively, each cluster represents a piece of the domain knowledge described by the context of its index. These clusters can also be viewed as a set of conditionally independent cases with each case mapped to an index that describes the context of the knowledge relevant to that case. The knowledge associated with each case can then be represented *independently* by a BN *conditioned* on its index. This independency of cases implies that the relationships among the corresponding attributes might be different for different cases. Thus, instead of assuming fixed relationships between attributes for the whole domain as in traditional BNs, these relationships can vary according to each different context of each case in the same domain. This conclusion is crucial, since it means that two variables X and Y might be directly dependent ($X \rightarrow Y$) in case C_i and independent in case C_j. Moreover, $X \rightarrow Y$ might occur in case C_i while $Y \leftarrow X$ occurs in case C_j. Even if the relationships in different cases are the same, the parameters that represent the strength of these relationships might be different.

As an example, consider a database of customers applying to a loan in a bank. Such a database might have two different patterns (i.e., cases) where each pattern represents a group of customers. The first group includes those customers who have a good balance in their checking and saving accounts. The decision to grant a loan to these customers might not be influenced by whether a customer has a guarantor, whether he/she has properties, or whether he/she is a citizen, but it might be highly affected by his/her residency time and somewhat by his/her credit history. The second group might include those people who do not have sufficient balance in their checking account and with poor credit history. For this group the situation is different since the bank decision will be highly affected by whether they have properties, whether they have a guarantor, whether they are citizens, as well as their residency time. The bank decision and its requirements may be considered as domain variables in a BN that have case dependent relationships among them Fig.(1).

Another example is cyclic knowledge. In a data set of patients suffering from diabetes a doctor can distinguish between two groups of patients; those who have just started taking a specific medication and found that it causes an improvement where the glucose level starts to decrease, and those who have been taking the medication for a while and

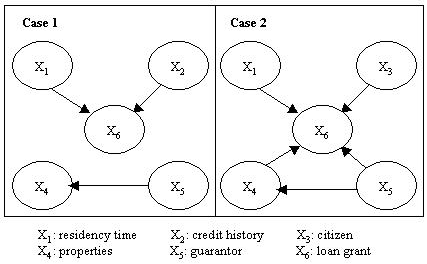

Figure 1: An example of case-dependent relationships

are excited by the improvement thus causing them to increase the dose of that medication or even taking an additional one. In this example the relation between medication level and health improvement is not purely unidirectional, but is case-dependent cyclic.

This kind of knowledge cannot be represented in traditional BNs. We argue that these relationships can be represented in our CBBN model and significantly improve its accuracy as a classifier. Moreover, our approach provides a novel procedure for selecting relevant attributes. In traditional GBN classifiers, a Markov blanket of the classification node is used as an attribute selection procedure. Often, this selection is useful and discards truly irrelevant attributes. However, it might discard attributes that are crucial for classification (Friedman, Geiger and Goldszmidt 1997). CBBN provides an alternative attribute selection procedure that better avoid discarding relevant attributes. The attributes that constitute an index for a cluster have fixed values for all objects in the cluster. We conclude that these attributes are irrelevant to the classification task in this cluster. Hence, the BN classifier learned from this cluster can safely exclude these attributes.[1]

CBBN Classification Model

Constructing a CBBN classification model consists of the following three phases:

Clustering and indexing phase

Suppose D is a training data set described by a set of categorical attributes $A_1, A_2, ..., A_n, C$ where C is the classification node. A clustering algorithm is used to partition D into a set of clusters $\mathbf{C} = \{C_1, C_2, \ldots, C_k\}$ characterized by a set of mutually exclusive indices $\mathbf{I} = \{I_1, I_2, \ldots, I_k\}$ respectively. This indexing scheme guarantees at most one mapping per a data object to the set **I**.

