Decision Trees for Hierarchical Multilabel Classification: A Case Study in Functional Genomics

Hendrik Blockeel[1], Leander Schietgat[1], Jan Struyf[1,2], Sašo Džeroski[3], and Amanda Clare[4]

[1] Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
{hendrik.blockeel, leander.schietgat, jan.struyf}@cs.kuleuven.be
[2] Dept. of Biostatistics and Medical Informatics, Univ. of Wisconsin, Madison, USA
[3] Department of Knowledge Technologies, Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Saso.Dzeroski@ijs.si
[4] Department of Computer Science, University of Wales Aberystwyth, SY23 3DB, UK
afc@aber.ac.uk

Abstract. Hierarchical multilabel classification (HMC) is a variant of classification where instances may belong to multiple classes organized in a hierarchy. The task is relevant for several application domains. This paper presents an empirical study of decision tree approaches to HMC in the area of functional genomics. We compare learning a single HMC tree (which makes predictions for all classes together) to learning a set of regular classification trees (one for each class). Interestingly, on all 12 datasets we use, the HMC tree wins on all fronts: it is faster to learn and to apply, easier to interpret, and has similar or better predictive performance than the set of regular trees. It turns out that HMC tree learning is more robust to overfitting than regular tree learning.

1 Introduction

Classification refers to the task of learning from a set of classified instances a model that can predict the class of previously unseen instances. Hierarchical multilabel classification differs from normal classification in two ways: (1) a single example may belong to multiple classes simultaneously; and (2) the classes are organized in a hierarchy: an example that belongs to some class automatically belongs to all its superclasses.

Examples of this kind of problems are found in several domains, including text classification [1] and functional genomics [2]. In functional genomics, an important problem is predicting the functions of genes. Biologists have a set of possible functions that genes may have, and these functions are organized in a hierarchy. It is known that a single gene may have multiple functions.

Hierarchical multilabel classification (HMC) can be performed by just learning a binary classifier for each class separately, but this has several disadvantages.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 18–29, 2006.

First, it is *less efficient*, because the learner has to be run $|C|$ times, with $|C|$ the number of classes, which can be hundreds or thousands. Second, it often results in learning from *strongly skewed class distributions*: among hundreds of classes, there are likely to be some that occur infrequently. Many learners have problems with strongly skewed class distributions [3]. Third, *hierarchical relationships* between classes are not taken into account. The constraint that an instance belonging to a class must belong to all its superclasses is not automatically imposed. Finally, from the knowledge discovery point of view, the learned models identify features relevant for one class, rather than identifying features with high *overall relevance*.

Some authors have therefore studied HMC as a separate learning task, and developed learners that learn a single model that predicts all the classes of an example at once (see below). These learners include a few decision tree approaches, but for these no in-depth empirical study has been presented up till now. In this paper we perform such an in-depth study, using datasets from functional genomics. Our study yields novel insights about the suitability of decision trees for HMC, and in particular gene function prediction.

In Section 2 we discuss previous work; in Section 3 we present the system used for the empirical study described in Section 4. In Section 5 we conclude.

2 Related Work

Much work in hierarchical multilabel classification (HMC) has been motivated by text classification. Rousu et al. [1] present the state of the art in this domain, which consists mostly of Bayesian and kernel-based classifiers.

Another application domain of HMC is functional genomics: a typical learning task is to learn a model that assigns to a gene a set of functions, selected from a hierarchy. Barutcuoglu et al. [2] recently presented a two-step approach where support vector machines are learned for each class separately, and then combined using a Bayesian learner so that the predictions are consistent with the hierarchical relationships; this solves one of the four issues mentioned above.

From the point of view of knowledge discovery, it is sometimes useful to obtain more interpretable models, such as decision trees, and that is the kind of approach we will study here.

