Learning Maximum Likelihood Semi-Naive Bayesian Network Classifier

Kaizhu Huang, Irwin King, and Michael R.Lyu

The Chinese University of Hong Kong
Department of Computer Science and Engineering
Shatin, N.T. Hong Kong SAR
{kzhuang, king, lyu}@cse.cuhk.edu.hk

Abstract— **In this paper, we propose a technique to construct a sub-optimal semi-naive Bayesian network when given a bound on the maximum number of variables that can be combined into a node. We theoretically show that our approach has a less computation cost when compared with the traditional semi-naive Bayesian network. At the same time, we can obtain a resulting sub-optimal structure according to the maximum likelihood criterion. We conduct a series of experiments to evaluate our approach. The results show our approach is encouraging and promising.**

Keywords— **Bayesian network, Semi-Naive, Bound, Integer programming .**

I. Introduction

Classification is a basic problem in data analysis and machine learning field. Learning accurate classifiers from data has been an active research topic in recent years. Different approaches have been proposed to learn a classifier from preclassified data sets. Among them are Statistical Neural networks [4], Decision trees [6], and Support Vector Machines [11].

Regarded as a knowledge representation method under uncertainty, Bayesian network did not come into classification experts' view until the discovery of Naive Bayesian network classifier (NB) [1], [14]. The NB network is a very simple Bayesian network, which assumes every variable (feature) of the data is independent given the class label. With this assumption the probability induction is made easily and efficiently. Figure 1 is an example of NB. In this example, given a set of symptoms, one wants to determine whether these symptoms give rise to a particular disease as shown in Fig. 1. Experts usually judge the probability of a disease's occurrence by examining the existence of some symptoms. Similarly in Naive Bayesian networks, according to the independency assumption, it is easy to write down the following equation:

$$P(Disease \mid s_1, s_2, s_3, s_4) \propto$$

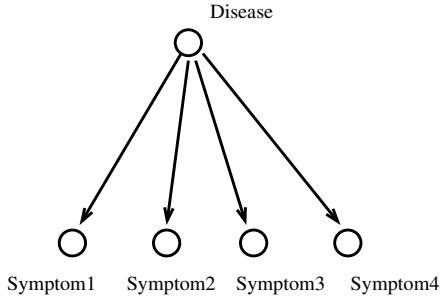

Fig. 1. An example of Naive Bayesian network

$$\prod_{i=1}^{4} P(s_i \mid Disease) P(Disease) \qquad (1)$$

Where, s_i represents the ith *Symptom*, $1 \leq i \leq 4$. Given an instance of each variable (symptom), for example *(true, false, true, false)*, according to the equation above, we can obtain the final probability of the disease hypothesis by calculating $P(Disease = true \mid s_1 = true, s_2 = false, s_3 = true, s_4 = false)$ and $P(Disease = false \mid s_1 = true, s_2 = false, s_3 = true, s_4 = false)$. This computation can be made easily and fast according to Eq. (1) under the independence assumptions. Then we take the judgment with larger probability value between $Disease = false$ and $Disease = true$ as the diagnosis. With such a simple structure, NB is surprisingly effective in many application domains even when compared with state-of-the-art classifiers [13]. This success triggered experts to explore more deeply into Bayesian networks as classifiers.

Since the strict assumption in NB can be violated strongly in many cases, researchers have wondered if the performance will be better when the strong independence assumption between variables in NB is relaxed. Then the so-called *semi-naive Bayesian network*(SNB) [5] [13] was invented. SNB constrains the network's structure by dividing the variables into several sets based on some criterions.

Inside each set, the variables are assumed dependent while inter-sets are independent, given the class label. Also other classifiers such as K2 [8], TANB [3] were discovered based on more complex structures.