In order to generate such an indexing scheme, algorithm *A* shown below begins by initializing **I** as set of k n-dimension vectors with "don't care" (i.e. 'x') values for all elements of each vector I_i. For a particular cluster C_i, the algorithm computes the probability distribution for each attribute, (i.e.,

[1]This attribute selection procedure might also be useful in simplifying the structure of the classifiers, especially in BAN and GBN, to avoid the overfitting problem reported by Cheng et al. (1999)

the frequencies of its possible values estimated from the data in this cluster). The algorithm proceeds to determine the value of each attribute that has the maximum frequency and assigns this value to this attribute in I_i if its frequency exceeds an indexing threshold α. The resulting assignment is then used as a description of the objects in C_i, thus the algorithm moves all objects that are not covered by I_i from C_i to the outliers cluster. The same procedure is repeated with each cluster. The algorithm then visits the outliers cluster to check for possible mappings of its objects back to the indexed clusters. These objects are retrieved from the outlier to be placed in a cluster if the objects are compatible to the cluster's description index.

In order to achieve mutual exclusion between the above assignments, algorithm *B* checks each two assignments for the mutual exclusion condition (at least one common attribute is assigned differently). If they do not satisfy this condition, it searches for the "don't care" attribute in both assignments that can be assigned differently in both of them such that a minimum number of objects is rejected from both clusters due to the new assignments. The algorithm then updates the members of all clusters, including the outliers, according to the new mutually exclusive assignments. Finally, to produce the index of each cluster, the algorithm simply discards any "don't care" attributes in each assignment.

Algorithm *A*: Clustering and Indexing
Input:
D: training data set
k: number of clusters
α: indexing threshold
Output:
C: set of k clusters $C_1, C_2, \ldots, C_k$
I: set of mutually exclusive indices $I_1, I_2, \ldots, I_k$
Outliers: possible outliers cluster
Notation:
$R(A_j)$: the domain of the attribute A_j
$a_{j,i(max)}$: a_j that maximizes $P(A_j = a_j|C_i)$
$P_{j,i(max)}$: $P(A_j = a_{j,i(max)}|C_i)$

Begin
Call clustering algorithm on D to form the set of clusters **C**
For each cluster C_i
  Initialize I_i as an n-dimensional vector with 'x' values
  For each attribute A_j
    Compute$P(A_j = a_j|C_i) \forall a_j \in R(A_j)$
    Find $a_{j,i(max)}$ and $P_{j,i(max)}$
    If $(P_{j,i(max)} > \alpha)$ assign $a_{j,i(max)}$ to j^{th} element in I_i
  Move the objects of C_i not covered by I_i to *Outliers*
For each cluster C_i
  Move from *Outliers* objects covered by I_i back to C_i
Call Algorithm *B* to get mutually exclusive vectors in **I**
For each cluster C_i and using its updated I_i
  Move objects of C_i not covered by I_i to the *Outliers*
For each cluster C_i and using its updated I_i
  Move from *Outliers* the objects covered by I_i back to C_i
End

Algorithm *B*: check and fix
Input:
C: a set of k data clusters
I: a set of k n-dimensional vectors
Output:
I: a set of mutually exclusive indices (updated **I**)
Notation:
$a_{t,i}$: the value of the attribute A_t in I_i
$I_{i(t)}$: the location of attribute A_t in I_i
$a_{t,i(max)}$: a_t with the maximizes $P(A_t = a_t|C_i)$
u_i: no. of uncovered objects by I_i in C_i
u_j: no. of uncovered objects by I_j in C_j

Begin
For $i = 1$ to $k - 1$
  For $j = i + 1$ to k
    If (I_i and I_j are not mutually exclusive) then
      For each attribute A_t ($t = 1, 2, ..., n$)
        If ($a_{t,i} = a_{t,j} =$'x') then
          Find $a_{t,i(max)}$ and $a_{t,j(max)}$
          If ($a_{t,i(max)} != a_{t,j(max)}$) then
            $I_{i(t)} = a_{t,i(max)}$ and $I_{j(t)} = a_{t,j(max)}$
            Find u_i and u_j
            Compute $s_t = u_i + u_j$
            Retrieve the original state of I_i and I_j
      Find the attribute A_p that minimizes s_t
      put $I_{i(p)} = a_{p,i(max)}$ and $I_{j(p)} = a_{p,j(max)}$
For each I_i
  If an attribute $A_j =$'x' then remove A_j from I_i
End

Learning Phase

We apply a BN learning algorithm to learn a local BN classifier B_i, where $i \in \{1, 2, ..., k\}$, from the data objects in each indexed cluster produced by algorithms *A* and *B*. This local classifier is defined over a subset $V_i \subset \mathbf{V}$. If $V(I_i)$ is the set of the attributes in I_i then $V_i = \mathbf{V} - V(I_i)$. We also learn a BN classifier, B_o, from the outliers cluster defined over the whole set **V**. The set of local classifiers together with the indecies constitute a CBBN classifier.

