UNIVERSITÀ DEGLI STUDI DI MILANO

PhD School in
Computer Science

Computer Science Department
"Giovanni Degli Antoni"

PhD in
Computer Science
XXXI Cycle

Hierarchical Ensemble Methods for Ontology-based Predictions in Computational Biology

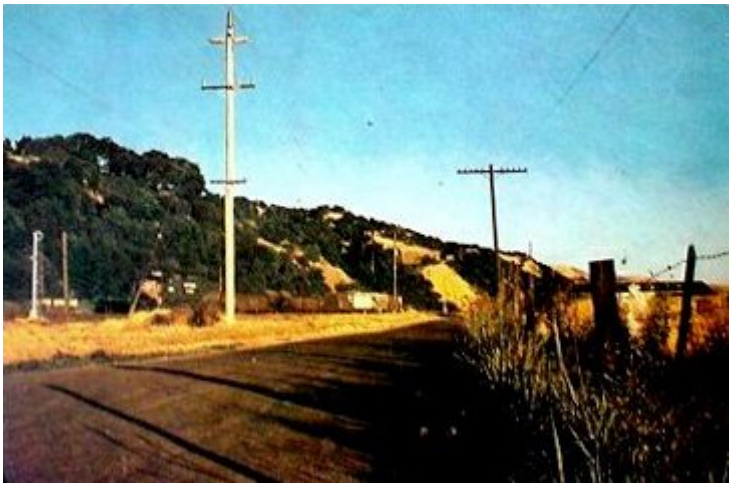
INF/01

PhD candidate:
Marco NOTARO

Advisor:
Prof. Giorgio VALENTINI
Co-Advisor:
Prof. Matteo RE
School Director:
Prof. Paolo BOLDI

Academic Year 2017/18

To Cudone, Genome and Souris,
without whom this PhD thesis
would have never been completed.



Stay Hungry. Stay Foolish.

Stewart Brand, *The Whole Earth Catalog*, 1974.

Acknowledgment

The first person I would like to thank is Giorgio Valentini, my supervisor. There are several reasons of why he needs to be the first of the list. Firstly, because he allowed me to join his lab and he never kicked me out. Secondly, because he was always available and happy to answer even my most dummy question. Thirdly, for the willingness to discuss by email, by phone or by Skype about different topics related to this thesis. And last but not least, for the large degree of freedom that he gave me and that I enjoyed during my awesome Ph.D. journey.

Thanks to Matteo, my co-advisor, for his precious advices. Thanks also to Marco (Frasca) and all the other members of *AnacletoLab*, Marco (Mesiti), Giuliano and Paolo, for having given me the possibility to contribute on other exciting projects during my Ph.D. years.

Thanks to Alberto Paccanaro, Francesco Masulli and Leif Peterson for having accepted to be the reviewers of this thesis and for their careful comments and suggestions.

Thanks to Peter Robinson and Max Scubach for their helpful comments and suggestions that led to publish a successful paper. Thanks also to Peter Hansen for his crazy tour around the Berlin Wall during my visit at Charité. Thanks to Carlos Cano and Michela Verbeni, for having kindly hosted me at CITIC-UGR and for having made my staying in Granada really enjoyable.

Thanks to the Bergamo friends Quadro, Giardo and Gio for the biking, skiing and mountain trips which were a "breath of fresh air" during my Ph.D. Thanks to Ale for the unforgettable moment of "*index 51*". Thanks to Marco (Granato), Ale (again) and Jessica for the hilarious beers after work. Thank to Giulia, my cousin, for having hosted me in her house in Milan whenever I left the lab late. Thanks to Civi, the senior Ph.D. student that everyone should know, for having answered even to the most unlikely question about Ph.D. Thanks to the EMBL friends, Jonas, Julia, Katrin, Rob, Ambra, Kamil, Marvin, Mie, Sabine, Greta and Max (Brambach) for all the amazing and funny moments spent in Heidelberg and in Zürich.

Thanks to EMBL Szilárd and Zürich University library to be handsome places where carrying-out research.

Thanks to Lorena for having made simple the messed-up academic bureaucracy.

A special thanks to my parents, for having always believed in my capacities, for having never stopped to inspire me in following my dreams and in pursuing my goals and for having always supported me during my Ph.D. years.

