Hierarchical Multi-classification with Predictive Clustering Trees in Functional Genomics

Clare, Amanda; Džeroski, Sašo; Struyf, Jan; Blockeel, Hendrik

Publication date:
2005

Citation for published version (APA):
Clare, A., Džeroski, S., Struyf, J., & Blockeel, H. (2005). *Hierarchical Multi-classification with Predictive Clustering Trees in Functional Genomics*.

Hierarchical Multi-classification with Predictive Clustering Trees in Functional Genomics

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

[1]Katholieke Universiteit Leuven, Dept. of Computer Science
Celestijnenlaan 200A, B-3001 Leuven, Belgium
{Jan.Struyf,Hendrik.Blockeel}@cs.kuleuven.be
[2]Jozef Stefan Institute, Dept. of Knowledge Technologies
Jamova 39, 1000 Ljubljana, Slovenia
Saso.Dzeroski@ijs.si
[3]The University of Wales, Aberystwyth, Dept. of Computer Science
Penglais, Aberystwyth, Ceredigion, SY23 3DB, Wales, UK
afc@aber.ac.uk

Abstract. This paper investigates how predictive clustering trees can be used to predict gene function in the genome of the yeast *Saccharomyces cerevisiae*. We consider the MIPS FunCat classification scheme, in which each gene is annotated with one or more classes selected from a given functional class hierarchy. This setting presents two important challenges to machine learning: (1) each instance is labeled with a set of classes instead of just one class, and (2) the classes are structured in a hierarchy; ideally the learning algorithm should also take this hierarchical information into account. Predictive clustering trees generalize decision trees and can be applied to a wide range of prediction tasks by plugging in a suitable distance metric. We define an appropriate distance metric for hierarchical multi-classification and present experiments evaluating this approach on a number of data sets that are available for yeast.

1 Introduction

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. 1. Many yeast genes are annotated with more than one functional class.

This classification setting presents two main challenges to machine learning: (1) each instance (gene) is labeled with a set of classes instead of just one class, and (2) the classes are structured in a hierarchy; ideally the learning algorithm should also take this hierarchical information into account.

A simple approach is to ignore the hierarchy and to learn separate models for each class (indicating whether a single instance belongs to the class or not).

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

(b)

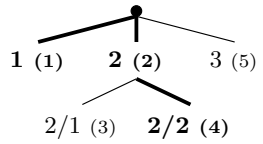

Fig. 1. (a) A part of the hierarchical FunCat classification scheme. (b) A toy hierarchy that will be used as example throughout the text. (Note that the class labels indicate the position in the hierarchy: 2/1 is a subclass of the class 2.)

In this work, we consider instead the task of learning one model for all classes. This has the advantage that the total size of the predictive theory is typically smaller, and that dependencies between different classes w.r.t. membership can be taken into account and may even be explicitated. Advantages of learning a single model for multiple related prediction tasks have been reported several times in the literature (see e.g., [6] for decision trees, [8, 1] for neural networks, [15] for text classification).

Taking into account the hierarchical structure of the classes is also important while learning. The hierarchy concisely conveys relevant information about the similarity and differences between classes and also expresses the constraint that an instance belonging to a class also belongs to the parent class. The combination of multi-classification and hierarchical classification is known as hierarchical multi-classification [3].

Blockeel et al. [3] show how predictive clustering trees (PCTs) can be applied to hierarchical multi-classification. PCTs form a very general framework for prediction that can be instantiated to a particular prediction task by defining a distance metric and prototype. The distance metric used in [3] is a generic distance between sets of classes that is subsequently instantiated for hierarchical classification by plugging in the weighted shortest path distance between individual classes. The set distance has however two major disadvantages: (1) the distance between two given sets is difficult to interpret (it involves the computation of a kernel), and (2) it is not guaranteed to be positive because the kernel matrix is not positive definite. The impact of the latter is difficult to evaluate.

In this work we take an approach similar to that of [3]. We also use PCTs, but we introduce a new distance metric that is specific to hierarchical multi-classification and that does not have the disadvantages of the distance metric used in [3].

