

**Predicting gene function in *S. cerevisiae*  
and *A. thaliana* using hierarchical  
multi-label decision tree ensembles**

*Leander Schietgat, Celine Vens,  
Jan Struyf, Hendrik Blockeel,  
Dragi Kocev, Saso Dzeroski  
Report CW 528, October 2008*



**Katholieke Universiteit Leuven**  
**Department of Computer Science**

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

# Predicting gene function in *S. cerevisiae* and *A. thaliana* using hierarchical multi-label decision tree ensembles

*Leander Schietgat, Celine Vens,  
Jan Struyf, Hendrik Blockeel,  
Dragi Kocev, Saso Dzeroski*  
*Report CW 528, October 2008*

Department of Computer Science, K.U.Leuven

## Abstract

Motivation: *S. cerevisiae* and *A. thaliana* are two well-studied organisms in biology. Despite the fact that their genomes have already been completed in 1996 and 2000 respectively, the functions of 30% to 40% of their open reading frames (ORFs) remain unclassified. Different machine learning methods have been proposed that annotate the ORFs automatically. However, it is unclear which method is to be preferred in terms of predictive performance, efficiency, interpretability, and usability. Moreover, different evaluation measures for predictive performance have been used in the literature, each showing a limited aspect of the method's performance. Results: We study the usefulness of decision tree based models for predicting the multiple functions of ORFs. First, we describe an algorithm for learning decision trees that can make predictions for the ORFs automatically. We present new results obtained with this algorithm, showing that the trees found by it exhibit clearly better predictive performance than the trees found by previously described methods, while yielding equally interpretable results. The predictive accuracy of our trees, however, is still below that of some recently proposed statistical learning methods. Ensembles of such trees, on the other hand, give even better predictive results, comparable with those of state-of-the-art methods (sometimes better, sometimes worse), while the ensemble method scales much better and is easier to use. We conclude that decision tree based methods are currently the most efficient, easy-to-use, and flexible approach to ORF function prediction, flexible in the sense that they cover the spectrum from maximally interpretable to maximally accurate models. Our evaluation makes use of precision-recall-curves. We argue that this is a better evaluation criterion than previously used criteria. Our evaluation method can be seen as an additional contribution to the field. Availability: The software is freely available on <http://www.cs.kuleuven.be/~dtai/clus/>.

**Keywords :** machine learning, prediction, functional genomics, *S. cerevisiae*, *A. thaliana*

**CR Subject Classification :** I.2.6

---

# Predicting gene function in *S. cerevisiae* and *A. thaliana* using hierarchical multi-label decision tree ensembles

---

Leander Schietgat\*  
Celine Vens\*  
Jan Struyf  
Hendrik Blockeel

LEANDER.SCHIETGAT@CS.KULEUVEN.BE  
CELINE.VENS@CS.KULEUVEN.BE  
JAN.STRUYF@CS.KULEUVEN.BE  
HENDRIK.BLOCKEEL@CS.KULEUVEN.BE

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium

Dragi Kocev  
Sašo Džeroski

DRAGI.KOCEV@IJS.SI  
SASO.DZEROSKI@IJS.SI

Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia

## Abstract

**Motivation:** *S. cerevisiae* and *A. thaliana* are two well-studied organisms in biology. Despite the fact that their genomes have already been completed in 1996 and 2000 respectively, the functions of 30% to 40% of their open reading frames (ORFs) remain unclassified. Different machine learning methods have been proposed that annotate the ORFs automatically. However, it is unclear which method is to be preferred in terms of predictive performance, efficiency, interpretability, and usability. Moreover, different evaluation measures for predictive performance have been used in the literature, each showing a limited aspect of the method's performance.

**Results:** We study the usefulness of decision tree based models for predicting the multiple functions of ORFs. First, we describe an algorithm for learning decision trees that can make predictions for the ORFs automatically. We present new results obtained with this algorithm, showing that the trees found by it exhibit clearly better predictive performance than the trees found by previously described methods, while yielding equally interpretable results. The predictive accuracy of our trees, however, is still below that of some recently proposed statistical learning methods. Ensembles of such trees, on the

other hand, give even better predictive results, comparable with those of state-of-the-art methods (sometimes better, sometimes worse), while the ensemble method scales much better and is easier to use. We conclude that decision tree based methods are currently the most efficient, easy-to-use, and flexible approach to ORF function prediction, flexible in the sense that they cover the spectrum from maximally interpretable to maximally accurate models.

Our evaluation makes use of precision-recall curves. We argue that this is a better evaluation criterion than previously used criteria. Our evaluation method can be seen as an additional contribution to the field.

**Availability:** The software is freely available on <http://www.cs.kuleuven.be/~dtai/clus/>.

**Keywords:** machine learning, prediction, functional genomics, *S. cerevisiae*, *A. thaliana*

## 1. Introduction

The completion of several genome projects in the past decade has generated the full genome sequence of many organisms. Identifying genes in the sequences and assigning biological functions to them has now become a key challenge in modern biology. This last step, which is the focus of our paper, is often guided by automatic discovery processes which interact with the laboratory experiments.

---

\* to whom correspondence should be addressed

More precisely, machine learning techniques are used to predict gene functions from a predefined set of possible functions (e.g., the functions in the Gene Ontology). Afterwards, the predictions with highest confidence can be tested in the lab. There are two characteristics of the function prediction task that distinguish it from common machine learning problems: (1) a single gene may have multiple functions; and (2) the functions are organized in a hierarchy: a gene that is related to some function is automatically related to all its parent functions (this is called the hierarchy constraint). This particular problem setting is known in machine learning as hierarchical multi-label classification (HMC), and recently, a number of approaches have been proposed to deal with it (Clare & King, 2003; Clare et al., 2006; Barutcuoglu et al., 2006; Cesa-Bianchi et al., 2006; Rousu et al., 2006; Vens et al., 2008). These approaches differ with respect to a number of characteristics: which learning algorithm they are based on, whether the hierarchy constraint is always met, and whether they can deal with hierarchies structured as a directed acyclic graph (DAG) e.g., the Gene Ontology, or are restricted to hierarchies structured as a rooted tree e.g., MIPS’s FunCat.