Finally, I would like to thank Elisa, for believing in me more than I believe in myself and

for never stopping to motivate and encourage me during my Ph.D. years. Thanks for all the motivational conversations about "what comes next the Ph.D.". Thanks for having borne the too many hours spent talking about my scripts' problems, just because: "talking helps to find the solution out!". Thanks for all the tips about how to "survive" in the research world (I got them and I hope that with some practice I will manage to put them into work). Thanks for our scientific debates that helped us to remember, despite anything, how wonderful the science world is. And never forget, to the question: "*What do you you prefer, DNA or RNA*"?, my answer will always be *RNA*, because there is a *U* in it.

Contents

List of Figures

List of Tables

List of Abbreviations

AFP Automated Protein Function Prediction

AUPRC Area Under the Precision-Recall Curve

AUROC Area Under the ROC Curve

BP Biological Process

C50 C5.0 Decision Tree

CAFA Critical Assessment of Function Annotation

CC Gene Ontology

DAG Directed Acyclic Graph

DESCENS DEScendant Classifier ENSemble

FunCat Functional Catalogue

GFP Gene Function Prediction

GLMNET Lasso and Elastic-Net Regularize Generalized Linear Models

GO Gene Ontology

GPAV Generalized Pool-Adjacent-Violators

HPO Human Phenotype Ontology

HTD-DAG Hierarchical Top-Down

ISO-TPR Isotonic True-Path-Rule

LDA Linear Discriminant Analysis

LOGIT Logit Boost

MEDLINE Medical Literature Analysis and Retrieval System Online

MeSH Medical Subject Headings

MF Molecular Function

ML Machine Learning

MLP Multi Layer Perceptron

NB Naive Bayes

NCBI National Center for Biotechnology Information

NLM United States National Library of Medicine

OBO Open Biomedical Ontologies

PFP Protein Function Prediction

PubMed Public/Publisher MEDLINE

PXR Precision at Different Recall Levels

RANKS RAnking of nodes with kernelized score functions

RF Random Forest

TPR-DAG True-Path-Rule

TREEBAG Bagging Ensemble of Decision Tree

UA Unweighted Average network integration

XGBOOST Extreme Gradient Boosting

Abstract

THE standardized annotation of biomedical related objects, often organized in dedicated catalogues, strongly promoted the organization of biological concepts into controlled vocabularies, i.e. ontologies by which related terms of the underlying biological domain are structured according to a predefined hierarchy. Indeed large ontologies have been developed by the scientific community to structure and organize the gene and protein taxonomy of all the living organisms from Archea to Metazoa, i.e. the Gene Ontology, or human specific ontologies, such as the Human Phenotype Ontology, that provides a structured taxonomy of the abnormal human phenotypes associated with diseases.

These ontologies, offering a coded and well-defined classification space for biological entities such as genes and proteins, favor the development of machine learning methods able to predict features of biological objects like the association between a human gene and a disease, with the aim to drive wet lab research allowing a reduction of the costs and a more effective usage of the available research funds.

Despite the soundness of the aforementioned objectives, the resulting multi-label classification problems raise so complex machine learning issues that until recently the far common approach was the "flat" prediction, i.e. simply training a classifier for each term in the controlled vocabulary and ignoring the relationships between terms. This approach was not only justified by the need to reduce the computational complexity of the learning task, but also by the somewhat "unstable" nature of the terms composing the controlled vocabularies, because they were (and are) updated on a monthly basis in a process performed by expert curators and based on biomedical literature, and wet and in-silico experiments.

In this context, two main general classes of classifiers have been proposed in literature. On the one hand, "hierarchy-unaware" learning methods predict labels in a "flat" way without exploiting the inherent structure of the annotation space. On the other hand, "hierarchy-aware" learning methods can improve the accuracy and the precision of the predictions by considering the hierarchical relationships between ontology terms. Moreover these methods can guarantee the consistency of the predicted labels according to the "true path rule", that is the biological and logical rule that governs the internal coherence of biological ontologies.

To properly handle the hierarchical relationships linking the ontology terms, two main classes of structured output methods have been proposed in literature: the first one is based on kernelized methods for structured output spaces, the second on hierarchical ensemble methods for ontology-based predictions. However both these approaches suffer of significant drawbacks. The kernel-based methods for structured output space are computationally intensive and do not scale well when applied to complex multi-label bio-ontologies. Most hierarchical ensemble methods

have been conceived for tree-structured taxonomies and the few ones specifically developed for the prediction in *DAG*-structured output spaces are, in most cases, unable to improve prediction performances over flat methods.

