Learning Augmented Bayesian Classifiers:
A Comparison of Distribution-based and Classification-based Approaches

Eamonn J. Keogh
Dept. of Information and Computer Science
University of California, Irvine
Irvine, California 92697 USA
eamonn@ics.uci.edu

Michael J. Pazzani
Dept. of Information and Computer Science
University of California, Irvine
Irvine, California 92697 USA
pazzani@ics.uci.edu

Abstract

The naïve Bayes classifier is built on the assumption of conditional independence between the attributes given the class. The algorithm has been shown to be surprisingly robust to obvious violations of this condition, but it is natural to ask if it is possible to further improve the accuracy by relaxing this assumption. We examine an approach where naïve Bayes is augmented by the addition of correlation arcs between attributes. We explore two methods for finding the set of augmenting arcs, a greedy hill-climbing search, and a novel, more computationally efficient algorithm that we call SuperParent. We compare these methods to TAN; a state-of the-art distribution-based approach to finding the augmenting arcs.

1 INTRODUCTION

The Bayesian classifier (Duda & Hart, 1973) is a simple classification method, which classifies an instance *j* by determining the probability of it belonging to class C_i. These probabilities are calculated as:

$$P(C_i \mid A_1 = V_{1_j} \,\&\, \cdots \,\&\, A_N = V_{N_j}), \qquad (1)$$

where an example is represented as attribute-value pairs of the form $A_i=V_i$. If there are *N* independent attributes, then the probability is proportional to:

$$P(C_i)\prod_k P(A_k = \mathrm{V}_{\mathrm{k_j}} \mid C_i) \qquad (2)$$

When this independence assumption is made, the classifier is called naïve (Simple, Idiots) Bayes (Ripley 1996). Naïve Bayes has been shown to be competitive with more complex, state-of-the-art classifiers (Dougherty 1995, Kohavi & Sahami 1995). This is surprising given the explicit assumption that all attributes are independent given the class. This assumption rarely holds in real world problems. There have been recent attempts to explain its surprisingly good performance (Domingos & Pazzani 1997) and to improve performance by relaxing the independence assumptions (Pazzani 1996, Friedman & Goldszmidt 1996, Sahami 1996, Kononenko 1991).

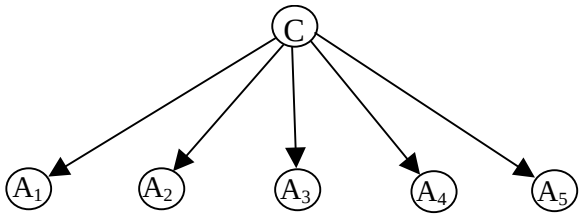

Figure 1: An example of a Naïve Bayes Network

The work of Friedman and Goldszmidt is particularly interesting. They compared naïve Bayes to Bayesian networks (Pearl 1988), a much more powerful and flexible representation of probabilistic dependence. Surprisingly, using unrestricted Bayesian networks did not generally lead to improvements in accuracy and even reduced accuracy in some domains. This prompted them to propose a compromise representation, combining some of Bayesian networks ability to represent dependence, with the simplicity of naïve Bayes. This representation is called *augmented naïve Bayes*, and is defined by the following conditions:

- Each attribute has the class attribute as a parent.
- Attributes may have one other attribute as a parent.

The latter condition means that if there is an arc from A_i to A_j, the two attributes are not independent given the class. Instead the influence of A_j on the class probabilities depends on the value of A_i. Figure 2 shows an example of an augmented Bayes network.

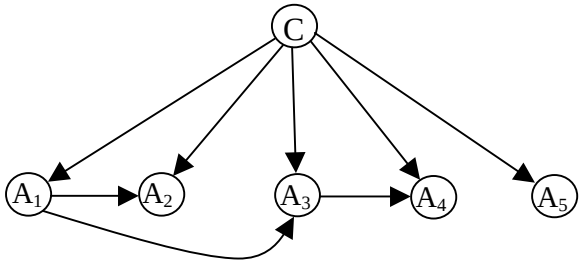

Figure 2: An example of an Augmented Bayes Network

Finding the best augmented Bayes network (in the sense of best approximation of an unrestricted Bayes network) is an NP-hard problem. Friedman and Goldszmidt deal with this difficulty by restricting the network to be a tree topology (in addition to the arcs from the parent to each node), and utilizing a result by Chow and Liu (1968) to efficiently find the best tree-augmented naïve Bayes (TAN).