The structure complexity in Bayesian network can be defined as the number of the parameters which are needed to quantify the network. In the NB example of Fig. 1, to quantify the network, we only need to record the parameters: $P(s_1|Disease)$, $P(s_2|Disease)$, $P(s_3|Disease)$, $P(s_4|Disease)$, and $P(Disease)$. For binary symptoms and disease example, there are $(2\times(2-1))\times 4+(2-1)=9$ values (parameters) we need to record. However if Fig. 1 is represented as a complete graph, we have to record $P(s_1,s_2,s_3,s_4|Disease)$ and $P(Disease)$ which will have $(2\times(2^4-1))+(2-1)=31$ parameters. To understand the complexity of the Bayesian network, we can simply regard "a network with more edges will be possibly more complex than the one with fewer edges". (It is not always true especially when the variables can take on different number of values.) NB can be considered as the simplest Bayesian network while a complete graph can be regarded as the most complex network.

Theoretically a more complex structure will approximate the training dataset more accurately. So it seems that a more complex structure will have a more accurate classifier. However it is absolutely not the case. It is shown that complex structure will often cause an over-fitting problem, that is the classifier learns the training data perfectly while having a high error rate in predicting a new data [3]. It seems that we are facing a dilemma: if we prefer the simple structure, the restriction caused by its simplicity may be violated frequently; if we prefer a complex structure, the over-fitting problem may occur.

One of the trade-off strategies is to restrict the network's complexity first and then to explore the best structure which can approximate the dataset. In fact this strategy has been done recently in [9]. They proposed a bounded tree-width graph approach. Tree-width can be considered to be one less than minimum possible value of the number of nodes involved in the maximum completely connected subnetwork of specific networks, which are transformed from the original network [12]. And this tree-width bound can avoid the network into a complex network. They firstly prove that it is NP problem to find the optimal $l-treewidth$ structure, where l is greater than 1. And then they give out an approximation solution based on a combination technique: Integer Programming (IP) technique. It is believed that their approach is the first combinatorial formulation of the learning problem. However their approximation is somewhat far away from the optimal solution. It is reported that their approximation bound to the optimal solution is about 1/324 when the tree-width is equal to 3.

In this paper, we use this strategy in building an optimal *K-bounded-large-node semi-naive Bayesian network (BLN-SNB).* K-bounded-large-node means that "the cardinality of every subset in SNB is not greater than the value K". Detailed issues about this can be seen in Section II. At the same time we found that even though in [9] they cannot find an accurate approximation to the hypertree, their methods can be used in searching an accurate BLN-SNB. In this way we restrict the network in a not so complex SNB structure and then we try to find the optimal structure in this restriction.

One interesting observation is that our proposed SNB has a polynomial time cost in searching a sub-optimal structure. We do not need a great number of iterations on the training dataset as in traditional SNB [5]. Also we do not just combine pairs of attributes as in [13] since in our approach we can combine any number of variables fewer than a bound. At the same time, in [5] there is no evidence that shows a sub-optimal or optimal structure can be maintained while our approach is shown to be suboptimal given a bound on the cardinality of the subset based on the maximum log likelihood criterion.

In the following we first give the BLN-SNB model definition. Then we reduce the optimization problem of this model into a *K-regular semi-naive network* which means each subset of SNB has the same cardinality K. After that we transform the searching procedure into an integer programming (IP) problem in a similar method as [9] and we approximate the IP solution in a linear programming (LP) method which is polynomial time in computational complexity. We show the computational complexity analysis in Section IV. And in Section V, we show our experimental results. Finally we conclude our paper with discussion and conclusion sections.