To overcome these limitations, in this thesis novel "ontology-aware" ensemble methods have been developed, able to handle *DAG*-structured ontologies, leveraging previous results obtained with "true-path-rule"-based hierarchical learning algorithms. These methods are highly modular in the sense that they adopt a "two-step" learning strategy: in the first step they learn separately each term of the ontology using flat methods, and in the second they properly combine the flat predictions according to the hierarchy of the classes.

The main contributions of this thesis are both methodological and experimental. From a methodological standpoint, novel hierarchical ensemble methods are proposed, including: a) *HTD-DAG* (Hierarchical Top-Down algorithm for *DAG* structured ontologies); b) *TPR-DAG* (True Path Rule ensemble for *DAG*) with several variants; c) *ISO-TPR*, a novel ensemble method that combines the True Path Rule approach with Isotonic Regression. For all these methods a formal proof of their consistency, i.e. the guarantee of providing predictions that "respect" the hierarchical relationships between classes, is provided.

From an experimental standpoint, extensive genome and ontology-wide results show that the proposed methods: a) are competitive with state-of-the-art prediction algorithms; b) are able to improve flat machine learning classifiers, if the base learners can provide non random predictions; c) are able to predict new associations between genes and human abnormal phenotypes, a crucial step to discover novel genes associated with human diseases ranging from genetic disorders to cancer; d) scale nicely with large datasets and bio-ontologies.

Finally **HEMDAG**, a novel R library implementing the proposed hierarchical ensemble methods has been developed and publicly delivered.

Chapter 1

Introduction

In modern biomedical research the integration, manipulation and exchange of information between researchers are based on the availability of controlled vocabularies used to unambiguously annotate different concepts (i.e. molecular function) related to entities (genes and proteins, just to cite a few) under investigation.

The constant increasing in the volume and in the variety of genomic, metabolomic, transcriptomic and proteomic data is a another characteristic trait of the modern biomedical sciences. Relevant and exciting problems in this area are, for example, the assignment of functions to macromolecules, especially proteins, and the association of abnormal phenotypes to genes. Both these tasks involve complex multi-class, multi-label and multi-path problems that can be modeled as classification or ranking problems. From a computational standpoint both these problems are challenging for the following reasons:

1. *multi-class*: the number of ontology classes is usually large. Hundreds for the Functional Catalogue (FunCat) [1] and thousands for both the Gene Ontology (*GO*) [2] and the Human Phenotype Ontology (*HPO*) [3]. It follows that we need computational methods that scale well with the number of classes;

2. *multi-label*: each gene or gene product may be annotated to more than one class at the same time;

3. *multi-sources*: high-throughput biotechnologies produce an increasing number of genomic, transcriptomic and proteomic data. Hence, in order to exploit all the information available for each gene, we need to integrate different source of data in order to achieve more accurate predictions [4–7];

4. *dependencies among labels*: annotations are dependent from each other because the ontology terms are hierarchically organized (a direct acyclic graph for both *GO* and *HPO*, a forest of trees for the FunCat [8]). The terms specificity depend on the level they are located: on the top-level of the hierarchy we find the more general terms, whereas on the lower level we find the most informative classes from a bio-medical standpoint (i.e. leaves node);

5. *classes are unbalanced*: usually the classes are severely unbalanced, with a small core number of "positive" annotations, and a large number of "negative" annotations;

6. *multiple definitions for negative examples*: since typically we assess only the positive membership to a functional class, the negative instances are not uniquely defined. Nevertheless different strategies have been proposed in literature to properly select them [9–11];

7. *different reliability of functional labels*: functional terms are supported by different degrees of evidence. This means that each label is assigned to a gene with a specific level of reliability;

8. *complex and noisy data*: data are usually complex, (e.g., high-dimensional, large-scale, graph-structured) and noisy;