Testing Phase

We test the newly learned CBBN classification model on the given test data set T. Basically, we map each test object $(a_1, a_2, \ldots, a_n)$ in T to an index in **I** by comparing the attributes assignment in both of them. We then compute $P(C|a_1, a_2, \ldots, a_n)$ from the local BN classifier characterized by that index and assign to *C* the value that maximizes P. Because of the mutual exclusion property of our indexing scheme, an object can map to at most one local classifier B_i. If an object cannot be mapped to any index in **I**, we map it to B_o as the default classifier. Finally, we compute the accuracy by comparing the predicted values of *C* found above with its true values in T.

Experimental Results

Experiment Settings

We have learned classifiers of different structures (i.e., naive, TAN, BAN, BAN*, GBN and GBN*) from a set of twenty-five benchmark databases. These classifiers have been built based on BN approach and based on our CBBN approach. Moreover, the structures of local classifiers have been learned using different learning algorithms. In particular, we used the MDL score algorithm to learn BAN and GBN, and CBL2 algorithm to learn BAN* and GBN*. For TAN classifier, we used Chow and Liu (1968) algorithm to learn a tree-like structure. When comparing CBBN classifiers and BN classifiers, we do that for corresponding structure types.

The data sets were obtained from the UCI machine learning repository(*www.ics.uci.edu*). In all data sets, objects with missing attribute values have been removed and numerical attributes have been categorized. To avoid differences in data cleaning, we had to recompute the results of BN classifiers instead of using results from previous work. However, our recomputed results are still close to the ones reported in (Cheng and Greiner 1999; Friedman, Geiger and Goldszmidt 1997).

For data clustering in CBBN model, we used the clustering algorithm, *k-modes* (Huang 1998), that extends the popular clustering algorithm, k-means, to categorical domains. The biggest advantage of this algorithm is that it is scalable to very large data sets in terms of both number of records and number of clusters. Another advantage of k-modes algorithm is that the modes provide characteristic descriptions of the clusters. These descriptions are important in characterizing clusters in our CBBN approach.

The k-modes algorithm, as many clustering algorithms, requires that the user specify the number of clusters k. In this work, we have determined an acceptable range of k for each data set. More specifically, k can take integer values between $k_{min} = 2$ and k_{max} which is the maximum number of clusters estimated such that each cluster has a number of objects sufficient to learn a BN classifier. We then ran our experiments at three different values of k (k_{min}=2, k_{max}, and $k_{arb} \in]k_{min}, k_{max}[$) and compare the accuracy of CBBN classifiers in each case to that of BN classifiers and machine learning (ML) classifiers (C4.5 and Instance-Based (IB) classifiers).

Classification Accuracy

Tables (1, 2, and 3) show our classification accuracy for BN, ML and CBBN classifiers. Because of space limitations, we only show the results for $k = k_{arb}$. Similar results have been obtained for other values of k (i.e., k_{min} and k_{max}).

The experimental results have shown that classifiers learned using our CBBN approach are either superior to or competitive with BN classifiers. This confirms our theoretical intuition in that better representation of the dependency relationships results in more accurate classifiers. However, the amount of improvement in the classification accuracy of CBBN models over BN models differs from one data set to another depending on how good the chosen clustering scheme and how rich the original data set with case-dependent relationships. In the worst case, as we can see from the experimental results, CBBN classifiers perform as well as BN classifiers. When case-dependent relationships matter, as an example, in the german loan approval data set, we noticed a cyclic relationship between three variables (balance, loan, and business). This relationship appears as follows: in one case, (balance $\rightarrow$ loan $\rightarrow$ business) while in another case, (balance $\leftarrow$ business). The results have also shown that CBBN classifiers are either superior to or competitive with ML classifiers.

In order to compare CBBN classifiers vs. BN classifiers and ML classifiers, we considered the *average improvement in accuracy* and the *winning count* over all data sets. Comparisons for all different structures have shown that CBBN classifiers have considerable average improvement in accuracy over BN classifiers and ML classifiers, and they beat them in most of the data sets.