Decision trees are well-known classifiers which can handle large datasets, produce accurate results and are interpretable for domain experts. They have been applied in several machine learning tasks (Quinlan, 1993). In (Vens et al., 2008) it was shown how they can be extended to the HMC setting. An HMC decision tree learner was presented that takes into account the hierarchy constraint and that is able to process DAGs. In this article, we show that this method outperforms previously published results for *S. cerevisiae* and *A. thaliana*. Moreover, we are able to further increase the predictive performance by upgrading our method to an ensemble technique, if the user is willing to (partly) give up on interpretability. Ensemble techniques are learning methods that construct a set of classifiers and classify new data instances by taking a vote over their predictions. Another contribution of this paper is a theoretical investigation of the used evaluation measures for HMC learners. We show that precision-recall curves are much more suited for this type of problems than commonly used measures such as accuracy, precision and ROC curves.

This article is organized as follows. Section 2 discusses related work. Section 3 presents the used algorithms and Sect. 4 contains an extensive experimental evaluation. Section 5 concludes.

## 2. Related work

A number of HMC approaches have been proposed in the area of functional genomics. Several approaches predict functions of unannotated yeast genes based on known functions of genes that are nearby in a protein interaction network (Chen & Dong, 2004; Karaoz et al., 2004; Troyanskaya et al., 2003). This is a form of lazy learning, whereas the focus of this article is on learning global predictive models.

(Lanckriet et al., 2004) represent the data by means of a kernel function and construct support vector machines for each gene function separately. They only predict top-level classes in the hierarchy, and as such, avoid hierarchy constraint issues. (Barutcuoglu et al., 2006) recently presented a two-step approach where unthresholded support vector machines are learned for each gene function separately (allowing violations of the hierarchy constraint), and then combined using a Bayesian learner so that the predictions are consistent with the hierarchical relationships.

The disadvantage of using support vector machines is the lack of interpretability: it is very hard to find out why a support vector machine assigns certain classes to an example, especially if a non-linear kernel is used. Therefore, we concentrate on approaches that learn decision trees.

(Clare, 2003) presents an HMC decision tree method in the context of predicting gene functions of *S. cerevisiae*. She adapts the well-known decision tree algorithm C4.5 (Quinlan, 1993) to cope with the issues introduced by the HMC task. First, where C4.5 normally uses class entropy for choosing the best split, her version uses the sum of entropies of the class variables. Second, she extends the method to predict classes on several levels of the hierarchy, assigning a larger cost to misclassifications higher up in the hierarchy. The resulting tree is transformed into a set of rules, and the best rules are selected, based on a significance test on a validation set. Note that this last step violates the hierarchy constraint, since rules predicting a class can be dropped while rules predicting its subclasses are kept. The non-hierarchical version of her method was later used to predict gene functions for *A. thaliana* (Clare et al., 2006). Here the annotations are considered one level at the time, which also results in violations of the hierarchy constraint.

(Blockeel et al., 2002) present an HMC decision tree method, based on the predictive clustering tree framework (Blockeel et al., 1998). Resulting predictions fulfil the hierarchy constraint. (Blockeel et al., 2006) apply a fine-tuned version of the method on functional

genomics data. (Vens et al., 2008) extend the algorithm towards hierarchies structured as DAGs and show that learning one decision tree for predicting all classes simultaneously, outperforms learning one tree per class (even if those trees are built taking into account the hierarchy).

### 3. Approach

In Sect. 3.1, we discuss the approach that was presented in (Vens et al., 2008) and in Sect. 3.2 we show how we extend it to an ensemble.

#### 3.1. Using predictive clustering trees for HMC tasks

The approach that we present is based on decision trees and is set in the predictive clustering tree (PCT) framework (Blockeel et al., 1998). This framework views a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all training examples, which is recursively partitioned into smaller clusters while moving down the tree. PCTs can be applied to both clustering and prediction tasks.

The PCT framework is implemented in the CLUS system, which is available at <http://www.cs.kuleuven.be/~dtai/clus>.

Before explaining the approach in more detail, we show an example of a (partial) predictive clustering tree predicting the functions of *S. cerevisiae* using homology data from (Clare, 2003) (Fig. 1). The homology features are based on a sequence similarity search for each gene in yeast against all the genes in SwissProt. The functions are taken from the FunCat classification scheme (Mewes et al., 1999). Each internal node of the tree contains a test on one of the attributes in the dataset. Here, the attributes are binary and have been obtained after preprocessing the relational data with a frequent pattern miner. The root node, for instance, tests whether there exists a SwissProt protein that has a high similarity ( $e\text{-value} < 1.0 \cdot 10^{-8}$ ) with the gene under consideration  $G$ , is classified into the rhizobiaceae group and has references to a database called “interpro”. In order to predict the functions of a new gene, the gene is routed down the tree according to the outcome of the tests. When a leaf node is reached, the gene is assigned the functions that are stored in it. Only the most specific functions are shown in the figure. In the rest of this section, we explain how PCTs are constructed. A detailed explanation is given in (Vens et al., 2008).