To deal with the issues described above, several computational approaches were proposed in literature ranging from sequence-based methods [12–14] to network-based methods [15–17]. However, both these strategies predict labels independently from each other, neglecting the structural information between classes. It follows that these approaches can introduce major inconsistencies in the classification, due to the violation of the *true path rule*, that governs both the *GO* [18] and the *HPO* [19] bio-ontologies. According to this rule, also known as *annotation propagation rule*, if a gene is annotated with a given functional term, it must be annotated with all the parent terms and with all its ancestors in a recursive way. On the contrary if a gene is not annotated to a term, it cannot be annotated to its offspring. Moreover, flat predictions are difficult to interpret because they are inconsistent with one another. For example, if we use the *HPO* to predict gene-phenotype relationships, a flat classifier can associate to a human gene the *HPO* term *Hyperplasia of metatarsal bones*, but not the parent term *Abnormality of the metatarsal bones*, introducing inconsistency since *Hypoplasia of metatarsal bones* is a subclass of *Abnormalities of the metatarsal bones*. Again, if we adopt the *GO* to catalog the protein functions, a method that claim, for example, that a protein has a *tyrosine binding* activity but not an *amino acid binding* activity is clearly incorrect and a molecular biologist attempting to interpret these results would likely not trust either annotations [20]. Besides inconsistency, flat methods do not exploit the a priori knowledge about the constraints of the hierarchical labeling thus potentially suffering a reduction in the accuracy of the predictions.

To fill this gap two main classes of structured output prediction methods have been proposed in literature, i.e. methods able to exploit in the learning process the hierarchical structure of terms.

The first category of methods exploits joint input and output kernelization techniques based on large margin methods for structured and interdependent output variables [21–24]. The second general class of structured output methods is based on ensembles of learning machines able to exploit the hierarchical relationship between classes [8,20,25–27]. More in general, both these families of structured output methods, exploit the hierarchical relationships between terms to significantly improve the performances of the "flat" approach [28,29]. In addition, the results of the two international challenge for the Critical Assessment of Functional Annotation, (*CAFA* [30] and *CAFA*2 [31]) emphasized the need of structured-output learning algorithms for predicting a

subset of *GO* terms for a given protein or a subgraph of *HPO* term for a given human gene. In particular, it is worth noting that one of the best performing method of the *CAFA*2 challenge improved flat predictions with an approach similar to that adopted by hierarchical ensemble methods [32]. More precisely, the hierarchical ensemble methods adopt a two-step learning strategy: in the first step a learning machine is trained to learn a specific ontology term and then, in a second step, the resulting prediction are "reconciled" by taking into account the topology of the ontology [33–36] Theoretical studies [37] as well as applications in several domains [38] showed the effectiveness of adopting a hierarchical instead of a flat approach [39]. Indeed hierarchical classification, and in particular ensemble methods for hierarchical classification, have been successfully applied in several domains, ranging from gene [40–42], to protein [43–45] and enzyme [46] function prediction, from speech [47, 48] to document [49, 50] classification, from image [51, 52] to music [53, 54] and video categorization [55, 56].

Even if structured output methods have been successfully applied in the context of biological ontologies, kernel based methods are not able to nicely scale with large ontologies and large data. On the other hand, most hierarchical ensemble methods have been applied to tree-structured ontologies, such as the FunCat, and are not directly applicable to DAG structured ontologies such as the Gene Ontology or the Human Phenotype Ontology [6, 8]. Moreover the few ensemble methods applied in the context of DAG-structured ontologies showed serious problems to improve over flat classification methods [20].

For this reason in this thesis novel ontology-aware ensemble methods are proposed, able to handle DAG-structured taxonomies, to provide consistent predictions and to significantly improve flat prediction methods. Large experimental results, involving genome-wide and ontology-wide prediction of protein function for six model organisms, show the effectiveness of the proposed approach for the prediction of protein function in the context of the Gene Ontology, and human genome-wide experiments for the prediction of abnormal phenotypes coded in the Human Phenotype Ontology show that the newly proposed hierarchical ensemble methods achieve state-of-the-art results in this challenging prediction task [28].

The thesis is structured as follows. In Chapter 3 we provide a short overview of the main classes of computational approaches used to predict the protein function in order to properly contextualize the hierarchical ensemble methods in the state-of-the-art scenario. In Chapter 4 we describe in detail the proposed hierarchical ensemble algorithms and we also prove that our methods always provide consistent predictions that obey to the *true path rule*, that is crucial to assure biologically predictions coherent with the topology of the ontology. In Chapter 5 we compare the proposed hierarchical ensemble algorithms with an array of different flat approaches in order to show that our approaches can improve flat predictions independently of the choice of the base learner. To accomplish this goal we applied our hierarchical ensemble methods to the *GO* term prediction by considering six different model organisms, ranging from nematodes to mammals. In Chapter 6 we present a genome and ontology-wide experimental comparison of our hierarchical ensembles with state-of-the-art methods for *HPO* term prediction and we show that the pro-

