TREE^2 - Decision Trees for Tree Structured Data

Björn Bringmann and Albrecht Zimmermann

Institute of Computer Science, Machine Learning Lab,
Albert-Ludwigs-University Freiburg, Georges-Köhler-Allee 79,
79110 Freiburg, Germany
{bbringma, azimmerm}@informatik.uni-freiburg.de

Abstract. We present TREE^2, a new approach to *structural classification*. This integrated approach induces decision trees that test for pattern occurrence in the inner nodes. It combines state-of-the-art tree mining with sophisticated pruning techniques to find the most discriminative pattern in each node. In contrast to existing methods, TREE^2 uses no heuristics and only a single, statistically well founded parameter has to be chosen by the user. The experiments show that TREE^2 classifiers achieve good accuracies while the induced models are smaller than those of existing approaches, facilitating better comprehensibility.

1 Introduction

Classification is one of the most important data mining tasks. Whereas traditional approaches have focused on flat representations, using feature vectors or attribute-value representations, there has recently been a lot of interest in more expressive representations, such as sequences, trees and graphs [1,2,3,4,5]. Motivations for this interest include drug design, since molecules can be represented as graphs or sequences. Classification of such data paves the way towards drug design on the screen instead of extensive experiments in the lab. Regarding documents, XML, essentially a tree-structured representation, is becoming ever more popular. Classification in this context allows for more efficient dealing with huge amounts of electronic documents.

Existing approaches to classifying structured data (such as trees and graphs) can be categorized into various categories. They differ largely in the way they derive structural features for discriminating between examples belonging to the different classes.

A first category can be described as a pure *propositionalization approach*. The propositionalization approach typically generates a very large number of features and uses an attribute-value learner to build a classifier. The resulting classifiers are often hard to understand due to the large number of features used which are possibly also combined in a non-trivial way (e.g. in a SVM).

A second class of systems can be described as the *association rule approach*, e.g. Zaki [4]. Even though the resulting rules often yield high predictive accuracy, the number of generated rules typically explodes, making the resulting classifier difficult to understand.

A. Jorge et al. (Eds.): PKDD 2005, LNAI 3721, pp. 46–58, 2005.

Both the association rule and propositionalization approaches consider feature generation and classification in two independent steps. *Integrated approaches* form a third category of systems that integrates feature construction with classification. This category includes inductive logic programming systems, such as FOIL [6] and PROGOL [7], as well as the DT-GBI approach of Motoda *et al.* [5]. For those approaches to be computationally feasible they have to perform heuristic search, possibly generating non-optimal features.

All techniques mentioned above share the need to specify a number of user-defined parameters, which is often non-trivial.

In this work we present a different approach called TREE^2. It is motivated by recent results on finding correlated patterns, allowing to find the k best features according to a convex optimization criterion such as χ^2 or *Information Gain* [8]. Rather than generating a large number of features or searching for good features in a heuristic manner, TREE^2 searches for the best features to be incorporated in a decision tree by employing a branch-and-bound search, pruning w.r.t. the best pattern seen so far. As in DT-GBI, a decision tree is induced but at each node, the *single* best feature is computed. There are several advantages: TREE^2 is an *integrated approach*, has stronger guarantees than GBI, only one parameter has to be set (the significance level), and the resulting classifiers are far smaller and easier to understand than those of the propositionalization and association rule approaches.

The paper is organized as follows: in Section 2 we describe earlier work on the topic and relate it to our approach; in Section 3, we discuss technical aspects of our method and outline our algorithm; in Section 4, the experimental evaluation is explained and its results discussed. We conclude in Section 5 and point to future work directions.

2 Related Work

Structural classification has been done with different techniques. Firstly, there are several propositionalization approaches, e.g. [2] and [3]. While details may differ, the basic mechanism in these approaches is to first mine all patterns that are unexpected according to some measure (typically frequency). Once those patterns have been found, instances are transformed into bitstrings, denoting occurrence of each pattern. Classifiers are trained using this bitstring representation. While these approaches can show excellent performance and have access to the whole spectrum of machine learning techniques there are possible problems. Obviously the decision which patterns to consider special, e.g. by fixing a minimum frequency, will have an effect on the quality of the model. The resulting feature set will probably be very large, forcing pruning of some kind. Finally, interpretation of the resulting model is not easy, especially if the classifier is non-symbolic, e.g. a SVM.