Clare and King [4] presented a decision tree method for multilabel classification in the context of functional genomics. In their approach, a tree predicts not a single class but a vector of boolean class variables. They propose a simple adaptation of C4.5 to learn such trees: where C4.5 normally uses class entropy, their version uses the sum of entropies of the class variables. Clare [5] extended the method to predict classes on several levels of the hierarchy, assigning a larger cost to misclassifications higher up in the hierarchy, and presented an extensive evaluation on twelve datasets from functional genomics. We use this method as a reference to validate our own approach; we further refer to it as C4.5H.

Blockeel et al. [6] independently proposed a decision tree learner for HMC that is based on the concept of predictive clustering trees [7], where decision

trees are viewed as cluster hierarchies, and present preliminary experiments in text classification and functional genomics as a proof of concept. The approach has been used in some later work [8,9].

Until now these approaches have been evaluated mainly from the biologists' point of view, who commented on the discovered rules and their accuracy. No thorough performance evaluation from a machine learning point of view (what are the advantages over learning a single HMC tree over learning several regular trees?) has been made. Such an evaluation is in fact not trivial, when domain experts want to see as few rules as possible that predict as many classes as possible as correctly as possible. This work is the first thorough empirical comparison between HMC tree learning and learning multiple regular trees.

3 The Clus-HMC Approach

We first define the HMC task more formally; next, we describe the Clus-HMC system in detail.

3.1 Formal Task Description

We define the hierarchical multilabel classification task as follows:

Given: an instance space X and class hierarchy $(C, \leq_h)$, where C is a set of classes and $\leq_h$ is a partial order structured as a rooted tree, representing the superclass relationship (for all $c_1, c_2 \in C$: $c_1 \leq_h c_2$ if and only if c_1 is a superclass of c_2); a set T of examples (x_i, S_i) with $x_i \in X$ and $S_i \subseteq C$ such that $c \in S_i \Rightarrow \forall c' \leq_h c : c' \in S_i$; and some quality criterion q (which typically rewards models with high predictive accuracy and low complexity)
Find: a function $f : X \rightarrow 2^C$ (where 2^C is the power set of C) such that $c \in f(x) \Rightarrow \forall c' \leq_h c : c' \in f(x)$ and f maximizes q.

3.2 Clus-HMC: An HMC Decision Tree Learner

Fig. 1 presents the Clus-HMC algorithm. It is a variant of the standard greedy top-down algorithm for decision tree induction [10,11]. It takes as input a set of training instances T. The main loop of the algorithm searches for the best acceptable attribute-value test that can be put in a node. If such a test t^* can be found then the algorithm creates a new internal node labeled t^* and calls itself recursively to construct a subtree for each subset in the partition $\mathcal{P}^*$ induced by t^* on the training instances. If no acceptable test can be found, then the algorithm creates a leaf.

Up till here, the description is no different from that of a standard decision tree learner. However, decision tree learners normally predict only one target attribute, whereas an HMC tree needs to predict a set of classes. To achieve this, the following changes are made to the learning procedure [6].

First, the example labels are represented as vectors with boolean components; the i'th component of the vector is 1 if the example belongs to class c_i and 0

procedure Clus-HMC(T) **returns** tree
1: $(t^*, h^*, \mathcal{P}^*) = (none, \infty, \emptyset)$
2: **for each** possible test t
3: $\quad \mathcal{P} =$ partition induced by t on T
4: $\quad h = \sum_{T_k \in \mathcal{P}} \frac{|T_k|}{|T|} Var(T_k)$
5: $\quad$ **if** $(h < h^*) \wedge acceptable(t, \mathcal{P})$
6: $\quad\quad (t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: **if** $t^* \neq none$
8: $\quad$ **for each** $T_k \in \mathcal{P}^*$
9: $\quad\quad tree_k =$ Clus-HMC(T_k)
10: $\quad$ **return** node$(t^*, \bigcup_k \{tree_k\})$
11: **else**
12: $\quad$ **return** leaf$(\bar{v})$

Fig. 1. The Clus-HMC induction algorithm

otherwise. It is easily checked that the arithmetic mean of a set of such vectors contains as i'th component the proportion of examples of the set belonging to class c_i. We define the variance of a set of examples as the average squared Euclidean distance between each example's label and the set's mean label.

