An Integrated Approach to Learning Bayesian Networks of Rules

Jesse Davis[1], Elizabeth Burnside[1], Inês de Castro Dutra[2], David Page[1], and Vítor Santos Costa[2]

[1] Department of Biostatistics and Medical Informatics, University of Wisconsin-Madison, USA
[2] COPPE/Sistemas, UFRJ, Centro de Tecnologia, Bloco H-319, Cx. Postal 68511 Rio de Janeiro, Brasil

Abstract. Inductive Logic Programming (ILP) is a popular approach for learning rules for classification tasks. An important question is how to combine the individual rules to obtain a useful classifier. In some instances, converting each learned rule into a binary feature for a Bayes net learner improves the accuracy compared to the standard decision list approach [3,4,14]. This results in a two-step process, where rules are generated in the first phase, and the classifier is learned in the second phase. We propose an algorithm that interleaves the two steps, by incrementally building a Bayes net during rule learning. Each candidate rule is introduced into the network, and scored by whether it improves the performance of the classifier. We call the algorithm SAYU for Score As You Use. We evaluate two structure learning algorithms Naïve Bayes and Tree Augmented Naïve Bayes. We test SAYU on four different datasets and see a significant improvement in two out of the four applications. Furthermore, the theories that SAYU learns tend to consist of far fewer rules than the theories in the two-step approach.

1 Introduction

Inductive Logic Programming (ILP) is a popular approach for learning in a relational environment. Given a set of positive and negative examples, an ILP system finds a logical description of the underlying data model that differentiates between the positive and negative examples. Usually this description is a set of *rules* or clauses, forming a logic program. In this case, unseen examples are applied to each clause in succession, forming a *decision list*. If the example matches one of the rules, it receives a positive label. If the example does not match any rule, it receives the negative classification. In an ideal world, where the rules would perfectly discriminate between the two classes, the decision list would represent an optimal combination scheme. In practice, it is difficult to find rules that do not cover any negative examples. As the precision of the individual rules declines, so does the accuracy of the decision list, as it maximizes the number of false positives.

The key question becomes how to combine a set of rules to obtain a useful classifier. Previous work has shown that an effective approach is to treat each learned rule as an attribute in a propositional learner, and to use the classifier to determine the final label of the example [3,4,14]. This methodology defines a two-step process. In the first step, an ILP algorithm learns a set of rules. In the second step, a classifier combines the learned rules. One weakness of this approach is that the rules learned in the first step are being evaluated by a different metric than how they are ultimately scored in the second step. ILP traditionally scores clauses through a coverage score or compression metric. Thus we have no guarantee that the rule learning process will select the rules that best contribute to the final classifier.

We propose an alternative approach, based on the idea of *constructing the classifier as we learn the rules.* In our approach, rules are scored by how much they improve the classifier, providing a tight coupling between rule generation and rule usage. We call this methodology *Score As You Use* or *SAYU*. Recently Landwehr, Kersting and De Raedt[10] have also provided a tight coupling between rule generation and rule usage, by integrating FOIL and Naïve Bayes, although their scoring function for rules is not exactly the improvement in performance of the Naïve Bayes classifier that the rule provides. The relationship to this important work is discussed in Section 5.

In order to implement SAYU, we first defined an interface that allows an ILP algorithm to control a propositional learner. Second, we developed a greedy algorithm that uses the interface to decide whether to retain a candidate clause. We implemented this interface using Aleph to learn ILP rules, and Bayesian networks as the combining mechanism. Previous experience has shown good results in using Bayes nets as a combining mechanism [3,4,14]. We used two different Bayes net structure learning algorithms, Naïve Bayes and Tree Augmented Naïve Bayes (TAN) [6] as propositional learners. Our results show that, given the same amount of CPU time, SAYU clearly outperforms the original two-step approach. Furthermore, SAYU learns smaller theories. These results were obtained even though SAYU considers far fewer rules than standard ILP.

2 Implementing SAYU

SAYU requires an interface to propose rules to the propositional learner. Additionally, SAYU needs to know the score of each clause in order to help guide rule search. The interface consists of the following three methods.