A second group of approaches is similar to the *associative classification* approach [9]. Again, outstanding patterns are mined but each of them has to associate with the class value. Zaki *et al.*'s XRULES classifier is of this variety. Each

pattern is then considered as a rule predicting its class. Usually, the resulting rule set has to be post-processed and/or a conflict resolution technique employed. As in the propositionalization techniques, the choice of constraints under which to mine is not straight-forward and choosing the resolution technique can strongly influence performance, as has been shown e.g. in [10,11]. Additionally, the resulting classifier often consists of thousands of rules, making interpretation by the user again difficult.

Finally, there exist integrated techniques that do not mine *all* patterns, but construct features during building the classifier. Since structural data can be represented in predicate logic, techniques such as FOIL [6] and PROGOL [7] are capable of doing that. While ILP approaches are elegant and powerful, working on large datasets can be too computationally expensive. An approach such as DT-GBI [5], on the other hand, constructs the features it uses for the tests of the induced decision tree by doing graph-mining. What is common to these approaches is that feature induction is usually done in a heuristic way, often by greedy maximization of a correlation measure during beam search. Responsibility of deciding the parameters governing this search is placed upon the user. For instance, in FOIL decisions have to be made on the beam size and the maximum number of literals that are allowed in the rule body. Similarly, DT-GBI requires the user to specify beam size, the maximum number of specializations in each node, and possibly a minimum frequency that should not be violated. As Motoda shows in his work [5], finding the right value for the beam size and the maximum number of specializations requires essentially a meta-search in the space of possible classifiers.

In contrast, the only parameter to be specified for TREE^2 is the cut-off value for growing the decision tree. By basing this value on the p-values for the χ^2-distribution, the user has a well-founded guide-line for choosing this value.

While all the above techniques focus on directly using structural information for classification purposes, a different approach is exemplified by [12]. Instead of explicitly representing the structures used, kernels are employed that quantify similarities between entities. While the resulting classifiers are very accurate, the use of e.g. a graph kernel together with an SVM make analyzing the model difficult.

3 Methodology

In this section we explain the pattern matching notion used by the TREE^2 approach, discuss upper bound calculation, the main component of the principled search for the most discriminating pattern, and formulate the algorithm itself.

3.1 Matching Embedded Trees

Several representations for structured data such as graphs, trees and sequences exist. In this paper we will focus on tree structured data, like XML, only. Thus, we need a notion for matching tree structured data.

A rooted k-tree t is a set of k nodes V_t where each $v \in V_t$, except one called root, has a parent denoted $\pi(v) \in V_t$. We use $\lambda(v)$ to denote the label of a node and an operator $\prec$ to denote the order from left to right among the children of a node. The transitive closure of π will be denoted π^*. Let $\mathcal{L}$ be a formal language composed of all labeled, ordered, rooted trees and $\mathcal{D} \subset \mathcal{L}$ a database. To count trees $t \in \mathcal{D}$ containing a pattern p we define a function $d_t : \mathcal{L} \rightarrow \{0, 1\}$ to be 1 iff p matches the tree t and 0 otherwise.

Several notions of tree matching exist. As in Zaki *et al.*'s work [4] we used a notion called *tree embedding* which is defined as follows:

Definition 1. *A tree t is embedded in a tree t' iff a mapping $\varphi : V_t \rightarrow V_{t'}$ exists such that* $\forall u, v \in V_t : \lambda(u) = \lambda(\varphi(u)) \wedge u \prec v \Leftrightarrow \varphi(u) \prec \varphi(v) \wedge \pi^*(u) = v \Leftrightarrow \pi^*(\varphi(u)) = \varphi(v)$.

An example of an embedded tree is given in Figure 1.

