CTC - Correlating Tree Patterns for Classification

Albrecht Zimmermann Björn Bringmann

Machine Learning Lab, Albert-Ludwigs-University Freiburg,
Georges-Köhler-Allee 79, 79110 Freiburg, Germany
E-mail: {azimmerm,bbringmann}@informatik.uni-freiburg.de

Abstract

We present CTC*, a new approach to* structural classification. *This approach uses the predictive power of tree patterns correlating with the class values. It combines state-of-the-art tree mining with sophisticated pruning techniques to find the k most discriminative pattern in a dataset. In contrast to existing methods,* CTC *uses no heuristics and the only parameters to be chosen by the user are the maximum size of the rule set and a single, statistically well founded cut-off value. The experiments show that* CTC *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 [4, 5, 1, 13, 3]. 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 large amounts of electronic documents.

Existing approaches to classify 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 construction of the used features requires the user-defined some parameters (such as the minimum frequency), which are often difficult to determine a priori, but have a profound effect on the resulting accuracies. Moreover, 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, e.g. the XRULES classifier [13], can be described as the *association rule approach*. Even though the resulting rules often yield high predictive accuracy, the number of generated rules typically explodes, making the resulting classifier difficult to understand. Moreover, as in the propositionalization approach, the user has to specify a number of parameters, which is often non-trivial.

Both the association rule and propositionalization approaches consider feature generation and classification in two independent steps. Indeed, features are first generated, and then post-processed, possibly using a propositional learner. *Integrated approaches* form a third category of systems that integrates feature construction with classification. This third category includes inductive logic programming systems, such as FOIL [11] and PROGOL [9], as well as the DT-GBI approach of Motoda *et al.* [3]. For those approaches to be computationally feasible they have to perform heuristic search, possibly generating non-optimal features. The ILP and DT-GBI also share the need to specify a number of user-defined parameters.

In this work we present an approach called CTC that is situated between the association rule technique and integrated systems. It is motivated by recent results on finding correlated patterns, allowing to find the k *best*, i.e. most discriminating, features according to a convex maximization criterion such as χ^2 [8]. Rather than generating the complete set of patterns satisfying a given criterion and post-processing them, or searching for good features in a heuristic manner, CTC computes the set of k best patterns by employing a branch-and-bound search. Pruning is performed

w.r.t. the kth best pattern seen so far and/or a user-specified cut-off value. The resulting ordered rule set can be used directly for classification using different rule conflict resolution strategies. There are several advantages over existing techniques: CTC guarantees that the best rules are found, setting it apart from heuristic solutions such as FOIL, PROGOL, and DT-GBI. Only two parameters have to be specified (the maximum size of the rule set and the significance level), decoupling the success of the classifier from decisions about parameters influencing the search process. The resulting classifiers are also far less complex than classifiers built with associative classification approaches such as the XRULES approach while attaining the similarly good accuracies.

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 results are 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. [5] and [1]. 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 meaningful, 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, requiring 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 [7]. Again, unexpected patterns are mined but each of them has to associate with the class value. An example is Zaki *et al.*'s XRULES classifier. Each pattern is 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, 14]. 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 generates features during classifier construction. Since structural data can be represented in predicate logic, techniques such as FOIL [11] and PROGOL [9] can be used for the task of structural classification. While ILP approaches are elegant and powerful, working on large datasets can be too computationally expensive. One way to handle this is to define a language bias that restricts the hypothesis space. Selecting the right language bias is not a straight-forward decision and can have a strong effect on the resulting classifier. Approaches such as DT-GBI [3], on the other hand, construct the features used, e.g. for the tests of the induced decision tree, by graph-mining. These approaches have in common 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 *et al.* show in their work [3], 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, only two parameters have to be specified for CTC. The first one is the maximum rule size, giving the user an intuitive way to decide on the complexity of the resulting model. The second one, the cut-off value below which to not consider rules interesting anymore, is optional. By setting this value the user can enforce the significance of included rules. By basing it on e.g. the p-values for the χ^2-distribution, the user has a well-founded guide-line for choosing this value.

