

A NEW ALGORITHM FOR LEARNING BAYESIAN CLASSIFIERS FROM DATA

A. KLEINER

Staffordshire University
Beaconside, Stafford ST18 0AD, UK
a.kleiner@staffs.ac.uk

B. SHARP

Staffordshire University
Beaconside, Stafford ST18 0AD, UK
b.sharp@staffs.ac.uk

Abstract

We introduce a new algorithm for the induction of classifiers from data, based on Bayesian networks. Basically this problem has already been examined from two perspectives: first, the induction of classifiers by learning algorithms for Bayesian networks, second, the induction of classifiers based on the naive Bayesian classifier. Our approach is located between these two perspectives; it eliminates the disadvantages of both while exploiting their advantages. In contrast to recently appeared refinements of the naive Bayes classifier, which captures single correlations in the data, we have developed an approach which captures multiple correlations and furthermore does a trade-off between complexity and accuracy. In this paper we evaluate the implementation of our approach with data sets from the machine learning repository and data sets artificially generated by Bayesian networks.

Keywords: Machine Learning, Naive Bayes Classifier, Bayesian Networks, MDL principle

INTRODUCTION

In this paper we introduce a new algorithm for the induction of classifiers from data, based on Bayesian networks. Basically this problem has already been examined from two perspectives: first from the induction of classifiers by learning algorithms for Bayesian networks, second, the induction of classifiers based on the naive Bayesian classifier. Our approach is located between these two perspectives; it eliminates the disadvantages of both while exploiting their advantages.

The first induction of classifiers involves a search over all possible networks and has been successfully solved in [2] and [4]. However, it can be considered as unsupervised learning [7] since it does not distinguish between attribute variables and the class variable. Thus the results for a classification task are not sufficiently accurate. The second induction approach is based on the refinement of the naive Bayes classifier, which has already proved its power for classification in many applications [13]. Due to the fact, that this classifier comes with the strong assumption of independence, refinements are achieved by relaxing this assumption.

Significant work in that field is found in [16], [14] and [7]. The latter approach improves the naive Bayes classifier by capturing single dependencies between the attributes. Our approach is motivated by this one but extends it by two new features. These two features allow the possibility of learning multiple correlations between attributes and the trade-off between complexity and accuracy. We argue that both features are important, on one hand, because data from real world applications is likely to have multiple correlations between its variables and on the other hand, because the application of classifiers to real world problems requires a fast computation. This computation, however, depends strongly on the complexity of the classifier. To realize these two features, we adopted the principle of *maximum description length* [2], which is a technique used in the general learning of Bayesian networks.

We shall denote variables that refer, for example, to attributes in a classification task, with capital letters, such as A, B, C and particular configurations of these variables in lowercase, such as a, b, c . A set of variables is denoted in bold, for example, $\mathbf{U} = \{A, B, C\}$. The classification problem then can be stated as follows. A training set of cases has to be partitioned into a fixed number of

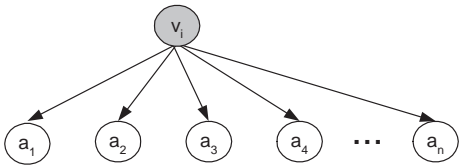


Figure 1: The structure of the naive Bayes classifier

classes. These “mapping” from cases to classes depends on particular configurations of the attributes and has to be learned by the classifier. A case is represented by the attributes (A_1, A_2, \dots, A_n) and the class V . Every attribute A_i can be in a certain state $A_i = a_i$ from its domain of N_{A_i} possible states. Each configuration \mathbf{A} of these attributes belongs to a class v_i from the set of classes V . The task is to learn a target mapping for each configuration to one of these classes. Finally, the quality of the induced classifier can be assessed by its ability to classify unknown configurations to an appropriate v_i .

NAIVE BAYESIAN CLASSIFIER

Among other techniques, the naive Bayesian classifier (or simply naive Bayes) is one of the most powerful tools in machine learning. It can compete with other classifiers, such as backpropagation or ID3, though its structure is less complex. Its power for text classification has been proven in [15], [11] and [13].