II. BLN-SNB Model Definition

See Fig. 2, our BLN-SNB model is defined as:

BLN-SNB Model definition: Given a dataset D with a class C, n variables $A_1, A_2, \ldots, A_n$ and a bound K, BLN-SNB is a maximum likelihood Bayesian network which satisfies the following conditions:

1. It is composed of m large nodes $AS_1, AS_2, \ldots, AS_m$, $1 \le m \le n$, each large node AS_l is a subset of $\{A_1, A_2, \ldots, A_{n-1}, A_n\}$.
2. There is no coverage among the large nodes and

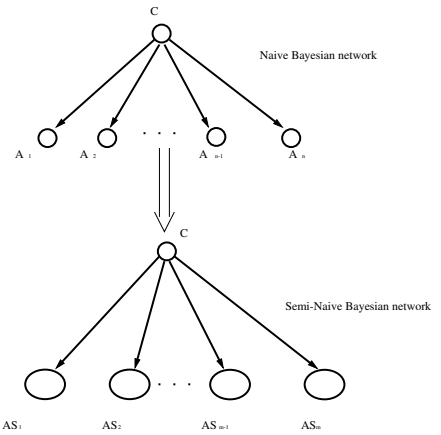

Fig. 2. Semi-Naive Bayesian network:AS_i is the combination of some variables, and $AS_i \cap AS_j = \phi, i \neq j$

their union forms the variables set.

$$\begin{aligned} &AS_i \cap AS_j = \phi, \\ &\quad \text{for} \quad i \neq j, \text{and} \quad 1 \leq i, j \leq m, \\ &AS_1 \cup AS_2 \cup \ldots \cup AS_m \\ &= \{A_1, A_2, \ldots, A_n\} \end{aligned} \tag{2}$$

3. Given the class label C, AS_i is independent with AS_j for $i \neq j$.

$$\begin{aligned} &P(AS_i, AS_j \mid C) = P(AS_i \mid C)P(AS_j \mid C) \\ &\quad \text{for} \quad i \neq j, \text{and} \quad 1 \leq i, j \leq m \end{aligned} \tag{3}$$

4. The cardinality of each large node AS_l ($1 \leq l \leq m$) is no greater than K.

Item 4 above is used to control the network complexity. We can see that if K is scaled up into n, it will be a complete graph. This structure is obviously a perfect approximation to the data with a certainly heavy over-fitting problem as well. On the other hand, if K is set to 1, it is degraded into Naive Bayesian network.

III. Maximum Likelihood Bounded-Large-Node Semi-Naive Bayesian network

A. Reducing BLN-SNB Optimization Problem

Lemma 1: The log likelihood of a SNB, represented by l_{SNB} can be written into the following form:

$$l_{SNB} = -\sum_{i=1}^{m} H(AS_i), \tag{4}$$

where $H(AS_i)$ is the entropy of variable subset AS_i. The entropy among a k-variable subset $\{X_1, X_2, \ldots, X_k\}$ can be defined as:

$$\begin{aligned} &H(X_1, X_2, \ldots, X_k) \\ &= -\sum_{x_1, \ldots, x_k} P(x_1, \ldots, x_k) \log P(x_1, \ldots, x_k) \end{aligned} \tag{5}$$

where low-case character x_i represents the assignment of the value to the variable X_i, $1 \leq i \leq k$.

Lemma 2: Let μ and μ' are two SNBs over dataset D. If μ' is coarser than μ then, μ' provides a better approximation than μ over D.

The *Coarser* concept can be defined in this way. If μ' can be obtained by combining the large nodes of μ without splitting the large node of μ, then μ' is coarser than μ. The details of the proof of Lemma 1 and Lemma 2 are shown in the Appendix.

According to Lemma 2, given a bound K, we should not separate the variables set into too many *small* subsets. Or it is more possible that we can combine some of these small subsets into a new subset whose cardinality is no greater than K, thus the new SNB will be coarser than the old one. From this viewpoint, we reduce the searching space of BLN-SNB into a K-regular SNB space since there are no possibility that a SNB coarser than K-regular SNB exists in the K-bound. Even though it is reasonable to search the maximum likelihood SNB in the K-regular-SNB space, we will not say that: a K-regular SNB is absolutely better than a non-K-regular SNB with the biggest cardinality no more than K. It is obvious some non-K-regular SNBs cannot be combined into a K-regular SNB. Thus in such a way, we reduce the searching space into a sub-space of K-bound SNB.