3 Methodology

In this section we explain the pattern matching notion used by the CTC 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 types of structured data exist, such as graphs, trees and sequences. 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

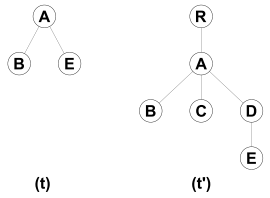

Figure 1. The tree t is embedded in t'.

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. The frequency of p in $\mathcal{D}$ can then be defined as $\sigma_{\mathcal{D}}(p) =_{\text{def}} \Sigma_{t \in \mathcal{D}} d_t(p)$.

Several notions of tree matching exist. As in Zaki *et al.*'s work [13] 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

$$\forall u, v \in V_t : \lambda(u) = \lambda(\varphi(u)) \wedge$$
$$u \prec v \Leftrightarrow \varphi(u) \prec \varphi(v) \wedge$$
$$v \in \pi^*(u) \Leftrightarrow \varphi(v) \in \pi^*(\varphi(u)).$$

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

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 [4]) and even different representations could be used with CTC. 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

CTC aims at finding a compact set of tree patterns whose occurrence in a tree allows to reliably predict the tree's class. One approach to do this is the one chosen in XRULES, namely to mine all tree patterns that occur frequently in the trees belonging to a given class and have a certain minimum strength (e.g. confidence) when evaluated on the entire dataset.

There are a few problems with this framework though. The first one lies in the fact that it is difficult to choose a "good" minimum support. If the threshold is set too high, valuable patterns will probably be missed by the mining process. If it is set too low, the resulting pattern set grows too large and probably includes many uninformative patterns that have to be removed in a post-processing step.

	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

Table 1. A Contingency Table

Also, if confidence is used to assess the quality of rules as in the XRULES classifier, it 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 occurence 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 causal relationship between the pattern and the class label.

Example 1 *Consider 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 entropy w.r.t. T, would give it a score of* 0.079.

We organize the observed frequencies in a contingency table, cf. Table 1. Since the two variables x_T and y_T are sufficient for calculating the value of a correlation measure on this table, we will view these measures as real-valued functions whose domain is $\mathbb{N}^2$ for the remainder of this paper. While we restrict ourselves to binary class problems in this work, a multi-class problem can easily be transformed into a set of two-class problems, making our technique applicable to these settings.

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

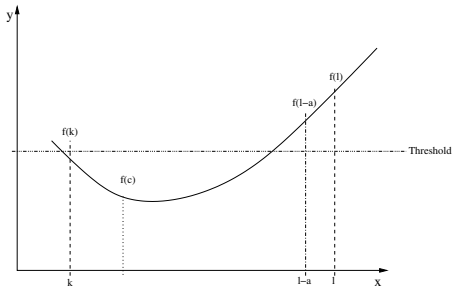

Figure 2. A Convex Function

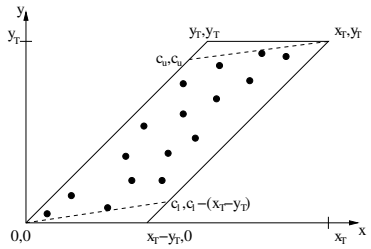

Figure 3. Convex Hull of σ's Domain

graph of $f(x)$ in Figure 2. 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.

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). 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. The tuple $\langle x_T, y_T\rangle$ corresponds to 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. Following the *Occam's Razor* argument, that 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, 14]

Example 2 *Continuing our example from the previous page, 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, e.g. *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 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 CTC algorithm

The CTC algorithm (shown as Algorithm 1) constructs an ordered list of at most k rules, all of which have a significance value equal to or above the user-specified cut-off value τ_{user}. During mining, only patterns that have a chance of exceeding both τ_{user} and the kth-best significance score seen so far are specialized. Since correlation measures are neither monotone nor anti-monotone, CTC calculates an upper bound on the value specializations of a pattern can achieve and prunes the search space using this upper bound and the kth-best correlation score calculated so far. In this way, we separate the success of the technique from user decisions about the search strategy. The only decisions that a user has to make are the ones w.r.t. the maximal size of the rule list and the cut-off value for inclusion in the rule list. 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.