The Bayesian approach to achieve a mapping between classes and attributes, is to identify the class with the highest probability for a particular configuration of the attributes. In statistical terms, the thereby identified class is named as the *maximum a posteriori* (MAP) hypothesis:

$$v_{MAP} = \operatorname{argmax}_{v_i \in V} P(v_i | a_1, a_2, \dots, a_n) \quad (1)$$

Applying Bayes theorem, this yields to:

$$v_{MAP} = \operatorname{argmax}_{v_i \in V} \frac{P(a_1, a_2, \dots, a_n | v_i) P(v_i)}{P(a_1, a_2, \dots, a_n)} \quad (2)$$

and due to the constant presence of $P(a_1, a_2, \dots, a_n)$ this becomes:

$$v_{MAP} = \operatorname{argmax}_{v_i \in V} P(a_1, a_2, \dots, a_n | v_i) P(v_i) \quad (3)$$

This describes an approach for a correct classification of attributes with respect to their probabilities, estimated from the training data. The estimation of these probabilities, however, becomes intractable with increasing number of attributes, since the number of possible configurations of these attributes, also known as “atomic events”, grows drastically. To overcome this problem, the naive Bayes comes with the “naive” underlying assumption, that every attribute A_i is independent from the others, thereby the number of required probability values its

largely reduced. Under the assumption of independence, the conjunction of the attributes can be decomposed in a product of the probabilities of each single attribute: $P(a_1, a_2, \dots, a_n | v_i) = \prod_j P(a_j | v_i)$, which yields to the naive Bayes classifier:

$$v_{NB} = \operatorname{argmax}_{v_i \in V} P(v_i) \prod_j P(a_j | v_i) \quad (4)$$

In other words, this learning method involves a learning step, where the estimates for all $P(v_i)$ and $P(a_j | v_i)$ are determined by their frequencies in the training set by simply counting their occurrences. An induced classifier can then be used to classify any configuration of the attributes by multiplying for every class v_i the probabilities $P(a_j | v_i)$ of each attribute and selection of that class, which yields to the highest probability.

The performance of this simple approach has been measured in various applications. One interesting example is the classification of newsgroups, as reported in [11]. In this work, 20 newsgroups, each with 1000 articles, have been classified. The classes v_i were given by the names of this 20 news groups, for example *comp.sys.ibm.pc.hardware*, and the attributes by words from the English language appearing in those articles. The experiment lead to an amazing result of 89% accuracy, in contrast to a random classification with expected 5% accuracy. Noteworthy, however, is that the assumption of conditional independence was not necessarily kept by the data. One can imagine, that in the case of classification of texts in natural language, conditional dependencies must exist. For instance, it is likely to find the word “Intelligence” after the word “Artificial” or to find the word “Naive” before the word “Bayes”. However, recent results showed that the naive Bayes classifier performs well even with violation of this assumption.

This leads to the obvious question, whether we can achieve even better performance by using networks which consider dependencies in the data. Bayesian Networks [17] provide a method to represent such dependencies between variables and there are approaches to learn their structure and parameters from data.

LEARNING BAYESIAN NETWORKS FOR CLASSIFICATION

Bayesian Networks

A Bayesian network B for a set of random variables \mathbf{U} is defined by a structure S , describing a directed acyclic graph, and a set of parameters Θ , quantifying this structure. The structure is represented by arcs between the random variables X_1, X_2, \dots, X_n in \mathbf{U} , which indicate direct dependencies between them. Furthermore, the set of parameters Θ provides for every configuration of a variable X_i and the configurations of its parents $pa(X_i)$ a

value $\theta_{X_i|pa(X_i)}$ which is equal to the probability $P(X_i | pa(X_i))$ of this particular configuration. Thus the joint probability distribution above \mathbf{U} can be reconstructed by the multiplication of each nodes probabilities:

$$P_B(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P_B(X_i | pa(X_i)) \quad (5)$$

If $pa(X_i)$ consisted only of the class variable V for every $i \in 1, 2, \dots, n$ and $pa(V) = 0$, the above would describe a Bayesian network for a naive Bayes classifier. However, we are able to express far more complex relationships within \mathbf{U} . Basically these relationships are about dependence and independence between these variables. Let $\mathbf{A}, \mathbf{B}, \mathbf{C}$ be subsets in \mathbf{U} . Then there is conditional independence between \mathbf{A} and \mathbf{C} given, if $P(A | B) = P(A | B, C)$ holds, whenever $P(B, C) > 0$. That is, when the state of \mathbf{B} is known, no knowledge about \mathbf{C} will alter the probability of \mathbf{A} [10]. Of course this implies, that this holds for every possible configuration $\mathbf{a}, \mathbf{b}, \mathbf{c}$ of the subsets $\mathbf{A}, \mathbf{B}, \mathbf{C}$. In Bayesian networks, this independence is encoded by the following definition: Every variable X_i is independent of its nondescendants given its parents [17].