Thus according to Lemma 1 and Lemma 2 the BLN-SNB problem defined in Section II is transformed into the following:

BLN-SNB Problem: From attributes set, finding $m = [n/K]$ K-cardinality subsets, which satisfy the SNB conditions, to maximize the log likelihood as shown in Eq. (4). Here $[x]$ means rounding the x to the nearest integer.

B. Transforming into IP problem

It is obvious that the BLN-SNB problem is a combinatorial problem. However it is not acceptable that we use a greedy search method to find the optimization solution. It can be easily calculated that the greedy search cost will be $\frac{n!}{(K!)^{[n/K]}}$. For a simple example $n = 18, k = 3$, the cost will be up to 13 billion!

In fact we can write the BLN-SNB into the following IP problem:

$$\text{Min} \sum_{V_1, V_2, \ldots, V_K} x_{V_1, V_2, \ldots, V_K} H(V_1, V_2, \ldots, V_K) \tag{6}$$

$$(\forall V_K) \sum_{V_1,V_2,\ldots,V_{K-1}} x_{V_1,V_2,\ldots,V_K} = 1 \quad (7)$$

$$x_{V_1,V_2,\ldots,V_K} = \{0,1\} \quad (8)$$

Here $V_1, V_2, \ldots, V_k$ represents any K variables. Eq. (7) describes that: for any variable, it can just belong to one subset, i.e., when it comes out in one subset, it must not be in another subset. $H(V_1, V_2, \ldots, V_K)$ representing the entropy of variable set $\{V_1, V_2, \ldots, V_K\}$, can be easily calculated from the data.

IP problem can be solved in many methods such as *Cutting Plane*, *Simulating Annealing*, etc. A tutorial note about IP can be obtained in [7]. Here we approximate the solution of IP via LP problem which can be solved in a polynomial time. By relaxing $x_{V_1,V_2,\ldots,V_K} = \{0,1\}$ into $0 \leq x_{V_1,V_2,\ldots,V_K} \leq 1$, the IP problem is transformed into a LP problem. Then a rounding procedure is conducted on the solution of LP. We assume the set of all $x_{V_1,V_2,\ldots,V_K}$ as X.

1. Until all the variables are covered.
2. Set the maximum $x_{V_1,V_2,\ldots,V_K}$ to value 1 in X, record its subscript as $\{V_{M_1}, V_{M_2}, \ldots, V_{M_K}\}$, delete this $x_{\{V_{M_1},V_{M_2},\ldots,V_{M_K}\}}$ from X.
3. Set all $x_{V_1,V_2,\ldots,V_K}$ to 0 , which have the coverage with $\{V_{M_1}, V_{M_2}, \ldots, V_{M_K}\}$. Delete all these x from X.
4. Goto 1.

As discussed in [7], a LP solution provided much information for the one of IP. Approximating IP solution by LP may reduce the accuracy of the SNB while it can decrease the computational cost. Shown in our experiments, a LP approximation really stands for much information of the IP solution.

C. When n/K is not an integer

We may notice that if n cannot be divided by K exactly, i.e., (n mod K)= $l \neq 0$, we will not be able to find a K-Regular-SMB since one subset will have only l variables. In solving this problem, we modified the method into the following:

1. Assume (n mod K)= $l \neq 0$, among all the l-subset of variables set, select the one which has the minimum entropy. We assume this l-subset is $AS_{mi} = \{A_{mi_1}, A_{mi_2}, \ldots, A_{mi_l}\}$. Let $B = \{A_1, A_2, \ldots, A_{n-1}, A_n\} \backslash AS_{mi}$.
2. Perform the optimization on the attributes set B as shown in the last section.

Actually, the solution of the modification approach above is in some sense a local minimum solution of LP problem. From Lemma 1 we know that to maximize the log likelihood, the entropy of every subset should be as little as possible. That is why we choose the minimum entropy among all the l-subsets in the beginning.