posed approaches can accurately predict novel candidate gene–*HPO* term associations [28]. In Chapter 7 we discuss the achieved results and we outline possible future developments. Finally in Appendix A and B we provide supplementary materials about the experiments performed respectively in Chapter 5 and 6, whereas in Appendix C we show a step-by-step application of **HEMDAG** (the R software library implementing the hierarchical ensemble methods proposed in Chapter 4) both for the hierarchical prediction of associations between human gene and abnormal phenotype and for protein function prediction. The **HEMDAG** (Hierarchical Ensemble Methods per *DAG*) library is publicly available both from CRAN and BIOCONDA repository under the GNU General Public License, version 3 (GPL-3.0). The **HEMDAG** package is available for *Linux*, *Windows* and *Macintosh* operating systems.

Chapter 2

Ontologies in Biology and Medicine

Ontologies are controlled vocabularies enabling reasoning at multiple levels in the study and comparison of complex concepts and entities and are widely used in biomedical research. What makes ontologies attractive for the usage in biomedical research is their ability to provide standard identifiers for classes and relations that represent complex molecular, physiological and pathological phenomena; to provide a vocabulary for the considered domain and all its sub domains; to manage metadata that can describe the intended meaning of the terms and relations between them and, last but not least, their ability to provide machine-readable axioms and definitions that enable computational access to some aspects of the meaning of terms and relations. Each of these features enables applications that facilitate data integration, data access and analysis.

The use of ontologies began in the biological sciences around 1998 with the development of the Gene Ontology (*GO*). By 2007, there was sufficient interest and activity in the area to deserve national and international coordination efforts such as the Open Biomedical Ontologies (*OBO*) Foundry or the National Center for Biomedical Ontologies.

At today many ontologies are routinely used in biomedical research. Gene Ontology is used for the description of gene-related functions and processes and is capable to describe them at molecular and subcellular level. The Human Phenotype Ontology (*HPO*) provides a standardized vocabulary of phenotypic abnormalities encountered in human disease. The Medical Subject Headings (*MeSH*) is the medical vocabulary resource curated by the *US* National Library of Medicine (*NLM*) with the aim to provide a hierarchically-organized terminology for indexing and cataloging of biomedical information such as MEDLINE/PUBmed and other *NLM* databases and for the usage in patients diagnosis and treatment-related data management in public and private hospitals.

The number of actively maintained biomedical ontologies is raising fast. The interested reader can find an updated list at the *OBO* Foundry web site (link). Independently by the entities specifically modeled by the ontology two broad classes of biomedical ontologies can be defined on the basis of the eventual availability of information about the relationships between the terms of the considered ontology. In absence of these information the ontology is considered "flat" or unstructured. When formal definitions of between terms relationships are available (here the word "formal" refer to the fact that some ontologies have diverse types of terms relationships that are not semantically equivalent) the ontology is said to be "structured". Structured ontology can,

in turn, be divided into two broad categories based on the type of hierarchical structure adopted to describe the functional relationships between terms: tree structured ontologies and directed acyclic graphs (*DAG*) structured ontologies. Given the availability of biomedical ontologies it is possible to predict the functional role of a broad collection of entities (i.e. genes, transcripts, genetic variants, proteins) in both physiological and pathological processes with positive impact on the development of novel clinical protocols or the improvement-refinement of existing ones.

Of great help in this kind of problems is the application of machine learning (*ML*) methods able to properly integrate and exploit diverse information about the entities whose functional prediction is the goal of the learning task. From a learning perspective the availability of relationships between the terms of the ontology, acting as labels in a multi-class learning problem, on one hand allow the usage of *ML* methods able to exploit the interdependencies between the labels, and this usually translates in better prediction performances, but on the other hand poses some challenges specifically related to the adaptation of the learning scheme to different types of structured output space (trees or *DAG*s).

Chapter 3

Protein Function Prediction Methods

MACHINE learning methods for Automated Function Prediction, can be schematically grouped in the following four large families:

1. sequence-based methods;
2. network-based methods;
3. kernel-based output structured methods;
4. hierarchical ensemble methods.

This grouping is neither exhaustive nor strict, meaning that certain methods do not belong to any of these groups, and others belong to more than one.

