Learning Bayesian Belief Network Classifiers: Algorithms and System

Jie Cheng* **Russell Greiner**
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1 Canada
Email: {jcheng, greiner}@cs.ualberta.ca

Abstract

This paper investigates the methods for learning predictive classifiers based on Bayesian belief networks (BN) -- primarily unrestricted Bayesian networks and Bayesian multi-nets. We present our algorithms for learning these classifiers, and discuss how these methods address the overfitting problem and provide a natural method for feature subset selection. Using a set of standard classification problems, we empirically evaluate the performance of various BN-based classifiers. The results show that the proposed BN and Bayes multi-net classifiers are competitive with (or superior to) the best known classifiers, based on both BN and other formalisms; and that the computational time for learning and using these classifiers is relatively small. These results argue that BN based classifiers deserve more attention in the data mining community.

1 Introduction

Many tasks – including fault diagnosis, pattern recognition and forecasting – can be viewed as *classification*, as each requires identifying the class labels for instances, each typically described by a set of features (attributes). Learning accurate classifiers from pre-classified data is a very active research topic in machine learning and data mining. In the past two decades, many algorithms have been developed for learning decision-tree and neural-network classifiers. While Bayesian networks (BNs) (Pearl 1988) are powerful tools for knowledge representation and inference under conditions of uncertainty, they were not considered as classifiers until the discovery that Naïve-Bayes, a very simple kind of BNs that assumes the attributes are independent given the class node, are surprisingly effective (Langley *et al.* 1992).

This paper further explores this role of BNs. Section 2 provides the framework of our research, introducing Bayesian networks and describing standard approaches to learning simple Bayesian networks, then briefly describing five classes of BNs – Naïve-Bayes, tree augmented Naïve-Bayes (TANs), BN augmented Naïve-Bayes (BANs), Bayesian multi-nets and general BNs (GBNs). Section 3 describes methods for learning GBNs and Bayesian multi-nets. This section also describes our approaches to avoiding overfitting and to selecting feature subsets. Section 4 presents and analyzes our experimental results, over a set of standard learning problems obtained from the UCI Machine Learning Repository (Murphy and Aha, 1995).

* Now at IBM Toronto Lab, Canada.

2 Framework

2.1 Bayesian networks

A Bayesian network $B = \langle N, A, \Theta \rangle$ is a directed acyclic graph (DAG) $\langle N, A \rangle$ where each node $n \in N$ represents a domain variable (eg, a dataset attribute), and each arc $a \in A$ between nodes represents a probabilistic dependency, quantified using a conditional probability distribution (CP table) $\theta_i \in \Theta$ for each node n_i (see Pearl 1988). A BN can be used to compute the conditional probability of one node, given values assigned to the other nodes; hence, a BN can be used as a classifier that gives the *posterior probability distribution* of the class node given the values of other attributes. Typically, the associated classifier system would then return class node value with the largest posterior probability. A major advantage of BNs over many other types of predictive models, such as neural networks, is that the Bayesian network structure represents the inter-relationships among the dataset attributes (see Figure 10). Human experts can easily understand the network structures and if necessary modify them to obtain better predictive models. By adding decision nodes and utility nodes, BN models can also be extended to *decision networks* for decision analysis (Neapolitan, 1990).

Applying Bayesian network techniques to classification involves two sub-tasks: BN learning (training) to get a model and BN inference to classify instances. In Section 4, we will demonstrate that learning BN models can be very efficient. As for Bayesian network inference, although it is NP-hard in general (Cooper, 1990), it reduces to simple multiplication in our classification context, when all the values of the dataset attributes are known.

2.2 Learning Bayesian networks

The two major tasks in learning a BN are: learning the graphical structure, and then learning the parameters (CP table entries) for that structure. As it is trivial to learn the parameters for a given structure that are optimal for a given corpus of complete data – simply use the empirical conditional frequencies from the data (Cooper and Herskovits 1992)[1] – we will focus on learning the BN structure.