The min. average improvement (9.661%) in CBBN over BN classifiers was recorded in naive classifiers. The reason for that is the restricted structure of the naive BN. However, the improvement is due to the ability of CBBN to estimate the parameters accurately from the relevant knowledge to identify and get rid of irrelevant attributes. By contrast, BAN and GBN classifiers recorded higher average improvements (15.562% and 13.714%) in CBBN over BN. We argue that these two classifiers allow unrestricted relationships between attributes, hence increasing the chance to capture case-dependent relationships.

BAN and BAN* classifiers in CBBN have the min. average classification error (4.554% and 5.058%), which means that their general accuracy is better than other classifiers. This is due to the fact that these classifiers allow unrestricted dependencies between attributes and at the same time considers the classification node as a parent for all other nodes. This is useful in some data sets when weak dependencies exist between attributes but cannot be captured unless the state of the classification node is given.

GBN classifier learned using MDL approach has the max. average error (12.137%) (i.e., the worst general accuracy). However, this accuracy is improved in GBN* using the CI test algorithm since it has only (8.614%) average error. This confirms that GBN classifiers built based on CI test learning algorithm perform better than those built based on search & score learning algorithms.

The indexing threshold α affects the size of the outliers clusters in a CBBN model. A large value for α is likely to lead to a small size for the outliers cluster, which is desirable, but will also make the descriptive attribute in the indices rare. By contrast, a small value of α will probably simplify the classifier structure by assigning more attributes to the index, but is likely to increase the size of the outliers cluster. So there is always a tradeoff. In our experiments, we adjust α such that the outliers do not exceed a predetermined percentage (10%) of the size of the training set.

Datasets						ML		naïve		TAN		BAN		BAN*		GBN		GBN*	
no.	**name**	**train**	**test**	**k**	**α**	**C4.5**	**IB**	**BN**	**CBBN**	**BN**	**CBBN**	**BN**	**CBBN**	**BN**	**CBBN**	**BN**	**CBBN**	**BN**	**CBBN**
1	australian	690	CV-5	3	0.80	85.217	81.739	86.087	93.333	81.159	87.391	86.957	96.232	87.246	**97.101**	86.232	94.493	88.986	95.652
2	breast	683	CV-5	3	0.80	94.436	96.047	97.218	96.779	95.900	95.608	96.633	**98.682***	96.779	**98.682***	96.925	97.804	95.022	97.657
3	car	1728	CV-5	4	0.85	69.329	66.204	85.185	94.907	94.097	**97.280**	90.451	96.586	94.039	96.933	86.400	92.882	86.111	93.403
4	chess	2130	1066	4	0.78	**99.390**	95.028	87.054	96.623	92.495	95.872	94.090	96.998	94.184	96.717	95.685	96.904	94.653	96.154
5	cleve	296	CV-5	2	0.90	73.986	77.027	83.446	90.541	79.730	93.581	79.392	93.243	82.095	**95.608**	81.081	93.581	84.459	92.905
6	crx	653	CV-5	3	0.95	86.217	77.489	86.064	93.109	83.920	94.334	86.524	94.793	88.055	**95.100**	85.758	93.415	86.217	94.181
7	diabetes	768	CV-5	3	0.75	76.172	71.484	74.219	81.901	75.000	87.891	75.520	87.500	77.083	**92.188**	75.391	85.938	81.250	90.365
8	DNA	2000	1186	4	0.70	92.580	75.801	95.278	94.688	93.592	96.374	90.135	**97.218**	88.533	96.121	73.946	82.125	79.089	93.086
9	flare	1066	CV-5	3	0.80	82.551	82.833	79.362	88.462	82.552	89.587	82.645	94.090	82.833	**94.934**	82.833	90.619	82.270	91.370
10	german	1000	CV-5	3	0.82	72.300	69.700	74.500	82.800	72.200	91.500	73.200	92.400	76.800	**94.800**	72.100	83.400	80.500	87.700
11	glass	214	CV-5	2	0.80	62.241	70.561	70.561	79.439	68.961	85.514	70.561	95.794	71.028	**96.262**	56.075	71.495	64.019	84.579
12	heart	270	CV-5	2	0.87	80.471	80.000	80.370	92.593	83.704	92.593	82.963	94.815	86.296	95.185	81.481	93.333	85.276	**96.667**
13	led24	200	3000	2	0.85	65.567	39.433	72.600	87.300	73.800	82.967	72.600	**95.767**	74.100	94.600	70.549	87.633	78.750	88.933
14	liver	345	CV-5	3	0.88	60.870	64.348	63.188	75.362	65.217	79.420	66.957	94.493	67.246	**95.072**	53.333	75.072	66.667	84.058
15	letter	15000	5000	10	0.85	77.700	72.800	74.980	88.520	83.460	94.920	76.640	91.440	79.300	**96.060**	75.000	86.880	78.733	87.760
16	mofn-3-7-10	300	1024	3	0.90	85.449	89.355	86.328	93.945	91.797	94.727	86.328	95.508	88.514	**96.680**	86.035	91.016	87.402	92.090
17	nursery	8640	4320	6	0.80	68.241	66.157	90.301	92.824	91.713	95.255	91.296	**97.593**	93.079	97.199	90.139	96.736	89.722	97.546
18	pima	768	CV-5	3	0.75	75.130	68.750	75.651	85.547	74.870	86.328	74.740	92.318	79.297	**93.359**	75.000	88.932	76.042	85.938
19	satimage	4435	2000	5	0.85	83.100	88.800	81.850	92.850	77.600	92.500	80.550	95.750	84.450	**96.050**	59.100	75.850	64.450	87.850
20	segment	1540	770	3	0.90	93.506	**96.104***	90.909	92.857	85.455	94.805	91.039	95.584	90.390	**96.104***	93.636	94.156	91.948	93.506
21	shuttle-small	3866	1934	5	0.77	99.121	**99.586**	98.242	96.381	98.914	97.156	98.910	98.190	97.208	96.794	99.121	97.880	97.001	97.466
22	soybean-large	562	CV-5	3	0.85	91.993	90.747	91.637	92.527	58.363	71.174	92.349	**96.263**	92.865	95.196	58.363	72.046	72.064	83.274
23	vehicle	846	CV-5	3	0.85	69.740	63.830	58.510	70.686	67.967	74.470	67.494	**93.498**	71.631	87.589	60.875	73.286	78.369	90.189
24	vote	435	CV-5	3	0.80	95.172	94.713	89.655	96.092	88.966	94.943	90.115	94.253	95.632	**97.241**	95.172	96.782	95.712	95.862
25	waveform21	300	4700	2	0.90	74.787	75.766	77.872	85.213	75.383	92.553	77.723	94.553	78.787	**94.574**	69.447	84.319	71.340	86.468
best classification accuracy count						1	1	0	0	0	1	0	5	0	**14**	0	0	0	1