Starting from the most general (empty) tree pattern, we canonically enumerate all tree patterns. The way this canonical expansion is performed has two benefits: Firstly, that each pattern is enumerated only once and secondly, that only patterns are enumerated that have a chance of exceeding the current threshold. For inclusion in the solution set S, the score of a pattern has to exceed the threshold τ. At the start of the algorithm this is the user-supplied cut-off value but as S is populated, the threshold is raised to the kth-best score seen so far. The possibility of a pattern exceeding this threshold in the future is assessed by checking the upper bound on future scores ub_σ against τ. Once the k best patterns have been found, each pattern is treated as a rule predicting the more frequent class among the training instances covered by it. In case of a tie, the majority class in the dataset is predicted.

Algorithm 1 The CTC algorithm

σ - correlation measure, τ_{user} - cut-off value, k - maximum size of rule set

1: $S = \emptyset$
2: ENUMERATEK-BESTSUBTREES($S, \top, \tau_{user}, \sigma$)
3: **return** S

ENUMERATEK-BESTSUBTREES(S, t, τ, σ)

1: **for all** canonical expansion t' of t **do**
2: **if** $\sigma(t') \geq \tau$ **then**
3: $S = S \cup \{t'\}$
4: **if** $|S| > k$ **then**
5: $S = S \setminus \arg\min_{s \in S} \sigma(s)$
6: $\tau = \min_{s \in S} \sigma(s)$
7: **if** $ub_\sigma(t') \geq \tau$ **then**
8: ENUMERATEK-BESTSUBTREES(S, t', τ, σ)

3.5 Discussion

CTC has several desirable properties. Firstly, by allowing the user to cap the size of the rule set, users have a intuitive tool for deciding which complexity they still consider useful. Secondly, by using correlation measures for quantifying the quality of patterns, we give the user a more solid theoretical foundation on which to base decisions about which found patterns to consider significant and use in the rule set. Thirdly, we avoid heuristics that force the user to decide on the values of parameters that could have a severe impact on the resulting model's quality. Contrary to heuristic approaches, we can guarantee that really the k *best* rules are found. Finally, by using the upper bounds on correlation measures for pruning during the mining process, we reduce the amount of patterns mined that are not included in the final classifier and avoid an additional post-processing step.

CTC mines a set of rules according to some a priori defined criterion and uses them in a classifier, combining them using some classification strategy. This is a characteristic it shares with associative classification approaches and that sets it apart from integrated approaches that generate features during classifier construction. But unlike associative classifiers is bases the selection of rules not on frequency but on a measure quantifying their discriminative power, similarly to the *Information Gain* or *Foil Gain* measure.

3.6 Classification Strategies

Once the rule set has been computed, the question is how to use it in the actual classification process. Many strategies can be found in the literature of which we describe four here that we used in the experiments.

If the rule set is ordered, as is the case with the rule set induced by CTC, the simplest strategy is to use the first rule matching an instance for classification. The rationale behind is that highly ranked rules have either larger coverage or stronger predictive power than lower ranked ones. The strategy is often referred to as using a *decision list* (DL). A second strategy calls for collecting all rules that match a given instance and combining their predictions. In the least complex version, each rule is given the same weight and for each class the number of rules predicting this class are counted. The class receiving the most votes is predicted. The common name for this strategy is *majority vote* (MV). A possible problem with this approach lies in the fact that rules that are ranked near the bottom are considered equally important to rules near the top, which is counter-intuitive. To alleviate it, discounting of rules becomes necessary. We used two such approaches in our experiments that both come from the field of using associative patterns for classification [6, 13]. One is the *average strength* method (AvgStr), introduced by Zaki *et al.* in [13]. For each class, the strength, e.g. confidence, w.r.t. this class of the rules matching the instance is added up and normalized by dividing by the number of rules. The class with highest average strength is predicted unless no class achieves higher than default strength in which case the majority class is predicted. Finally, the *weighted* χ^2 heuristic (WChi), introduced by Han *et al.* [6] discounts the χ^2 value for each rule against the maximum χ^2 value that rule could have attained. In all cases the majority class is predicted if no rule matches the instance to be classified.