There are two ways to view a BN, each suggesting a particular approach to learning. First, a BN is a structure that encodes the joint distribution of the attributes. This suggests that the best BN is the one that best fits the data, and leads to the *scoring-based* learning algorithms, that seek a structure that maximizes the Bayesian, MDL or Kullback-Leibler (KL) entropy scoring function (Heckerman 1995; Cooper and Herskovits 1992).

Second, the BN structure encodes a group of conditional independence relationships among the nodes, according to the concept of *d-separation* (Pearl 1988). This suggests learning the BN structure by identifying the conditional independence relationships among the nodes. Using some statistical tests (such as Chi-squared or mutual information), we can find the conditional independence relationships among the attributes and use these relationships as constraints to construct a BN. These algorithms are referred as *CI-based* algorithms or constraint-based algorithms (Spirtes and Glymour 1996; Cheng *et al.* 1997a).

Heckerman *et al.* (1997) compare these two general learning, and show that the scoring-based methods often have certain advantages over the CI-based methods, *in terms of modeling a distribution*. However, Friedman *et al.* (1997) show theoretically that the general scoring-based

[1] These parameters maximize the likelihood of the data for this structure, which corresponds to minimizes the KL-divergence.

methods may result in poor *classifiers* since a good classifier maximizes a different function – *viz.*, classification accuracy. Greiner *et al.* (1997) reach the same conclusion, albeit via a different analysis. Moreover, the scoring-based methods are often less efficient in practice.

This paper demonstrates that the CI-based learning algorithms can effectively learn BN *classifiers*.

2.3 Bayesian network classifiers

We will consider the following five classes of BN classifiers: Naïve-Bayes, Tree augmented Naïve-Bayes (TANs), Bayesian network augmented Naïve-Bayes (BANs), Bayesian multi-nets and general Bayesian networks (GBNs).

2.3.1 Naïve-Bayes

A Naïve-Bayes BN, as discussed in Duda and Hart (1973), is a simple structure that has the class node as the parent node of all other nodes (see Figure 1). No other connections are allowed in a Naïve-Bayes structure.

Naïve-Bayes has been used as an effective classifier for many years. Unlike many other classifiers, it is easy to construct, as the structure is given *a priori* (and hence no *structure* learning procedure is required). Naïve-Bayes assumes that all the features are independent of each other. Although this independence assumption is obviously problematic, Naïve-Bayes has surprisingly outperformed many sophisticated classifiers over a large number of datasets, especially where the features are not strongly correlated (Langley *et al.* 1992).

In recent years, a lot of effort has focussed on improving Naïve-Bayesian classifiers, following two general approaches: selecting feature subset (Langley and Sage 1994; Kohavi and John 1997; Pazzani 1995) and relaxing independence assumptions (Kononenko 1991; Friedman *et al.* 1997). Section 2.3.2 through Section 2.3.4 introduce BN models that extend Naïve-Bayes by allowing dependencies among the features.

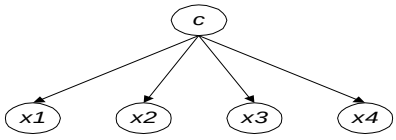

Figure 1: A simple Naïve Bayes structure

2.3.2 Tree Augmented Naïve-Bayes (TAN)

TAN classifiers extend Naïve-Bayes by allowing the attributes to form a tree – cf, Figure 2: here c is the class node, and the features x_1, x_2, x_3, x_4, without their respective arcs from c, form a tree. Learning such structures can be easily achieved by using a variation of the Chow-Liu (1968) algorithm. The performance of TAN classifiers is studied in Friedman *et al.* (1997) and Cheng and Greiner (1999).