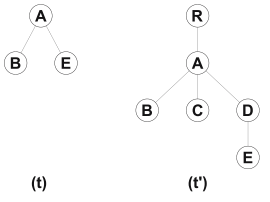


Fig. 1. The tree t is embedded in t'

	c_1	c_2	
T	y_T	$x_T - y_T$	x_T
$\neg T$	$m - y_T$	$n - m - (x_T - y_T)$	$n - x_T$
	m	$n - m$	n

Fig. 2. A Contingency Table

We use *tree embedding* to compare our approach with Zaki *et al.*'s technique. This notion is more flexible than simple subtrees and the mining process is still efficient. In general, other matching notions (see [1]) and even different representations could be used with our technique . This includes not only other notions of matching trees, but also graphs, sequences etc., since the general principles of our approach apply to all domains.

3.2 Correlation Measures

Popular approaches to finding relevant patterns in the data are based on the support-confidence framework, mining frequent patterns, in the hope of capturing statistically significant phenomena, with high predictive power. This framework has some problems though, namely the difficulty of choosing a "good" support and the fact that confidence tends to reward patterns occurring together with the majority class. To alleviate these problems, we use correlation measures for selecting discriminative patterns. A correlation measure compares the expected frequency of the joint occurrence of a pattern and a certain class value to the observed frequency. If the resulting value is larger than a certain threshold then the deviation from the independence assumption is considered statistically significant enough to assume a relationship between pattern and class label.

Example 1. *Consider as an example a database consisting of 50 instances, half of which are labeled with class label c_1, the other half with class label c_2. Assume furthermore a pattern T which occurs with support 10 in the database. If eight of the ten instances including T are labeled with c_1, then the χ^2 measure would give this deviation a score of* 4.5. Information Gain*, that quantifies only the changes in class distribution w.r.t. T, would give it a score of* 0.079.

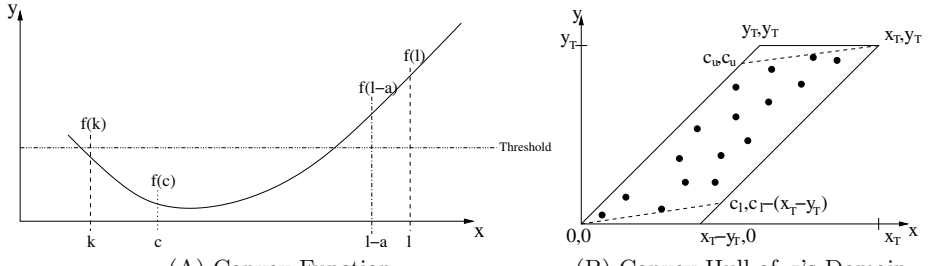
We organize the observed frequencies of a tree pattern T in a contingency table, cf. Figure 2, with x_T denoting the total number of occurences in the dataset and y_T the occurences in the subset corresponding to the first class. Since the two variables are sufficient for calculating the value of a correlation measure on this table, we will view these measures as real-valued functions $\sigma : \mathbb{N}^2 \mapsto \mathbb{R}$ for the remainder of this paper.

While calculating the correlation value of a given pattern is relatively simple, directed search towards better solutions is somewhat more difficult since correlation measures have no desirable properties such as *anti-monotonicity*. But if they are convex it is possible to calculate an upper bound on the score that can be achieved by specializations of the current pattern T and thus to decide whether this branch in the search tree should be followed.

3.3 Convexity and Upper Bounds

It can be proved that χ^2 and *Information Gain* are convex. For the proofs of the convexity of χ^2 and *Information Gain* we refer the reader to [8].

Convex functions take their extreme values at the points forming the convex hull of their domain D. Consider the graph of $f(x)$ in Figure 3(A). Assume the function's domain is restricted to the interval $[k, l]$ which also makes those points the convex hull of D. Obviously, $f(k)$ and $f(l)$ are locally maximal, with $f(l)$ being the global maximum. Given the current value of the function at $f(c)$ and assuming that it is unknown whether c increases or decreases, evaluating f at k and l allows to check whether it is possible for any value of c to put the value of f over the threshold.