The learning of structure and parameters

This problem can be solved by a search over all possible networks and an estimation of the parameters Θ . To identify that network, which matches the data best, a commonly used method is to calculate the *log-likelihood* for B given D . Let $B = \{S, \Theta\}$ be a Bayesian network for the data set D with $D = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N\}$ where \mathbf{d}_i assigns for every variable in B a value. Then

$$LL(B | D) = \sum_{i=1}^N \log P_B(\mathbf{d}_i) \quad (6)$$

measures the probability that the data D was generated from the network B . That means, the bigger $LL(B | D)$ the more likely the examined network can represent the underlying distribution of D . Unfortunately this measure is not appropriate in its pure form for learning Bayesian networks, since it favours complex networks, which lead to a higher log-likelihood than simple ones.

However, it has been shown, that the number of possible structures increases drastically with the number of nodes in the network. Heckerman commented on this: "If we consider Bayesian network models with n variables, the number of possible structure hypotheses is more than exponential in n " [9]. Therefore, it is impossible to consider all of these models during a search. A common technique to avoid the consideration of all possible solution of a problem is to perform a *greedy-search*, which usually leads to a local maximum in the search space. To apply such a greedy-search to a Bayesian network, a scoring function is necessary, which returns a value for locally applied changes.

Suppose A_i is a node in a network with n variables. Consequently there are $n - 1$ possible parents for this node and 2^{n-1} possible combinations of them. Instead of applying all these combinations for every node, a greedy-search could be implemented which performs the operations *add parent* and *delete parent*, guided by a scoring function. A greedy-search works on the principle of not reconsidering operations done in previous steps. This leads finally to a local optimal solution, as long as the score indicates the optimal operation for every step.

Since the scoring function needs to be applicable locally, the log-likelihood, which returns a value corresponding to the whole network, cannot be used. Furthermore, as mentioned above, this measure tends to favour complex structures which we want to avoid. To solve this problems, two metric functions have been introduced, namely the *Bayesian Information Criterion* (BIC) [4] and the *Minimum Description Length* (MDL) criterion [2]. Both of these functions return a score which maximises the log-likelihood, however with a restriction by the complexity. Since these functions are similar from their principle, we focus on the MDL score, which also motivates our approach.

Minimum description length (MDL) principle

The MDL score consists of two parts, which are the previously introduced log-likelihood and the complexity of the model. The approach selects a model within a trade-off between these two components. The complexity of a network can be expressed by the number of bits necessary for its representation. Suppose there are n nodes in a network each with k parents, then the parents of a node can be encoded with $k \log_2(n)$ bits. Furthermore, the conditional probability tables, associated with each node, have to be encoded as well. For a node in a Bayesian network, one probability is needed for every possible configuration of the parents and the node itself. The information necessary to encode a Bayesian network with n nodes is:

$$\sum_{i=1}^n [k_i \log(n) + d(val(X_i) - 1) \prod_{X_j \in pa(X_i)} val(X_j)] \quad (7)$$

Where d denotes the number of bits necessary to represent the numeric value of a probability, $val(X)$ the number of possible states node X can take and $pa(X)$ denotes its set of parents. This is the encoding scheme, suggested in [2]. Since this formula sums over all nodes in the network, it can easily be decomposed for a single node.

The second part in the MDL score is a measure of how well the network represents the data. However, the log-likelihood cannot be used directly, since it cannot be decomposed for each node. The metric used in [2] is supported by the two following facts:

- The *Kullback-Leibler Cross-Entropy* between the true distribution $P(X)$ and the distribution $Q(X)$, generated by a Bayesian network, shrinks as $Q(X)$ more closely approximates $P(X)$. Due to the fact, that a network which generates the true distribution, also encodes the data optimally, the cross-entropy can be used as a measure to identify this network.

