

Hierarchical Multilabel Classification Trees for Gene Function Prediction (Extended Abstract)

Hendrik Blockeel¹, Leander Schietgat¹, Jan Struyf^{1,4},
Amanda Clare², Sašo Džeroski³

¹ Dept. of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, 3001 Leuven, Belgium

² Dept. of Computer Science, University of Wales, Aberystwyth,
SY23 3DB, UK

³ Dept. of Knowledge Technologies, Jožef Stefan Institute,
Jamova 39, 1000 Ljubljana, Slovenia

⁴ Dept. of Computer Science, University of Wisconsin,
1210 West Dayton, Madison, WI 53706, USA

Abstract

Prediction of gene function is a so-called hierarchical multilabel classification (HMC) task: a single instance can be labelled with multiple classes rather than just one (i.e., a gene can have multiple functions), and these classes are organized in a hierarchy. Many machine learning methods focus on learning predictive models with a single target variable. One can then learn to predict all classes separately and combine the predictions afterwards. An alternative is to upgrade these methods towards the HMC context. In this paper we explore this alternative for classification trees. A comparison of learning HMC trees with learning normal classification trees shows that the former has clear advantages with respect to accuracy, efficiency, and interpretability. It seems worth investigating to what extent these results carry over to other machine learning methods.

1 Introduction

Gene function prediction is an example of what is known in machine learning as *hierarchical multilabel classification* (HMC): in contrast to normal classification, where a function is learned that maps a data instance to one class (from a pre-defined set of classes), a function needs to be learned that maps a data instance to multiple classes (a subset of the entire class set). In addition, the classes are organized in a hierarchy, imposing the constraint that when an instance is assigned to some class it should also be assigned to all its superclasses.

HMC has received some attention in the machine learning community the last few years, but still much less than normal classification, and mostly in the area of text classification [10]. Given that gene function prediction is also a HMC task, it seems natural to try to use existing HMC methods for gene function prediction. Only a few authors have taken such an approach [2, 6, 1]. Some of them [2, 6] propose to use classification trees, which have the advantage of yielding interpretable rules; this is the kind of approach we will study here. Standard tree learners such as CART [4] or C4.5 [9] assume a single target variable, but adaptations have been proposed that can handle multiple target variables [3, 6]. However, it has not yet been studied how these HMC-tree learners perform when compared with the straightforward approach of learning a separate tree for each class.

In this work we compare the performance of a single HMC tree, predicting the membership of 250 classes together, to the performance of a set of 250 single-classification (SC) trees, each predicting the membership of one single class. The results are surprising: learning a single HMC tree turns out to outperform learning a 250 SC trees on all dimensions: efficiency, interpretability, but also (and this is less expected) accuracy. We conjecture that the latter happens because different classes are correlated, thus they carry information on each other, improving the signal-to-noise ratio in the data.

2 Methods

We use Clus, a state-of-the-art decision tree learner based on the principles in [3] that can also learn HMC trees. A detailed algorithmic description of Clus can be obtained from the authors upon request; here we explain the basic principles of the method.

In the normal classification setting, decision tree learners work as follows. The algorithm considers the whole training set and looks for a “maximally informative” attribute in the example descriptions. The training set is then partitioned according to the values of this attribute. A “maximally informative” attribute means that the created subsets are on average maximally homogeneous with respect to the classes of the examples they contain. For a subset that is not entirely homogeneous, the same procedure is repeated: the most informative attribute for the examples within this subset is sought, the subset is partitioned according to it, etc. This continues until each subset contains examples of only one class, or some other stopping criterion is fulfilled (e.g., further partitioning will be statistically unreliable). The hierarchy of subsets thus created, together with the attributes defining these subsets, is a decision tree. The tree can be used for classification by sorting an instance into a leaf of the tree based on the attributes tested by the tree, and assigning to it the class that occurs most frequently in that leaf.