(A) Convex Function (B) Convex Hull of σ's Domain

Fig. 3. Convex Function and Convex Hull of the set of possible $\langle x'_T, y'_T \rangle$

For the two-dimensional case, the extreme values are reached at the vertices of the enclosing polygon (in our case the four vertices of the parallelogram in Figure 3(B)). This parallelogram encloses all possible tuples $\langle x'_T, y'_T \rangle$ that correspond to occurence counts of specializations of the current pattern T. The tuple $\langle 0, 0 \rangle$ corresponds to a pattern that does not occur in the dataset and therefore does not have to be considered in calculating the upper bound. $\langle x_T, y_T \rangle$ represents a valid pattern, but in the context of upper bound calculation denotes a specialization of the current pattern T that is equally good in discriminative power. Since general structures have a higher expected probability of being effective on unseen data, we prefer those and thus disregard this tuple as well. Thus the upper bound on $\sigma(T')$ is $ub_\sigma(T) = \max\{\sigma(y_T, y_T), \sigma(x_T - y_T, 0)\}$. For an in-depth discussion of upper bound calculation we refer the reader to [8,11].

Example 2. *Continuing our example from 3.2, this means that for σ being χ^2, $ub_{\chi^2}(T) = \max\{9.52, 2.08\}$, given $x = 10$, $y = 8$. Since 9.52 is larger than $\chi^2(x_T, y_T) = 4.5$ there might be a specialization of T that discriminates better than T itself and therefore exploring this search path is worthwhile.*

While this upper bound calculation is correct for *Information Gain*, an additional problem w.r.t. χ^2 lies in the fact that the information provided by the score of χ^2 is not always reliable. Statistical theory says that for a contingency table with one degree of freedom, such as the one we are considering here, the expected number of occurrences has to be greater than or equal to 5 for the χ^2 score to be reliable. This means that a χ^2-value on $\langle y_T, y_T \rangle$ or $\langle x_T - y_T, 0 \rangle$ is not necessarily reliable. Thus, upper bound calculation has to be modified to achieve reliability. Based on the size of the class and of $\mathcal{D}$, upper and lower bounds c_u, c_l on x'_T for which all four cells have an expected count of 5 can be calculated and the values of the tuples adjusted accordingly. Two of the new vertices are shown as $\langle c_u, c_u \rangle$ and $\langle c_l, c_l - (x_T - y_T) \rangle$.

3.4 The TREE^2 Algorithm

The TREE^2 algorithm (shown as Algorithm 1) constructs a binary decision tree in the manner of ID3 [13]. In the root node and each inner node, the occurrence of a tree pattern is tested against the instance to be classified. A resulting tree could look like the example given in Figure 4. In each node, the subtree having the best discriminative effect on the corresponding subset is found by a systematic branch-and-bound search. The mining process is shown in the subroutine ENUMERATEBESTSUBTREE. The space of possible patterns is traversed using canonical enumeration and the value of σ calculated for each candidate pattern. If this value lies above the best score seen so far, the current pattern is the most discriminating on this subset so far and the threshold is raised to its σ-value. An upper bound on the value specializations of the current pattern can achieve is calculated and pruning of the search space using this upper bound and the threshold is performed. In this way, we separate the success of the technique from user decisions about the search strategy. The only decision a user has to

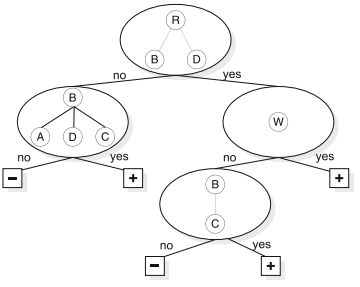


Fig. 4. A decision tree as produced by the TREE^2 algorithm

make is the one w.r.t. a stopping criterion for further growth of the tree. To this effect, a minimum value for the score of the correlation measure has to be specified, which can be based on statistical theory, thus giving the user a better guidance for making this decision.