- As shown in [5], the cross-entropy between a true distribution $P(X)$ and a distribution $Q(X)$ is minimal if the underlying network generating $Q(X)$ is a maximum weight spanning tree and the weights between each node X_i and X_j are defined by the mutual information between them.

The mutual information between two nodes X_i and X_j is defined as:

$$I(X_i, X_j) = \sum_{X_i, X_j} P(X_i, X_j) \log_2 \frac{P(X_i, X_j)}{P(X_i)P(X_j)} \quad (8)$$

which sums over all possible states of X_i and X_j . Given the mutual information between all variables, one can build a maximum weighted spanning tree and therefore approximate the maximum log-likelihood, with respect to the data. The mutual information, however, can be applied to every single node.

Lam and Bacchus evaluated their approach with networks of different sizes. In most of the cases the MDL score was able to reconstruct the original Bayesian network which generated the data for the learning process. In [3] they extended their approach to refining existing network structures, and particularly considered the encoding of changes between one model and a potentially better one.

However an evaluation in [6] showed, that the learning of classifiers with this approach leads to poor results. In that paper they argued, that the reason for this is the scoring function itself. Since the MDL score favours simple networks, it tends to reduce relations between attribute variables and the class variable. In particular, for problems with many attributes, classifiers produce poor predictions, since important attributes are cut from the class node and therefore not able to contribute directly to the classification. They considered learning with the MDL score as “unsupervised” learning, since the learning algorithm has no information given about which node represents the class. They also claimed, that the learning of a classification problem with the Bayesian information criterion (BIC) suffers from the same problem. For these reasons, the classification problem has been tackled using the naive Bayes classifier, which is described in the next section.

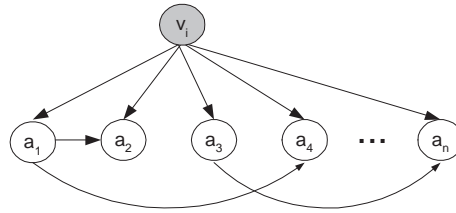


Figure 2: The structure of the tree augmented naive Bayes classifier (TAN)

IMPROVING NAIVE BAYES

As we saw in the previous section, standard learning algorithms for Bayesian networks are insufficient to solve classification tasks. Therefore several efforts have been undertaken to improve the naive Bayes classifier for the classification of correlated data. Since the naive Bayes classifier comes with the strong assumption of independence, these approaches are motivated in relaxing this assumption. Basically this leads to a search of correlation between the attributes and methods to reflect them in the classifier. In the literature three significant approaches can be found:

- Sub feature selection
- The joining of attributes
- The Tree Augmented Naive Bayes (TAN)

The first method is found in [14] who described a greedy-search algorithm which excludes strong correlated attributes from the classifier. This takes place in a forward selection manner, which starts with an empty set of attributes and incrementally adds new ones, unless a termination criterion is met. The selection of attributes takes place with respect to a metric, which identifies attributes with a crucial contribution to the classification. The metric they used was *leave-one-out cross validation*¹, since it is the most precise measure for the accuracy of a classifier.

The second approach is described in [16] and tackles the problem in the opposite way. Rather than excluding correlated attributes, this approach joins them together to achieve higher classification accuracy. They evaluated the selection of attributes in a forward and backward manner, with two possible operations, which are to add an attribute to the classifier (respectively delete) and to join an attribute with one in the classifier. They also used the leave-one-out technique to indicate whether a change was successful or not.

The latest work in this field is the *tree augmented naive Bayes* (TAN) approach, described in [7]. This approach performs better than the other two and is also

¹Described later in this paper

the motivation for our approach. It is based on the work from [5] and [8], which developed algorithms for building a maximum weighted spanning tree by the mutual information and conditional mutual information respectively. Note, as we saw above, the first one also motivated the MDL principle.

The TAN algorithm builds a network structure, depending on the mutual information between nodes. Basically it captures correlations between attributes by drawing arcs between them. However, this approach comes with intended restrictions, since it is goal orientated to classification tasks. The first restriction is that every attribute is connected to the class variable that yields to the structure of naive Bayes. The second restriction is that every attribute may own one more parent besides the class variable (see Figure 2). The so resulting structure improves the naive Bayes, since it can capture single relations between two attributes. On the other hand, it avoids a search through the space of all possible networks by these restrictions to the structure. The results reported by this approach are equal or better than results reported from the naive Bayes.