The heuristic for choosing the best test in a node of the tree is then minimization of the average variance in the created subsets (weighted according to the size of the subsets, see line 4 of Fig. 1). This corresponds to the heuristic typically used when learning regression trees and to CART's Gini index [10], and is in line with the "predictive clustering trees" view [7]. The heuristic ensures that examples labelled with similar sets of classes tend to go into the same subset.

In the HMC context, it makes sense to consider similarity on higher levels of the hierarchy more important than similarity on lower levels. To that aim, we can use for the variance a weighted Euclidean distance $d(x_1, x_2) = \sqrt{\sum_k w_k \cdot (v_{1,k} - v_{2,k})^2}$, where $v_{i,k}$ is the k'th component of the class vector v_i of instance x_i, and the weights w_k decrease with the depth of the class c_k in the hierarchy (e.g., $w_k = w_0^{\text{depth}(c_k)}$). Consider for example the class hierarchy shown in Fig. 2, and two examples (x_1, S_1) and (x_2, S_2) with $S_1 = \{1, 2, 2/2\}$ and $S_2 = \{2\}$. Using a vector representation with consecutive components representing membership of class 1, 2, 2/1, 2/2 and 3, in that order, $d(x_1, x_2) = d_{\text{Euclidean}}([1, 1, 0, 1, 0], [0, 1, 0, 0, 0]) = \sqrt{w_0 + w_0^2}$.

A decision tree normally stores in a leaf the majority class for that leaf; this class will be the tree's prediction for examples arriving in the leaf. But in our case, since an example may have multiple classes, there is no "majority class". Instead, the mean $\bar{v}$ of the vectors of the examples in that leaf is stored; in other words, for each class c_i, the proportion of examples belonging to c_i is kept. An example arriving in the leaf will be predicted to belong to class c_i if the i-th component of $\bar{v}$ is above some threshold t_i. To ensure that the predictions fulfill the constraint that whenever a class is predicted its superclasses are also predicted, it suffices to choose $t_i \leq t_j$ whenever $c_i \leq_h c_j$.

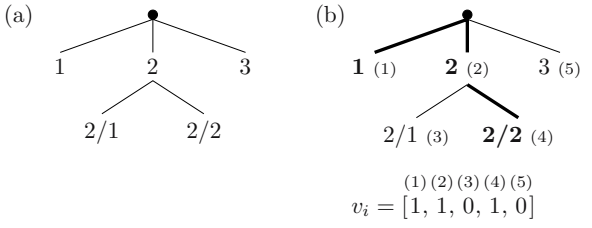

Fig. 2. (a) A toy hierarchy. Class label names reflect the position in the hierarchy, e.g., '2/1' is a subclass of '2'. (b) The set of classes {1,2,2/2}, indicated in bold in the hierarchy, and represented as a vector.

The predictive accuracy of the model is maximized by taking all $t_i = 0.5$, but as we are dealing with skewed class distributions, accuracy is not a very good evaluation criterion, and hence there is no good reason to try to maximize it. Precision-recall based evaluation is preferred in such cases [12]. Precision is the probability that a positive prediction is correct, and recall is the probability that a positive instance is predicted positive. When decreasing t_i from 1 to 0, an increasing number of instances is predicted as belonging to c_i, causing the recall for c_i to increase whereas precision may increase or decrease (with normally a tendency to decrease). Thus, a tree with specified threshold has a single precision and recall, and by varying the threshold for a single tree a precision-recall curve (PR curve) is obtained. Such curves allow us to evaluate the predictive performance of a model regardless of t.

Finally, the function *acceptable* in Fig. 1 verifies for a given test that the number of instances in each subset of the corresponding partition $\mathcal{P}$ is at least *mincases* (a parameter) and that the variance reduction is significant according to a statistical F-test.