IV. Computational Complexity Analysis of BLN-SNB

In this section, we conduct a simple computational complexity analysis for BLN-SNB.

A strong empirical evidence shows that classical LP optimization methods such as *simplex* only takes $O(w)$ iterations to find an optimal solution with w equality constraints [10]. Each iteration costs $O(wN)$ arithmetic operations where N is the number of variables to be solved. For our LP problem of Eq. (6), there are totally $N = C_n^K$ variables $x_{V_1,V_2,\ldots,V_K}$ which need to be solved and w is equal to n. Accordingly the computational cost in our optimization process is about $n^2 C_n^K$. In the other hand, $r^K C_n^K$ operations are needed to computing the K-variable entropy in 6. Here r is the maximum number of values a variable can take on. Accordingly the total cost will be $(n^2 + r^K) C_n^K$. It will be a $O(n^{K+2})$ time cost, when $K \ll n$.

However, in the traditional SNB [5], the computational cost is exponential. It is said that: *the number of iterations over the training dataset is approximately equal to the number of values of all attributes.* For a simple example in which every variable has r values, the combination cost will be r^n. it is an exponential cost. As the variable dimension grows, the cost difference between the BLN-SNB and Kononenko SNB will be bigger and bigger. Especially in order to resist the over-fitting problem the K has to be fixed at a small number.

On the other hand, the approaches by Pazzani [13] is impractical for even three attributes combination even though their approaches have a low cost report of $O(n^3)$ when combining two attributes. "Although it would be possible to consider joining three (or more attributes), the computational complexity makes it impractical for most databases" [13]. Thus the accuracy of Pazzani SNB may be limited in this sense. Table I shows the analysis result of the above. Here "Max ♯" means the maximum number of variables which involve in a large node.

TABLE I
Computation cost table

Methods	BLN-SNB	Kononenko	Pazzani
Cost	$O(n^{K+2})$	$O(r^n)$	$O(n^3)$
Max ♯	K	$N \geq 1$	2

V. Experimental results

To evaluate the performance of BLN-SNB approach, we conduct a series of experiments on *Tic-tac-toe* and *Vote* databases from UCI Machine learning Repository [15]. Since NB is a competitive model even when compared with the state-of-art classifier, we only conduct the performance comparison between our model and NB. In both datasets, we use a five fold cross validation described by Kohavi et. al. in [2]. We test 2-BLN-SNB and 3-BLN-SNB.

Table II describes the datasets used in our experiments. We build one BLN-SNB for each class in both data sets. When used in recognition, we output the class with the higher probability. From Table III, we can see that there is a significant increase in recognition rate when using BLN-SNB compared with NB in *Tic-tac-toe* dataset both in training accuracy and test accuracy. The performance of BLN-SNB in *Vote* is slightly high or nearly the same in test accuracy and significantly higher in training accuracy than NB.

TABLE II
Description of data sets used in the experiments

Dataset	Variables	Class	train	test
Tic-tac-toe	15	2	435	CV-5
Vote	9	2	958	CV-5

TABLE III
Recognition rate

Training accuracy			
DB	NB	BLN-SNB	
		K=2	K=3
Tic	71.30 ± 0.64	74.09 ± 1.22	81.47 ± 2.21
Vote	90.75 ± 0.27	94.02 ± 0.76	96.03 ± 0.85
Testing accuracy			
DB	NB	BLN-SNB	
		K=2	K=3
Tic	70.77 ± 1.38	72.65 ± 1.58	78.39 ± 3.00
Vote	90.11 ± 1.74	90.26 ± 1.81	90.25 ± 2.04

From the experiments, we found that in all the CV-5 training process, that LP solution is part of IP solution only happens 3 times in all of 20 times training. This means that our LP approximation to the IP solution is reasonable. See Table IV, we show the IP and LP solutions only in one of CV-5 training in 2-SNB. In Vote database of Table IV, the BLN-SMB LP solution of Class 1 is not the integer solution. It is then rounded into the integer solution as in the right two columns of Table IV according to our rounding scheme. In the last line of Class 1 in Table IV $x_4 = 1$ means $H(X_4)$ is the minimum entropy in all the 1-subset in Class 1. It comes from the modified approach introduced in Subsection III-C since the result of 15 mod 2 = 1 is not equal to zero.