PCTs (Blockeel et al., 1998) can be constructed with a standard “top-down induction of decision trees”

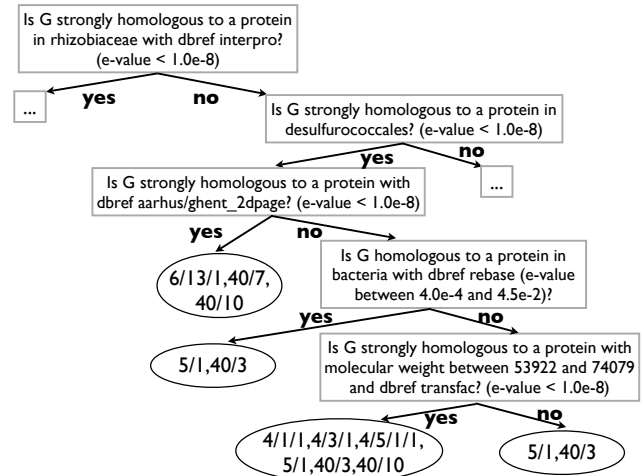


Figure 1. Example of a predictive clustering tree, where the functions of a gene  $G$  are predicted, based on homology data. The functions are taken from the FunCat classification scheme and are hierarchical: if for example function 4/3/1 (tRNA synthesis) is predicted, then function 4/3 (tRNA transcription) and function 4 (transcription) are predicted as well.

(TDIDT) algorithm, similar to CART (Breiman et al., 1984) or C4.5 (Quinlan, 1993). The algorithm takes as input a set of training instances (i.e. the genes and their annotations). It searches for the best acceptable test that can be put in a node. If such a test can be found then the algorithm creates a new internal node and calls itself recursively to construct a subtree for each subset (cluster) in the partition induced by the test on the training instances. To select the best test, the algorithm scores the tests by the reduction in variance (which is to be defined further) they induce on the instances. Maximizing variance reduction maximizes cluster homogeneity and improves predictive performance. If no acceptable test can be found, that is, if no test significantly reduces variance, then the algorithm creates a leaf and labels it with a representative case, or prototype, of the given instances.

To apply PCTs to the task of hierarchical multi-label classification, the variance and prototype are instantiated as follows (Vens et al., 2008).

First, the set of labels of each example is represented as a vector with binary components; the  $i$ 'th component of the vector is 1 if the example belongs to class  $c_i$  and 0 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 distance between

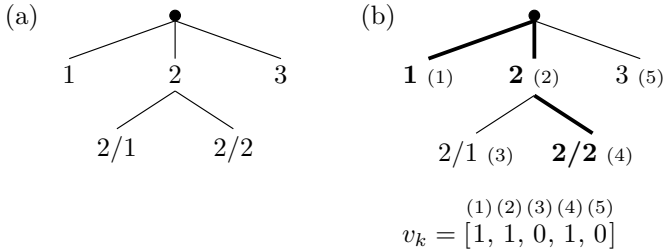


Figure 2. (a) A toy hierarchy. Class label names reflect the position in the hierarchy, e.g., ‘2/1’ is a subclass of ‘2’. (b) The set of classes  $\{1, 2, 2/2\}$ , indicated in bold in the hierarchy, and represented as a vector.

each example’s class vector  $v_k$  and the set’s mean class vector  $\bar{v}$ , i.e.,

$$\text{Var}(S) = \frac{\sum_k d(v_k, \bar{v})^2}{|S|}.$$

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 use a weighted Euclidean distance

$$d(v_1, v_2) = \sqrt{\sum_i w(c_i) \cdot (v_{1,i} - v_{2,i})^2},$$

where  $v_{k,i}$  is the  $i$ ’th component of the class vector  $v_k$  of an instance  $x_k$ , and the class weights  $w(c)$  decrease with the depth of the class in the hierarchy. In our implementation  $w(c) = w_0 \cdot \text{avg } w(p_j(c))$ , where  $p_j(c)$  denotes the  $j$ ’th parent of class  $c$  and  $0 < w_0 < 1$ . Note that our definition of  $w(c)$  allows the classes to be structured in a DAG, as is the case with the Gene Ontology. 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,

$$d([1, 1, 0, 1, 0], [0, 1, 0, 0, 0]) = \sqrt{w_0 + w_0^2}.$$

The heuristic for choosing the best test for a node of the tree is then maximization of the variance reduction as discussed before, with the above definition of variance.

Second, a classification tree 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, the notion of ‘majority class’ does not apply in a straightforward manner. Instead, the mean  $\bar{v}$  of the class vectors of the examples in that leaf is stored. Recall that

$\bar{v}_i$  is the proportion of examples in the leaf belonging to  $c_i$ . An example arriving in the leaf can therefore be predicted to belong to class  $c_i$  if  $\bar{v}_i$  is above some threshold  $t_i$ , which can be chosen by a domain expert. To ensure that the predictions fulfil the hierarchy constraint (whenever a class is predicted its superclasses are also predicted), it suffices to choose  $t_i \leq t_j$  whenever  $c_i$  is a superclass of  $c_j$ . The PCT that is shown in Fig. 1 has a threshold of  $t_i = 0.4$  for all  $i$ .

We call the resulting instantiation of the PCT algorithm in the CLUS system CLUS-HMC.

### 3.2. Ensembles of PCTs

Ensemble methods are learning methods that construct a set of classifiers for a given prediction task and classify new examples by combining the predictions of each classifier. In this paper, we consider bagging, an ensemble learning technique that has primarily been used in the context of decision trees.