The version of Clus-HMC described here is exactly the same as in Struyf et al. [8], except that the latter commits to a treshold of 0.5 whereas our version does not. Commitment to a fixed threshold causes models to be partially ordered in a precision-recall evaluation (e.g., a model may have higher recall but lower precision than another model), which is undesirable.

4 Experiments in Yeast Functional Genomics

The goals of our experiments are twofold. First, we wish to validate the "predictive clustering trees" approach to HMC, as implemented in Clus-HMC, by comparing its precision-recall behaviour to C4.5H, the state of the art in decision tree based HMC. Second, and most importantly, we evaluate the strengths and weaknesses of HMC tree learning as compared to learning a separate tree for each class. The expectation here is that HMC tree learning is faster and yields a tree that is more complex than an individual single-class tree, but less complex than the whole set of trees; one would further hope that this simplicity does not come at the cost of worse predictive performance.

```
1 METABOLISM
1/1 amino acid metabolism
1/2 nitrogen and sulfur metabolism
...
2 ENERGY
2/1 glycolysis and gluconeogenesis
...
```

Fig. 3. A small part of the hierarchical FunCat classification scheme

Table 1. Dataset properties: number of instances $|D|$, number of attributes $|A|$

Dataset	$\|D\|$	$\|A\|$	Dataset	$\|D\|$	$\|A\|$
D_1 Sequence (seq)	3932	478	D_7 DeRisi et al. (derisi)	3733	63
D_2 Phenotype (pheno)	1592	69	D_8 Eisen et al. (eisen)	2425	79
D_3 Secondary structure (struc)	3851	19628	D_9 Gasch et al. (gasch1)	3773	173
D_4 Homology search (hom)	3867	47034	D_{10} Gasch et al. (gasch2)	3788	52
D_5 Spellman et al. (cellcycle)	3766	77	D_{11} Chu et al. (spo)	3711	80
D_6 Roth et al. (church)	3764	27	D_{12} All microarray (expr)	3788	551

4.1 Datasets

Saccharomyces cerevisiae (baker's or brewer's yeast) is one of biology's classic model organisms, and has been the subject of intensive study for years. Its genes have annotations provided by the Munich Information Center for Protein Sequences (MIPS) under their FunCat scheme for classifying the functions of the products of genes. FunCat is a hierarchical system of functional classes. A small part of this hierarchy is shown in Fig. 3. Many yeast genes are annotated with more than one functional class.

We use the 12 datasets from [5]. An overview of these datasets is given in Table 1. The different datasets describe different aspects of the genes in the yeast genome. Five types of bioinformatic data for yeast are considered: sequence statistics (D_1), phenotype (D_2), predicted secondary structure (D_3), homology (D_4), and expression as measured with microarray chips (D_5 – D_{12}). The biologists' motivation for this is that different sources of data should highlight different aspects of gene function. More information on how the datasets were constructed and relevant references to the literature can be found in [5].[1]

Each gene in the datasets is annotated with one or more classes selected from the MIPS FunCat hierarchical classification scheme. The annotations and classification scheme available on 4/24/2002 were used. The hierarchy has 250 classes: 17 at the first level, 102 at the second, 89 at the third, and 42 at the fourth level.

4.2 Method

Clare [5] presents models trained on 2/3 of each dataset and tested on the remaining 1/3. In our experiments we use exactly the same training and test sets.

[1] Available together with the datasets at `http://www.aber.ac.uk/compsci/Research/bio/dss/yeastdata/`

To evaluate C4.5H, we computed the precision and recall of Clare's models. These models were presented as rules derived from the trees.

Clus-HMC results were obtained as follows. The weights used for the weighted Euclidean distance were chosen as $w_k = w_0^{\text{depth}(c_k)}$, with w_0 set to 0.75, and *mincases* was set to 5. The F-test stopping criterion takes a "significance level" parameter s, which was optimized as follows: for each out of 18 available values for s, Clus-HMC was run on 67% of the training set and its PR curve for the remaining 33% was constructed. The model having the largest area under this validation PR curve was then used to construct a PR curve for the test set. PR curves were constructed with non-linear interpolation between points [12].