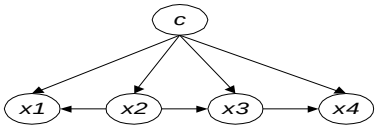

Figure 2: A simple TAN structure

2.3.3 BN Augmented Naïve-Bayes (BAN)

BAN classifiers extend TAN classifiers by allowing the attributes to form an arbitrary graph, rather than just a tree (Friedman *et al.* 1997) – see Figure 3. Learning such structures is less efficient. Friedman *et al.* (1997) presents a minimum description length scoring method for learning BAN. Cheng and Greiner (1999) study a different algorithm based on conditional independence (CI) tests. Both papers also investigate the performance of BAN classifiers.

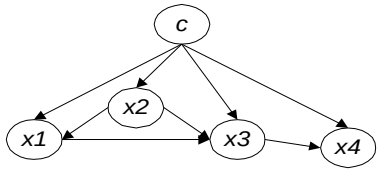

Figure 3: A simple BAN structure

2.3.4 Bayesian Multi-net

Bayesian Multi-nets were first introduced in (Geiger and Heckerman, 1996) and then studied in (Friedman *et al.*, 1997) as a type of classifiers. A Bayesian multi-net is composed of the prior probability distribution of the class node and a *set* of local networks, each corresponding to a value that the class node can take (see Figure 4). Bayesian multi-nets can be viewed as a generalization of BANs. A BAN forces the relations among the features to be the same for all the values that the class node takes; by contrast a Bayesian multi-net allows the relations among the features to be different – i.e., for different values the class node takes, the features can form different local networks with different structures. In a sense, the class node can be also viewed as a parent of all the feature nodes since each local network is associated with a value of the class node. Note that these multi-net structures are strictly more expressive than Naïve-Bayes, TAN or BAN structures.

To motivate this, consider the tasks in pattern recognition – different patterns may have different relationships among features.

As multi-net structure imposes no restrictions on the relationships among the attributes, they are a kind of *unrestricted* BN classifier. However, while multi-net is more general than BAN, it is often less complex than BAN since some of the local networks can be simpler than others, while BAN needs to have a complex structure in order to express all the relationships among the features.

Figure 4 A simple Bayesian multi-net

2.3.5 General Bayesian Network (GBN)

GBN is another kind of unrestricted BN classifier, however, of a different flavor. A common feature of Naïve Bayes, TAN, BAN and multi-net is that the class node is treated as a special node – the parent of all the features. However, GBN treats the class nodes as an ordinary node (see Figure 5), it is not necessary a parent of all the feature nodes. The learning methods and the performance of GBN for classification are studied in (Friedman *et al.*, 1997; Cheng and Greiner 1999).

Comparison: To compare GBNs and Bayesian multi-nets, observe that GBNs assume that there is a single probabilistic dependency structure for the entire dataset; by contrast, multi-nets allow different probabilistic dependencies for different values of the class node. This suggests that GBN classifiers should work better when there is a single underlying model of the dataset and multi-net classifier should work better when the underlying relationships among the features are very different for different classes.

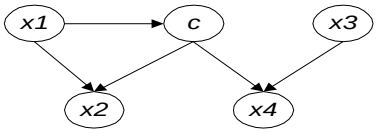

Figure 5: A simple GBN

2.4 Motivations

This work continues our earlier work presented in (Cheng and Greiner 1999). In that paper, we studied the CI-based methods for learning GBN and BAN and showed that our CI-based methods appear not to suffer from the drawbacks of scoring-based methods (see Section 2.2). With a wrapper algorithm (see Section 3.3), these more general types of BN classifiers do work well. This paper continues our research in BN classifiers in the following aspects.

1. Our earlier work suggested that the more general forms of BN classifiers can capture the relationships among the features better and therefore make more accurate predictive models. However, it did not consider an important class of BN classifiers – Bayesian multi-net. Here we evaluate its learning efficiency and performance for classification.
2. A node ordering specifies an order of the nodes, with the understanding no node can be an ancestor of a node that appears earlier in the order. Our earlier work assumed this node ordering was given, which greatly simplified the task of learning the appropriate structure (for the CI-based GBN and BAN learners). Here we investigate the effect of such orderings by learning the BN classifiers with and without node orderings.
3. The learned GBN structure immediately identifies the relevant feature subset – the *Markov blanket* (Section 3.4) around the class node. Here we study the effectiveness of such feature subsets by using it to simplify Bayesian multi-net classifiers.