Table 1: Classification Accuracy ($k=k_{arb}$)

CBBN →	naïve	TAN	BAN	BAN*	GBN	GBN*
win/BN	88.000	92.000	96.000	96.000	96.000	100.000
imp/BN	9.661	11.756	15.562	13.729	13.714	12.080
error	10.589	9.650	5.058	4.554	12.137	8.614

Table 2: CBBN Classifiers vs. BN Classifiers ($k=k_{arb}$)

CBBN →	naïve	TAN	BAN	BAN*	GBN	GBN*
win/C4.5	88.000	84.000	88.000	92.000	80.000	88.000
imp/C4.5	12.260	13.856	20.041	20.667	10.490	15.245
win/IB	92.000	84.000	88.000	96.000	84.000	84.000
imp/IB	17.826	19.265	25.987	26.555	16.095	20.955

Table 3: CBBN Classifiers vs. ML Classifiers ($k=k_{arb}$)

Time Cost

Suppose N is the number of data objects in the training set, r is the maximum number of possible values for an attribute, and t is the number of iteration required for k-modes to converge. We are interested in the construction time of the model. In BN models, the construction time is only the learning time which is $O(Nn^4r^n)$ for unrestricted BNs and only $O(Nn^2)$ for tree-structure networks. In our CBBN model, the construction time is the summation of the clustering time, the total learning time, and the indexing time.