The $Init()$ function initializes the propositional learner with a table containing only the class attribute. The $NewAttribute(NewFeature)$ function introduces $NewFeature$ into the training set and learns a new classifier incorporating this attribute. It returns a score for the new network on a set of examples. The $Commit()$ function permanently incorporates the most recently evaluated feature into the classifier.

We use the interface to design a greedy learning algorithm, where the ILP system proposes a rule and converts it into a new attribute. The rule is then incorporated into the learner. If the rule improves the score, it is retained by the classifier. Otherwise, we discard the rule and revert back to the old classifier.

```
Input: Stop Criteria, Scoring Function
Output: Propositional Classifier
$CurrentScore = Init()$;
while Stop criteria not met do
    Choose a positive example as a seed and saturate the example;
    repeat
        $NewFeature$ = Generate new clause according to saturated example;
        $NewScore = NewAttribute(NewFeature)$;
        if $NewScore$ exceeds $CurrentScore$ then
            $Commit()$;
        end
    until $NewScore$ exceeds $CurrentScore$;
end
```

Algorithm 1: Implementing SAYU

Our implementation depends on the ILP system and on the propositional learner. Following previous work, we used saturation based learning ILP systems, in the style of the MDIE algorithm used in Progol [12] and Aleph [19]. In MDIE, the ILP search proceeds by randomly choosing an unexplained seed, and saturating that seed to obtain its most specific, or *saturated* clause. It then searches the space of clauses that generalize the saturated clause until finding the *best* clause.

The algorithm we used for this work follows the same principles, with one major difference: we search for the first *good* clause for each seed, instead of continuing search until finding the best clause. The main reason for picking the first good clause is that the best clause may be hard to find, thus exhaustively searching for the best clause may end up wasting our time on a single seed. Our implementation is shown in Algorithm 1.

Next, we present our propositional learning algorithms. Bayesian learning algorithms have several important advantages for our purposes. First, they allow us to give examples a probability. Second, Naïve Bayes is a well known approach that often performs well, and is particularly suitable for incremental learning. The drawback of Naïve Bayes is that it assumes that all of the rules are independent, given the class value. We evaluate Naïve Bayes against Tree Augmented Naïve Bayes (TAN) [6]. TAN networks can be learned efficiently, and can represent a limited set of dependencies between the attributes.

Finally, we need to define a scoring function. The main goal is to use the scoring function for both learning and evaluation. Furthermore, we wish to be able to handle datasets that have a highly skewed class distribution. In the presence of skew, precision and recall are often used to evaluate classifier quality. In order to characterize how the algorithm performs over the whole precision recall space, we follow Goadrich et.al. [7], and adopt the area under the precision-recall curve as our score metric. When calculating the area under the precision-recall curve, we integrate from recall levels of 0.2 or greater. Precision-recall curves can be misleading at low levels of recall as they have high variation in that region.

3 Methodology

We evaluated our algorithm with four very different datasets, corresponding to different applications of ILP. Two of the applications are relatively novel, the *Mammography* and *Yeast Proteins* datasets. The other application, *Carcinogenesis*, is well-known in the Inductive Logic Programming Community. Finally, we used the Univeristy of Washington *Advised By* dataset that is becoming a popular benchmark in Statistical Relational Learning [9,17].

Mammography. The *Mammography* dataset was the original motivation of this work. The National Mammography Database (NMD) standard established by the American College of Radiology. The NMD was designed to standardize data collection for mammography practices in the United States and is widely used for quality assurance. The database consisted of 47,669 mammography examinations on 18,270 patients. The dataset contains 435 malignant abnormalities and 65,365 benign abnormalities. It is important to note that the data consists of a radiologist's interpretation of a mammogram and not the raw image data. A mammogram can contain multiple abnormalities. The target predicate we are trying to predict is whether a given abnormality is benign or malignant. We randomly divided the abnormalities into ten roughly equal-sized sets, each with approximately one-tenth of the malignant abnormalities and one-tenth of the benign abnormalities. We ensured that all abnormalities belonging to a given patient appeared in the same fold [2].