Recently, an extension to C4.5 [11] has been introduced by Clare [9] that is also capable of hierarchical multi-classification. This is accomplished by adapting the definition of entropy to take into account both the multi-class aspect and the hierarchical relationship between the classes. In the experimental evaluation included in this paper we compare our approach to the results presented in [9].

This paper is organized as follows. We first define hierarchical multi-classification more formally (Section 2). Then follows a brief description of PCTs (Section 3). Section 4 shows how PCTs can be instantiated for hierarchical multi-classification. This approach is validated experimentally in Section 5. Section 6 discusses further work, and Section 7 states the main conclusions.

2 Hierarchical Multi-classification

We represent a hierarchy on a set of classes C as a tree, defined by a set of top-level classes $T \subseteq C$ and a *parent* function that maps the children of a tree node onto the node (it is not defined for the top-level classes). A *valid set* is a set of classes that is closed with respect to the *parent* function, i.e., if $c \in S$ then $parent(c) \in S$ or $c \in T$. 2^C denotes the power set of C and $V(C) \subseteq 2^C$ denotes the set of valid sets constructed from C.

Example 1. In Fig. 1.b, $C = \{1, 2, 2/1, 2/2, 3\}$, $T = \{1, 2, 3\}$, $parent(2/2) = 2$, and $\{1, 2, 2/2\} \in V(C)$ is a valid set of classes. Note that a valid set of classes always corresponds to a subtree of the hierarchy. In the case of hierarchical single-classification, the subtree reduces to a path.

The problem of hierarchical multi-classification can now be stated as follows.
Given:
- an instance space X
- a set of labeled instances $D \subseteq X \times V(C)$
- a class space C
- a quality criterion Q
- a hierarchy H defined on C

Find: a function $h : X \rightarrow V(C)$ that maps an instance x onto a valid set of classes S, so that h maximizes the quality criterion Q.

The hierarchy concisely conveys relevant information about the similarity and differences between classes. Intuitively, the distance between two classes is smaller if they are closer to each other in the hierarchy. Further, the siblings of a node should be equidistant, and the distance from a node to its parent is the same for all the nodes on a given level. The distance metric that we will introduce in Section 4 fulfills these criteria.

The quality criterion Q can, but need not be based on the distance. For instance, it could be just the average precision with which all the different classes are predicted, or it could take into account the fact that predicting $\{2, 2/1\}$ is a smaller mistake for an instance that is labeled $\{2, 2/2\}$ than for an instance labeled $\{1\}$.

We finally remark that by representing the labels as subtrees of the hierarchy, the natural constraints on class membership, i.e., anything belonging to a specific class (e.g., 2/1) automatically belongs to the more general ancestor classes (e.g., 1) are automatically honored. This would not be guaranteed if independent models were learned for all different classes.

3 Predictive Clustering Trees

A variety of algorithms for predictive modeling exists. Among the better known are algorithms that induce decision trees [7, 11]. Compared to other well-known techniques such as neural networks [2], decision trees have the advantage of being more interpretable: they clearly explicitate the factors that influence the outcome most strongly.

Decision trees are most often used in the context of classification or single-target regression; i.e., they represent a model in which the value of a single variable is predicted. However, as a decision tree naturally identifies partitions of the data (course-grained at the top of the tree, fine-grained at the bottom), one can also consider a tree as a hierarchy of clusters [10]. A good cluster hierarchy is one in which individuals that are in the same cluster are also similar with respect to a number of observable properties. This leads to a simple method for building trees that allow the prediction of multiple target attributes at once. If we can define a distance measure on tuples of target variable values, we can build decision trees for multi-target prediction. Similarly, if a distance on hierarchical target values is defined, we can build decision trees for hierarchical classification. The methodology has been used successfully for a variety of applications such as conceptual clustering [5], simultaneous prediction of multiple parameters [6], and ranking tasks [13].