In this paper we take a different approach to constructing tree-augmented Bayesian networks. Rather than directly attempting to approximate the underlying probability distribution, we concentrate solely on using the same representation to improve classification accuracy. We show two methods for constructing augmented Bayesian networks, a hill climbing greedy search, and a novel, more efficient search algorithm, which we call SuperParent. We compare these to Friedman and Goldszmidt's approach and show that approximating the underlying probability distribution is not the best way to improve classification accuracy.

2 SEARCHING FOR AUGMENTED BAYES NETWORKS

Following Friedman and Goldszmidt we restrict the allowed topology of an augmented Bayes network in the following manner. Each node may have at most one (non-class) parent[1]. There are two reasons for this restriction. First, it reduces the search space. Second, the probability estimates for a node become more unreliable as additional parents are added, because the size of the conditional probability tables increases exponentially with the number of parents. Restricting the number of parents to two mitigates problems in estimating probabilities from data while allowing some amount of dependence among variables to be represented.

This restriction limits the maximum number of correlation arcs added to $N - 1$, where N is the number of attributes. The minimum number of correlation arcs added is zero, which corresponds to a naïve Bayes classifier, a special case of an augmented Bayes network.

We present the following definition to facilitate discussion of augmented Bayes networks.

Definition 1 (Orphan)
In augmented Bayes network, a node without a parent, other than the class node, is called an *orphan*.

Table 1 shows an outline of the algorithm for building an augmented Bayes network using hill climbing search (HCS). The network is initialized to naïve Bayes, and the set of orphans, O, is initialized to the full set of nodes A_1, A_2,…, A_N. Each possible arc from A_i to A_j ($A_i \neq A_j$, $A_j \in O$) is evaluated, using leaving-one-out cross validation to estimate the accuracy of the network with that arc added. If no arc produces an improvement in classification accuracy, the current classifier is returned. Otherwise, the arc that gave the most improvement is retained, and the node pointed to by the arc is removed from O. This process is repeated until O contains just one node, or there are no arcs which can be added to improve classification accuracy.

```
0. Initialize network to naive Bayes.
1. Evaluate the current classifier.
2. Consider adding every legal arc to the
   current classifier.
3. If there is an arc addition, which
   improves accuracy, then add the arc
   which improves accuracy the most to the
   current network, and go to 2.
Else: Return current classifier.
```

Table 1: An outline of hill climbing search (HCS)

$O(N^2)$ classifiers are constructed and evaluated for each arc added to the network, and $O(N)$ arcs may be added. So the complexity for HCS is $O(N^3)$. In Section 2.1 we discuss two optimizations which greatly speed up the search process without effecting the returned network.

2.1 EFFICIENT EVALUATION OF CHANGES TO CLASSIFIERS

To speed up the process of evaluating many classifiers we have implemented the following two optimizations, the first of which was introduced in Pazzani (1996).

We order the training instances, so that examples misclassified by the previous classifier are tested first. This allows the algorithm to abandon testing of a classifier as soon as the number of misclassified examples is greater than the current *best-so-far* classifier. This technique allows the algorithm to perform a fraction of the tests required by full search, but return the same result. The fraction of examples actually tested on a given dataset is approximately equal to the error rate of naïve Bayes on that dataset.