To compare HMC tree learning to regular tree learning, we can use Clus on both sides. Indeed, the Clus system can be used for single classification just by reducing the class vector to a single component; its heuristic then reduces to CART's Gini index [10], and it performs comparably to regular decision tree learners. We refer to this version as Clus-SC. The results for Clus-SC were obtained in the same way as for Clus-HMC, but with a separate run for each class (including separate optimization of s for each class).

With 250 classes, each of which has its own PR curve, there is the question of how to evaluate the overall performance of a system. We can construct a single "average" PR curve for all classes together by counting instance-class-couples instead of instances. An instance-class couple is (predicted) positive if the instance has (is predicted to have) that class. The definition of precision and recall is then as before.

4.3 Results

Comparison with C4.5H. For each of the 12 datasets, average PR curves were generated and plotted against the points obtained by C4.5H. As we are comparing curves with points, we speak of a "win" for Clus-HMC when its curve is above C4.5H's point, a "loss" when it is below. Under the null hypothesis that both systems perform equally well, we expect as many wins as losses. For the average PR curves, we found 12 wins out of 12 for Clus-HMC. 4 representative plots are shown in Fig. 4. We have also included results for "Clus05", the predecessor of Clus-HMC where the s parameter was optimized for maximal precision given a fixed threshold of 0.5 [8]. Without the F-test optimization, the points of Clus05 would be on the Clus-HMC curve; small differences are due to the slightly different optimization criteria. It can clearly be seen that committing to a threshold of 0.5 kept Clus05 from achieving maximal precision.

We also made a class-by-class comparison: for each dataset and for each class for which C4.5H produced rules, we compared its PR to the Clus-HMC curve. Here we found 25 wins and 6 losses. Fig. 5 details the performance on the gasch1 dataset for the 7 classes predicted by C4.5H. Class 6/13/1 is the only class where Clus-HMC did not yield any classifiers strictly better than C4.5H. For Class 1 the C4.5H point is slightly above the Clus-HMC curve, yet Clus-HMC yields one classifier with the same precision but more than twice the recall.

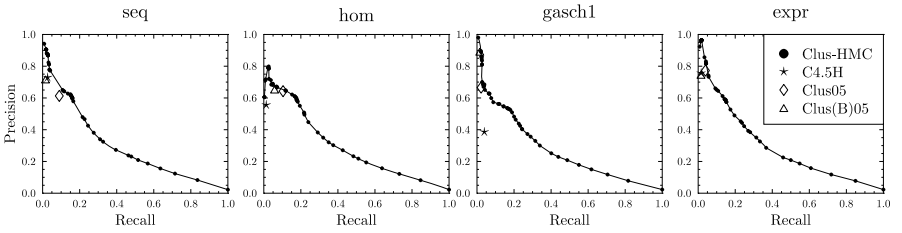

Fig. 4. Average precision/recall over all classes for Clus-HMC, C4.5H, and two versions of Clus-HMC's predecessor [8]

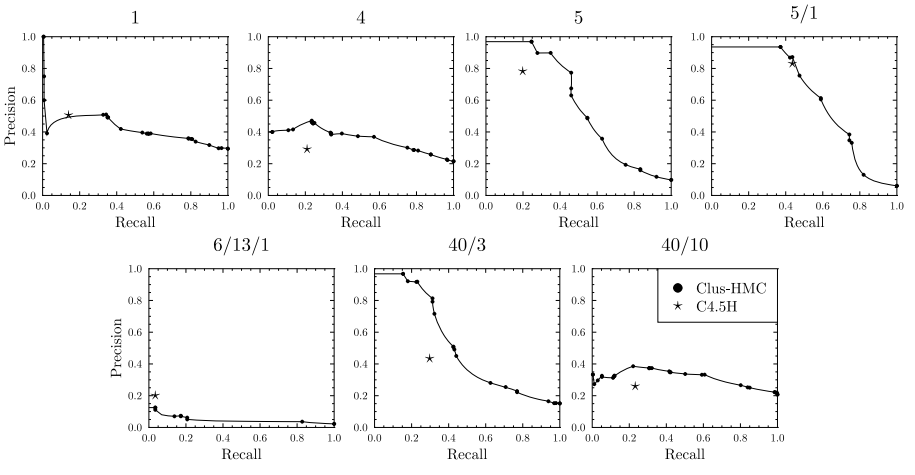

Fig. 5. Class by class comparison between Clus-HMC and C4.5H (gasch1)