The algorithm for inducing these so-called predictive clustering trees (PCTs) is essentially a standard TDIDT (Top-Down Induction of Decision Trees) algorithm such as C4.5 [11]. The general idea is to recursively partition a set of data into clusters in such a way that the intra-cluster variation is minimized. (The heuristic for selecting the test to include in a node of the tree is the sum of the intra-cluster variations of the subsets induced by the test.) Intra-cluster variation is defined as the sum of squared distances between the members of the cluster and its prototype p, where the latter is defined as $p = \arg\min_q \sum_i d(x_i, q)^2$, i.e., roughly the point that is closest to all the instances in the cluster, according to the distance defined. This prototype may or may not be a valid prediction. For instance for 0-1 prediction the prototype could be the mean of all target values, e.g., 0.8, but when making a prediction for a specific instance this has to be converted into a valid prediction (0 or 1). The result of the induction process is a decision tree in which each leaf contains (a prediction derived from) the prototype of the examples covered by that leaf.

A detailed description of PCTs can be found in [5]. The main point to be made here is that the proposed method for inducing PCTs relies entirely on the definition of the distance measure, the prototypes, and the mapping of prototypes onto valid predictions. These issues are the focus of the following section.

4 PCTs for Hierarchical Multi-classification

In this section, we show how PCTs can be applied to hierarchical multi-classification. As indicated above, this comes down to defining a suitable distance metric and prototype.

4.1 Representing a Valid Set of Classes as a Vector

In the hierarchical multi-classification setting, each instance i is annotated with a valid set of classes $C_i \in V(C)$ selected from a hierarchically structured set of classes C. The distance metric and prototype that we will use are based on a vector representation of C_i. This representation is constructed as follows. The vector v_i representing C_i is a vector with $|C|$ components. Each component corresponds to a class of C. The components of v_i that correspond to classes in C_i take the value 1, the others are set to 0.

Example 2. Consider the class hierarchy shown in Fig 1.b and suppose that a given instance i is annotated with the valid set $C_i = \{1, 2, 2/2\}$. Assuming that $v_{i,k}$ corresponds to the class at position k in the preorder traversal of the hierarchy (as indicated by the numbers between parenthesis in Fig 1.b), the vector representing C_i is $v_i = [1, 1, 0, 1, 0]$.

4.2 The Distance Metric and Prototype

Because each valid set of classes is represented as a vector, we can use the Euclidean distance as distance metric and define the distance between two valid sets C_i and C_j as the Euclidean distance between their vector representations.

$$d(C_i, C_j) = d_{\text{Euclidean}}(v_i, v_j) = \sqrt{\sum_k w_k \cdot (v_{i,k} - v_{j,k})^2} \tag{1}$$

The hierarchical relationship among the classes can be taken into account by setting the weights w_k in (1) to appropriate values. If the weight for classes deeper down the hierarchy is smaller than that of classes closer to the top, then the distance between two top-level classes will be large and the distance between sibling classes deeper down the hierarchy will be small. In the experimental evaluation, we will use weights that decrease exponentially with hierarchy depth: $w_k = w_0^{\text{depth}(c_k)}$, with w_0 a parameter, which we set ad-hoc to 0.75. It can be easily verified that this choice fulfills the criteria listed in Section 2.

Example 3. Consider two instances, the first one annotated with $C_i = \{1, 2, 2/2\}$ and the second one with $C_j = \{2\}$. The distance between C_i and C_j is $d(C_i, C_j) = d_{\text{Euclidean}}([1, 1, 0, 1, 0], [0, 1, 0, 0, 0]) = \sqrt{w_0 + w_0^2}$. Note that this distance can be interpreted as the square-root of the sum of a penalty w_0 because class 1 does not occur in C_j and a penalty w_0^2 because class 2/2 does not occur in C_j.

Consider a set of vectors V. The prototype p_V of V corresponding to the Euclidean distance is the vector mean of V, i.e., $p_V = \sum_{v_i \in V} v_i / |V|$. If D is a set of instances and V is the set of vectors representing their target values then each component of p_V represents the proportion of the instances in D that belong to the corresponding class.

Table 1. Data set properties. $|D|$ is the number of instances (genes) and $|A|$ the number of attributes.

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

Example 4. Consider the instances of Example 3. The prototype is $p_V = ([1,1,0,1,0] + [0,1,0,0,0])/2 = [0.5,1,0,0.5,0]$ and indicates that all instances belong to class 2 and 50% belong to the classes 1 and 2/2.