Algorithm 1 The TREE^2 algorithm

TREE$^2(\mathcal{D}, \sigma, \tau_{user}$, DT)
1: p_{split} = ENUMERATEBESTSUBTREE$(\top, 0, \sigma, \tau_{user}, \varnothing)$
2: **if** $p_{split} \neq \varnothing$ **then**
3: Add node including p_{split} to the DT
4: TREE$^2(\ \{T \in \mathcal{D} | p_{split}$ embedded in $T\}\ , \sigma, \tau_{user}$, DT)
5: TREE$^2(\ \{T \in \mathcal{D} | p_{split}$ not embedded in $T\}\ , \sigma, \tau_{user}$, DT)
6: **return** DT

ENUMERATEBESTSUBTREE$(t, \tau, \sigma, \tau_{user}, p)$
1: **for all** canonical expansions t' of t **do**
2: **if** $\sigma(t') > \tau \ \wedge \ \sigma(t') \geq \tau_{user}$ **then**
3: $p = t', \ \tau = \sigma(t')$
4: **if** $ub_\sigma(t') \geq \tau$ **then**
5: $p =$ ENUMERATEBESTSUBTREE$(t', \tau, \sigma, \tau_{user}, p)$
6: **return** p

TREE^2 has several desirable properties. Firstly, the resulting classifier is integrated in the sense that it uses patterns directly, thus circumventing the need for the user to restrict the amount of features and making the resulting classifier more understandable. Secondly, by using correlation measures for quantifying the quality of patterns, we give the user a sounder theoretical foundation on which to base the decision about which learned tests to consider significant and include in the model. Thirdly, we avoid using heuristics that force the user to decide on the values of parameters that could have a severe impact on the resulting model's accuracy. Using principled search guarantees that TREE^2 finds the best discriminating pattern for each node in the decision tree w.r.t. the correlation measure used. Finally, as the experiments show, the resulting decision tree is far smaller than the rule sets produced by XRULES classifier [4], while achieving comparable accuracy, and is therefore more easily interpretable by human users.

4 Experimental Evaluation

For the experimental evaluation, we compared our approach to XRULES and a decision tree base-line approach on the XML data used in Zaki *et al.*'s publication [4]. Furthermore, we compared TREE^2 to a base-line approach using frequency mining for a SVM classifier and two PROGOL results on the regression-friendly subset of the Mutagenesis dataset.

XML Data. The XML data used in our experiments are log files from web-site visitors' sessions. They are separated into three weeks (CSLOG1, CSLOG2, and CSLOG3) and each session is classified as its producing visitor coming either from an `.edu` domain or from any other domain. Characteristics of the datasets are shown in Table 1. For the comparison we built decision trees with the

Table 1. Characteristics of Datasets (taken from [4])

DB	#Sessions	edu	other	%edu	%other
CSLOG1	8074	1962	6112	24.3	75.7
CSLOG2	7409	1687	5722	22.8	77.2
CSLOG12	13934	2969	10965	21.3	78.7
CSLOG3	7628	1798	5830	23.6	76.4

χ^2 distribution's significance value for 90%, 95% and 99% respectively. In each setting we used one set of data for training and another one for testing. Following Zaki's notation, CSLOGx-y means that we trained on set x and tested on set y. For the base-line approach we mined the 100 patterns having the highest discriminative effect on the data, transformed the data into bitstring instances according to the found patterns, and built decision trees using all 100 patterns in one run (*C4.5 - 100*) and the 50 best patterns in another run (*C4.5 - 50*) with the WEKA [14] implementation of the C4.5 [15] algorithm. We compare the accuracies of the resulting classifiers against each other as well as the complexity of the model which we measure by the number of rules used by XRULES, and by the number of leaves in the decision trees, which corresponds to the number of rules that can be derived from the trees, respectively.