Comparing the interpretability and precision/recall of individual rules (where a rule describes a single leaf of the tree), Clus-HMC also performs well. For instance, in the gasch1 dataset, for the class 40/3 (with prior frequency 14%), C4.5H learned two rules:

```
IF 29C_Plus1M_sorbitol_to_33C_Plus_1M_sorbitol___15_minutes <= 0.03 AND
  constant_0point32_mM_H2O2_20_min_redo <= 0.72 AND
  1point5_mM_diamide_60_min <= -0.17 AND
  steady_state_1M_sorbitol > -0.37 AND
  DBYmsn2_4__37degree_heat___20_min <= -0.67
THEN 40/3

IF Heat_Shock_10_minutes_hs_1 <= 1.82 AND
  Heat_Shock_030inutes__hs_2 <= -0.48 AND
  29C_Plus1M_sorbitol_to_33C_Plus_1M_sorbitol___5_minutes > -0.1
THEN 40/3
```

They have a precision/recall of 0.52/0.26 and 0.56/0.18, respectively. Clus-SC's most precise rule for 40/3, obtained by selecting a high threshold, is

```
IF Nitrogen_Depletion_8_h <= -2.74 AND Nitrogen_Depletion_2_h > -1.94 AND
   1point5_mM_diamide_5_min > -0.03 AND 1M_sorbitol___45_min_ > -0.36 AND
   37C_to_25C_shock___60_min > 1.28
THEN 40/3
```

with a precision/recall of 0.97/0.15. The second point on the PR curve turns out to represent the same rule with the last condition dropped; this rule scores 0.92/0.18. The best model consisting of two rules scores 0.92/0.23, and the best 3-rule model 0.81/0.31.

These results confirm that under precision-recall evaluation Clus-HMC performs at least as well as C4.5H, and can thus be considered a state-of-the-art HMC tree learner.

Comparison with Single Classification. We now turn to a comparison of HMC tree learning with learning separate trees for each class, using the Clus-HMC and Clus-SC instantiations. To limit the total runtime of Clus-SC, 40 classes were sampled from the hierarchy – 10 for each level.

For each dataset two average PR curves were generated: one for single classification, one for multilabel classification. Fig. 6 shows a few representative graphs. The single classification curve usually lies completely below the multilabel classification curve. Table 2 shows the difference in area under the PR curve (AUPRC) between Clus-HMC and Clus-SC; all differences are positive.

Fig. 7 shows some representative PR curves for gasch1. There are noticeable differences: sometimes Clus-SC performs better, sometimes Clus-HMC. So for individual classes the outcome is less clear-cut, but on average, Clus-HMC performs slightly better.

This is unexpected: one would think that Clus-SC has the advantage because it can learn a different optimal model for each class. Our original conjecture was that this was due to Clus-HMC performing better on the lower levels of the hierarchy.[2] But a per-level computation of the AUPRC difference (see again Table 2) does not confirm this: Clus-HMC tends to perform better overall, there is no clear correlation with depth in the hierarchy.

Further investigation revealed that Clus-SC tends to overfit more than Clus-HMC. Subtracting the area under the PR curve (AUPRC) obtained on the test set from that on the training set gives an indication of how strongly the approach overfits. Clus-SC scored a difference of 0.219, Clus-HMC 0.024.[3]

In hindsight, it makes sense that Clus-HMC overfits less than Clus-SC: overfitting 250 target values is simply more difficult than overfitting a single target

[2] Level four classes are very infrequent and therefore difficult to learn, but in Clus-HMC the parent classes may help in keeping the instances from class x/y/z together in the tree, and within a node with mainly x/y/z instances, a class x/y/z/u has a higher relative frequency.