Bagging (Breiman, 1996a) is an ensemble method where the different classifiers are constructed by making bootstrap replicates of the training set and using each of these replicates to construct one classifier. Each bootstrap sample is obtained by randomly sampling training instances, with replacement, from the original training set, until an equal number of instances is obtained. The individual predictions given by each classifier can be combined by taking the average (for numeric targets) or the majority vote (for nominal targets). (Breiman, 1996a) has shown that bagging can give substantial gains in predictive performance of decision tree learners. Also in the case of learning PCTs for predicting multiple targets at once, decision tree methods benefit from the application of bagging (Kocev et al., 2007).

The algorithm for bagging the PCTs takes an extra parameter  $k$  as input that denotes the number of trees in the ensemble. In order to make predictions, the average of all class vectors predicted by the  $k$  trees in the ensemble is computed, and then the threshold is applied as before. This ensures that the hierarchy constraint holds.

Remark that, by introducing bagging in the PCT algorithm, the interpretability is significantly decreased: instead of one interpretable tree, we obtain a collection of trees whose predictions are averaged. Thus, there is a clear trade-off between predictive performance and interpretability to be considered by the user. However, methods exist that overcome the comprehensibility issue of ensembles. For example, (Van Assche & Blockeel, 2007) learn a single decision tree that ap-

proximates an ensemble of trees. The resulting tree is more accurate than a single tree learned directly from the data (but less accurate than the ensemble).

We call the resulting instantiation of the bagging algorithm around the CLUS-HMC algorithm CLUS-HMC-ENS.

## 4. Experimental evaluation

In this section, we want to answer the following questions:

1. How well does CLUS-HMC perform on functional genomics data and what is the improvement that can be obtained by using CLUS-HMC-ENS? Does giving up in interpretability pay off?
2. How does the predictive performance of the proposed algorithms compare to results reported in the biomedical literature?

In order to answer the second question, we compare our results to the results reported by (Clare & King, 2003) and (Barutcuoglu et al., 2006) on *S. cerevisiae*, and by (Clare et al., 2006) on *A. thaliana*. The methods used in these studies were discussed in Sect. 2. The datasets that we use in our evaluation are exactly those datasets that are used in the mentioned articles.

### 4.1. Datasets

Next to predicting gene functions of two organisms (*S. cerevisiae* and *A. thaliana*), we consider two annotation schemes in our evaluation: FunCat (developed by MIPS (Mewes et al., 1999)), which is a tree-structured class hierarchy and the Gene Ontology (GO) (Ashburner et al., 2000), which forms a directed acyclic graph instead of a tree: each term can have multiple parents (GO's "is-a" relationship between terms is used).

#### 4.1.1. SACCHAROMYCES CEREVISIAE.

The first dataset we use ( $\mathbf{D}_0$ ) is described by (Barutcuoglu et al., 2006) and is available from these authors upon request.

$\mathbf{D}_0$  is a combination of different data sources. The input feature vector for a gene consists of pairwise interaction information, membership to colocalization locale, possession of transcription factor binding sites, and results from microarray experiments. Features containing less than two non-zero entries were removed, resulting in a dataset with 5930 features. The 3465 genes are annotated with function terms from a

subset of 105 nodes from the Gene Ontology's *biological process* hierarchy.

We also use the 12 yeast datasets ( $\mathbf{D}_1 - \mathbf{D}_{12}$ ) from (Clare, 2003).<sup>1</sup> The datasets describe different aspects of the genes in the yeast genome. They include five types of bioinformatics data: sequence statistics, phenotype, secondary structure, homology, and expression. The different sources of data highlight different aspects of gene function. The genes are annotated with functions from the FunCat classification schemes. Only annotations from the first four levels are given.

$\mathbf{D}_1$  (seq) records sequence statistics that depend on the amino acid sequence of the protein for which the gene codes. These include amino acid ratios, sequence length, molecular weight and hydrophobicity.

$\mathbf{D}_2$  (pheno) contains phenotype data, which represents the growth or lack of growth of knock-out mutants that are missing the gene in question. The 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.

$\mathbf{D}_3$  (struc) stores features computed from the secondary structure of the yeast proteins. The secondary structure is not known for all yeast genes; however, it can be predicted from the protein sequence with reasonable accuracy, using Prof (Ouali & King, 2000). Due to the relational nature of secondary structure data, Clare performed a preprocessing step of relational frequent pattern mining;  $\mathbf{D}_3$  includes the constructed patterns as binary attributes.

$\mathbf{D}_4$  (hom) includes for each yeast gene, information from other, homologous genes. Homology is usually determined by sequence similarity. PSI-BLAST (Altschul et al., 1997) was used to compare yeast genes both with other yeast genes, and with all genes indexed in SwissProt v39. This provided for each yeast gene, a list of homologous genes. For each of these, various properties were extracted (keywords, sequence length, names of databases they are listed in, ...). Clare pre-processed this data in a similar way as the secondary structure data, to produce binary attributes.

$\mathbf{D}_5, \dots, \mathbf{D}_{12}$ . Many microarray datasets exist for yeast, and several of these were used. Attributes for these datasets are real valued, representing fold changes in expression levels.

<sup>1</sup>The datasets can be downloaded from <http://www.aber.ac.uk/compsci/Research/bio/dss/>.

#### 4.1.2. ARABIDOPSIS THALIANA.

We use six datasets from (Clare et al., 2006), originating from different sources: sequence statistics, expression, predicted SCOP class, predicted secondary structure, InterPro and homology. The datasets can be downloaded from the same url. Each dataset comes in two versions: with annotations from the FunCat classification scheme and from the Gene Ontology’s *biological process* hierarchy. Again, only annotations for the first four levels are given. We use the manual annotations for both schemes.