Yeast Protein. Our second task consists of learning whether a yeast gene codes for a protein involved in the general function category of *metabolism*. We used for this task the MIPS (Munich Information Center for Protein Sequence) Comprehensive Yeast Genome Database, as of February 2005 [11]. Positive and negative examples were obtained from the MIPS function category catalog. The positives are all proteins/genes that code for metabolism, according to the MIPS functional categorization. The negatives are all genes that have known functions in the MIPS function categorization and do not code for metabolism. Notice that the same gene may code for several different functions, or may code for different sub-functions. We used information on gene location, phenotype, protein class, and enzymes. We also used gene-to-gene interaction and protein complex data. The dataset contains 1,299 positive examples and 5,456 negative examples. We randomly divided the data into ten folds. Each fold contained approximately the same number of positive and negative examples.

Carcinogenesis. Our third dataset concerns the well-known problem of predicting carcinogenicity test outcomes on rodents [18]. This dataset has a number of attractive features: it is an important practical problem; the background knowledge consists of a number of non-determinate predicate definitions; experience suggests that a fairly large search space needs to be examined to obtain a good clause. The dataset contains 182 positive carcinogenicity tests and 148 negative tests. We randomly divided the data into ten folds. Each fold contained approximately the same number of positive and negative examples.

Table 1. Mammography. All metrics given are averages over all ten folds.

Algorithm	Clauses in Theory	Number Predicates per Clause	Clauses Scored
Aleph	99.6	2.8213	620000.0
SAYU-NB	39.1	1.4655	85342.9
SAYU-TAN	32.8	1.4207	20944.4

Advised By. Our last dataset concerns learning whether one entity is advised by other entity, and it is based on real data obtained by Richardson and Domingos from the University of Washington CS Department [17]. The example distribution are skewed, we have 113 positive examples versus 2,711 negative examples. Following the original authors, we divide the data in 5 folds, each one corresponding to a different group in the CS Department.

4 Experimental Setup and Results

On the first three datasets we perform stratified, ten-fold cross validation in order to obtain significance results. On each round of cross validation, we use five folds as a training set, four folds as a tuning set and one fold as a test set. We only saturate examples from the training set. Since the Advised By dataset only has five folds, we used two folds for a training set and two folds as a tuning set. The communication between the Bayes net learner and the ILP algorithm is computationally expensive. The Bayes net algorithm might have to learn a new network topology and new parameters. Furthermore, inference must be performed to compute the score after incorporating a new feature. The SAYU algorithm is strictly more expensive than standard ILP as SAYU also has to prove whether a rule covers each example in order to create the new feature. To reflect the added cost, we use a time-based stop criteria for the new algorithm. In effect, we test whether, given an equal amount of CPU time, the two-step approach or SAYU performs better.

To obtain a performance baseline we first ran a set of experiments that use the original two-step process. In all experiments we use Srinivasan's Aleph ILP System [19] as the rule learning algorithm. First, we used Aleph running under `induce_cover` to learn a set of rules for each fold. `Induce_cover` implements a a variant of Progol's MDIE greedy covering algorithm, where we do not discard previously covered examples when we score a new clause. Second, we selected the rules using a greedy algorithm, where we pick the rule with the highest m-estimate such that it covers an unexplained training example. Subsequently, we converted each rule into a binary feature for a Naïve Bayes and TAN classifier. In the baseline experiments, we used both the training and tuning data to construct the classifier and learn its parameters. Furthermore, we recorded the CPU time that it took for each fold to run to completion. This time was used as the stop criteria for the corresponding fold when evaluating the integrated approach. To offset potential differences in computer speeds, all of the experiments for a given dataset were run on the same machine.

