Comparing Bayesian Network Classifiers

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

Abstract

In this paper, we empirically evaluate algorithms for learning four Bayesian network (BN) classifiers: Naïve-Bayes, tree augmented Naïve-Bayes (TANs), BN augmented Naïve-Bayes (BANs) and general BNs (GBNs), where the GBNs and BANs are learned using two variants of a conditional independence based BN-learning algorithm. Experimental results show the GBNs and BANs learned using the proposing learning algorithms are competitive with (or superior to) the best classifiers based on both Bayesian networks and other formalisms, and that the computational time for learning and using these classifiers is relatively small. These results argue that BN classifiers deserve more attention in machine learning and data mining communities.

1 INTRODUCTION

Classification is a fundamental task in fault diagnosis, pattern recognition and forecasting. In general, a classifier is a function that chooses a class label (from a group of predefined labels) for instances 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 classifiers have been developed, such as decision tree based classifiers and neural network based 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 classification node, are surprisingly effective (Langley *et al.* 1992).

This paper further explores this role of BNs. Section 2 gives the framework of our research work on BN classifiers. Section 3 defines four classes of BNs, i.e., Naïve-Bayes, tree augmented Naïve-Bayes (TANs) (Friedman *et al.*, 1997), BN augmented Naïve-Bayes (BANs) and general BNs (GBNs), and describes methods for learning each. We implemented these systems to see how they would work. Section 4 presents and analyzes the experimental results, and also proposes an algorithm for further improving BN classifiers.

2 FRAMEWORK

2.1 LEARNING BNS

A Bayesian network $B = < N, A, \theta >$ is a directed acyclic graph (DAG) with a conditional probability distribution for each node. Each node $n \in N$ represents a domain variable, and each arc $a \in A$ between nodes represents a probabilistic dependency. When learning Bayesian networks from datasets, we use nodes to represent dataset attributes.

For a BN, a *Markov boundary* of a node is a subset of nodes that "shields" the node from being affected by any node outside the boundary. One *Markov boundary* of a node *n* is the union of the *n*'s parents, *n*'s children, and the parents of *n*'s children. It is often called the *Markov blanket* of *n*. When using a BN classifier, the *Markov blanket* of the classification node forms a natural feature selection. All features outside the *Markov blanket* can be safely deleted from the BN, which can produce a much smaller BN without compromising the classification accuracy.

BNs can be viewed in two different ways. First, a BN can be viewed as a structure that encodes the joint distribution of the attributes. Hence, a BN can be used as a classifier that gives the posterior probability distribution of the classification node given the values of other attributes. BN-learning can also be achieved by searching for a structure that fits the data well. The heuristic search is often guided by a Bayesian, MDL or entropy scoring function. These algorithms are generally referred as *scoring-based* algorithms (Heckerman 1995; Cooper and Herskovits 1992).

Second, a BN can be viewed as a structure that encodes a group of conditional independence relationships among the nodes according to the concept of *d-separation* (Pearl 1988). This suggests that a BN structure can be learned by learning the conditional independence relationships among the nodes. Using some statistical tests (such as Chi-squared test and mutual information test), 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).

Chow and Liu (1968) developed a famous algorithm for learning tree-like BNs. Interestingly, it has the features of both scoring-based methods and the CI-based methods – although the general idea behind this algorithm is to find a structure with the best score (Kullback-Leibler cross-entropy), it ends up with analyzing the pair-wise dependencies. Unfortunately, for unrestricted BN learning, no such connection can be found between the scoring-based methods and the CI-based methods. The merit of this algorithm is that it only needs $O(N^2)$ pair-wise dependency calculations and no heuristic search is needed. This algorithm is the base of the TAN learning algorithm.

A comparison of the scoring-based methods and the CI-based methods is presented in (Heckerman et al., 1997), which shows that the scoring-based methods have certain advantages over the CI-based methods. However, Friedman *et al.* (1997) show that the general scoring-based methods may result in poor classifiers by using theoretical analysis. Moreover, the scoring-based methods are often less efficient in practice.

2.2 SIMPLE BN CLASSIFIERS

2.2.1 Naïve-Bayes

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

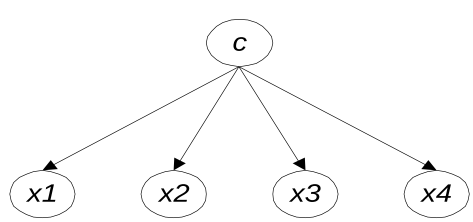

Figure 1: A simple Naïve Bayes structure

Naïve-Bayes has been used as an effective classifier for many years. It has two advantages over other classifiers. First, it is easy to construct – no learning procedure is required. Second, the classification process is very efficient since it assumes that all the features are independent of each other. Although this assumption is obviously problematic, Naïve-Bayes can surprisingly outperform many sophisticated classifiers on datasets where the features are not strongly correlated.