We argue, however, that this approach comes with two crucial disadvantages:

- Only single correlations between attributes can be captured, due to the restriction to one additional parent besides the class node
- Parents are chosen with respect to the maximum log-likelihood, but the resulting complexity is not considered.

Certainly the first is reasonable, since networks with $n > 2$ parents are more complex. However, suppose a network, where the configurations of attribute A_4 depends on the configuration of A_1, A_2, A_3 , which are independent from each other. The TAN Bayes would add the node with maximal influence on A_4 as parent and ignore the influence of the other two. In the worst case, these ignored attributes would even be connected as children to other nodes.

The second disadvantage becomes significant if there are nodes with plenty of states. The algorithm would favour networks, which increase $LL(D | B)$, regardless of their complexity. Note, that the size of a node's conditional probability table (CPT), which holds all possible configurations of the node itself and its parents, grows drastically with the number of states of each parent. For a node with i states and two parents with j and k states, the CPT consists of $i * j * k$ entries. We argue further that the predictions of the classifier become less reliable with more complex nodes. This comes from the fact that the bigger the CPT, the more probabilities are need to

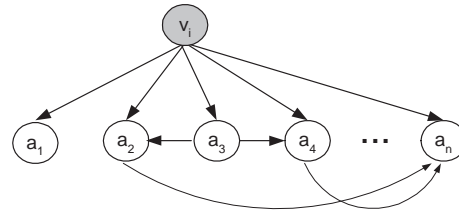


Figure 3: An example network, as it might result from our approach

be estimated from the data. These configurations, however, are less likely to be found in the data with increasing complexity. Thus their estimates are insufficient and lead to poor predictions. To overcome this problem, Friedman and Goldszmidt introduced a smoothing operation to fill the gap of unreliable probabilities.

LEARNING CLASSIFIERS WITH MULTIPLE CORRELATIONS AND LESS COMPLEXITY

Our approach can be found between the TAN architecture and the MDL approach for learning Bayesian networks. Furthermore it combines the advantages of both and excludes their disadvantages, which were previously identified in this paper. The algorithm can be characterised by two main features as follows. First, since our algorithm is supposed to be applied for classification tasks, and the class variable is usually known, we limit the number of possible networks to those whose attributes have the class node as parent. Second, we start to search for relations between these attributes, using the MDL score which favours simple relations with maximum contribution to the log-likelihood.

The second feature applies a greedy search for each node, to avoid the huge search space of all valid parent combinations. The metric for this greedy search is the trade-off between mutual information and the complexity caused by the change. We used the modified version of the mutual information from [2], which defines a weight $W(A_i, pa(A_i))$ for attribute A_i and its parents $pa(A_i)$ with

$$\sum_{A_i, pa(A_i)} P(A_i, pa(A_i)) \log_2 \frac{P(A_i, pa(A_i))}{P(A_i)P(pa(A_i))}. \quad (9)$$

Against this information measure stands the complexity $C(A_i, pa(A_i))$, resulting from this parent configuration with

$$val(A_i) \prod_{A_j \in pa(A_i)} val(A_j), \quad (10)$$

where $val(X)$ the number of possible states for X denotes. The score in our greedy search consists of these two

components. Suppose two parent configurations $pa_1(A_i)$ and $pa_2(A_i)$ for attribute A_i , where pa_2 is extended by one more attribute than pa_1 . For a comparison of these two parent configurations, we compute the value pairs (W_1, C_1) and (W_2, C_2) for both respectively and compare the relatively growth of complexity and weight:

$$\alpha \frac{W_2}{W_1} > \frac{C_2}{C_1} \quad (11)$$

Where α denotes a weighting factor to control the trade-off between complexity and information gain. Using this formula a new parent configuration is accepted, if the relative information exceeds the relative complexity.