**D<sub>13</sub>** (seq) records sequence statistics in exactly the same way as for *S. cerevisiae*. **D<sub>14</sub>** (exprindiv) contains 43 experiments from NASC’s Affymetrix service “Affywatch”<sup>2</sup>, taking the signal, detection call and detection p-values. **D<sub>15</sub>** (scop) consists of SCOP superfamily class predictions made by the Superfamily server (Gough et al., 2001). **D<sub>16</sub>** (struc) was obtained in the same way as for *S. cerevisiae*. **D<sub>17</sub>** (interpro) includes features from several motif or signature finding databases, like PROSITE, PRINTS, Pfam, ProDom, SMART and TIGRFAMs, calculated using the EBI’s stand-alone InterProScan package (Zdobnov & Apweiler, 2001). To obtain features, the relational data was mined in the same manner as the structure data. **D<sub>18</sub>** (hom) was obtained in the same way as for *S. cerevisiae*, but now using SWISSPROT v41.

## 4.2. Methodology

### 4.2.1. EVALUATION MEASURE.

We will report our results with precision-recall curves. Precision is the probability that a positive prediction is correct, and recall is the probability that a positive instance is predicted positive. Remember that every leaf in the tree contains a vector  $\bar{v}$  with for each function the probability that the gene has this function. When decreasing CLUS-HMC’s prediction threshold  $t_i$  from 1 to 0, an increasing number of instances is predicted as belonging to class  $c_i$ , causing the recall for  $c_i$  to increase whereas precision may increase or decrease (with normally a tendency to decrease). Thus, a tree (or ensemble) with specified threshold has a single precision and recall, and by varying the threshold a precision-recall curve (PR curve) is obtained. Such curves allow us to evaluate the predictive performance of a model regardless of  $t$ . In the end, a domain expert can choose a threshold according to the point on the curve which is most interesting to him.

Our decision to conduct a precision-recall based eval-

---

<sup>2</sup>NASC Affywatch: <http://affymetrix.arabidopsis.info/AffyWatch.html>

uation is based on two observations: (1) it allows us to exploit the fact that our algorithms output class probabilities, which means that we can evaluate them regardless of any prediction threshold, and (2) it suits the characteristics of typical HMC datasets, in which it is often the case that individual classes have few positive instances (i.e. typically only a few genes have a particular function). This implies that for most classes, the number of negative instances by far exceeds the number of positive instances. We are more interested in recognizing the positive instances (i.e. that a gene has a given function), rather than correctly predicting the negative ones (i.e. that a gene does not have a particular function). Although ROC curves are better known, they are less suited for this task, exactly because they reward a learner if it correctly predicts negative instances (giving rise to a low false positive rate). This can present an overly optimistic view of the algorithm’s performance (Davis & Goadrich, 2006).

With hundreds of 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 transforming the multi-label problem into a binary single-label one, i.e. by counting instance-class-couples instead of instances (Vens et al., 2008). An instance-class couple is (predicted) positive if the instance has (is predicted to have) that class, it is (predicted) negative otherwise. The definition of precision and recall is then as before. Note that a domain expert can still choose an appropriate threshold for an average PR curve.

### 4.2.2. PARAMETER INSTANTIATIONS FOR CLUS-HMC AND CLUS-HMC-ENS.

In our implementation,  $w_0$  is set to 0.75 and the parameter  $k$ , which denotes the number of trees used in the ensemble, was set to 50.

## 4.3. Results

In this section we will first check if ensembles improve the predictive performance of CLUS-HMC in gene function prediction and if so, quantify this difference. Then, we will evaluate CLUS-HMC and CLUS-HMC-ENS against two state-of-the-art systems in gene function prediction. On the one hand, we will compare CLUS-HMC to C4.5H, because they both build a single decision tree which is easy to interpret, and then, we will compare CLUS-HMC-ENS to Bayesian-corrected SVMs, because both models give up on interpretability in order to reach maximum performance.

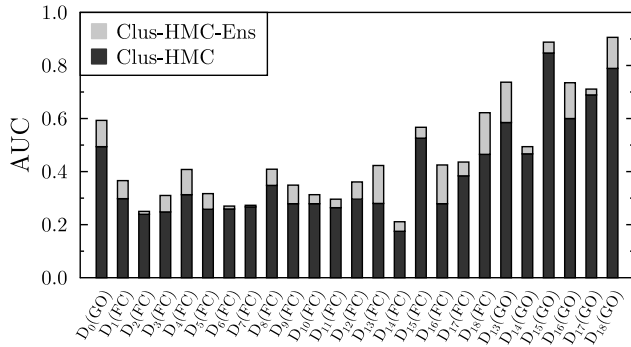


Figure 3. Area under PR curves for the CLUS-HMC and CLUS-HMC-ENS algorithms. The grey surface represents the gain in AUC obtained by CLUS-HMC-ENS.

#### 4.3.1. COMPARISON BETWEEN CLUS-HMC AND CLUS-HMC-ENS.

The datasets originating from (Clare & King, 2003; ?) (i.e., datasets  $D_1$  to  $D_{18}$ ) are divided into a training set (2/3) and a test set (1/3). We use exactly the same splits. For dataset  $D_0$  we randomly construct a training and test set with the same ratio.

For each of the datasets, the area under the curve (AUC) of the average PR curves of CLUS-HMC and CLUS-HMC-ENS is shown in Fig. 3. We see that for every dataset, there is an increase in AUC when using ensembles. The average gain in AUC is 0.070; the maximal gain is 0.157. Representative PR curves can be found in Fig. 5 (left) and 6. Hence, as expected, the improvement in predictive performance that is obtained by using ensembles carries over to the HMC setting.