The intra-cluster variation of D, which is used in the heuristic when building PCTs, can now be computed as the sum of squared distances between the members of V and its prototype p_V.

Using the Euclidean distance has the advantage over other distance metrics defined on sets [12] that both the distance and the prototype can be computed efficiently. This is important in the context of PCTs because the distance and prototype are used in the computation of the heuristic and this heuristic must be evaluated for each possible split of the instances considered by the system.

4.3 Mapping a Prototype to a Prediction

The prediction associated with a leaf is the set of classes that occur in at least 50% of the training examples belonging to the leaf. Note that this set is always a valid set. It can be computed based on the prototype as the set of classes that correspond to the components that are greater or equal to 0.5.

5 Experimental Evaluation

In this section we present experiments evaluating PCTs for hierarchical multi-classification, i.e., PCTs with the prototype and distance metric discussed in the previous section plugged in. We first describe the data sets used in the evaluation, then we define the experimental setup and finally we present and discuss the obtained results.

5.1 Data Sets

We use the 12 data sets that were also used by Clare [9] (Table 1). The reason is that we will compare PCTs for hierarchical multi-classification to the hierarchical extension to C4.5 [11] presented in [9].

The data sets describe different aspects of the genes in the *Saccharomyces cerevisiae* genome (baker's or brewer's yeast). Each gene included in the data

sets is annotated with one or more classes selected from the MIPS FunCat hierarchical classification scheme. The annotations and classification scheme that was available on 24/4/02 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.

Five types of bioinformatic data for yeast are considered in the data sets: sequence statistics, phenotype, predicted secondary structure, homology, and expression. Different sources of data should highlight different aspects of gene function. Below, we describe each data set in turn. Note that the relevant references to the literature have been omitted because of space restrictions. These references are available in [9], which can be obtained together with the data sets themselves at `http://www.aber.ac.uk/compsci/Research/bio/dss/yeastdata/`.

(D_1, sec) Sequence statistics are recorded that depend on the amino acid sequence of the protein produced by the gene. These include amino acid ratios, sequence length, molecular weight and hydrophobicity. Some of the properties were calculated using Expasy's ProtParam tool, some were listed by MIPS as part of the description of the sequence such as the chromosome on which the gene was located, and some were simply calculated directly. Attributes are mostly real valued, although some (like chromosome number or strand) are discrete.

(D_2, pheno) Phenotype data represents the growth or lack of growth of knock-out mutants that are missing the gene in question. A gene is removed or disabled and the resulting organism is grown with a variety of media to determine what the modified organism might be sensitive or resistant to. Phenotype data was taken from EUROFAN, MIPS and TRIPLES. Attributes for this dataset are discrete, and the dataset is sparse, since not all knock-outs have been grown under all conditions.

(D_3, seq) The secondary structure of a protein is also known to influence the function of the protein. Secondary structure is caused by hydrogen bonding along the protein's backbone, and there are two main classes of secondary structure elements, the alpha helix and the beta sheet. Structure that can't be classified as alpha or beta is usually termed "coil". Yeast does not have known structure for all of its genes; however secondary structure can be predicted from protein sequences with reasonable precision. The program Prof was used to generate predicted secondary structure for each gene. Due to the relational nature of this type of data a preprocessing step of relational association mining was employed to generate frequent associations from the data. The discovered associations are included as binary attributes.

(D_4, hom) Genes are homologous when they share a common ancestor. In determining the function of a yeast gene, information about a homologous gene from another species can provide clues to the possible role of the gene. Homology is usually determined by sequence similarity. If two genes have similar sequences they are deemed homologous, and standard software exists for finding all such similar sequences in a large database. PSI-BLAST was used to compare yeast genes both with other yeast genes, and with all genes whose proteins are indexed in SwissProt version 39, a database of well-annotated genes from all species. This provided for each yeast gene, a list of homologous genes, and for each of these

homologous genes various properties were extracted, such as keywords, sequence length and the names of the other databases they were known to be listed in. This relational dataset was then mined for frequent associations in the same way as the secondary structure data, to produce binary attributes.