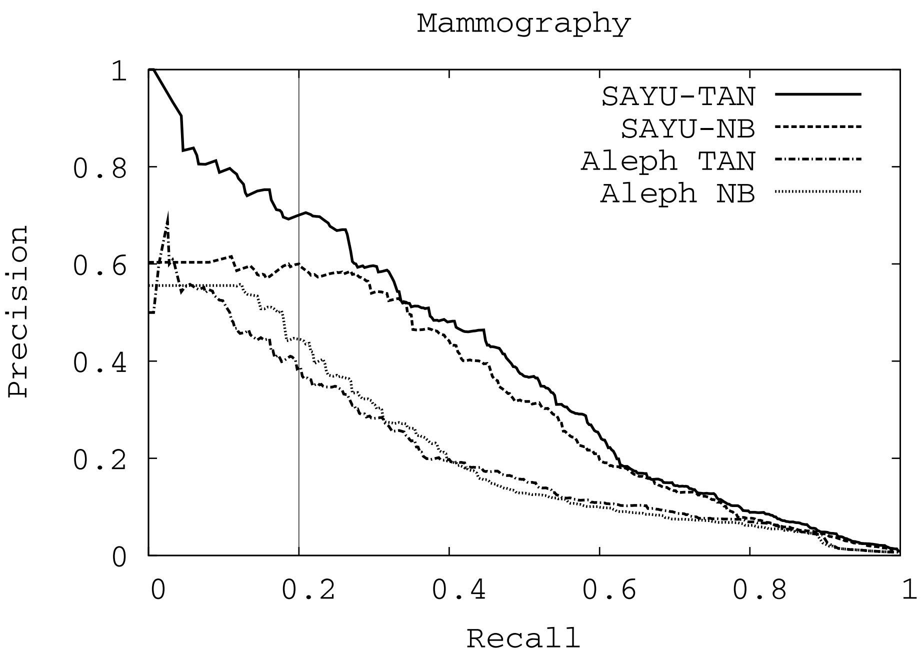

Fig. 1. Mammography Precision-Recall Curves

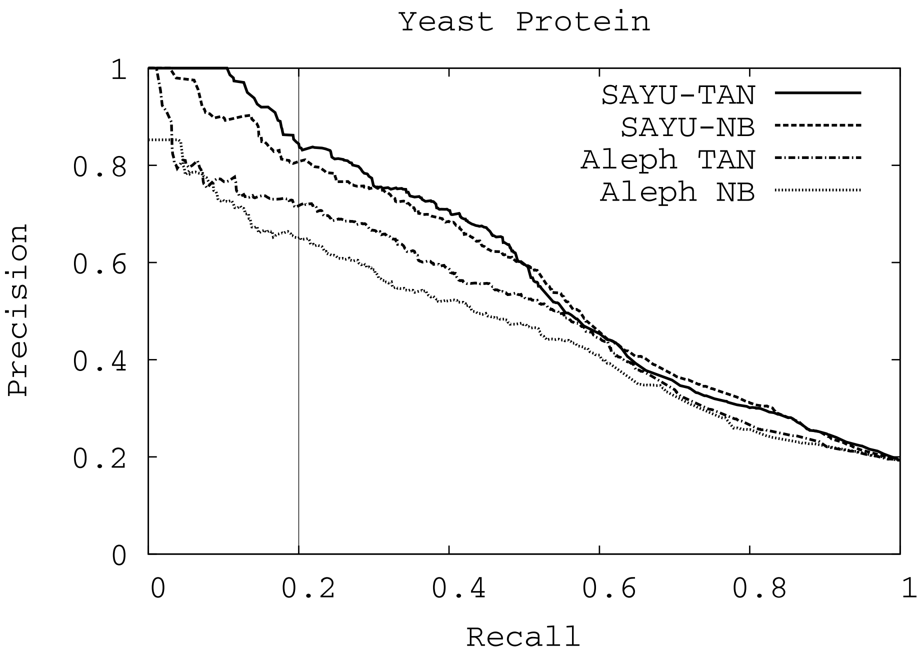

Fig. 2. Yeast Protein Function Precision-Recall Curves

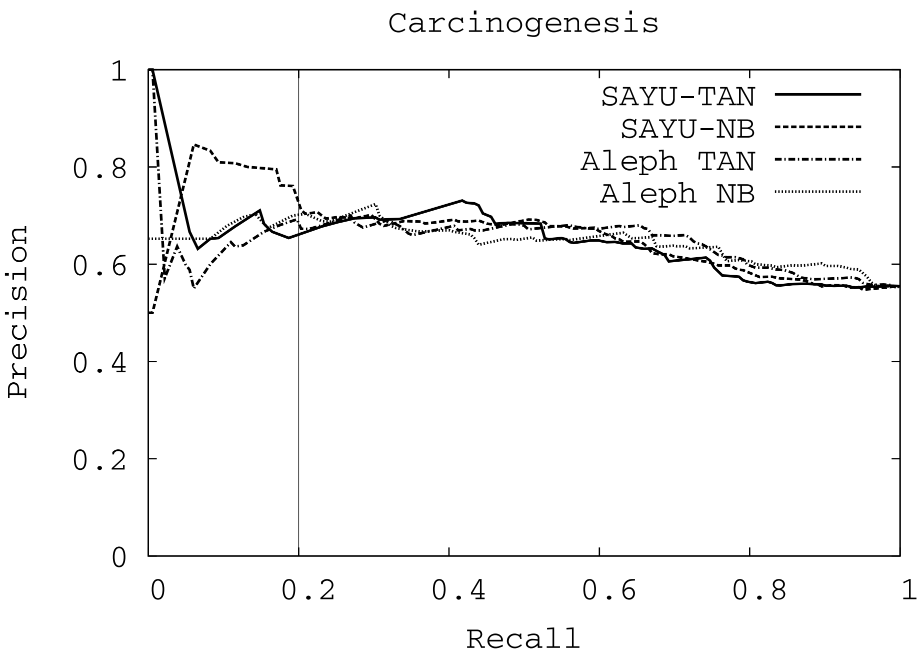

Fig. 3. Carcinogenesis Precision-Recall Curves

Table 2. Yeast Protein. All metrics given are averages over all ten folds.

Algorithm	Clauses in Theory	Number Predicates per Clause	Clauses Scored
Aleph	169.5	2.9345	915654.3
SAYU-NB	13.9	1.1367	190320.4
SAYU-TAN	12.5	1.152	131719.8

Table 3. Carcinogenesis. All metrics given are averages over all ten folds.

Algorithm	Clauses in Theory	Number Predicates per Clause	Clauses Scored
Aleph	185.6	3.5889	3533521.1
SAYU-NB	8.7	1.6897	874587.7
SAYU-TAN	12.1	1.9504	679274.6

For SAYU, we use only the training set to learn the rules. We use the training set to learn the structure and parameters of the Bayes net, and we use the tuning set to calculate the score of a network structure. Again, we use Aleph to perform the clause saturation and propose candidate clauses to include in the Bayes Net. In order to retain a clause in the network, the area under the precision-recall curve of the Bayes net incorporating the rule must achieve at least a two percent improvement over the area of the precision-recall curve of the best Bayes net.