3 Learning Unrestricted BN Classifiers

This section presents algorithms for learning general Bayesian networks and Bayesian multi-nets. It also presents the wrapper algorithm that can wrap around these two learners to help find good settings for the "independence test threshold", and an algorithm for learning multi-nets using feature subsets.

Figure 6 and Figure 7 sketches the algorithms for learning multi-nets and GBNs. They each use the CBL_i algorithms, which are general purpose BN-learning algorithms: one for the case when node ordering is given (the CBL_1 algorithm – Cheng *et al.* 1997a); the other for the case when node ordering is not given (the CBL_2 algorithm – Cheng *et al.* 1997b).

Both CBL_1 and CBL_2 are CI-based algorithms that use information theory for dependency analysis. CBL_1 requires $O(N^2M)$ mutual information tests to learn a general BN over N attributes from M training cases, and CBL_2 requires $O(N^5M)$ mutual information tests. The efficiency of these algorithms is achieved by a three-phase BN learning algorithm: *drafting*, which is essentially the

Chow-Liu (1968) tree construction algorithm; *thickening*, which adds edges to the draft; and *thinning*, which removes unnecessary edges. As these learners use a finite set of samples, they need to use some threshold $\tau \in \Re^+$ when determining whether some statistical condition is met (see below). Modulo this issue, these algorithms are guaranteed to learn the optimal structure, when the underlying model of the data satisfies certain benign assumptions. For the correctness proof, complexity analysis and other detailed information, please refer to (Cheng *et al.*, 1997a; Cheng *et al*., 1997b).[2]

3.1 MN_i(*S*: training set; *F*: feature set; [*O*: Node Ordering]): *returns* Bayesian Multi-net

1. Partition the training set into subsets S_i, by the values of the class node.
2. For each training subset S_i,
   Call BN-structure learning algorithm CBL_i on S, F (and O if *i=1*)
   Compute the parameters (using observed frequencies) of each local network.
3. Estimate the prior probability distribution of the class node.

Figure 6 The MN_i Algorithm

3.2 GBN_i(*S*: training set; *F*: feature set; [*O*: Node Ordering]): *returns* General Bayesian network

1. Call BN-structure learning algorithm CBL_i on S, F (and O if *i=1*)
2. Find the *Markov blanket* $B \subseteq F$ of the class node.
3. Delete all the nodes that are outside the Markov blanket.
4. Compute the parameters (using observed frequencies)

Figure 7 The GBN_i Algorithm

3.3 The wrapper algorithm

Unlike Naïve-Bayes and TAN learners, there is no restriction on the structures that the GBN learner and multi-net learner can learn. Therefore, it is possible that a BN model will *overfit* – ie, fit the training set too closely instead of generalizing, and so will not perform well on data outside the training samples. In (Cheng and Greiner 1999), we proposed a wrapper algorithm to determine the best setting for the threshold τ; we observed that this increased the prediction accuracy up to 20% in our experiments. Suppose X-learner is a learning algorithm for classifier X, the wrapper algorithm can wrap around X-learner in the following way.

When the training set is not large enough, k-fold cross validation should be used to evaluate the performance of each classifier.

This wrapper algorithm is fairly efficient since it can reuse all the mutual information tests. Note that mutual information tests often take more than 95% of the running time of the BN learning process (Cheng *et al.* 1997b).

[2] Here, that the GBN-learner will produce the same answers if it omitted steps 2 and 3 – i.e., if it considered the entire BN, rather than just a subset. We explicitly compute the Markov blanket as it will be used by the subsequent W-MN-FS and W-MN-FS-O learners .