$(D_5, \ldots, D_{12})$ The use of microarrays to gather information on the expression of genes is currently popular in biology and bioinformatics. Microarray chips now provide the means to test the expression levels of genes across an entire genome in a single experiment. Many expression data sets exist for yeast, and several of these were used. Attributes for these datasets are real valued, representing fold changes in expression levels.

5.2 Method

Implementation The distance metric and prototype for hierarchical multi-classification introduced in Section 4 are implemented in the CLUS system[1]. CLUS is a system for building PCTs and is essentially a propositional version of the TILDE system [4, 5].

CLUS has a parameter that controls the minimum number of instances in each leaf. This parameter was set ad-hoc to 5. CLUS only considers tests that yield a significant reduction in intra-cluster variation. The significance level of this test was tuned for each experiment using a 3 fold cross-validation on the training set to maximize average class-wise precision (see further). Other parameters were set to their default values.

Obtaining Validated Predictions As already said above, we are interested in comparing our method to the hierarchical extension of C4.5 [11] introduced by [9]. In [9], the predictions are validated on a separate validation set to obtain a higher precision at the expense of coverage. For each class predicted by a leaf of the decision tree, a significance test is performed. Suppose that the leaf covers N validation instances and that the proportion of instances belonging to the predicted class is a (the precision of the prediction). The test then computes the probability that the proportion of instances of the predicted class in a random sample of size N is greater than a (using the hypergeometric distribution). If this probability is above the significance level, then the prediction is considered insignificant and removed.

In [9], the significance level was set to 0.05 and the Bonferroni correction was used. The latter divides the significance level by the number of tests performed (in this case the sum of the predicted number of classes over all leaves of the decision tree). Such a correction is advised if the number of tests is large. We use the same significance level and perform experiments with and without the Bonferroni correction. Note that, because of the validation step, the predictions are no longer guaranteed to be valid sets.

[1] CLUS is available from the authors upon request.

Table 2. Average precision and coverage for all data sets (in percent).

	Precision			Coverage		
Name	Clus(B)	Clus	C4.5H	Clus(B)	Clus	C4.5H
seq	72	61	71	8.35	80.18	14.16
pheno	67	67	68	3.09	3.09	3.26
struc	51	68	58	19.91	29.71	2.05
hom	65	64	55	35.51	81.41	12.06
cellcycle	79	82	54	0.86	0.86	71.34
church	72	75	53	3.50	8.18	58.64
derisi	75	77	61	2.90	2.90	8.39
eisen	84	88	48	5.73	5.73	37.63
gasch1	89	67	38	4.20	15.77	47.24
gasch2	96	96	60	3.09	3.09	64.06
spo	86	79	46	2.29	3.70	12.82
expr	75	77	75	7.26	27.28	5.56

Data Set Partition The experiments are based on a three-way split of each data set: a training set, a validation set and a test set. The test set contains 33% of the data. The remaining 66% are split again using a 66%/33% split to create the training and validation set. We use the same split as is used in [9]. The training set is used to induce the PCT, the validation set is used to remove predicted classes that are not significant and the test set is used to measure the predictive precision and coverage.

Note that there is a large number of classes (250) and that validated predictions are only obtained for some of them. Predictive precision is computed for each predicted class individually (class-wise precision), whereas average precision is computed over all predicted classes. Coverage is defined as the proportion of instances for which at least one class is predicted.

5.3 Results

Table 2 presents the obtained average precision and coverage for each data set. It contains results for Clus(B), the Clus system with Bonferroni correction enabled, for Clus (no Bonferroni correction) and for C4.5H, the hierarchical extension of C4.5.

The average precision obtained by Clus(B) and Clus are generally higher than those obtained with C4.5H (Clus(B) yields a higher precision on 9 data sets and Clus on 10), but this increased precision comes in most cases at the expense of a lower coverage (the coverage obtained with Clus(B) is lower than that of C4.5H in 9 data sets). Note however that domain experts prefer precise predictions over a high coverage in this domain.