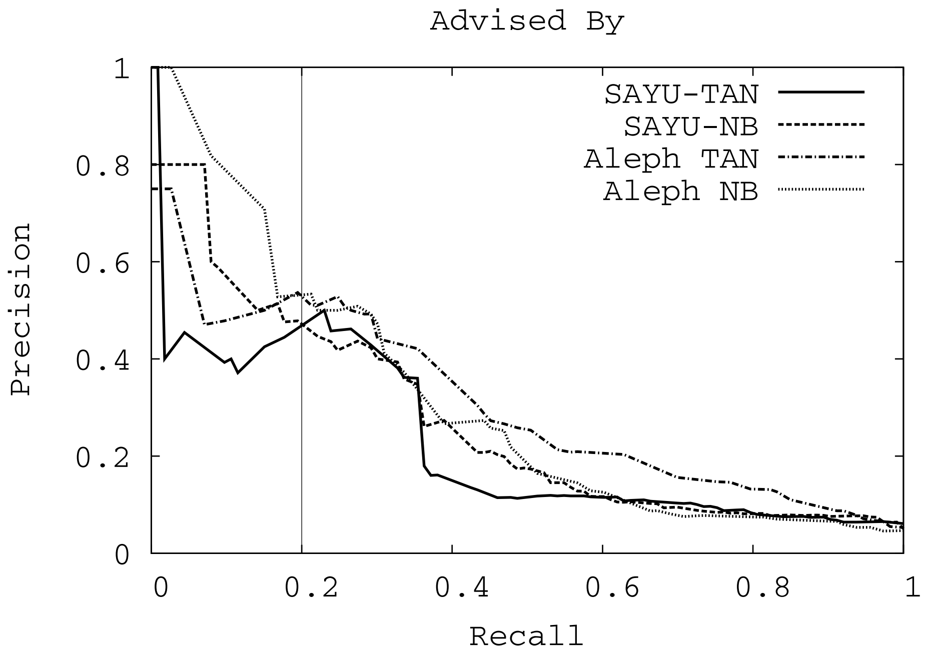

Fig. 4. Advised By Precision-Recall Curves

Figures 1 through 4 show precision-recall curves for all four datasets. In all graphs, curves were generated by pooling results over all ten folds. SAYU-NB refers to the integrated approach of incrementally learning a Naïve Bayes net. SAYU-TAN refers to the integrated approach of incrementally learning a TAN network. Aleph-NB refers to the two-step approach consisting of rule learning with Aleph and rule combination with Naïve Bayes. We use Aleph-TAN to represent learning rules with Aleph and then layering a TAN network over them.

The Mammography dataset (Figure 1) shows a clear win for SAYU over the original two-step methodology. We used the paired *t-test* to compare the areas under the curve for every fold, and we found the difference to be statistically significant at the 99% level of confidence. The difference between using SAYU-TAN and SAYU-NB is not significant. The difference between using TAN and Naïve Bayes to combine the Aleph learned rules is also not significant. Moreover, the results using SAYU match our best results on this dataset [2], which had required more computational effort.

The Yeast Protein dataset (Figure 2) also shows a win for SAYU over the original two-step methodology. The difference is not as striking as in the Mammography dataset, mostly because Aleph TAN learning did very well on one of the folds. In this case Aleph TAN is significantly better than Aleph NB with 98% confidence. SAYU-TAN learning is significantly better than Aleph NB with 99% confidence, and Aleph TAN with 95% confidence. SAYU-NB is better than

Aleph NB with 99% confidence. However, it is not significantly better than Aleph TAN (only at 90% confidence), despite the fact that SAYU-NB beats two-step TAN on nine out of ten folds.

The results for Carcinogenesis (Figure 3) are ambiguous: no method is significantly better than the other. One possible explanation is that precision-recall might not be an appropriate evaluation metric for this dataset. Unlike the other datasets, this one only has a small skew in the class distribution and there are more positive examples than negative examples. A more appropriate scoring function for this dataset might be the area under the ROC curve. We ran SAYU using this metric and again found no difference between the integrated approach and the two-step method. We believe an essential piece of future work is to run a simulation study to try better discern the conditions under which the SAYU algorithm provides an advantage over the two-step approach.

As we had discussed before, implementing SAYU is costly, as we now need to build a new propositional classifier when evaluating each rule. Moreover, the differences in scoring methods may lead to learning very different sets of clauses. Tables 1 through 3 display several statistics for the first three datasets. We have omitted the statistics for the Advised By dataset in interest of space. First, we look at the average number of clauses in a theory in the two-step approach, and compare it with SAYU-NB and SAYU-TAN. Second, we compare average clause length, measured by the number of literals per clause body. Finally, we show the average number of clauses scored in each fold. Table 1 shows that SAYU's theories contain far fewer clauses than the two-step algorithm in Mammography. Moreover, if finds shorter clauses, some even with a single attribute. The two columns are very similar for SAYU-NB and SAYU-TAN. The last column shows that the cost of using SAYU is very high on this dataset: we only generate a tenth of the number of clauses when using Naïve Bayes. Results for SAYU-TAN are even worse as it only generates 3% as many clauses as the original Aleph run. Even so, the SAYU-based algorithms perform better.

The Yeast dataset (Table 2) tells a similar story. Again, the SAYU-based approaches require fewer clauses to obtain a better result. Again, SAYU generates smaller clauses with the average clause length lower than for Mammography. The cost of implementing SAYU was less in this case. We believe this is because of the cost of transporting the bitmaps (representing the new feature) through the Java-Prolog interface is smaller, since the dataset is not as large. Finally, Carcinogenesis (Table 3) again shows SAYU-based approaches learning smaller theories with shorter clauses, and paying a heavy price for interfacing with the propositional learning. Carcinogenesis is the smallest benchmark, so its cost is smaller than Mammography or Yeast Protein.