To prevent the need for completely recomputing equation 1 for each classifier built, we do the following. In the first step of the algorithm, we store the results of equation 2 in a J by I matrix, (J is the number of instances in the training set, I is the number of distinct classes) where the each element is the probability that example j belongs to class C_i. When testing a new classifier which has an addition arc from node A_b to node A_a, we adjust the matrix by multiplying the element (i, j) by

$$\frac{P(A_a = V_{a_j} | C_i \ \& \ A_b = V_{b_j})}{P(A_a = V_{a_j} | C_i)} \quad (3)$$

This equation simultaneously factors out the effect of the "orphan" node A_a, and factors in the effect of the arc from node A_b to node A_a. This effectively means the time taken to evaluate one instance of a classifier is independent of the

[1] Friedman and Goldszmidt's TAN networks also have this restriction, as a consequence of the underlying tree topology.

number of attributes. So the speed up achieved by this optimization is approximately *N*, the number of attributes.
To test the utility of these two optimizations we performed the following experiment. We ran the optimized and unoptimized versions of the HCS algorithm, 10 times, on several datasets from the UCI repository (Details about the datasets can be found in tables 3 and 6). Table 2 shows the average speedup achieved. As explained above, the speedup achieved depends on both the number of attributes and the accuracy achievable by naïve Bayes. Nevertheless the optimizations are clearly useful in general.

DataSet	Speedup achieved
Iris	41.0
Soybean-Large	312.3
Segment	199.2
Vote	135.7

Table 2: The average speedup achieved on various datasets by the two optimizations described in section 2.1

2.2 EXPERIMENTAL RESULTS

Our experimental methodology is closely modeled on that of Friedman and Goldszmidt (1996). We tested 13 data sets from the UCI repository (Merz et al, 1997) and one artificial data set. The accuracy of each learning method on each domain was determined by running 5*2-fold cross validation (Dietterich 1996). All classification algorithms were trained and tested on exactly the same cross validation folds. Following Friedman and Goldszmidt, instances with missing values were deleted from the database and continuous values were discretized, using only the training data, by Fayyad & Irani's (1993) entropy based method. In our comparison, we use the unsmoothed version of TAN (We also implemented and tested the "smoothed" version of TAN, and obtained results similar to Friedman & Goldszmidt. We omit these results for brevity and clarity). In the unsmoothed TAN and in our work, we replace zero probabilities with a small epsilon (.0001). The three algorithms used in this experiment are: **Naive** - naïve Bayes; **TAN** - Friedman and Goldszmidt's unsmoothed Tan network; **HCS** - Augmented Bayes networks built using hill climbing greedy search.

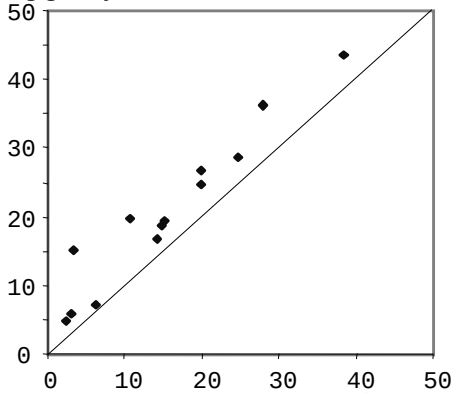

Figure 3: Scatter plot comparing the error of HCS to TAN on 13 datasets from the UCI repository. Points above the diagonal line correspond to datasets where HCS performs better. Points on the diagonal line indicate no difference in performance.

As an extreme example of attribute dependence we create an artificial noisy two-class dataset called *exclusive-or*. This dataset has 500 instances, and 10 attributes. After the introduction of class noise, the exclusive-or of two of the attributes predicts the class with 70% accuracy. The other 8 attributes are completely irrelevant. Table 3 summarizes the datasets.

Dataset	# Attributes	# Classes	# Instances
Vehicle	18	4	846
Post-op	9	3	90
Lung	56	3	32
Australia	14	2	690
Hepatitis	19	2	270
Vote	16	2	435
Heart	13	2	270
Soybean-Large	35	19	562
Pima	8	2	768
Breast	10	2	683
Iris	4	3	150
Segment	19	7	1540
E.coli	7	8	336
exclusive-or	10	2	500

Table 3. Descriptions of domains used.