Wrapper (X-learner: LearningAlgorithm, *D*: Data)

1. Partition the input training set $D = T \cup H$ into internal training set *T* and internal holdout set *H*.
2. Call X-learner on the internal training set *T* *m* times, each time using a different threshold setting τ_i; this produces a set of *m* classifiers { BN_i }
3. Select a classifier $BN^* = \langle N, A^*, \theta^* \rangle \in \{BN_i\}$ that performs best on the *holdout* set *H*.
4. Keep this classifier's structure $\langle N, A^* \rangle$ and re-learn the parameters (conditional probability tables) Θ' using the whole training set *D*.
5. Output this new classifier.

Figure 8 The wrapper algorithm

3.4 Feature Subset Selection

Overfitting often happens when there are too many "parameters", for a given quantity of data. Here, this can happen if there are too many nodes, and hence too many CPtable entries. One way to reduce the chance of this happening is by considering only a subset of the features; this is called "feature selection", and is an active research topic in data mining. For example, Langley and Sage (1994) use forward selection to find a good subset of attributes; Kohavi and John (1997) use best-first search, based on accuracy estimates, to find a subset of attributes.

A byproduct of GBN learning is that we can get a set of features that are on the Markov blanket of the class node. The *Markov blanket* of a node *n* is the union of *n*'s parents, *n*'s children, and the parents of *n*'s children. This subset of nodes can "shields" *n* from being affected by any node outside the blanket. When using a BN classifier on complete data, the Markov blanket of the class node forms a natural feature selection, as all features outside the Markov blanket can be safely deleted from the BN. This can often produce a much smaller BN without compromising the classification accuracy.

To examine the effectiveness of such feature subset, we use it to simplify the multi-net learner. The algorithm is described below.

3.5 MN-FS_i(*S*: training set; *F*: feature set; [*O*: Node Ordering]): *returns* Bayesian Multi-net

1. Call *Wrapper(GBN_i)* with the training set *S* and all features *F*.
2. Get the Markov blanket $B \subseteq N$ of the class node.
3. Call *Wrapper (MN_i)* with the training set *S* and the feature subset *B*.
4. Output the multi-net classifier.

Figure 9 The MN-FS_i algorithm

4 Empirical study

4.1 Methodology

Our experiments involved five datasets downloaded from the UCI machine learning repository -- see Table 1. When choosing the datasets, we selected datasets with large numbers of cases, to allow us to measure the learning and classification efficiency. We also preferred datasets that have few or no continuous features, to avoid information loss in discretization and to be able to compare the learning

accuracy with other algorithms fairly. When we needed to discretize the continuous features, we used the discretization utility of MLC++ (Kohavi *et al.* 1994) on the default setting.

Table 1: Datasets used in the experiments.

Dataset	Attributes.	Classes	Instances	
			Train	Test
Adult	13	2	32561	16281
Nursery	8	5	8640	4320
Mushroom	22	2	5416	2708
Chess	36	2	2130	1066
DNA	60	3	2000	1186

The experiments were carried out using our Bayesian Network PowerPredictor 1.0. For each data set, we learned six BN classifiers: Wrapper(GBN) = W-GBN, Wrapper(GBN) with ordering = W-GBN-O, Wrapper(multi-net) = W-MN, Wrapper(multi-net) with ordering = W-MN-O, Wrapper(multi-net) with feature selection = W-MN-FS and Wrapper(multi-net) with feature selection with ordering = W-MN-FS-O. The ordering for the Chess data set is the reversed order of the features that appear in the data set since it is more reasonable, the ordering we use for other data sets are simply the order of the features that appear in the data set. For the GBN learner, we also assume that the class node it is a root node in the network.

The classification process is also performed using BN PowerPredictor. The classification of each case in the test set is done by choosing, as class label, the value of class variable that has the highest posterior probability, given the instantiations of the feature nodes. The classification accuracy is defined as the percentage of correct predictions on the test sets (i.e., using a 0-1 loss function).