2.2.2 Improving on Naïve-Bayes

In recent years, a lot of effort has been made on improving Naïve-Bayesian classifiers, following two general approaches: selecting feature subset and relaxing independence assumptions.

Feature subset select

Langley and Sage (1994) use forward selection to find a good subset of attributes, then use this subset to construct a *selective Bayesian classifier*. Kohavi and John (1997) use best-first search, based on accuracy estimates, to find a subset of attributes. Their algorithm can wrap around any classifiers, such as the decision tree classifiers and the Naïve-Bayesian classifier. Pazzani's algorithm (Pazzani 1995) performs feature joining as well as feature selection to improve the Naïve-Bayesian classifier.

Independence assumption relax

Kononenko's algorithm (Kononenko 1991) partitions the attributes into disjoint groups and assumes independence only between attributes of different groups. Friedman *et al.* (1997) studied TAN, which allows tree-like structures to be used to represent dependencies among attributes (see Section 3.2). Their experiments show that TAN outperforms Naïve-Bayes, while maintains computational simplicity on learning. The paper also presents methods for learning GBNs and BANs as classifiers using a minimum description length (MDL) scoring method. Singh and Provan (1996) combine the two approaches by first performing feature selection using information theoretic methods and then using a scoring based BN learning method to learn a *selective BN* from the subset of features.

2.3 MOTIVATIONS

Although all these algorithms have proven to be more accurate than Naïve Bayes classifiers, there is still a lot of work to be done. We are particularly interested in the following questions.

1. Since "*using MDL (or other nonspecialized scoring functions) for learning unrestricted Bayesian networks may result in poor classifier…*" (Friedman *et al.* 1997), a natural question is "Will non-scoring methods (i.e., condition independence (CI) test based methods, such as mutual information test and chi-squared test based methods) learn good classifiers?"

2. If we treat the classification node as an ordinary node and learn an unrestricted BN, we get a natural feature subset – the *Markov blanket* (section 2.1) around the classification node. Can we take this approach to select features (rather than earlier approaches that perform feature selection before the BN-learning)?

3. GBNs and BANs have more parameters, which increases the risk of overfitting given relatively small training sets. How can we cope with it? (Overfitting is a phenomenon that occurs when a model tries to fit the training set too closely instead of generalizing. An overfitted model will not perform well on data outside the training samples.)

4. Efficiency is a major reason for learning and using TAN or Naïve-Bayes. How expensive is it to learn and use GBN and BAN classifiers?

In this paper, we investigate these questions using empirical study. We use two variants of a general BN-learning algorithm (based on conditional-independence tests) to learn GBNs and BANs. We empirically compared these classifiers with TAN and Naïve-Bayes using seven datasets from the UCI Machine Learning Repository (Murphy and Aha, 1995).

3 LEARNING BAYESIAN NETWOK CLASSIFIERS

Learning Bayesian network classifiers involves two steps: structure learning and parameter (conditional probability tables) learning. We will focus on structure learning methods for different Bayesian network classifiers in the subsections below. These methods are all implemented on top of the *PowerConstructor 2.0* program (Cheng 1998), which implements two BN-learning algorithms, one for the case when node ordering is given (the CBL1 algorithm – Cheng *et al.* 1997a); the other for the case when node ordering is not given (the CBL2 algorithm – Cheng *et al.* 1997b). (A node ordering specifies a total order of the nodes; we insist that no node can be an ancestor of a node that appears earlier in the order.)

In this paper, we treat the classification node as the first node in the ordering. The order of other nodes is arbitrary; we simply use the order they appear in the dataset. Therefore, we only need to use the CBL1 algorithm, which has the time complexity of $O(N^2)$ on the mutual information tests (N is the number of attributes in the dataset) and linear on the number of cases. The efficiency is achieved by directly extending the Chow-Liu tree construction algorithm (Chow and Liu 1968) to a three-phase BN learning algorithm: *drafting*, which is essentially the Chow-Liu algorithm, *thickening*, which adds edges to the draft, and *thinning*, which verifies the necessity of each edge. For the correctness proof, complexity analysis and other detailed information, please refer to (Cheng, 1997a).

The parameter learning uses frequency estimates from the data.

3.1 NAÏVE-BAYES

The procedure of constructing Naïve-Bayes is given as follows.