Table 6 in the Appendix summarizes the accuracy of each algorithm on each dataset[2]. The best accuracy achieved on each dataset is shown in bold. Runners up, which did not differ at the 5% confidence level using a paired two tailed *t*-test are also shown in bold. We confirmed Friedman and Goldszmidt's ranking of the Naïve and TAN classifiers. Figure 3 provides a visual comparison of HCS and TAN. It shows that the HCS approach is usually more accurate than the TAN approach.
The data set *exclusive-or* is an excellent example of how augmenting arcs allow a Bayesian classifier to learn a non-linearly separable function. Naïve Bayes performs at chance levels on this data set. TAN does better, each is sometimes able to find the correct arc to connect the two dependent attributes. However, even if one considers just the folds where TAN finds the correct arc, they still do slightly worse than HCS. This is because both add 8

[2] In general our initial experimental results on **TAN** and *smoothed* TAN replicate the results of Friedman and Goldszmidt on the same data sets. There are no substantial differences between their implementation of *smoothed* TAN and our implementation. On a few domains our implementation of unsmoothed TAN is significantly more accurate. For example on Soybean-Large they report 58.17 but we report 86.07. We surmise that this is due to our method of handling of zero counts in probability estimation. We replace zero probabilities with a very small epsilon. This is particularly important on the Soybean-Large dataset because quite often an instance to be classified will have at least one zero probability in the calculation for every class.

additional unnecessary arcs. These arcs are fitted to noise, and do not generalize to the test data. In contrast HCS always finds the correct arc to add in the first iteration, and then it usually halts (occasionally it adds a second arc due to a chance pattern occurring in a cross-validated fold). This allows HCS to achieve the maximum accuracy possible on the data set. This increased representation flexibility of HCS has an additional bonus. If we see an arc in a network returned by HCS it indicates that modeling the relationship between these attributes is important to increase predictive accuracy of the model. Viewing an arc in a network returned by TAN, we have no such assurance, because TAN always returns N -1 augmenting arcs, even when the attributes are completely independent. This makes our approach more useful when comprehensibility and insight is important. In Section 4, we further explore a comparison of augmenting the naïve Bayesian classifier modeling the probability distribution and our approach.

3 SUPERPARENT

We have demonstrated that the addition of arcs using full greedy search can mitigate the strong independence assumptions of naïve Bayes and improve its classification accuracy. Here, we introduce SuperParent (SP) a more efficient heuristic search that attempts to have the same improvement in accuracy with a less expensive search technique. The general idea is that in hill climbing search as described so far, we attempt to find the best arc to add. Here, we break this up into two steps, first finding a good parent and then finding the best child of that parent.

Definition 2 (SuperParent)

Given an augmented Bayes network, if we extend arcs from node A_x to every orphan, node A_x is said to be a *SuperParent*.

Definition 3 (FavoriteChild)

Given an augmented Bayes network, if we extend an arc from node A_x to each orphan in turn, and test the effect on predictive accuracy, the node pointed to by the best arc is said to be the *FavoriteChild* of A_x.

Table 4 shows an outline of the SuperParent algorithm. The network is initialized to naïve Bayes, and the list of orphans, O, is initialized to the full set of nodes A_1, A_2,…, A_N. The effect on classification accuracy, of making each node a SuperParent is assessed. The best such node we designated A_{SP}. Next, the algorithm finds FavoriteChild of A_{SP}, by assessing the effect of adding a single arc from A_{SP} to each orphan. If the addition of the arc from A_{SP} to the FavoriteChild improves the classification accuracy and $|O| > 1$, then the FavoriteChild is removed from O, the arc is added to the current classifier, and the SuperParent cycle begins again. If there was no improvement, or $|O| = 1$, the current classifier is returned.

```
0. Initialize network to naive Bayes.
1. Evaluate the current classifier.
2. Consider making each node a SuperParent.
   Let A_SP be the SuperParent which
   increases accuracy the most.
3. Consider an arc from A_SP to each orphan.
   If the best such arc improves accuracy,
   keep it and go to 2.
Else: Return the current classifier.
```

Table 4: An outline of the SuperParent algorithm (SP)

We defer a detailed discussion of our experimental results until section 3.2. However, Figure 4 demonstrates that SuperParent has essentially the same accuracy as HCS, even though it utilizes a more efficient search.