TABLE IV
LP solution and Rounded IP solution(K=2) of $Vote$

LP solution		Rounded LP solution	
Class1	Class2	Class1	Class2
$x_{1,2} = 1$	$x_{1,13} = 1$	$x_{1,2} = 1$	$x_{1,13} = 1$
$x_{3,10} = .5$	$x_{2,9} = 1$	$x_{3,10} = 1$	$x_{2,9} = 1$
$x_{5,8} = 1$	$x_{3,14} = 1$	$x_{5,8} = 1$	$x_{3,14} = 1$
$x_{6,12} = .5$	$x_{4,6} = 1$	$x_{6,12} = 0$	$x_{4,6} = 1$
$x_{6,14} = .5$	$x_{7,8} = 1$	$x_{6,14} = 1$	$x_{7,8} = 1$
$x_{7,13} = 1$	$x_{10,11} = 1$	$x_{7,13} = 1$	$x_{10,11} = 1$
$x_{9,15} = 1$	$x_{12,15} = 1$	$x_{9,15} = 1$	$x_{12,15} = 1$
$x_{10,11} = .5$		$x_{10,11} = 0$	
$x_{12,14} = .5$		$x_{12,14} = 0$	
$x_{3,11} = .5$		$x_{3,11} = 0$	
$x_{11,14} = 0$		$x_{11,14} = 1$	
$x_4 = 1$	$x_5 = 1$	$x_4 = 1$	$x_5 = 1$

VI. Discussion

We can see that the complete graph and the Naive Bayesian network are special cases of our BLN-SMB. In BLN-SMB when K is equal to n, it is a complete graph. When K is equal to 1, it is degraded into the Naive Bayesian network.

At the same time, our approach is a bound strategy. To resist the over-fitting problem, the K has to be chosen as some small value. In the $Vote$ dataset of our experiments, even when K is chosen as 2 or 3, the BLN-SNB has a tendency towards overfitting. This shows that the Naive Bayesian network may be the better model for this dataset.

VII. Conclusion

In this paper,we proposed a bounded-Large-Node Semi-Naive Bayesian network model. When compared with the traditional Semi-Naive Bayesian network, our model can be solved in a polynomial time and also can maintain a sub-optimal fitness in Semi-Naive network domain. Our experiments show that our approach can both increase the training accuracy and testing accuracy compared with Naive Bayesian network.

VIII. Appendix

Proof for Lemma 1:

Proof: Let there are n variables which are represented respectively by A_i, $1 \leq i \leq n$. And according to the SNB assumption, the variable set can be partitioned into m subsets without coverage among them. We assume the subsets respectively as B_i, $1 \leq i \leq m$. We use low-case characters represent the assignments of values to the variables. So b_i is a vector which represents an assignment of values to the variables in B_i. We use $(B_1, \ldots, B_m)$ as the short form of $(B_1, B_2, \ldots, B_{m-1}, B_m)$. The log likelihood over a data set can be written into the following:

$$\begin{aligned} & L_{SNB} \\ = & \sum_{(A_1,\ldots,A_n)} P(a_1,\ldots,a_n)\log P(a_1,,\ldots,a_n) \\ = & \sum_{(B_1,\ldots,B_m)} P(b_1,\ldots,b_m)\log P(b_1,\ldots,b_m) \\ = & \sum_{(B_1,\ldots,B_m)} P(b_1,\ldots,b_m)\sum_{i=1}^{m}\log P(b_i) \\ = & \sum_{i=1}^{m}\sum_{(B_1,\ldots,B_m)} P(b_1,\ldots,b_m)\log P(b_i) \\ = & \sum_{i=1}^{m}\sum_{B_i} P(b_i) log P(b_i) \\ = & -\sum_{i=1}^{m} H(B_i) \end{aligned} \quad (9)$$

■

Proof for Lemma 2:

Proof: We just consider a simple case, a general case proof is much similar.