1. Let the classification node be the parent of all other nodes.
2. Learn the parameters and output the Naïve-Bayes.

3.2 TREE AUGMENTED NAÏVE-BAYES (TAN)

Letting $X = \{x_1, ... x_n, c\}$ represent the node set (where c is the classification node) of the data. The algorithm for learning TAN classifier (Friedman *et al.* 1997) first learns a tree structure over $X \setminus \{c\}$, using mutual information tests. It then adds a link from the classification node to each feature node in the manner as we construct a Naïve-Bayes (i.e., the classification node is a parent of all other nodes.) A simple TAN structure is shown in Figure 2. (Note that features x_1, x_2, x_3, x_4 form a tree.)

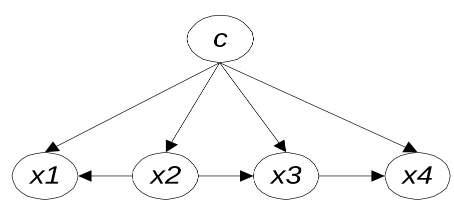

Figure 2: A simple TAN structure

The learning procedure can be described as follows.

1. Take the training set and $X \setminus \{c\}$ as input.
2. Call the modified Chow-Liu algorithm. (The original algorithm is modified by replacing every mutual information test $I(x_i, x_j)$ with a conditional mutual information test $I(x_i, x_j \mid \{c\})$.
3. Add c as a parent of every x_i where $1 \leq i \leq n$.
4. Learn the parameters and output the TAN.

This algorithm, which is modified from the Chow-Liu algorithm, requires $O(N^2)$ numbers of conditional mutual information tests.

This algorithm is essentially the first phase of the BAN-learning algorithm (Section 3.3).

3.3 BN AUGMENTED NAÏVE-BAYES (BAN)

BAN classifier has been studied in several papers, such as (Friedman *et al.* 1997). The basic idea of this algorithm is just like the TAN learner of Section 3.2, but Step 2 calls an unrestricted BN-learning algorithm instead of a tree-learning algorithm (see Figure 3).

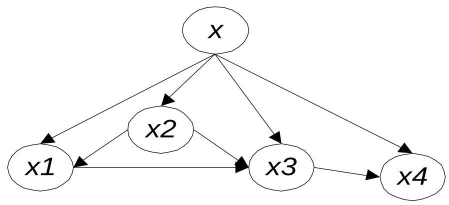

Figure 3: A simple BAN structure

Letting $X = \{x_1,...x_n, c\}$ represent the feature set (where c is the classification node) of the data, the learning procedure based on mutual information test can be described as follows.

1. Take the training set and $X \setminus \{c\}$ (along with the ordering) as input.
2. Call the modified CBL1 algorithm. (The original algorithm is modified in the following way: replace every mutual information test $I(x_i, x_j)$ with a conditional mutual information test $I(x_i, x_j \mid \{c\})$; replace every conditional mutual information test $I(x_i, x_j \mid Z)$ with $I(x_i, x_j \mid Z + \{c\})$, where $Z \subset X \setminus \{c\}$.
3. Add c as a parent of every x_i where $1 \le i \le n$.
4. Learn the parameters and output the BAN.

Like the TAN-learning algorithm, this algorithm does not require additional mutual information tests, and so it requires $O(N^2)$ mutual information tests.

3.4 GENERAL BAYESIAN NETWORK (GBN)

Unlike the other BN-classifier learners, the GBN learner treats the classification nodes as an ordinary node (see Figure 4). The learning procedure is described as follows.

1. Take the train set and the feature set (along with ordering) as input.
2. Call the CBL1 algorithm.
3. Find the *Markov blanket* of the classification node.
4. Delete all the nodes that are outside the *Markov blanket*.
5. Learn the parameters and output the GBN.

This algorithm has the time complexity of $O(N^2)$ on the mutual information tests.

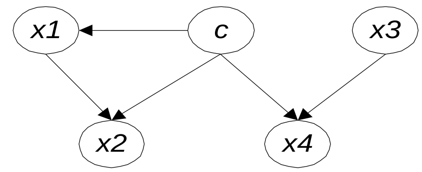

Figure 4: A simple GBN

4 EXPERIMENTS

4.1 METHODOLOGY

Our experiments were carried out using seven datasets downloaded from the UCI machine learning repository. When choosing the datasets, we tried to use datasets with large numbers of cases so that we can measure the learning and classification efficiency. We also tried to use the datasets that have few or no continuous features. The reason is that we want 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.

The datasets we used are summarized in Table 1. (CV5 stands for five-fold cross validation.)