By disabling the Bonferroni correction the coverage increases, but not for all data sets. There are 4 data sets where Clus yields a larger coverage than C4.5H: seq, struc, hom, and expr. For 3 of these, the precision obtained by Clus is also higher than that of C4.5H. On the other hand, on the 8 data sets where C4.5H has a higher coverage than Clus, the precision obtained with C4.5H is only higher in one case (pheno).

Table 3 lists the class-wise precision for each data set for Clus and C4.5H. In most cases, the set of classes predicted by Clus and C4.5H is similar. In

Table 3. Class-wise precision. For each predicted class, the prior probability is given together with the precision obtained by CLUS and C4.5H (in percent).

pheno	Prior	CLUS	C4.5H
3/1/3	2		67
30	8	67	
30/1	6	67	69

cellcycle	Prior	CLUS	C4.5H
4	20		34
5	9	82	33
5/1	5	73	64
40	59	91	61
40/3	14	82	57

derisi	Prior	CLUS	C4.5H
2/13	2		63
5	9	76	58
5/1	5	69	54
40	59	88	
40/3	15	76	64

gasch1	Prior	CLUS	C4.5H
1	28	51	50
4	20		29
5	9	86	78
5/1	6	87	83
6/13/1	3		20
40	59	89	
40/3	15	89	43
40/10	20		25

expr	Prior	CLUS	C4.5H
3	17		44
5	9	100	87
5/1	6	93	78
40	59	72	
40/3	15	100	80

struc	Prior	CLUS	C4.5H
8/4	2		73
40	58	68	
67	8		55
67/28	1		44

church	Prior	CLUS	C4.5H
1	28		36
5	9	62	55
5/1	5	56	61
40	59	88	65
40/3	15	76	

eisen	Prior	CLUS	C4.5H
5	12	85	64
5/1	7	79	56
40	74	98	
40/3	19	88	55
40/10	27		39
40/16	12		38

gasch2	Prior	CLUS	C4.5H
2/16	1		0
5	9	95	56
5/1	6	90	100
40	59	100	64
40/3	15	98	56
40/16	9		29

spo	Prior	CLUS	C4.5H
5	9	77	56
5/1	5	72	79
40	59	88	
40/3	14	79	47
40/10	21		26

seq	Prior	CLUS	C4.5H
1	27	56	48
5	9	88	78
5/1	5	83	77
8	13	37	
29	3		81
40	57	60	
40/2	4	50	14
40/3	14	80	81
67	8	66	78

hom	Prior	CLUS	C4.5H
1	27	61	
1/5/1	7		60
4	20	73	
4/5	14	56	
4/5/1	10	41	
4/5/1/4	9	37	
5	9	84	
5/1	5	84	78
6	15	100	
6/13	4	100	
6/13/1	3	100	
8	12	86	
8/4	2	86	100
29	3	64	55
40	57	64	
40/3	14	71	35
40/7	4	100	
40/10	20	64	
40/16	9	64	
67	8	65	88

5 data sets, C4.5H predicts more classes than CLUS (usually one or two extra classes are predicted). CLUS usually yields higher class-wise precisions. These observations are consistent with the higher average precision obtained by CLUS and the larger coverage obtained with C4.5H.

The result most in favor of the CLUS system is obtained on the homology data set. Here, CLUS predicts 19 classes, including one level 4 class. C4.5H predicts 6 classes for this data set. As discussed above, both the average precision and the coverage obtained on this data set is higher for CLUS than for C4.5H. The PCT for this data set is shown in Fig. 2. The bottom-right leaf for example represents a cluster in which the genes are predicted to have two functions: 40/3 and 5/1. Note that the tests in the nodes above the leaf provide a description of the cluster. We have compared our PCT to the rules found in [9]. There are a number of similarities, but the knowledge discovered by both systems can be considered complementary.