#### 4.3.2. COMPARISON WITH C4.5H/M.

We now concentrate on the comparison of the results obtained by our algorithms to those obtained by other decision tree-based algorithms. For the datasets that are annotated with FunCat classes ( $D_1 - D_{18}$ ), we will compare to the hierarchical extension of C4.5 (Clare & King, 2003), which we will refer to as C4.5H. For the datasets with GO annotations ( $D_{13} - D_{18}$ ), we will use the non-hierarchical version (Clare et al., 2006), as C4.5H cannot handle hierarchies structured as a DAG. We refer to this system as C4.5M.

For their experiments on *A. thaliana*, (Clare et al., 2006) only report results per level of the hierarchy. In order to obtain these results, they learn a separate classifier per level, removing from their training and test set those genes that do not have annotated functions at that level. In our opinion, this gives a biased

result: when annotating a new gene, it is not known in advance at which levels of the hierarchy it will have functions. Therefore, we reran C4.5H/M to learn one classifier that uses all training data and tested it on the complete test set.

For evaluating their systems, (Clare et al., 2006) report average precision. Indeed, as the biological experiments required to validate the learned rules are costly, it is important to avoid false positives. However, precision is always traded off by recall: a classifier that predicts one example positive, but misses 1000 other positive examples may have a precision of 1, although it can hardly be called a good classifier. Therefore, we also computed the average recall of the models obtained by C4.5H/M. These models were presented as rules derived from the trees, which enables us to plot only one point in PR space.

For each of the datasets  $D_1 - D_{18}$ , these PR points are plotted against the average PR curves for CLUS-HMC. As we are comparing curves with points, we speak of a “win” for CLUS-HMC when its curve is above C4.5H/M’s point, and of a “loss” when it is below the point. Under the null hypothesis that both systems perform equally well, we expect as many wins as losses. We observed that only in one case out of 24, for dataset  $D_{16}$  with FunCat annotations, C4.5H/M outperforms CLUS-HMC. For all other cases there is a clear win for CLUS-HMC. In order to illustrate the gain that is obtainable by ensembles, we also included the results for CLUS-HMC-ENS. Representative PR curves can be found in Fig. 5 (left) and 6.

For each of these datasets, we also compared the precision of C4.5H/M, CLUS-HMC and CLUS-HMC-ENS, at the recall obtained by C4.5H/M. The results can be found in Fig. 4. The average gain in precision w.r.t. C4.5H/M is 0.209 for CLUS-HMC and 0.276 for CLUS-HMC-ENS.

We can conclude that if interpretable models are to be obtained, CLUS-HMC is the system that yields the best predictive performance. Compared with other existing methods, we are able to obtain the same precision with higher recall, or the same recall with higher precision. Moreover, the hierarchy constraint is always fulfilled, which is not the case for C4.5H/M.

*Comparing individual rules.* Every leaf of a decision tree corresponds to an *if ... then ...* rule. When comparing the interpretability and precision/recall of these individual rules, CLUS-HMC also performs well. For instance, take FunCat class 29, with a prior frequency of 3%. Figure 5 (right) shows the PR evaluation for the algorithms for this class using homology dataset

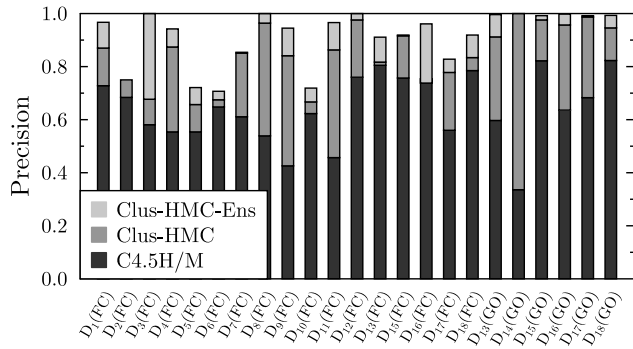


Figure 4. Precision of the C4.5H/M, CLUS-HMC and CLUS-HMC-ENS algorithms, at the recall obtained by C4.5H/M. The dark grey surface represents the gain in precision obtained by CLUS-HMC, the light grey surface represents the gain for CLUS-HMC-ENS.  $D_{14}(FC)$  was not included, since C4.5H did not find significant rules. For  $D_{16}(FC)$ , C4.5H scored a slightly better precision (see Fig. 6), hence the lack of dark grey surface.

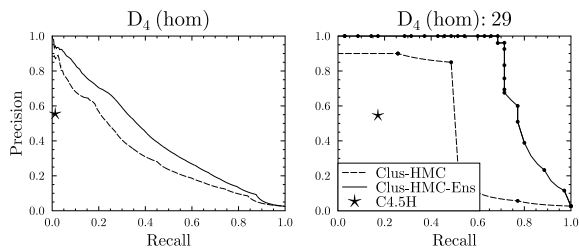


Figure 5. Left: Average precision/recall over all classes for C4.5H, CLUS-HMC and CLUS-HMC-ENS on  $D_4$  with FunCat annotations. Right: Precision-recall curve for class 29 on  $D_4$  with FunCat annotations.