O(N) classifiers are constructed and evaluated for each arc added to the network, and O(N) arcs may be added. So the complexity for SP is O(N^2). We also utilize the two optimizations mentioned for HCS.

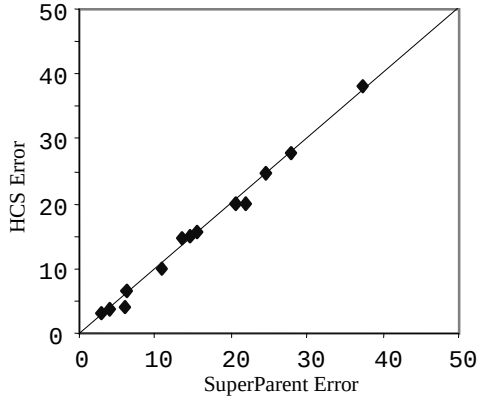

Figure 4: Scatter plot comparing SuperParent to HCS on 13 datasets from the UCI repository. Points on the diagonal line indicate no difference in performance.

3.1 HOW SUPERPARENT WORKS

If an attribute is truly independent of all other attributes, then the following equality is true.

$$P(C_i)\prod_k P(A_k = V_{k_j} | C_i) = P(C_i)\prod_k P(A_k = V_{k_j} | C_i \& A_{SP}) \quad (4)$$

Therefore making that attribute the SuperParent should not effect the classification accuracy. Because we are estimating the probabilities with a finite amount of data, we can expect small fluctuations in classification accuracy. If however, the right-hand side of equation 4 differs from the left-hand side by a substantial amount, we can infer that A_{SP} is related to at least one other attribute. By making each attribute a SuperParent in turn, we can detect which attributes make good parents of other (as yet, unknown) attributes. This is what we do in line 2 of Table 4.

Choosing the attribute, which, as SuperParent, most increases the classification accuracy, ensures that we have an attribute that is strongly correlated some other attribute, or attributes. We still need to find the attribute to which it is most strongly correlated. This we do in line 3 of Table 3.

Simply stated, the SuperParent algorithm is a heuristic, which repeatedly finds a good parent node, then that node's best child, until there is no more improvement to be had. The heuristic is not admissible. That is to say it is not guaranteed to produce the same set of augmenting arcs as HCS.

3.2 EXPERIMENTAL RESULTS WITH SUPERPARENT

Table 5 in the Appendix also lists the results of running SuperParent under the same conditions as Table 1 (in fact SuperParent was tested on exactly the same folds). Only on the dataset Segment is HCS significantly more accurate than SuperParent. These results are also summarized in Figure 4. It shows that there is little difference between the two algorithms. This indicates that searching for a good parent in the manner that SuperParent does is a useful heuristic for finding arcs to add.

4 WHY IS A CLASSIFICATION APPROACH BETTER?

There are two differences between Friedman and Goldszmidt's approach and the search approaches we proposed in the previous sections. First, they use a different criteria for deciding which arcs to add to the naïve Bayesian classifier. Second, Friedman and Goldszmidt's approach always adds *N*-1 arcs while our approach stops adding arcs when no improvement will occur. In order to gain an understanding of why our algorithm yields more accurate classifiers than the earlier approach, we consider modifications to each method.