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
Car	6	4	1728	CV5
Flare	10	3	1066	CV5
Vote	16	2	435	CV5

Brief descriptions of the seven datasets are given below.

Adult dataset: The data was extracted from the census bureau database. Prediction task is to determine whether a person makes over 50K a year. The discretization process ignores “fnlwgt” (which is one of the 14 attributes). We therefore omit "flnwgt" and use the remainder 13 attributes in our experiments.

Car dataset: car evaluation based on the six features of a car.

Chess: chess end-game result classification based on board-descriptions.

Flare: classifying the number of times of occurrence of certain type of solar flare.

Mushroom: classifying whether a type a mushroom is edible. Missing values are treated as having the value "?" in our experiments.

Nursery: Ranking nursery-school applications based on 8 features.

Vote: Using voting records to classify Congressmen as democrat and republican.

Our experiments were carried out as follows. We first used the four algorithms presented in Section 3 to learn the four classifiers (one of each type) and then export the BNs to *Bayesian Interchange Format* (BIF v0.15) files. The GBN and BAN classifiers were learned using the default threshold setting of the *PowerConstructor*. (This threshold determines how much mutual information between two nodes is considered as significant.) TAN and Naïve-Bayes learning algorithms have no such threshold.

To test these classifiers on the test sets, we use a modified version of *JavaBayes* (Cozman 1998). We added some classes to the *JavaBayes* v0.341 so that it can read the test datasets and perform classification given a BN.

The classification accuracy was measured by the percentage of correct predictions on the test sets.

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

4.2 RESULTS

The prediction accuracy and standard deviation of each classifier is given in Table 2. We ordered the datasets by their training sets from large to small. The best results of each dataset are emphasized using a boldfaced font. The best results reported in the literature on these data sets (as far as we know) are also listed.

From Table 2 we can see that the GBN, BAN and TAN have overall better performance than the Naïve-Bayes in these experiments. BAN did best on four of the datasets and GBN and TAN each did best on two of the datasets.

On the data sets "Nursery" and "Car", the GBN classifier was inferior to the Naïve-Bayes. The reason is, in both cases the GBN actually reduced to the Naïve-Bayes with missing links (the reduced Naïve-Bayes is sometimes called *selective Naïve-Bayes*). This reveals that the features in the two datasets are *almost* independent to each other. (Note that using *selective Naïve-Bayes* is not appropriate in such situations.) However, when using BAN or TAN classifiers, the weak dependencies among the features given the classification node can be successfully captured. These weak dependencies can improve the prediction accuracy significantly, as we can see from Table 2. Since in GBN the classification node is treated in the same way as the feature nodes, these weak dependencies cannot be captured. This suggests that treating classification node differently is reasonable, at least in some domains.

Table 2: Experimental Results

	GBN (No. Selected Fea./Total No. Fea.)	**BAN**	**TAN**	**Naïve-Bayes**	**Best reported**
Adult	**86.11±0.27** (8/13)	85.82±0.27	86.01±0.27	84.18±0.29	85.95
Nursery	89.72±0.46 (6/8)	**93.08±0.39**	91.71±0.42	90.32±0.45	N/A
Mushroom	99.30±0.16 (5/22)	**100**	99.82±0.08	95.79±0.39	99.64±0.31
Chess	**94.65±0.69** (19/36)	94.18±0.72	92.50±0.81	87.34±1.02	99.53±0.21
Car	86.11±1.46 (5/6)	**94.04±0.44**	**94.10±0.48**	86.58±1.78	N/A
Flare	82.27±1.45 (1-3/10)	82.85±2.00.	**83.49±1.29**	80.11±3.14	83.40±1.67
Vote	95.17±1.89 (10-11/16)	**95.63±3.85**	94.25±3.63	89.89±5.29	96.3±1.3

As we mentioned earlier, the GBN and BAN classifiers were learned using the default threshold setting given by *PowerConstructor*. Based on our experience, this threshold setting is appropriate for most domains when the dataset is large enough. When the dataset is small, too high a setting will cause missing edges, too low a setting will cause overfitting. Both situations will decrease the prediction accuracy of GBN and BAN. From Table 2, we can see that GBN or BAN can produce outstanding results when the datasets are large enough. For example, on the "Adult" data, the GBN gives the best result as far as we know; on the "mushroom" data, the BAN gives 100% prediction accuracy. On the same dataset, the GBN gives 99.3% of accuracy using only 5 of the 22 features. However, when the datasets are small, there is evidence that show the threshold is not appropriate. For example, on the "Flare" data, the GBN uses only 1 to 3 of the 10 features, which is probably too harsh. An advantage of the TAN classifier is that it does not need threshold. But we believe that with a proper threshold setting, the unrestricted BN classifiers (GBN and BAN) should outperform the TAN classifier even when the dataset is small. We already have some evidence that shows the performance of GBN and BAN can be improved on some of the seven datasets by searching an optimal threshold. We propose a wrapper for this task in Section 4.3.