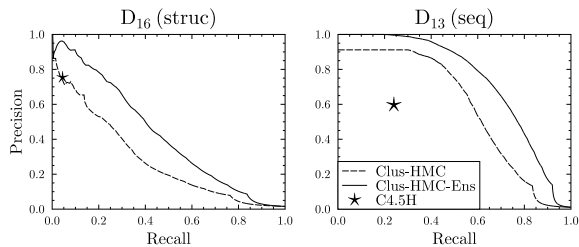


Figure 6. Left: Average precision/recall over all classes for C4.5H, CLUS-HMC and CLUS-HMC-ENS on  $D_{16}$  with FunCat annotations. Right: Average curve for C4.5M, CLUS-HMC and CLUS-HMC-ENS on  $D_{13}$  with GO annotations.

```

if the ORF is NOT homologous to another yeast protein ( $e \geq 0.73$ )
and homologous to a protein in rhodospirillaceae ( $e < 1.0 \cdot 10^{-8}$ )
and NOT homologous to another yeast protein ( $5.0 \cdot 10^{-4} < e < 3.3 \cdot 10^{-2}$ )
and homologous to a protein in anabaena ( $e \geq 1.1$ )
and homologous to another yeast protein ( $2.0 \cdot 10^{-7} < e < 5.0 \cdot 10^{-4}$ )
and homologous to a protein in beta_subdivision ( $e < 1.0 \cdot 10^{-8}$ )
and NOT homologous to a protein in sinorhizobium with keyword transmembrane ( $e \geq 1.1$ )
and NOT homologous to a protein in entomopoxvirinae with dbref pir ( $e \geq 1.1$ )
and NOT homologous to a protein in t4-like_phages with molecular weight between 1485 and 38502 ( $4.5 \cdot 10^{-2} < e < 1.1$ )
and NOT homologous to a protein in chroococcales with dbref prints ( $1.0 \cdot 10^{-8} < e < 4.0 \cdot 10^{-4}$ )
and NOT homologous to a protein with sequence length between 344 and 483 and dbref tigr ( $e < 1.0 \cdot 10^{-8}$ )
and homologous to a protein in beta_subdivision with sequence length between 16 and 344 ( $e < 1.0 \cdot 10^{-8}$ )
then class 29/0/0/0 "transposable elements, viral and plasmid proteins"
    
```

Figure 7. Rule found by C4.5H on the  $D_4$  homology dataset.

```

if the ORF is NOT homologous to a protein in rhizobiaceae_group with dbref interpro ( $e < 1.0 \cdot 10^{-8}$ )
and NOT homologous to a protein in desulfurococcales ( $e < 1.0 \cdot 10^{-8}$ )
and homologous to a protein in ascomycota with dbref transfac ( $e < 1.0 \cdot 10^{-8}$ )
and homologous to a protein in viridiplantae with sequence length  $\geq 970$  ( $e < 1.0 \cdot 10^{-8}$ )
and homologous to a protein in rhizobium with keyword plasmid ( $1.0 \cdot 10^{-8} < e < 4.0 \cdot 10^{-4}$ )
and homologous to a protein in nicotiana with dbref interpro ( $e < 1.0 \cdot 10^{-8}$ )
then class 29/0/0/0 "transposable elements, viral and plasmid proteins"
    
```

Figure 8. Rule found by CLUS-HMC on the  $D_4$  homology dataset.

$D_4$ . The PR point for C4.5H corresponds to one rule, shown in Fig. 7. This rule has a precision/recall of 0.55/0.17. CLUS-HMC’s most precise rule for 29 is shown in Fig. 8. This rule has a precision/recall of 0.90/0.26.

Note from Fig. 5 (right) that an even higher precision can be obtained with CLUS-HMC-ENS, although the rules which lead to this prediction are more complex.

### 4.3.3. COMPARISON WITH BAYESIAN-CORRECTED SVMs.

(Barutcuoglu et al., 2006) have used dataset  $D_0$  to evaluate their method, which consists of Bayesian-corrected SVMs (see Sect. 2). We will further refer to this method as BSVM. They report classwise area under the ROC curve for 105 nodes of the Gene Ontology. Although precision-recall based evaluation is more suited for HMC problems, we adopt the same evaluation metric for this comparison.

(Barutcuoglu et al., 2006) build a bagging procedure around their system and report out-of-bag error estimates (Breiman, 1996b) as evaluation, which removes the need for a set-aside test set. Out-of-bag error estimation proceeds as follows: for each example in the

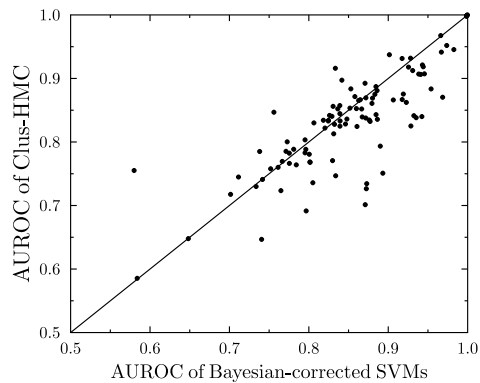


Figure 9. Class-wise out-of-bag AUROC comparison between CLUS-HMC-ENS and Bayesian-corrected SVMs.

original training set, the predictions are made by aggregating only over those classifiers for which the example was not used for training. This is the out-of-bag classifier. The out-of-bag error estimate is then the error rate of the out-of-bag classifier on the training set.

Fig. 9 compares the classwise out-of-bag AUROC estimates for CLUS-HMC-ENS and BSVM outputs. BSVM scored better on 66 of the 105 functions, while CLUS-HMC-ENS scored better on the remaining 39 cases. Thus, while for a randomly chosen function BSVM has the highest probability of performing best ( $p = 0.0002$ ), it is advisable to try both approaches whenever possible, as either one may give the most accurate model for that particular function. Among these two approaches, CLUS-HMC-ENS is the most scalable one: it could be run on the full hierarchy, while BSVM could only model a small part of the GO hierarchy (in these experiments, 105 nodes). It is also more efficient.<sup>3</sup> The trees in the ensemble, or the single tree found by CLUS-HMC, are easy to interpret and give insight in how the model makes its predictions. Finally, the software producing them is easy to use and does not require deep knowledge of the methods used to produce the results.

<sup>3</sup>Run on a cluster of AMD Opteron processors (1.8 - 2.4GHz, >2GB RAM) on dataset  $D_{16}$  (which contains the full hierarchy instead of a subset) having 14804 attributes and 7778 examples annotated with 629 different classes, CLUS-HMC-ENS required 34.8 hours while BSVM required 190.5 hours for learning the model (CLUS-HMC-ENS is a factor 5.5 faster).