1. We modify our search algorithm so that it always adds *N*-1 arcs (SP_{full}). This is done by always adding the arc that results in the most accurate classifier, but not requiring that an arc improve the accuracy when compared to not adding an arc. Comparing this modified algorithm to SP and TAN will allow us to determine if SP's superior performance is due to its ability to add fewer arcs.
2. We create a variant of Friedman and Goldszmidt's approach that does not necessarily add every arc in the tree to the Bayesian classifier (sTAN). In particular, once the tree is formed, we use a hill climbing search procedure to add arcs to the Bayesian classifier. We select from only those edges that appear in the tree and we consider both possible arc directions for all edges. Like our original hill-climbing search procedure, this stops when no arc addition results in a more accurate classifier. Because this variant of TAN is free from limitations with regard to the number of arcs that can be added, and can choose the direction of the arcs, we can use it to ascertain how whether the arcs selected by TAN improve accuracy when compared to SP.

We ran an experiment on ten datasets using the three above algorithms together with unsmoothed TAN and the normal SuperParent algorithm. We used the same methodology as mentioned in section 2.2, in particular, all classification algorithms were trained and tested on exactly the same cross validation folds. Table 5 summarizes the results.

Dataset	Tan	sTAN	SP_{full}	SP
Vote	93.9	94.2	**95.3**	**95.7**
Australia	80.4	82.2	82.9	**85.2**
Pima	75.5	76.6	**77.6**	**78.2**
Breast	**96.0**	**96.2**	**96.3**	**96.1**
Heart	73.5	74.7	75.2	**76.1**
Hepatitis	83.5	83.0	**85.5**	**84.3**
Vehicle	63.5	65.4	**69.2**	**70.3**
Soybean	82.0	84.7	**87.5**	**88.4**
E.coli	80.9	81.7	**84.5**	**84.3**
Post-op	70.0	70.3	**73.3**	**72.1**
Mean	*79.92*	*80.90*	*82.73*	*83.07*

Table 5: Experimental results of comparing various algorithms.

The last row contains the mean accuracy for each column, although we echo the often-stated caution of its debatable significance. The best accuracy for a given dataset is reported in bold text. Where a runner-up does not differ at the 5% confidence level (using a paired two-tailed t-test), it too is recorded in bold.

SP is clearly the superior classifier. SP_{full} does not do quite as well, but it is still significantly better than TAN on 8 of the datasets. The difference between its mean accuracy and SP's is only 11% of the difference between TAN and SP. We take this, as strong evidence that the (possible) disparity in the *number* of arcs added to the augmented Bayes classifier by the different approaches is not the main reason for SP's superior performance.

Given these results, we surmise that the overall difference in performance between TAN and SP is due to the fact that SP generally chooses a different set of augmenting arcs to add.

5 RELATED WORK

There has been much recent work on extensions to the naïve Bayes classifier. Kohavi (1994) has shown that although irrelevant features should theoretically not hurt the accuracy of naïve Bayes, in practice irrelevant features do degrade performance. He deals with this problem using wrappers for subset selection. Langley and Sage (1994) use a similar approach. Although neither approach directly deals with the problem of related attributes they can have the effect of mitigating the problem. If two attributes are related, then naïve Bayes will overweight the evidence from the two attributes. Deleting one of the two attributes may help.

Kononenko (1991) and Pazzani (1996) deal with attribute dependence by merging two (or more) related attributes into a new compound attribute, which replaces the original attributes in the classifier. While Kononenko uses a

statistical test to decide which attributes to merge, Pazzani achieves much better results by using predictive accuracy as a merging criteria.

6 CONCLUSIONS

In this paper, we have shown that it is possible to build classifiers that are superior to naïve Bayes, by finding related attributes and considering their dependence when classifying new instances. We further demonstrate two useful optimizations that greatly speed up the process of finding correlated attributes.

Acknowledgments

This research was supported by NSF grant IRI-9310413, AFOSR grant F49620-96-1-0224, and Sun Microsystems.

References

Chow, C. & Lui, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Trans. On Info Theory* 14:462-467.

Domingos, P, & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29, 103-130.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Proc of the 12th Int. Conference on Machine Learning*. (pp. 194-202). Tahoe , Ca: Morgan Kaufmann.