The greedy search needs a sorted list for each node, indicating which of the other nodes are worth of becoming parents. Thus we generate a list for each node, consisting of every possible parent A_j and rank them by their weight $W(A_i, V, A_j)$. Given this list, the algorithm successively adds a node from this list as parent and keeps it as parent, if the achieved weight W exceeds the complexity C . In addition to the operations *add arc* and *delete arc* we use the operation *reverse arc*, which is necessary in the case that a node A_i favours another node A_j as parent, but A_i has already been chosen as parent for A_j . In this case we reverse this arc, depending on the weight of both. The computational complexity of this greedy search is $O(n^2)$ for n attributes, since in the worst case all possible $n - 1$ parents are considered by every node. The algorithm for n attributes can be summarized as follows:

- Generate for every attribute a parent list P , corresponding to a naive Bayesian classifier with $P_V = \{\}$ and $P_i = V$ for every $i \in \{1..n\}$
- Repeat for every A_i with $i \in \{1..n\}$:
 - Generate a list L** , consisting of $n - 1$ entries which store the weights $W(A_i, V, A_j)$, for every possible parent node A_j .
 - Sort L** with ascending W .
 - Repeat** for all $A_j \in P$
 - Compute** $W_1 = W(A_i, P_{A_i})$ and $C_1 = C(A_i, P_{A_i})$
 - Add** A_j to P_{A_i} (add arc)
 - Compute** $W_2 = W(A_i, P_{A_i})$ and $C_2 = C(A_i, P_{A_i})$
 - If** $\alpha \frac{W_2}{W_1} < \frac{C_2}{C_1}$
 - Remove** A_j from P_{A_i} (delete arc)
 - ElseIf** $A_i \in P_{A_j}$
 - If** $W(A_i, P_{A_i}) > W(A_j, P_{A_j})$
 - Reverse arc**
 - end**
 - end**
 - end**
 - end**
- Return classifier

EXPERIMENTS

Methodology

The naive Bayes, the TAN Bayes and our approach, named multiple Bayes, have been implemented for an

evaluation with different training sets. We used on the one hand training sets from the machine learning repository [1] and on the other hand with Bayesian networks artificially generated data. The latter comes with the advantage, that the underlying network is already known and thus the induced classifier can be compared with it. We build Bayesian networks with the commercial package *Netica* and sampled sufficient cases from it.

In line with other research papers, the accuracy of each classifier has been determined by the *leave-one-out cross validation* [12]. In contrast to the less precise *hold-out* method, where a classifier is induced with $\frac{2}{3}$ of the training data and its accuracy measured with the other $\frac{1}{3}$, this method induces a classifier with all samples less one and measures its accuracy with that sample, left out during the training. This process is repeated for all samples in the training data, and the accuracy calculated by correctly classified samples divided by all samples. A detailed examination on the evaluation of classifiers is found in [12].

Results

Generally we expected from our results a better or equal accuracy as the naive Bayes classifier. Furthermore we expected that the complexity of the generated classifier would be located between naive Bayes and TAN. Table 1 shows the properties of the data set and training set. The results with this data is found in table 2, which lists the accuracy for the naive Bayes classifier, the tree augmented naive Bayes and multiple Bayes.

As it can be seen in table 2, the multiple Bayes approach has achieved for the first two data sets an accuracy equal to the naive Bayes. The TAN classifier, however, achieved in both sets less accuracy than the others. These sets have not been chosen arbitrary, they both have very little correlations between their attributes. Thus our classifier preferred the most simple structure, which is the naive Bayes (no additional parents), and achieved thus the same accuracy. Since the TAN approach ignores the balance between complexity and accuracy, it builds a classifier, based on weak correlations between the attributes. The resulting parent-child connections are poorly supported by the data, which explains the loss of accuracy.

The third data set comes with 16 attributes. As we discovered with our classifier, two from this 16 attributes are significantly influenced by more than three others. This stands in contrast to the other 14 attributes, which are maximal with two correlated. Thus our classifier returned a structure where 14 attributes are connected to two or less parents, these two strong correlated nodes, however, with more than three. Since the TAN architecture allows a maximum of one parent for each node, it was not able to capture these multiple correlations. Due

	# Var.	# Cases	# Av. states
breast	11	699	10
balance	5	625	5
votes	17	435	3
ABN	5	1000	2

Table 1: Properties of the used datasets

	Naive Bayes	TAN	Multi Bayes
breast	97,42±0,60	92,56±1,00	97,42±0,60
balance	92,16±1,10	85,28±1,42	92,16±1,10
votes	90,34±1,41	89,20±1,61	92,42±0,60
ABN	70,00±1,45	70,90±1,44	73,70±1,41

Table 2: Accuracy of the tested classifiers

to this, our approach was able to achieve a higher accuracy. Furthermore this could be reached with a slightly more complex classifier, than the one returned by TAN, since not all attributes had been connected with a parent.