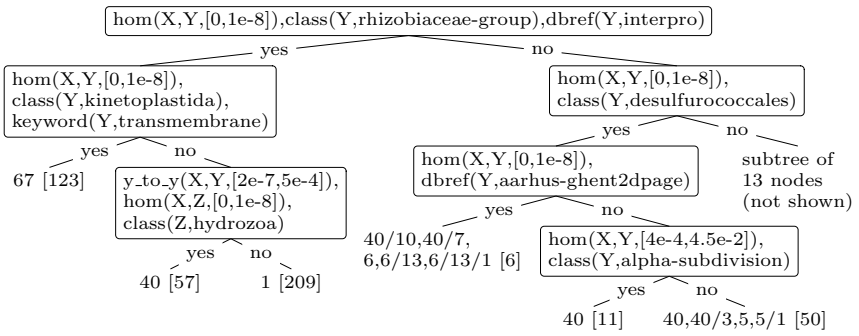

Fig. 2. Part of the PCT obtained for the homology data set. Recall that the attributes are binary relational features. Each node contains such a feature, expressed in first order logic. The variable X represents the given gene and Y and Z refer to homologous genes. Details can be found in [9]. Each leaf of the PCT shows the predicted set of classes together with the number of training examples belonging to the leaf.

6 Further Work

A first item for further work is investigating the trade-off between coverage and precision. Ideally, it should be possible to specify this trade-off by means of a parameter. This is already possible to some extent by altering the significance level used in the validation step, but also the test selection and pruning mechanism of the induction algorithm influence this trade-off. These effects should be studied further.

Ženko et al. [14] propose a system for building predictive clustering rules. By plugging in the distance metric and prototype introduced in this paper, this system will also be suitable for hierarchical multi-classification. Rules are better suited than trees in situations where a high precision is required and a coverage less than 100% can be tolerated.

It would be interesting to evaluate our approach in other domains where hierarchically structured classes occur. E.g., in ecological modeling, samples of soil or river water are collected and the species occurring in these samples are often classified using a hierarchical scheme. Our method could be used to cluster such samples.

7 Conclusions

Predictive clustering trees form a generic framework for prediction that can be instantiated to a particular task by defining the distance metric and prototype. We have introduced such a distance metric and prototype for the task of hierarchical multi-classification. This task occurs in several domains, most notably functional genomics, where each gene is annotated with a set of classes selected from a hierarchical classification scheme.

We have experimentally validated our approach (implemented in the CLUS system) on 12 data sets that are available for the yeast *Saccharomyces cerevisiae* by means of a comparison to C4.5H, a hierarchical extension to C4.5 that has recently been proposed. Our results show that CLUS generates precise predictions. In further work, we will investigate the trade-off between precision and coverage further.

References

1. B. Bakker and T. Heskes. Task clustering for learning to learn. In *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence*, pages 33–40, Amsterdam, 2001.
2. C. M. Bishop. *Neural Networks for Pattern Recognition*. University Press, Oxford, 1999.
3. H. Blockeel, M. Bruynooghe, S. Džeroski, J. Ramon, and J. Struyf. Hierarchical multi-classification. In *Proceedings of the ACM SIGKDD 2002 Workshop on Multi-Relational Data Mining (MRDM 2002)*, pages 21–35, 2002.
4. H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.
5. H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63, 1998.
6. H. Blockeel, S. Džeroski, and J. Grbović. Simultaneous prediction of multiple chemical parameters of river water quality with tilde. In *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, volume 1704 of *Lecture Notes in Artificial Intelligence*, pages 32–40. Springer, 1999.
7. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.
8. R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.
9. A. Clare. *Machine Learning and Data Mining for Yeast Functional Genomics*. PhD thesis, University of Wales, Aberystwyth, 2003.
10. P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, 1996.
11. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann, 1993.
12. J. Ramon and M. Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37:765–780, 2001.
13. L. Todorovski, H. Blockeel, and S. Džeroski. Ranking with predictive clustering trees. In *Proceedings of the 13th European Conference on Machine Learning*, volume 2430 of *Lecture Notes in Artificial Intelligence*, pages 444–455. Springer-Verlag, 2002.
14. B. Ženko, S. Džeroski, and J. Struyf. Learning predictive clustering rules, 2005. Submitted to the Workshop on Knowledge Discovery in Inductive Databases at the 16th European Conference on Machine Learning (ECML).
15. K. Wang, S. Zhou, and S.C. Liew. Building hierarchical classifiers using class proximity. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 363–374. Morgan Kaufmann, 1999.