[3] To make sure that this overfitting behaviour is not an artifact of our particular implementation, we also ran M5' from Weka [13] on the same datasets. M5' did not produce better results on the test set than Clus-SC and overfitted even more, with an AUPRC difference of 0.387. The tendency of M5' to overfit more than Clus-SC is consistent with our previous experience and with an earlier analysis by [14].

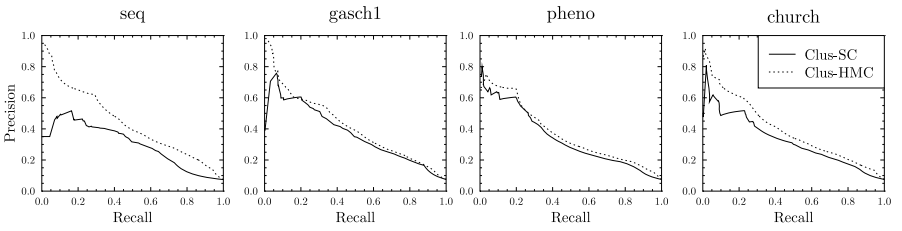

Fig. 6. Average precision/recall over all classes for Clus-SC and Clus-HMC

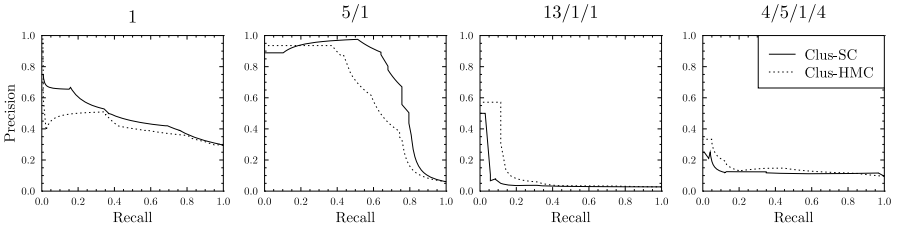

Fig. 7. Class by class comparison between Clus-SC and Clus-HMC (gasch1)

value. This is also visible in the tree sizes: Clus-HMC trees contain on average 24 nodes, whereas Clus-SC learns per dataset 250 trees with an average size of 33 nodes. In addition, Clus-HMC naturally takes dependencies between different classes into account (e.g., the constraints imposed by the hierarchy).

Clus-HMC runs slower than Clus-SC because it takes information about $|C|$ classes into acount, but on the other hand Clus-SC needs to be run $|C|$ times. On our twelve datasets Clus-HMC was 4.5 to 65 times faster than running Clus-SC for 250 classes, with an average speedup factor of 37.

Table 2. Overall and level-wise comparison of the area under the PR curve (AUPRC) of Clus-HMC and Clus-SC. Numbers are AUPRC(Clus-HMC) – AUPRC(Clus-SC).

Dataset	All	level 1	level 2	level 3	level 4
seq	0.142	0.135	0.086	0.056	0.025
pheno	0.030	0.025	0.001	0.022	0.010
struc	0.061	0.035	0.001	0.043	0.039
hom	0.057	0.058	0.032	0.028	0.036
cellcycle	0.038	0.037	-0.035	0.007	0.007
church	0.070	0.066	0.052	0.031	0.019
derisi	0.112	0.124	0.029	0.020	0.046
eisen	0.067	0.085	0.023	0.011	0.021
gasch1	0.044	0.027	0.041	0.032	0.018
gasch2	0.096	0.103	0.043	0.055	0.013
spo	0.095	0.102	0.022	-0.012	0.040
expr	0.099	0.071	0.062	0.031	0.021

5 Conclusions

We have conducted an empirical study of decision tree approaches to hierarchical multilabel classification.