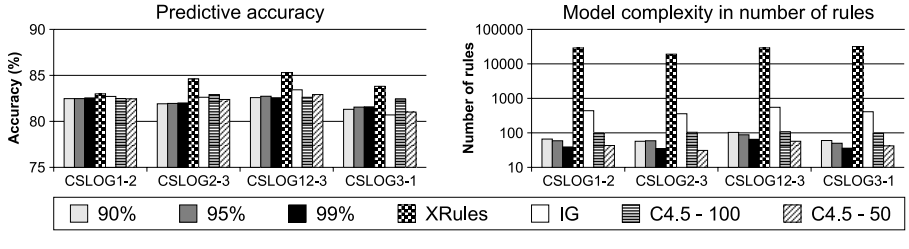


Fig. 5. Accuracies and size in rules of the different approaches

Results are summarized in Figure 5. As can be seen, the accuracies of the induced classifiers do not vary much. The only approach that significantly outperforms (by 2-3%) the other techniques on all but the CSLOG1-2 setting, is XRules. At the same time, the size of XRules' models is also significantly greater. While the Tree^2 trees induced with *Information Gain* have several hundred nodes and all trees induced with χ^2 (both Tree^2 and base-line) between 35 and 103 nodes, the smallest XRules model consists of more than 19000 rules. Patterns tested against in the inner decision tree nodes consist of 3-7 nodes only. Since this is similar to the size of patterns used in XRules' rules, complexity is really reduced and not just pushed inside the classifier. In comparing the other approaches, several things are noticeable. Raising the threshold from the 90% to the 95% significance level for χ^2-induced Tree^2 trees does not decrease accuracy (even improving it slightly in 3 cases). Raising it further to the 99% level has no clear effect. The tree size decreases, though, on average by 7.5 nodes from the 90% to the 95% setting. Raising the significance level further to 99% decreases the tree size by 18 nodes on average.

For the base-line approach we mined patterns correlating strongly with the classes and trained a classifier on them. This approach achieves competitive results w.r.t the accuracy. The clear drawback is that deciding on the number of features to use is not straightforward. Using only 50 instead of 100 features produces all kinds of behavior. In some cases the accuracy does not change. In other cases the classifier using 50 features outperforms the one using 100 or vice versa. Also, the base-line approach using 100 patterns tends to use most of these, even if Tree^2 trees of similar quality are much smaller.

Finally, using *Information Gain* as quality criterion shows mainly one thing - that it is difficult to make an informed decision on cut-off values. The accuracies and sizes shown refer to decision trees induced with a cut-off value of 0.001. For one thing, the resulting trees grow far bigger than the χ^2-trees. Additionally, the accuracies in comparison with the χ^2 approach vary, giving rise to one worse tree, one of equal quality and two better ones. None of the differences in accuracy is significant though. Inducing decision trees with a cut-off value of 0.01 lowers accuracy by 1.5 to 3 percentage points, with the induced trees still being larger than the χ^2 trees.

Mutagenicity Data. For this setting, we chose the regression-friendly subset of the well known Mutagenicity dataset used in [16]. We compare with the results of the ILP system Progol reported in [16,17] and the results of the base-line approach reported in [3]. Since the Mutagenicity dataset consists of molecules represented as graphs, a transformation from the SMILES representation into so-called fragment-trees is used that is explained following this paragraph.

The Smiles Encoding. The SMILES language [18] is used by computational chemists as a compact encoding of molecular structure. It is supported by many tools as OpenBabel or Daylight ([19,20]). The language contains symbols for atoms, bonds, branches, and can express cycles. Using a decomposition-algorithm by Karwath and De Raedt [21], a SMILES-String can, after some

reformatting, be decomposed into a so-called *fragment tree*. Since there is no *unique* SMILES-string for a molecule, the fragment tree is not unique either. The decomposition-algorithm recursively splits the string into *cycles* $\{_xT\}_x$ and *branches* $A(B)C$. In the resulting fragment-tree the leaves contain pure cycles or linear fragments without further branches. The inner nodes of such a tree contain fragments still containing branches while the root node is the whole molecule. The edge labels denote the type of decomposition (i.e. the part of the branch or the number of the cycle). Thus, the leaves of a fragment-tree contain a lot of information decomposed into very small fragments. As in [3] we drop the edge labels and labeled all but the leaf nodes with a new, unique label. Hence, the tree-structure represents the abstract structure of the molecule with the chemical information in the leaves.