4 Experimental Evaluation

For the experimental evaluation, we compared our approach to XRULES and an integrated approach introduced in earlier work [2] on the XML data used in Zaki *et al.*'s publication [13].

The integrated approach, TREE^2, induces a binary decision tree. In each inner node the occurrence of a discriminating patterns in the data is tested. Pattern mining for the tests is based on the same principles as in the CTC classifier, namely the best (most discriminative) pattern in the subset of data corresponding to the node is found by optimal branch-and-bound search. Tree growth is stopped based on a single cut-off value, similar to the parameter τ_{user} in CTC.

Basing selection of rules on a correlation measure and inclusion on a well-founded cut-off value should lead to the induction of rule sets with high predictive accuracy that are smaller than those produced by XRULES. Since TREE^2 generates features when they are needed, the resulting models will likely be smaller than those of CTC but the use of the larger rule sets and strategies for combining those rules could lead to better performance.

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

Table 2. Characteristics of Datasets (taken from [13])

Setting	CTC	XRULES	TREE^2	CTC$_{Val}$
CSLOG1-2	592	28911	66	130
CSLOG2-3	497	19098	57	150
CSLOG12-3	981	29098	103	170
CSLOG3-1	546	31661	60	220

Table 3. Size of the induced Models for CTC, XRULES, and TREE^2

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 based on whether the visitor came either from an `.edu` domain or from any other domain. Characteristics of the datasets are shown in Table 2. For the mining process we set the maximum size of the rule set to 1000 and the cut-off value to 3.84, the 90%-p value for the χ^2 distribution. For the comparison with TREE^2 we built decision trees with the same cut-off value. As we showed in earlier work ([14, 2]), using a different correlation measure, e.g. *Information Gain*, gives rise to more complex classifiers that do not perform significantly better. In each setting we used one set of data for training and another one for testing. Following Zaki's notation, CSLOGx-y denotes that we trained on set x and tested on set y.

Table 3 shows the complexity of the resulting models. The first column lists the setting for which the corresponding model was induced. The second column reports the number of rules mined by CTC for the parameter setting given above, column three shows the number of rules for XRULES, and the fourth column the number of inner nodes for TREE^2. It is interesting to note that for all four settings the dataset supports less than 1000 rules that pass the 90% significance test. As expected, the size of the CTC's rule set is larger than the tree size of the integrated approach. On the other hand, XRULES produces rule sets that are two orders of magnitude larger than those induced by CTC.

To explore the effect that varying the number of rules used for classification has on predictive accuracy, we evaluated subsets of the rule sets induced by CTC on the whole test sets. For those subsets the l highest-ranked rules induced for a particular setting were selected. The smallest

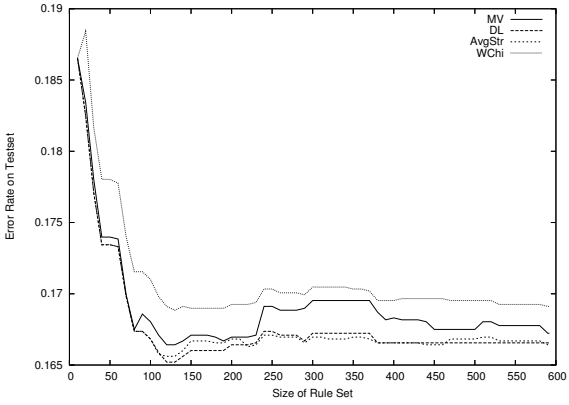

Figure 4. Error rates for different classification strategies for the CSLOG1-2 setting