In [3] it is explained how a similar procedure can be used when there are multiple target variables. Informativeness is then measured as reduction of variance, where variance is defined as the mean squared distance between an

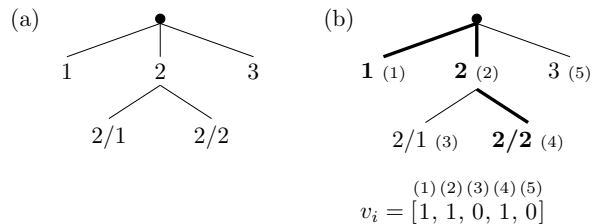


Figure 1: (a) A toy hierarchy. Class label names reflect the position in the hierarchy, e.g., ‘2/1’ is a subclass of ‘2’. (b) The set of classes $\{1, 2, 2/2\}$, indicated in bold in the hierarchy, and represented as a vector.

example and the mean of the subset it belongs to, and the notion of “distance” is left to be defined by the user. In [2] it is discussed how to instantiate this procedure for HMC. The multiple target variables then indicate class membership (variable v_i is 1 if an instance belongs to class c_i and 0 otherwise; we call the vector of all v_i the class vector). Several distance measures can be used; we here choose a weighted Euclidean distance between the class vectors, with the weight decreasing exponentially with the depth of the class in the hierarchy (reflecting that errors higher up in the hierarchy are considered more costly). Consider for example the class hierarchy shown in Fig. 1, 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, $d(x_1, x_2) = d_{\text{Euclidean}}([1, 1, 0, 1, 0], [0, 1, 0, 0, 0]) = \sqrt{w_0 + w_0^2}$, with $w_i = w_0^{\text{depth}(c_i)}$.

The heuristic is a first point of difference of Clus as compared to standard tree learners. A second point is deciding, once an instance has been sorted into a leaf, what classes to assign to it. The standard procedure is to assign the class that is most frequent in a leaf, but in our case we may have to assign multiple classes. It is then reasonable to predict that an instance belongs to some class c_i if the proportion of c_i -examples in the leaf, call this p_i , exceeds some threshold t_i . Clus does not fix t_i in advance, but stores the proportions p_i in the trees that it learns, leaving the thresholds to be determined afterwards (see also experiments).

Finally, Clus uses a stopping criterion inspired by that of normal tree learners: it uses a parameter *mincases* that specifies the minimal number of instances that a leaf needs to contain, and a statistical *F*-test that checks whether the reduction of variance obtained with a split is “significant” at a certain significance level (which is also a parameter of the system).

In the following, Clus-HMC refers to Clus used as a hierarchical multilabel classification system, learning trees that predict all 250 classes at once with the weighted Euclidean distance mentioned before. Clus-SC refers to the approach of learning a single classification tree for each class separately; this version reduces roughly to the CART decision tree learner.

```

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

```

Figure 2: A small part of the hierarchical FunCat classification scheme.

3 Data and experimental setup

We use 12 datasets from <http://www.aber.ac.uk/compsci/Research/bio/dss/yeastdata/> [5]. The different datasets describe different aspects of the genes in the genome of *Saccharomyces cerevisiae*. Five types of bioinformatic data 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.

The number of examples in each dataset ranges from 1592 to 3932, the number of attributes from 27 to 47034. Each gene in the datasets is annotated with one or more classes selected from the FunCat hierarchical classification scheme from the Munich Information Center for Protein Sequences (MIPS) as available on 4/24/2002. The hierarchy has 250 classes distributed over 4 levels. A small part of it is shown in Fig. 2.

Each of the 12 datasets was split in 2/3 training data and 1/3 test data; Clus-HMC and Clus-SC were trained on the training data and evaluated on the test data. Both were used with default parameters except for a parameter f (the significance level for the statistical F -test), which affects the size and accuracy of the trees that are learned; f was optimized in both cases in exactly the same way, using a validation set internal to the training set.