Consider one partition as $\mu = (B_1, B_2, \ldots, B_m)$ and another partition as

$$\mu_1 = (B_1, B_2, \ldots, B_{m-1}, B_{m1}, B_{m2})$$

,where we have:

$$\begin{aligned} & B_{m1} \cap B_{m2} = \phi \quad \text{and} \\ & B_{m1} \cup B_{m2} = B_m \end{aligned}$$

According to the proof of Lemma 1 above, we have:

$$\begin{aligned} L_{SNB_\mu} &= -\sum_{i=1}^{m} H(B_i) \\ &= -\sum_{i=1}^{m-1} H(B_i) - H(AS_m) \end{aligned} \quad (10)$$

According to Entropy theory, $H(XY) \leq H(X) + H(Y)$. We can write Eq. (10) into:

$$\begin{aligned} L_{SNB_\mu} &= -\sum_{i=1}^{m-1} H(B_i) - H(B_m) \\ &\geq -\sum_{i=1}^{m-1} H(B_i) - H(B_{m1}) - H(B_{m2}) \\ &= L_{SNB_{\mu_1}} \end{aligned} \quad (11)$$

■

References

[1] R. Duda and P. Hart, *Pattern classification and scene analysis,* John Wiley & Sons, 1973.

[2] R. Kohavi, "A study of cross validation and bootstrap for accuracy estimation and model selection", *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, San Francisco, CA:Morgan Kaufmann, pp. 338-345, 1995.

[3] N. Friedman, D. Geiger and M. Goldszmidt, "Bayesian Network Classifiers", *Machine Learning*, vol. 29, pp. 131-161, 1997.

[4] M. Pankaj and W. W. Benjamin , *Artificial neural networks : concepts and theory,* Los Alamitos, Calif. : IEEE Computer Society Press, 1992.

[5] I. Kononenko, "Semi-naive Bayesian classifier', *Proceedings of sixth European Working Session on Learning,* Springer-Verlag, pp. 206-219, 1991.

[6] J. R. Quinlan, *C4.5 : programs for machine learning,* San Mateo, Calif.:Morgan Kaufmann Publishers, 1993.

[7] T. Michael, "A Tutorial on Integer Programming", *http:// mat.gsia.cmu.edu/ orclass/ integer/ integer.html*, The Operations Research Faculty of GSIA.

[8] G. F. Cooper and E. Herskovits, "A Bayesian Method for the induction of probabilistic networks from data", *Machine Learning*, vol. 9, pp. 309-347, 1992.

[9] Karger David and Srebro Nathan, "Learning Markov networks: maximum bounded tree-width graphs", *Symposium on Discrete Algorithms,* pp. 392-401, 2001.

[10] B. Demitris and N. T. John, *Induction of Linear Optimization*, Athena Scientific, Belmont, Massachusetts, 1997.

[11] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

[12] H. L. Bodlaender, "A tourist guide through treewidth", *Acta Cybernetica*, vol. 11", pp. 1-21, 1993.

[13] M. J. Pazzani, "Searching dependency in Bayesian classifiers", *Learning from data: Artificial intelligence and statistics V*, New York, NY:Springer-Verlag, editor D. Fisher and H.-J. Lenz, pp. 239-248, 1996.

[14] P. Langley, W. Iba and K. Thompson, "An analysis of Bayesian classifiers", *In Proceedings of AAAI-92*, pp. 223-228, 1992.

[15] M. Murphy, "UCI repository of machine learning databases", *http: //www.ics.uci.edu/ mlearn/ MLRepository.html.*