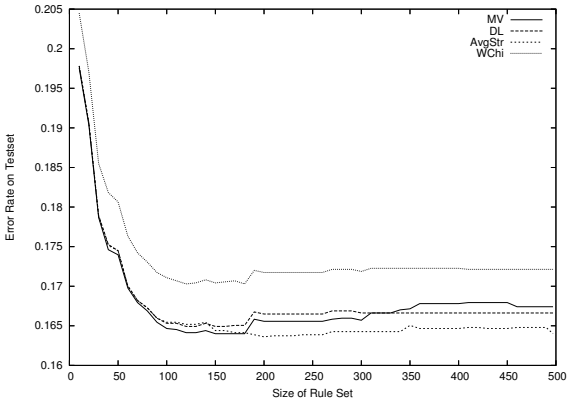

Figure 5. Error rates for different classification strategies for the CSLOG2-3 setting

subset had size 10 and we increased l in increments of 10 up to the total number of rules induced for the respective setting. Figures 4-7 show the resulting error rates for the four different settings. In each diagram, **MV** denotes the majority voting strategy, **DL** the decision list approach, **AvgStr** that Zaki *et al.*'s average strength heuristic was used, and **WChi** the use of the weighted χ^2 heuristic introduced by Han *et al.*. It is noticeable that using less than 100 rules causes relatively high error rates while significantly enlarging the rule set past 200 rules causes error rates to increase again. This means that lower-ranked rules are very likely the result of overfitting.

By using validation sets one can determine the size of the rule set giving best performance. We used half of the corresponding test sets as validation sets. The resulting (best) amount of rules for each setting is reported in the last column of Table 3, denoted by CTC$_{Val}$. By using the respective number of rules for each setting and classifying

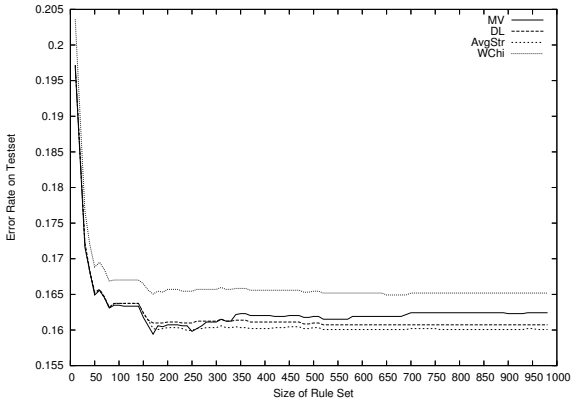

Figure 6. Error rates for different classification strategies for the CSLOG12-3 setting

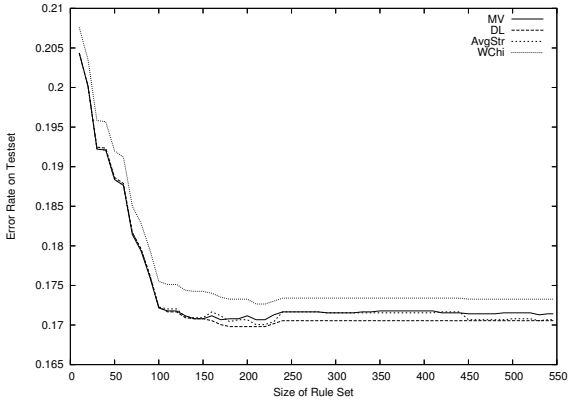

Figure 7. Error rates for different classification strategies for the CSLOG3-1 setting

the other half of the test set we arrive at predictive accuracy estimates that we report in Table 4.

As can be seen, CTC performs well in all settings. For the first setting, CSLOG1-2, the differences between the different classifiers (XRULES, the four variants of CTC, and TREE^2) are not significant at the 5% level. For the settings CSLOG2-3 and CSLOG12-3, XRULES and the first three CTC variants (MV, DL, AvgStr) are significantly better than TREE^2 while the difference between them is not significant. CTC with the weighted χ^2 heuristic performs significantly worse than XRULES and is not significantly better than TREE^2. Finally for the last setting, CSLOG3-1, the rule-based classifiers outperform TREE^2 while there is no significant difference between them.

As these results show, the increase in complexity from TREE^2 to CTC (two to four times as many patterns) is accompanied by an increase in performance while XRULES does not gain an advantage from using more than 100 times as many rules as CTC. This is particularly obvious when comparing XRULES' results to the results of CTC using the average strength strategy, which is the strategy employed in XRULES as well. It is interesting to see that the weighted χ^2 heuristic does not perform as strongly as the other strategies and that the decision list produces results that are very similar to the average strength method. While majority vote performs also well for the rule sets found by using a validation set, Figures 4 and 5 show that the danger of overfitting is more pronounced for this technique.