In all datasets, the theory found by SAYU consists of significantly fewer and shorter clauses. Even with the simpler classifier, SAYU does at least as well as the two-step approach. Furthermore, SAYU achieves these benefits despite evaluating significantly fewer rules than Aleph.

Subsequent to these experiments, we have more recently run a further experiment on the "Advised-By" task of Domingos, used to test learning in Markov

Logic Networks (MLN) [9]. The task is to predict students' advisors from web pages. Using the same folds for 5-fold cross-validation used in [9], SAYU with either TAN or Naïve Bayes achieves higher area under the PR curve than MLN; specifically, SAYU-TAN achieves 0.414, SAYU-NB achieves 0.394, and MLN achieves 0.295 (taken from [9]). We do not know whether the comparison with MLN is significant, because we do not have the per-fold numbers of MLN. SAYU-TAN, SAYU-NB, Aleph-TAN and Aleph-NB all achieve roughly the same areas, and the differences among them are not significant.

All our results show no significant benefit from using SAYU-TAN over SAYU-NB. We believe there are two reasons for that. First, the SAYU algorithm itself might be searching for independent attributes for the classifier, especially when we are using SAYU-NB. Second, Naïve Bayes is computationally more efficient, as the network topology is fixed. In fact, only the conditional probability table corresponding to the newly introduced rule must be built in order to evaluate the new rule. Thus, SAYU-NB benefits from considering more rules.

5 Related Work

The present work builds upon previous work on using ILP for feature construction. Such work treats ILP-constructed rules as Boolean features, re-represents each example as a feature vector, and then uses a feature-vector learner to produce a final classifier. To our knowledge, Pompe and Kononenko [14] were the first to apply Naïve Bayes to combine clauses. Other work in this category was by Srinivasan and King [18], who use rules as extra features for the task of predicting biological activities of molecules from their atom-and-bond structures. More generally, research on propositionalization of First Order Logic [1] is similar in that it converts the training sets to propositions and then applies feature vector techniques in the learning phase.

There also has been significant research on alternatives to the standard decision list approach. One can use formalisms such as relational trees [13] to change the structure of rules themselves. A popular alternative to decision lists is *voting*. Voting has been used in ensemble-based approaches, such as bagging [5] and boosting [8,16]. Boosting relies on the insight that one should focus on misclassified examples. Search is directed by having a sequence of steps, such that at each consecutive step misclassified examples become more and more valuable. We do not change example weights at each step. Instead, we rely on the classifier itself and trust the tuning data to give us approximate performance of the global system. On the other hand, we do try to focus search on examples where we perform worse, by skewing seed selection.

ROCCER is a more recent example of a two-step algorithm that starts from a set of rules and tries to maximize classifier performance [15]. ROCCER takes a set of rules, and returns a subset that corresponds to a convex hull in ROC space. ROCCER relies on the Apriori algorithm to obtain the set of rules.

To our knowledge, the first work to replace a two-step approach with a tight coupling between rule learning and rule usage is the work appearing earlier this year (done in parallel with ours) by Landwehr, Kersting and De Raedt [10]. That work presented a new system called nFOIL. The significant differences in the two pieces of work appear to be the following. First, nFOIL scores clauses by conditional log likelihood rather than improvement in classifier accuracy or classifier AUC (area under ROC or PR curve). Second, nFOIL can handle multiple-class classification tasks, which SAYU cannot. Third, the present paper reports experiments on data sets with significant class skew, to which probabilistic classifiers are often sensitive. Finally, both papers cite work last year showing that TAN outperformed Naïve Bayes for rule combination [4]; the present paper shows that once clauses are scored as they are actually used, the advantage of TAN seems to disappear. More specifically, TAN no longer significantly outperforms Naïve Bayes. Hence the present paper may be seen as providing some justification for the decision of Landwehr et al. to focus on Naïve Bayes.