Figure 6 shows a molecule on the left-hand side which could be encoded by the SMILES-string $N-c1ccc(cc1)-O-c2ccc(cc2)-[Cl]$. This string represents the same as $N\{_0cccc(cc\}_0)O\{_1cccc(cc\}_1)[Cl]$. The corresponding fragment-tree is shown on the right-hand side of Figure 6.

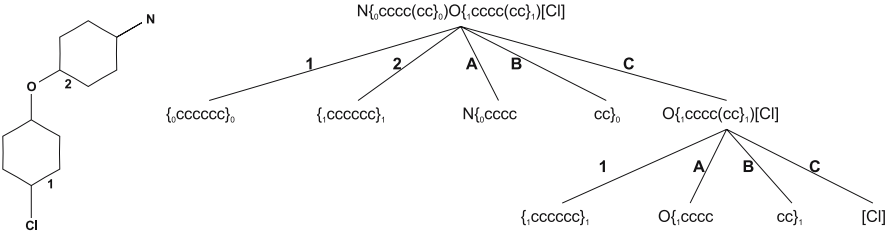


Fig. 6. A molecule with the encoding $N-c1ccc(cc1)-O-c2ccc(cc2)-[Cl]$ and the corresponding fragment-tree

Experimental Results. Predictive accuracy for each approach was estimated using ten-fold cross-validation. Reported are average accuracies and standard deviation (if known). For TREE^2, trees were induced at the 95% significance level for χ^2 and with a cut-off value of 0.01 for *Information Gain*. The results reported in [16] were achieved using PROGOL and working only on structural information, in [17], numerical values suggested by experts were used as well. This work reports only an average accuracy. The resulting accuracies and the size of the corresponding theories are shown in Table 2.

As can be seen, for both measures TREE^2 gives similar results to the purely structural PROGOL approach, with the differences being not significant. At the same time, the χ^2 induced model is far smaller than the other two. Again, the patterns tested against in the inner nodes are not overly complex (5-11 nodes). When PROGOL uses the expert-identified attributes as well, its accuracy increases. Since we do not have access to the standard deviation of these experiments, we cannot make a significance statement. Finally, the base-line approach,

Table 2. Accuracies and complexity of the models on the mutagenicity dataset

Approach	Predictive Accuracy	Average Size of the Model
TREE^2 χ^2	80.26±7.14	2.3 Nodes
TREE^2 *IG*	81.76±9	11.8 Nodes
PROGOL '94 [16]	80±3	9 Clauses
PROGOL '95 [17]	84	4 Clauses
FREQUENT SMILES [3]	86.70	214 Patterns

which mined all patterns frequent in one class and not exceeding a given frequency in the other class, and built a model using these features in an SVM, significantly outperforms the TREE^2 classifiers. On the other hand, by using a SVM, the results will hardly be interpretable for humans anymore and the amount of patterns used is larger than in the TREE^2 models by two orders of magnitude.

5 Conclusion and Future Work

We presented TREE^2, an integrated approach to structural classification. The algorithm builds a decision tree for tree structured data that tests for pattern occurrence in the inner nodes. Using an optimal branch-and-bound search, made possible by effective pruning, TREE^2 finds the most discriminative pattern for each subset of the data considered. This allows the user to abstract the success of the classifier from decisions about the search process, unlike in existing approaches that include heuristics. Basing the stopping criterion for growing the decision tree on statistically well founded measures rather than arbitrary thresholds whose meaning is somewhat ambiguous gives the user better guidance for selecting this parameter. It also alleviates the main problem of the support-confidence framework, namely the generation of very large rule sets that are incomprehensible to the user and possibly include uninformative rules w.r.t. classification.

As the experiments show, TREE^2 classifiers are effective while being less complex than existing approaches. While using χ^2 for assessing the quality of discriminative patterns, raising or lowering the significance threshold affects the induced trees in an expected manner. In contrast, using *Information Gain* is more difficult, since selecting the cut-off value has no statistical foundations. While base-line approaches, that separate feature generation and classifier construction, achieve very good results, it is not entirely clear how to justify the selected the number of features mined. Furthermore, there exists a gap in interpretability since the classifier used might combine the mined features in a way that is not easily accessible to the user.