The experiments were performed using a Pentium II 300 MHz PC with 128MB of RAM, running MS-Windows NT 4.0.

4.2 Results

Table 2 provides the prediction accuracy and standard deviation of each classifier. We ordered the datasets by their training sets from large to small. The best results of each dataset are emphasized using a boldfaced font. Table 2 also gives the best results reported in the literature on these data sets (as far as we know). To get an idea of the structure of a learned BN classifier, please see Figure 6.

From Table 2 we can see that all six unrestricted BN classifiers work quite well. Bayesian multi-net works better on Nursery and Mushroom; while GBN works better on DNA. The two types of classifiers have similar performance on Adult and Chess. This suggest that some data sets are more suitable for multi-net classifiers while others are more suitable for GBN, depending on whether the underlying relationships among the features are different for different class node values.

We can also see that the feature ordering does not make much difference to the performance of the classifiers. We also tried to provide the BN learners with obviously wrong ordering. Its effect to the classifier's performance is very small. However, with wrong ordering, the classifiers tend to be more complex.

By comparing the performance of the multi-nets *without* feature selection to the multi-nets *with* feature selection, we can see that the difference is quite small. However, the multi-nets with feature selection are much simpler. By comparing the running time of learning these classifiers (see Table 3), we can see that multi-nets with feature selection can be learned faster.

Table 2 The results of unrestricted BN classifiers

(The numbers in the parentheses are the number of selected features / total number of features)

	W-GBN	W-GBN-O	W-MN	W-MN-O	W-MN-FS	W-MN-FS-O		Best-reported
Adult	**86.33±0.53** (7/13)	85.88±0.53 (8/13)	84.83±0.55	85.54±0.54	85.79±0.54 (7/13)	85.46±0.54 (8/13)		85.95
Nursery	91.92±0.81 (8/8)	91.60±0.83 (8/8)	97.13±0.50	**97.31±0.48**	Same as W-MN	Same as W-MN-O		N/A
Mushroom	98.67±0.43 (7/22)	98.74±0.42 (5/22)	99.96±0.07	**100**	98.67±0.43 (7/22)	99.11±0.35 (5/22)		100
Chess	93.53±1.48 (11/36)	93.62±1.47 (11/36)	**96.44±1.11**	94.56±1.36	93.25±1.51 (11/36)	93.43±1.49 (11/36)		99.53±0.21
DNA	95.70±1.15 (14/60)	**96.63±1.03** (15/60)	94.10±1.34	93.51±1.40	95.36±1.20 (14/60)	95.70±1.15 (15/60)		96.12±0.6

Table 3 gives the total learning time of each BN classifier using the wrapper algorithm. Because the feature ordering makes little difference on the efficiency, we only give the running time of the learning procedure without the ordering. (In practice, CBL1 and CBL2 are both linear in the number of instances and appear $O(N^2)$ in the number of features.) The table shows that all BN classifiers can be learned efficiently as the longest learning time is less than 25 minutes. Note that the running time for learning the multi-nets with feature selection includes the running time for learning GBN in the first step of the feature subset selection algorithm (see Section 3.4). In general, the wrapper algorithm is about 3 to 5 times slower than only using the learner alone, even though the wrapper algorithm usually tries 7 to 15 different models before it output the best performer.

Table 3: Running time (CPU seconds) of the classifier learning procedures.

	W-GBN	W-MN	W-MN-FS
Adult	1046	1466	1200
Nursery	54	79	N.A.
Mushroom	322	533	345
Chess	84	163	109
DNA	210	1000	266

In our experiments, we found that the classification process is also very efficient. PowerPredictor can perform 200 to over 1000 classifications per second depending on the complexity of the classifier.