6 Conclusions and Future Work

Prior work has shown that combining ILP-induced rules by a learned Bayesian network can improve classification performance over an ordinary union of the rules [3,4]. Nevertheless, in that earlier work, rules were scored using a standard ILP scoring function (compression), and the Bayesian network was constructed afterward. The present paper proposes an approach that integrates rule learning and Bayesian network learning. Each candidate rule is temporarily added to the current set of rules, and a Bayesian network is learned over these rules. The score of the rule is the improvement in performance of the new Bayesian network over the previous best network. Performance is measured as area under the precision-recall curve, omitting recalls between 0 and 0.2. (Precision-recall curves have high variation in that region.)

This paper shows that the new integrated approach results in significantly improved performance over the prior, two-step approach on two of three datasets, and no significant change on a third dataset. In addition, on all three datasets, the integrated approach results in a simpler classifier—and hence potentially improved comprehensibility—as measured by the average number and length of learned clauses.

Acknowledgments. Support for this research was partially provided by U.S. Air Force grant F30602-01-2-0571. Inês Dutra and Vítor Santos Costa were visiting UW-Madison. Vítor Santos Costa was partially supported by the Fundação para a Ciência e Tecnologia. We would like to thank Mark Goadrich and Rich Maclin for reading over drafts of this paper.

References

1. E. Alphonse and C. Rouveirol. Lazy propositionalisation for relational learning. In Horn W., editor, *ECAI'00, Berlin, Allemagne*, pages 256–260. IOS Press, 2000.
2. J. Davis, E. Burnside, I. C. Dutra, D. Page, R. Ramakrishnan, V. Santos Costa, and J. Shavlik. View learning for statistical relational learning: With an application to mammography. In *IJCAI05*, Edinburgh, Scotland, 2005.

3. J. Davis, I. C. Dutra, D. Page, and V. Santos Costa. Establishing Entity Equivalence in Multi-Relation Domains. In *International Conference on Intelligence Analysis*, Vienna, Va, May 2005.
4. J. Davis, V. Santos Costa, I. M. Ong, D. Page, and I. C. Dutra. Using Bayesian Classifiers to Combine Rules. In *3rd MRDM*, Seattle, USA, August 2004.
5. I. C. Dutra, D. Page, and J. Shavlik V. Santos Costa. An empirical evaluation of bagging in inductive logic programming. pages 48–65, September 2002.
6. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian networks classifiers. *Machine Learning*, 29:131–163, 1997.
7. M. Goadrich, L. Oliphant, and J. Shavlik. Learning Ensembles of First-Order Clauses for Recall-Precision Curves: A Case Study in Biomedical Information Extraction. In *Proceedings of the 14th ILP*, Porto, Portugal, 2004.
8. S. Hoche and S. Wrobel. Relational learning using constrained confidence-rated boosting. In *ILP01*, volume 2157, pages 51–64, September 2001.
9. S. Kok and P. Domingos. Learning the structure of Markov Logic Networks. In *National Conference on Artificial Intelligene (AAAI)*, 2005.
10. N. Landwehr, K. Kersting, and L. De Raedt. nFOIL: Integrating Naive Bayes and FOIL. In *National Conference on Artificial Intelligene (AAAI)*, 2005.
11. H. W. Mewes, D. Frishman, C. Gruber, B. Geier, D. Haase, A. Kaps, K. Lemcke, G. Mannhaupt, F. Pfeiffer, C. Schüller, S. Stocker, and B. Weil. Mips: a database for genomes and protein sequences. *Nucleic Acids Research*, 28(1):37–40, Jan 2000.
12. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
13. J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *KDD '03*, pages 625–630. ACM Press, 2003.
14. U. Pompe and I. Kononenko. Naive Bayesian classifier within ILP-R. In L. De Raedt, editor, *ILP95*, pages 417–436, 1995.
15. R. Prati and P. Flach. Roccer: an algorithm for rule learning based on roc analysis. In *IJCAI05*, Edinburgh, Scotland, 2005.
16. J. R. Quinlan. Boosting first-order learning. *Algorithmic Learning Theory, 7th International Workshop, Lecture Notes in Computer Science*, 1160:143–155, 1996.
17. M. Richardson and P. Domingos. Markov logic networks, 2004.
18. A. Srinivasan and R. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. In *ILP97*, pages 89–104, 1997.
19. Ashwin Srinivasan. *The Aleph Manual*, 2001.