So far, we have restricted ourselves to a single representation, *trees*, a certain type of classifier, *decision trees*, and two measures. Future work will include evaluating other correlation measures and applying our approach to different

representations. Finally, the success of using effective conflict resolution strategies in the XRULES classifier suggests the expansion our approach to ensemble classifiers.

Acknowledgments. We would like to thank Mohammed J. Zaki for providing the datasets and the XRULES algorithm. Furthermore, we would like to thank Andreas Karwath, Kristian Kersting, Robert Egginton and Luc De Raedt for interesting discussions and comments to our work.

References

1. Kilpeläinen, P.: Tree Matching Problems with Applications to Structured Text Databases. PhD thesis, University of Helsinki (1992)
2. Kramer, S., Raedt, L.D., Helma, C.: Molecular feature mining in HIV data. In Provost, F., Srikant, R., eds.: Proc. KDD-01, New York, ACM Press (2001) 136–143
3. Bringmann, B., Karwath, A.: Frequent SMILES. In: Lernen, Wissensentdeckung und Adaptivität, Workshop GI Fachgruppe Maschinelles Lernen, LWA. (2004)
4. Zaki, M.J., Aggarwal, C.C.: XRules: an effective structural classifier for XML data. In Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C., eds.: KDD, Washington, DC, USA, ACM (2003) 316–325
5. Geamsakul, W., Matsuda, T., Yoshida, T., Motoda, H., Washio, T.: Performance evaluation of decision tree graph-based induction. In Grieser, G., Tanaka, Y., Yamamoto, A., eds.: Discovery Science, Sapporo, Japan, Springer (2003) 128–140
6. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning **5** (1990) 239–266
7. Muggleton, S.: Inverse entailment and PROGOL. New Generation Computing **13** (1995) 245–286
8. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: Proceedings of the Nineteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Dallas, Texas, USA, ACM (2000) 226–236
9. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In Agrawal, R., Stolorz, P.E., Piatetsky-Shapiro, G., eds.: KDD, New York City, New York, USA, AAAI Press (1998) 80–86
10. Mutter, S., Hall, M., Frank, F.: Using classification to evaluate the output of confidence-based association rule mining. In Webb, G.I., Yu, X., eds.: Australian Conference on Artificial Intelligence, Cairns, Australia, Springer (2004) 538–549
11. Zimmermann, A., De Raedt, L.: Corclass: Correlated association rule mining for classification. [22] 60–72
12. Gärtner, T., Lloyd, J.W., Flach, P.A.: Kernels and distances for structured data. Machine Learning **57** (2004)
13. Quinlan, J.R.: Induction of decision trees. Machine Learning **1** (1986) 81–106
14. Frank, E., Hall, M., Trigg, L.E., Holmes, G., Witten, I.H.: Data mining in bioinformatics using weka. Bioinformatics **20** (2004) 2479–2481
15. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
16. Srinivasan, A., Muggleton, S., King, R., Sternberg, M.: Mutagenesis: ILP experiments in a non-determinate biological domain. In Wrobel, S., ed.: Proceedings of the 4th International Workshop on Inductive Logic Programming. Volume 237., Gesellschaft für Mathematik und Datenverarbeitung MBH (1994) 217–232

17. King, R.D., Sternberg, M.J.E., Srinivasan, A.: Relating chemical activity to structure: An examination of ILP successes. New Generation Comput. **13** (1995) 411–433
18. Weininger, D.: SMILES, a chemical language and information system 1. Introduction and encoding rules. J. Chem. Inf. Comput. Sci. **28** (1988) 31–36
19. The OpenBabel Software Community: Open Babel. http://openbabel.sourceforge.net/ (2003)
20. Daylight Chemical Information Systems, Inc. http://www.daylight.com/ (2004)
21. Karwath, A., De Raedt, L.: Predictive graph mining. [22] 1–15
22. Suzuki, E., Arikawa, S., eds.: Discovery Science, 7th International Conference, DS 2004, Padova, Italy, October 2-5, 2004, Proceedings. In Suzuki, E., Arikawa, S., eds.: DS 2004, Padova, Italy, Springer (2004)