Minor contributions of this study are, in the context of the predictive clustering trees (Clus) approach to HMC, (1) the description of a better tuned version of Clus-HMC (one that does not use thresholds that are suboptimal for precision-recall evaluation), and (2) a comparison showing that this version is at least as good as, and perhaps slightly better than, earlier HMC tree learning systems: it tends to yield rules with higher precision and recall and similar interpretability.

The major contribution however is the comparison between (a) learning a single tree that predicts all classes at once with a HMC-oriented algorithm, and (b) learning a separate decision tree for each class. We find that learning a single HMC tree is much faster than learning many regular trees, and it has the additional advantage of identifying features that are relevant for all the functions together (instead of separate features for each function). Obviously, a single HMC tree is also much more efficient to apply than 250 separate trees. Somewhat less expectedly, the HMC tree has on average a comparable predictive performance for each single class as a regular tree optimized for just that class. Our conjecture that the information contained in the hierarchy improves classification for infrequent classes in the lower parts of the hierarchy, which would partially explain this, was not confirmed. Instead, it turns out that the HMC approach is less susceptible to overfitting than the single classification approach. Fitting a model to many classes is indeed harder than fitting it to one class. Further, the HMC approach naturally takes into account dependencies between class membership, which may help it in making the right decisions during learning.

Given that HMC decision trees can yield better efficiency and interpretability without suffering a decline in predictive accuracy compared to learning separate trees, their use should definitely be considered in HMC tasks where interpretable models are desired.

Acknowledgements

H.B. and J.S. are post-doctoral fellows of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). L.S. is supported by a PhD grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). The authors thank Maurice Bruynooghe and Elisa Fromont for valuable suggestions and proofreading.

References

1. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Learning hierarchical multi-category text classification models. In De Raedt, L., Wrobel, S., eds.: Proceedings of the 22nd International Conference on Machine Learning, ACM Press (2005) 744 – 751
2. Barutcuoglu, Z., Schapire, R.E., Troyanskaya, O.G.: Hierarchical multi-label prediction of gene function. Bioinformatics **22**(7) (2006) 830–836

3. Weiss, G.M., Provost, F.J.: Learning when training data are costly: The effect of class distribution on tree induction. J. Artif. Intell. Res. (JAIR) **19** (2003) 315–354
4. Clare, A., King, R.: Knowledge discovery in multi-label phenotype data. In De Raedt, L., Siebes, A., eds.: 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2001). Volume 2168 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2001) 42–53
5. Clare, A.: Machine learning and data mining for yeast functional genomics. PhD thesis, University of Wales, Aberystwyth (2003)
6. Blockeel, H., Bruynooghe, M., Džeroski, S., Ramon, J., Struyf, J.: Hierarchical multi-classification. In: Proceedings of the ACM SIGKDD 2002 Workshop on Multi-Relational Data Mining (MRDM 2002). (2002) 21–35
7. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: Proceedings of the 15th International Conference on Machine Learning. (1998) 55–63
8. Struyf, J., Džeroski, S., Blockeel, H., Clare, A.: Hierarchical multi-classification with predictive clustering trees in functional genomics. In: Progress in Artificial Intelligence: 12th Portugese Conference on Artificial Intelligence. Volume 3808 of Lecture Notes in Computer Science., Springer (2005) 272–283
9. Struyf, J., Vens, C., Croonenborghs, T., Dzeroski, S., Blockeel, H.: Applying predictive clustering trees to the inductive logic programming 2005 challenge data. In: Inductive Logic Programming, 15th International Conference, ILP 2005, Late-Breaking Papers, Institut für Informatik der Technischen Universität München (2005) 111–116
10. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth, Belmont (1984)
11. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann (1993)
12. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. Technical report, University of Wisconsin, Madison (2005)
13. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)
14. Torgo, L.: A comparative study of reliable error estimators for pruning regression trees. In Coelho, H., ed.: Proceedings of the Iberoamerican Conference on AI (IBERAMIA-98), Springer (1998)