From Table 2, we can also find that, on six of the seven datasets, at least one of the unrestricted BN classifiers (i.e., GBN and BAN) can give the best performance. This suggests that wrapping the two algorithms together and returning the winner will probably lead to better performance.

In our experiments, we also measured the running time of the classifier-learning procedures. The results are in Table 3.

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

	GBN	BAN	TAN
Adult	515	536	131
Nursery	10	11	10
Mushroom	136	134	38
Chess	41	56	37
Car	1	1	1
Flare	3	3	2
Vote	8	8	3

Table 3 gives the total learning time of each BN classifier, which includes the time for structure learning and parameter learning. It shows that all BN classifiers can be learned efficiently. The unrestricted BN classifier learning procedures are at most five times slower than the TAN learning procedure. The longest learning time is about 9 minutes when learning the BAN from the "Adult" data. Considering the training set has 32,561 cases and 13 attributes, the efficiency is quite satisfactory.

In our experiments, we found that the classification process is also very efficient. *JavaBayes* can perform 100 to 600 classifications per second depending on the complexity of the classifier. We also found that the GBNs are often faster than Naïve-Bayes on classification when the GBNs contain only a subset of the features.

4.3 IMPROVING BN CLASSIFIERS

The analysis of our experimental results give us two ideas to improve the unrestricted BN classifiers: 1. Automatic threshold selecting based on the prediction accuracy. 2. Wrapping the GBN and BAN together and returning the winner. (Another algorithm for automatic threshold selection on BN construction is presented in [Fung and Crawford 1990].)

We therefore propose a wrapper algorithm that adopts the two ideas.

1. Partition the input training set into internal training set and internal holdout set.
2. Call GBN-learner using different threshold settings and select a GBN that performs best on the holdout set.
3. Call BAN-learner using different threshold settings and select a BAN that performs best on the holdout set.
4. Determine which one of the two classifiers is the best according to their performance on the internal holdout set.
5. Keep this classifier's structure and re-learn the parameters (conditional probability tables) using the whole training set.
6. Output this new classifier.

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

This wrapper algorithm can be fairly efficient since it can reuse all the mutual information tests. Note that mutual information tests usually take more than 95% of the running time of the BN learning process. The accuracy estimation process will not be too slow either since a few hundred predictions can be made in one second.

Another possible way of improving BN classifiers is by providing domain knowledge, such as node ordering, forbidden links and cause & effect relationships. This can help the BN learners to find a better BN structure.

5 CONCLUSION

In this paper, we empirically evaluated and compared four BN classifiers. The unrestricted classifiers (GBN and BAN), learned using two variants of the CBL1 algorithm, give very encouraging results without using any wrapper function. After analyzing the experimental results, we proposed a wrapper algorithm that wraps around GBN and BAN. The implementation of this wrapper classifier is on going. We are confident that the wrapper classifier will give even better results.

From the experiments we can also see that the time expenses of unrestricted BN-learning are only a few times slower (at most) than that of the efficient TAN-learning. This is due to the three-phase learning mechanism used in the CBL1 algorithm.

Although we have not evaluated the scoring based BN-learning algorithms, based on the theoretical analysis of Friedman *et al.* (1997) and the empirical comparison with the results given in (Singh and Provan 1996; Friedman *et al.* 1997), we believe that methods based on CI tests (such as mutual information tests) are more suitable for BN classifier learning than the general scoring-based methods. In fact, mutual information test has been used widely in decision tree learning and feature selection.

As constraint based methods, domain knowledge can be easily combined as constraints with the CI test results in our BN classifier learning methods. This will probably lead to yet better classification accuracy.

Our experiments also show that the classification can be performed very efficiently with BN classifiers.

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

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.

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

Cozman, F.G. (1998). *JavaBayes* system. http://www.cs.cmu.edu/~javabayes/.

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).

Fung, R.M. and Crawford, S.L. (1990). Constructor: a system for the induction of probabilistic models. In P*roceedings of AAAI-90*.

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.

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.

Singh, M. and Provan, G.M. (1996). Efficient Learning of Selective Bayesian Network Classifiers. In *Proceedings of the ICML-96*.

Spirtes, P., Glymour, C. and Scheines, R. (1996). *Causation, Prediction, and Search*. http://hss.cmu.edu/html/departments/philosophy/TETRAD.BOOK/book.html.