## 5. Conclusions

In this article, we have presented a state-of-the-art HMC decision tree learner for functional genomics. The contributions are the use of ensembles of such trees, the introduction of precision-recall curves for classification in functional genomics and an experimental comparison between our method and two state-of-the-art learners in functional genomics.

Precision-recall curves give the domain expert more insight into the relation between precision and recall. We showed that PR-based evaluation measures are best suited for HMC problems, since they do not reward the negative predictions, i.e. predicting an example not having particular labels (like ROC curves do).

We showed that CLUS-HMC outperforms an existing decision tree learner (C4.5H) w.r.t. predictive performance, while still keeping interpretability. Moreover, it is possible to maximize predictive performance by constructing an ensemble of CLUS-HMC-trees. We show that the latter can produce competitive results with an approach based on SVMs, while still being efficient and easy to use.

## Acknowledgements

Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) to L.S. Research Fund K.U.Leuven to C.V and J.S. Research Foundation - Flanders (FWO-Vlaanderen) to H.B. The EU funded project IQ (Inductive Queries for Mining Patterns and Models). The authors thank Amanda Clare and Zafer Barutcuoglu for their cooperation. This research was conducted utilizing high performance computational resources provided by the K.U.Leuven, <http://ludit.kuleuven.be/hpc>.

## References

- Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids. Res.*, *25*, 3389–3402.
- Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J., Harris, M., Hill, D., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J., Richardson, J., Ringwald, M., Rubin, G., & Sherlock, G. (2000). Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genet.*, *25*, 25–29.

- Barutcuoglu, Z., Schapire, R., & Troyanskaya, O. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, *22*, 830–836.
- Blockeel, H., Bruynooghe, M., Džeroski, S., Ramon, J., & Struyf, J. (2002). Hierarchical multi-classification. *Proceedings of the ACM SIGKDD 2002 Workshop on Multi-Relational Data Mining (MRDM 2002)* (pp. 21–35).
- Blockeel, H., De Raedt, L., & Ramon, J. (1998). Top-down induction of clustering trees. *Proceedings of the 15th International Conference on Machine Learning* (pp. 55–63).
- Blockeel, H., Schietgat, L., Struyf, J., Džeroski, S., & Clare, A. (2006). Decision trees for hierarchical multilabel classification: A case study in functional genomics. *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases* (pp. 18–29).
- Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *24*, 123–140.
- Breiman, L. (1996b). Out-of-bag estimation. [ftp.stat.berkeley.edu/pub/users/breiman/OOBestimation.ps](http://ftp.stat.berkeley.edu/pub/users/breiman/OOBestimation.ps).
- Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Belmont: Wadsworth.
- Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, *7*, 31–54.
- Chen, Y., & Dong, X. (2004). Global protein function annotation through mining genome-scale data in yeast *saccharomyces cerevisiae*. *Nucleic Acids Research*, *32*, 6414–6424.
- Clare, A. (2003). *Machine learning and data mining for yeast functional genomics*. Doctoral dissertation, University of Wales, Aberystwyth.
- Clare, A., Karwath, A., Ougham, H., & King, R. D. (2006). Functional bioinformatics for *Arabidopsis thaliana*. *Bioinformatics*, *22*, 1130–1136.
- Clare, A., & King, R. D. (2003). Predicting gene function in *Saccharomyces cerevisiae*. *Bioinformatics*, *19*, ii42–49.
- Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning* (pp. 233–240).
- Gough, J., Karplus, K., Hughey, R., & Chothia, C. (2001). Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure. *Molecular Biology*, *313*, 903–919.
- Karaoz, U., Murali, T., Letovsky, S., Zheng, Y., Ding, C., Cantor, C., & Kasif, S. (2004). Whole-genome annotation by using evidence integration in functional-linkage networks. *Proceedings of the National Academy of Sciences*, *101*, 2888–2893.
- Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2007). Ensembles of multi-objective decision trees. *Proceedings of the 18th European Conference on Machine Learning* (pp. 624–631). Springer.
- Lanckriet, G. R., Deng, M., Cristianini, N., Jordan, M. I., & Noble, W. S. (2004). Kernel-based data fusion and its application to protein function prediction in yeast. *Proceedings of the Pacific Symposium on Biocomputing* (pp. 300–311).
- Mewes, H., Heumann, K., Kaps, A., Mayer, K., Pfeiffer, F., Stocker, S., & Frishman, D. (1999). MIPS: A database for protein sequences and complete genomes. *Nucleic Acids Research*, *27*, 44–48.
- Ouali, M., & King, R. (2000). Cascaded multiple classifiers for secondary structure prediction. *Protein Science*, *9*(6), 1162–76.
- Quinlan, J. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann.
- Rousu, J., Saunders, C., Szedmak, S., & Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, *7*, 1601–1626.
- Troyanskaya, O., Dolinski, K., Owen, A., Altman, R., & D., B. (2003). A bayesian framework for combining heterogeneous data sources for gene function prediction (in *saccharomyces cerevisiae*). *Proceedings of the National Academy of Sciences*, *100*, 8348–8353.
- Van Assche, A., & Blockeel, H. (2007). Seeing the forest through the trees: Learning a comprehensible model from an ensemble. *Proceedings of The 18th European Conference on Machine Learning* (pp. 418–429). Springer.
- Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, *73*, 185–214.

Zdobnov, E., & Apweiler, R. (2001). Interproscan - an integration platform for the signature-recognition methods in interpro. *Bioinformatics*, *17*, 847–848.