The last data set which we examined, was generated with an artificial Bayesian Network. The structure of this network was chosen to reflect multiple correlations as the previous example as well. We sampled 1000 cases from this network and used them with our classifier. The result was a classifier reflecting all correlations defined in the Bayesian network and thus succeeded with the highest accuracy.

CONCLUSION

We proposed a new architecture for the induction of classifiers, based on Bayesian networks. Essentially this was carried out by the adoption of the MDL principle to the naive Bayes classifier.

Our results show that our refined classifier yields in all cases an accuracy equal to or better that of naive Bayes, and furthermore it outperforms TAN in the case of data with weak or multiple correlations. Our assumption, that correlations are only worth modelling in a classifier if they are cheap in terms of complexity, has been reflected by this results. We intended to do further tests, to cover a wide range of different data sets.

The complexity of our algorithm is equal to that of other approaches and computationally tractable. However, the calculation of the mutual information seems to limit the speed of the induction process significantly. Thus our approach, and all other methods based on the mutual information, are likely to induce classifiers slowly, if attributes with numerous states are found in the data. To overcome this problem, a simpler method for the identification of correlations in the data has to be found. This method has to be capable to identify statistical correlations and non-statistical correlations, as for example given

by the parity problem, as well. This area is to be explored in the next stage.

ACKNOWLEDGEMENTS

We would like to thank David Emery and Bill Walley for their useful discussions and comments.

References

- [1] D. W. Aha and K. Murphy. UCI repository of machine learning databases, 1995. <http://www.ics.uci.edu/mllearn/MLRepository.html>.
- [2] F. Bacchus and W. Lam. Learnin Bayesian Belief Networks: An Approach based on the MDL Principle. In *Computational Intelligence*, volume 10, pages 269–293, 1994.
- [3] F. Bacchus and W. Lam. Using New Data to Refine a Bayesian Network. In *Uncertainty in Artificial Intelligence*, pages 383–390, 1994.
- [4] D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian Networks: The combination of knowledge and statistical data. In *Machine Learning*, volume 20, pages 197–243, 1995.
- [5] C. K. Chow and C. N. Lui. Approximating discrete probability distributions with dependence trees. In *IEEE Transactions on Info. Theory*, volume 14, pages 462–467, 1968.
- [6] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. In *Machine Learning*, volume 29, pages 131–163, 1997.
- [7] N. Friedman and M. Goldszmidt. Building Classifiers using Bayesian Networks. In *Thirteenth National Conf. on Artificial Intelligence*, 1996.
- [8] D. Geiger. An entropy-based learning algorithm of Bayesian conditional trees. In *UAI'92*, pages 92–97, 1992.
- [9] D. Heckerman. A Tutorial on learning with Bayesian networks. Techn. Report, Microsoft Research, Redmond, Washington, 1995.
- [10] F. V. Jensen. *An Introduction to Bayesian Networks*. UCL Press Limited University College London, England, 1996.
- [11] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. Computer Science Techn. Report CMU-CS-96-118, Carnegie Mellon University, 1996.
- [12] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI'95*, pages 1137–1143, 1995.
- [13] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339, San Fransisco, Calif., 1995. Morgan Kaufmann.
- [14] P. Langley and S. Saga. Induction of selective Bayesian classifiers. In *UAI'94*, pages 399–406, 1994.
- [15] D. Lewis. *Representation and learning in informational retrieval*. Dissertation, Dept. of Computer and Information Science, University of Massachusetts, United States of America, 1991. (COINS Technical Report 91-93).
- [16] M. J. Pazzani. Searching for dependencies in Bayesian classifiers. In *Proc. of the 5th Int. Workshop on Artificial Intelligence and Statistics*, 1995.
- [17] J. Perl. *Probalistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Fransisco, Calif., 1988.