5 Conclusion and Future Work

In this work, we presented CTC, a rule-based approach to structural classification. Using an optimal branch-and-bound search, the algorithm finds the k most discriminating patterns in a data set and uses them for prediction. This allows the user to separate the success of the classifier from decision about the search process, unlike in approaches that use heuristics. Basing the criterion for inclusion in the rule set 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, CTC classifiers are effective while being less complex than existing rule-based approaches. By having users supply a parameter restricting the maximal size of the induced rule set, we give them the opportunity to build models that they still consider comprehensible. Furthermore, evaluating the subset of the induced rule set consisting of the l highest-ranking rules on a validation set and selecting the l giving the best results offers a

Setting	XRULES	CTC$_{MV}$	CTC$_{DL}$	CTC$_{AvgStr}$	CTC$_{WChi}$	TREE^2
CSLOG1-2	82.99	83.23	83.31	83.31	83.01	82.47
CSLOG2-3	84.61	83.95	83.90	83.92	82.83	81.91
CSLOG12-3	85.30	84.27	84.24	84.29	83.53	82.58
CSLOG3-1	83.81	83.50	83.77	83.63	83.53	81.31

Table 4. Predictive Accuracy for XRULES, different classification strategies for CTC, and the TREE^2 classifier

straight-forward way of tuning the classifier's performance.

So far, we have restricted ourselves to a single representation, *trees*, a single measure, and evaluated four possible classification strategies. Future work will include evaluating other correlation measures and applying our approach to different representations. Finally, selecting the subset of rules to actually use in the classifier is done heuristically so far. To base model construction on optimal search seems to be a promising research direction.

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, and Luc De Raedt for interesting discussions and comments to our work.

References

[1] B. Bringmann and A. Karwath. Frequent SMILES. In *Lernen, Wissensentdeckung und Adaptivität, Workshop GI Fachgruppe Maschinelles Lernen, part of LWA 2004*, 2004.

[2] B. Bringmann and A. Zimmermann. TREE^2 - Decision trees for tree structured data. Submitted to PKDD '05.

[3] W. Geamsakul, T. Matsuda, T. Yoshida, H. Motoda, and T. Washio. Performance evaluation of decision tree graph-based induction. In G. Grieser, Y. Tanaka, and A. Yamamoto, editors, *Discovery Science*, pages 128–140, Sapporo, Japan, Oct. 2003. Springer.

[4] P. Kilpeläinen. *Tree Matching Problems with Applications to Structured Text Databases*. PhD thesis, University of Helsinki, 1992.

[5] S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in HIV data. In F. Provost and R. Srikant, editors, *Proc. KDD-01*, pages 136–143, New York, Aug. 26–29 2001. ACM Press.

[6] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 369–376, San José, California, USA, 2001. IEEE Computer Society.

[7] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *KDD*, pages 80–86, New York City, New York, USA, Aug. 1998. AAAI Press.

[8] S. Morishita and J. Sese. Traversing itemset lattices with statistical metric pruning. In *PODS*, pages 226–236, Dallas, Texas, USA, May 2000. ACM.

[9] S. Muggleton. Inverse entailment and PROGOL. *New Generation Computing*, 13(3&4):245–286, 1995.

[10] S. Mutter, M. Hall, and F. Frank. Using classification to evaluate the output of confidence-based association rule mining. In G. I. Webb and X. Yu, editors, *Australian Conference on Artificial Intelligence*, pages 538–549, Cairns, Australia, Dec. 2004. Springer.

[11] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[12] E. Suzuki and S. Arikawa, editors. *Discovery Science, 7th International Conference, DS 2004, Padova, Italy, October 2-5, 2004, Proceedings*, Padova, Italy, Oct. 2004. Springer.

[13] M. J. Zaki and C. C. Aggarwal. XRules: an effective structural classifier for XML data. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *KDD*, pages 316–325, Washington, DC, USA, Aug. 2003. ACM.

[14] A. Zimmermann and L. De Raedt. Corclass: Correlated association rule mining for classification. In Suzuki and Arikawa [12], pages 60–72.