We compared the size of the trees, the time to learn them, and their predictive performance. The latter was evaluated using precision-recall curves [7], as the classical accuracy measure is not very suitable for strongly skewed class distributions. Precision is the probability that a positive prediction is correct, and recall the probability that a positive instance is predicted positive. An instance-class couple is (predicted) positive if the instance has (is predicted to have) that class. We use a common threshold t for all classes. When decreasing t from 1 to 0, an increasing number of instance-class couples is predicted, yielding a curve in the precision-recall space. The higher this curve, the better the predictive performance.

4 Results

Averaged over all 250 classes, the HMC curve was almost consistently above the SC curve for all 12 datasets. Figure 3 shows some representative precision-recall curves. Moreover, HMC trees contained on average 24 nodes, SC trees 33. This

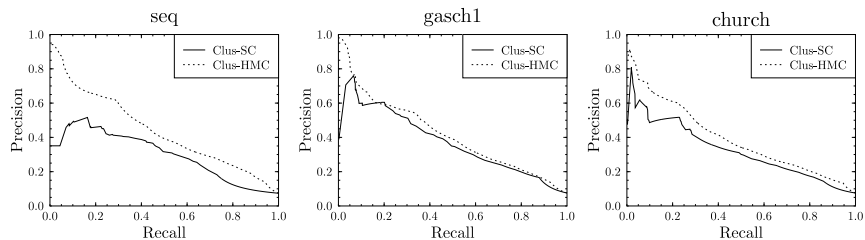


Figure 3: Representative PR-curves for Clus-SC and Clus-HMC.

combination is particularly surprising: the model predicting 250 classes (a more difficult task) is simpler and yet more accurate than models tuned specifically for one class. Further investigation revealed that trying to predict all classes together actually helps to avoid overfitting.

The following is an example of a rule returned by Clus-HMC:

```
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, 40/3, 5, 5/1
```

The rule identifies conditions when a gene has classes 40/3 and 5/1 (and hence also their superclasses). This rule has a precision/recall of 0.97/0.15 for class 40/3 and 0.94/0.37 for class 5/1.

Finally, learning a HMC tree was on average 37 times faster than learning 250 SC trees.

5 Conclusions and further work

Our experiments show that HMC trees are a useful tool for gene function prediction. Compared to learning normal classification trees for each of the classes, learning a HMC tree is much faster (and applying it as well), the resulting tree is not bigger, the tree identifies the attributes relevant for all classes together (instead of those relevant for one specific class), and its predictive performance does not suffer.

We intend to continue this work with a comparison of our approach with other HMC methods, which do not learn decision trees [10, 1] or use a different approach to construct the trees [6]. We also plan to evaluate the method on other datasets from functional genomics, and to map our data onto Gene Ontology [8]: for the current data, GO provides thousands of classes on 19 levels. It will be interesting to see how well the algorithm scales to this more advanced ontology. Moreover, using a widely-known ontology will make it easier to compare the results with other methods.

Acknowledgements

Hendrik Blockeel and Jan Struyf are post-doctoral fellows of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). Leander Schietgat 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 for valuable suggestions.

References

- [1] Zafer Barutcuoglu, Robert E. Schapire, and Olga G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.
- [2] 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.
- [3] 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.
- [4] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.
- [5] A. Clare. *Machine learning and data mining for yeast functional genomics*. PhD thesis, University of Wales, Aberystwyth, 2003.
- [6] A. Clare and R. D. King. Predicting gene function in *Saccharomyces cerevisiae*. *Bioinformatics*, 19(Suppl. 2):ii42–49, 2003.
- [7] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. Technical report, University of Wisconsin, Madison, 2005.
- [8] M. Ashburner et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genet.*, 25(1):25–29, 2000.
- [9] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann, 1993.
- [10] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Learning hierarchical multi-category text classification models. In L. De Raedt and S. Wrobel, editors, *Proceedings of the 22nd International Conference on Machine Learning*, pages 744 – 751. ACM Press, 2005.