The clustering time is $O(tknN)$ where $t, k, n << N$. It is obvious that we will not waste too much time in clustering because of the linearity of the running time of the clustering algorithm we chose with the size of the data set. The total learning time is the summation of the learning times needed to learn a BN classifier from each cluster. The repetition of the learning process is time consuming in CBBN models. However, as we can see the learning time in the algorithms mentioned above is linear with the number of data objects N and polynomial in number of attributes n. Since each cluster is smaller in size than the original data set and represented by a fewer number of nodes because of our indices, we conclude that the average time of learning from a cluster might be much smaller than learning from a whole large training set described by a large number of attributes. The only time left is the indexing time. From algorithms *A* and *B* this time can be estimated as $O(rnNk^2)$ in the worst case.

Based on the above discussion, for sparse BNs, we would expect a CBBN model to be expensive compared to a BN model because of the indexing time and the repeated learning time. For dense BNs, we would expect some of the time used in indexing and repeated learning in a CBBN model to be compensated by the lengthy time to learn such a complex BN model. For example, for the DNA data set with 60 attributes and 2000 data objects in the training set, it takes 117 CPU seconds to build a BN-TAN, while it takes 374 CPU seconds to build a CBBN-TAN from four clusters. For the same database, it takes 563 CPU seconds to build a BN-BAN*, while it takes 957 CPU seconds to build CBBN-BAN* which is only 1.7 times slower.

Conclusions and Future Work

In this paper, we have proposed a new approach to learn Bayesian classifiers from data. This approach uses a clustering technique to organize the data into semantically sound clusters, thereby representing the domain knowledge in a more expressive and accurate way. In particular, using our novel approach, we were able to learn Bayesian classifiers that can capture finer levels of dependency assertions than possible in traditional BNs. We have shown that being able to represent such dependency relationships more accurately can significantly improve the performance of our classifiers.

We plan to extend this work in the following directions: We would like to study Bayesian multi-net classifiers since we believe that they are a special case of our CBBN classifiers.

We believe that the semantics of the case indices in CBBNs are probabilistically sound. In fact, we will formally demonstrate that CBBNs are a special case of Bayesian Knowledge Bases (Santos Jr, Santos and Shimony 2003).

We will also explore multi-source data sets using our CBBN classifiers. In such data sets, assuming fixed relationships among attributes for the whole domain is inappropriate. Different sources (i.e., experts) might have different organizations for the domain, hence, they might allow different relationships among the corresponding attributes.

Finally, we suggested using the k-modes clustering algorithm and ran our experiments with three different values of k and with α adjusted by the user. Although we obtained good results in all runs, there is no guarantee that these are the best results possible. We would like to find a procedure to optimize k and α for the best classification accuracy.

Acknowledgements

This work was supported in part by Air Force Office of Scientific Research Grant No. F49620-03-1-0014.

References

Cheng, J., Bell, D., and Liu, W. 1997. Learning Belief Networks from Data: An Information Theory Based Approach. In *Proceedings of the Sixth ACM International Conference on Information and Knowledge Management*.

Cheng, J., and Greiner, R. 1999. Comparing Bayesian Network Classifiers. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*.

Cheng, J., and Greiner, R. 2001. Learning Bayesian Belief Network Classifiers: Algorithms and Systems. In *Proceedings of the Fourteenth Canadian Conference on Artificial Intelligence*.

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

Friedman, N., Geiger, D., and Goldszmidt M. 1997. Bayesian Network Classifiers. *Machine Learning* 29:131-161.

Han, J., and Kamber, M. 2001. *Data Mining Concepts and Techniques*. San Francisco:Morgan Kaufmann

Huang, Z. 1998. Extensions to the k-means Algorithm for Clustering Large Data Sets. *Data Mining and Knowledge Discovery* 2(3):283-304.

Lam, W., and Bacchus, F. 1994. Learning Bayesian Belief Networks: An Approach Based on the MDL Principle. *Computational Intelligence* 10(4)

Langley, P., Iba, W., and Thompson, K. 1992. Induction of Selective Bayesian Classifiers. In *Proceedings of the National Conference of Artificial Intelligence*, 223-228. Menlo Park, CA:AAAI Press.

Quinlan, J. R. 1993. *C4.5:Programs for Machine Learning*. San Mateo:Morgan Kaufmann

Santos, E. Jr.; Santos, E. S.; and Shimony, S. E. 2003. Implicitly Preserving Semantics During Incremental Knowledge Base Acquisition Under Uncertainty. *International Journal of Approximate Reasoning* 33(1):71-94