Dietterich, T. (1996) Statistical Tests for Comparing Supervised Classification Learning Algorithms. Technical Report. ftp://ftp.cs.orst.edu/pub/tgd/papers/stats.ps.gz

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

Fayyad, U. & Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence*, (pp. 1022-1027). San Francisco, CA.

Friedman, N. & Goldszmidt, M. (1996). Building classifiers using Bayesian networks. In *Proc, National Conference on Artificial Intelligence*, (pp. 1277-1284). Menlo Park, CA: AAAI Press.

Kohavi, R. (1994). Feature subset selection as search with probabilistic estimates. In *AAAI Fall Symposium on Relevance*.

Kononenko, I. (1991). Semi-naïve Bayesian classifier. In *Proceedings of the Sixth European Working Session on Learning*, 206-219. Berlin: Springer- Verlag.

Langley, P. & Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainly in Artificial Intelligence*. p 399-406. San Francisco, CA: Morgan Kaufmann.

Merz, C., Murphy, P. & Aha, D. (1997). UCI repository of machine learning databases. Dept of Information and Computer Science, University of California, Irvine. http://www.ics.uci.edu/mlearn/MLRepository.html.

Pazzani, M. (1996). Constructive induction of Cartesian product attributes. *Information, Statistics and Induction in Science*. Melbourne, Australia.

Pazzani, M. (1996). Searching for dependencies in Bayesian classifiers. In D. Fisher & H. J. Lenz (Eds.), *Learning from data: Artificial intelligence and statistics* V (pp. 239-248). New York, NY: Springer-Verlag.

Pearl, J. *1988 Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Ripley, B. (1996). Pattern recognition and neural networks. Cambridge University Press.

Sahami, M. (1996). Learning limited dependence Bayesian classifiers. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 335-338). Portland OR: AAAI Press.

Appendix

Dataset	Naïve	TAN	HCS	SP	TAN Time	SP Time
Vehicle	61.97 ± 1.58	63.47 ± 2.47	**70.17** ± 1.87	**70.25** ± 2.01	333	1390
Post-op	70.01 ± 0.83	70.06 ± 1.53	**72.82** ± 1.52	**72.13** ± 2.11	4	41
Lung	47.87 ± 7.34	54.82 ± 8.99	**58.34** ± 8.18	**59.12** ± 6.58	156	767
Australia	80.72 ± 0.68	80.42 ± 0.66	**84.74** ± 0.75	**85.20** ± 0.55	168	1299
Hepatitis	83.25 ± 1.37	83.50 ± 2.43	**84.75** ± 1.87	**84.25** ± 2.13	9	126
Vote	90.34 ± 0.78	93.91 ± 1.48	**95.58** ± 0.56	**95.71** ± 0.43	17	84
Heart	72.51 ± 3.30	73.52 ± 2.84	**78.73** ± 2.16	**76.10** ± 1.96	6	93
Soybean-Large	86.07 ± 1.19	82.04 ± 1.72	**88.83** ± 1.22	**88.41** ± 1.71	1046	13807
Pima	69.56 ± 1.35	75.47 ± 1.75	**78.00** ± 1.31	**78.22** ± 1.28	4	63
Breast	96.02 ± 0.45	96.45 ± 0.72	**97.41** ± 0.89	96.12 ± 0.81	21	172
Iris	**93.00** ± 1.00	**93.60** ± 0.95	**94.00** ± 1.35	**93.60** ± 1.25	3	10
Segment	90.92 ± 1.86	86.25 ± 1.65	**95.67** ± 1.07	94.45 ± 1.36	5491	62410
E.coli	80.21 ± 0.44	80.89 ± 0.69	**85.43** ± 0.75	**84.35** ± 0.34	16	91
exclusive-or	51.92 ± 2.30	54.52 ± 2.16	**68.22** ± 1.46	**70.71** ± 1.43	12	96

Table 6: Experimental results of comparing various algorithms. The best result and those not significantly worse than the best at the 5% confidence level are shown in bold. The last two columns contain the average time (in seconds) taken to build a classifier using TAN and SP