5 Conclusion

In this paper, we studied two types of unrestricted BN classifiers – general Bayesian networks and Bayesian multi-nets. The results show that our CI based BN learning algorithms are very efficient, and the learned BN classifiers can give very good prediction accuracy. This paper also presents an effective way for feature subset selection.

As we illustrate in Figure 6, the BN classifiers are also very easy to understand for human being. By checking and modifying the learned BN predictive models, domain experts can study the relationships among the attributes and construct better BN predictive models.

Based on these results we believe that the improved types of BN classifiers, such as the ones shown here, should be used more often in real-world data mining applications.

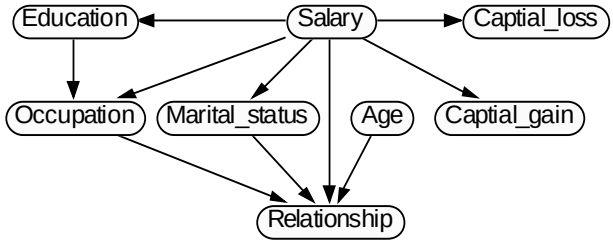

Figure 10 The W-GBN classifier for "Adult" data set

References

Cheng, J., Bell, D.A. and Liu, W. (1997a). An algorithm for Bayesian belief network construction from data. In *Proceedings of AI & STAT'97* (pp.83-90), Florida.

Cheng, J., Bell, D.A. and Liu, W. (1997b). Learning belief networks from data: An information theory based approach. In *Proceedings of ACM CIKM'97*.

Cheng, J. (1998). *PowerConstructor* System. http://www.cs.ualberta.ca/~jcheng/bnpc.htm.

Cheng, J., Greiner, R. (1999). Comparing Bayesian network classifiers. In *Proceedings of UAI-99*.

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

Cooper, G.F. (1990) Computational complexity of probabilistic inference using Bayesian belief networks, In *Artificial Intelligence*, 42 (pp. 393-405).

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

Duda, R. and Hart, P. (1973). *Pattern classification and scene analysis*. John Wiley & Sons.

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

Geiger, D. and Heckerman, D. (1996). Knowledge representation and inference in similarity networks and Bayesian multinets. In *Artificial Intelligence* 82 (pp. 45-74).

Greiner, R. Grove, A. and Schuurmans, D. (1997). Learning Bayesian nets that perform well. In *Proceedings of UAI-97*.

Heckerman, D. (1995). A tutorial on learning Bayesian networks. *Tech Report MSR-TR-95-06*, Microsoft Research.

Heckerman, D., Meek, C. and Cooper, G. (1997). A Bayesian approach to causal discovery. *Technical Report MSR-TR-97-05*. Microsoft Research.

Kohavi, R., John, G., Long, R. Manley, D. and Pfleger, K. (1994). MLC++: A machine learning library in C++. In *Proceedings of Sixth International Conference on Tools with Artificial Intelligence*. IEEE Computer Society.

Kohavi, R., John G. (1997) Wrappers for Feature Subset Selection. In *Artificial Intelligence journal*, special issue on relevance, Vol. 97, No. 1-2 (pp. 273-324).

Kononenko, I. (1991). Semi-naïve Bayesian classifier. In Y. Kodratoff (Ed.), *Proceedings of sixth European working session on learning (pp.206-219)*. Springer-Verlag.

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

Langley, P. and Sage, S. (1994). Induction of Selective Bayesian Classifiers. In *Proceedings of UAI-94*.

Murphy, P.M. and Aha, D.W. (1995). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Neapolitan, R.E. (1990), *Probabilistic reasoning in expert systems: theory and algorithms*, John Wiley & Sons.

Pazzani, M.J. (1995). Searching for dependencies in Bayesian classifiers. In *Proceedings of AI & STAT'95*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: networks of plausible inference*, Morgan Kaufmann.

Spirtes, P., Glymour, C. and Scheines, R. (1993). *Causation, Prediction, and Search*. http://hss.cmu.edu/html/departments/philosophy/TETRAD.BOOK/book.html.