It is important to note that, although protein functions are clearly dependent (e.g., a protein can carry-out many functions, which can be further specialized into sub-functions) the methods described in Section 3.2 and 3.1 predict biological functions independently from each other. In contrast, the approaches outlined in Section 3.3 and 3.4 explicitly take into account the hierarchical relationships between labels in order to improve classification performance [28, 57].

3.1 Sequence-Based Methods

Sequence-based algorithms represent the first attempts to computationally predict the protein functions [58–60]. Most of the computational approaches for annotating protein function follow the "*transfer-of-annotation*" paradigm where a gene product is compared against a database and annotated with the function of another protein on the basis of sequence similarity [61]. Such approaches, inspired by nearest-neighbor methodology, suffers of serious limitations in that they fail to exploit the inherent hierarchical structure of the annotation space. Nevertheless, it is important to note that in the *CAFA* challenge [30] and in the *CAFA*2 challenge [31] one of the best performing methods is represented by a sequence-based algorithm [62, 63]. Indeed, when the only information available is represented by a raw sequence of amino acids or nucleotides, sequence-based methods can be competitive with state-of-the-art machine learning methods by exploiting homology-based inference [64]. Finally, algorithms inferring similarities between

sequences are still nowadays standard approaches used for assigning functions to proteins in newly sequenced organisms [12, 65].

3.2 Network-Based Methods

The availability of large-scale networks, in which nodes are genes/proteins and edges their functional pairwise relationships, has promoted the development of several machine learning methods where novel annotations are inferred by exploiting the topology of the underlying biomolecular network. Network-based approaches are able to transfer annotations from previously labeled nodes to unlabeled ones by exploiting "proximity relationships" between connected nodes. These algorithms relied on the so called *guilt-by-association* (*GBA*) rule, which makes predictions assuming that interacting proteins are likely to share similar functions [17, 66, 67]. Indirect neighbors were also exploited to modify the notion of pairwise-similarities among nodes by considering the pairs of nodes connected through intermediate ones [68, 69]. Other approaches exploited the semantic similarity between ontology terms [70, 71] to derive functional similarity measures between genes to construct functional terms, using then supervised or semi-supervised learning algorithm to infer ontology annotations of genes [72–75]. Other strategies propagate the protein function in the network with an iterative process until convergence [76, 77], by tuning the amount of propagation allowed in the graph through Markov random walks [78, 79], by adopting Hopfield networks to handle the class imbalance problem [80, 81], by evaluating the functional flow through the graph nodes [82], by exploiting kernelized score functions [83] and by modeling protein memberships through Markov Random Fields [84] and Gaussian Random Field [85, 86]. Bengio at al. [87] showed that different graph-based algorithms can be cast into a common framework where a quadratic cost objective function is minimized. In this framework closed form solutions can be derived by solving a linear system of size equal to the cardinality of nodes (proteins), or using fast iterative procedures such as the Jacobi method [88]. Cesa-Bianchi et al [89] proposed a network-based approaches alternative to label propagation and exhibiting strong theoretical predictive guarantees in the so-called mistake bound model.

3.3 Kernel Methods for Structured Output Spaces

Structured output methods represent a learning problem using a joint representation of the input-output space, and try to learn a compatibility function $f(x, y)$ that quantifies how related is an input x (e.g. protein) with a label y of the output space (e.g. set of ontology terms) [21]. More precisely, given a feature space $\mathcal{X}$, a space of structured labels $\mathcal{Y}$ and a training set of labeled examples $(x_i, y_i)_{i=1}^n$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, the structured output methods learn a *compatibility function* $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$ which maps input-output pairs to a score that indicates how likely is a protein x to be associated with a set of ontology categories. The compatibility function is expressed as a linear function in a feature space representing the labels and inputs,

i.e. $f(x, y) = w^T \phi(x, y)$ where $\phi(x, y)$ is the joint input-output feature map and w is a weight vector. The predicted label $\bar{y}$ for an input x can be obtained by using the *argmax* operator:

$$\bar{y} = \arg \max_{y \in \mathcal{Y}_\mathcal{C}} f(x, y)$$

where $\mathcal{Y}_\mathcal{C} \in \mathcal{Y}$ is the set of all candidate labels.

In order to obtain correct classification, the compatibility value of the true label (i.e correct set of ontology annotations) of an input instance (i.e protein) needs to be higher than that of any other candidate label. This is captured by the following large-margin formulation [21]:

$$\min_{w,\epsilon} \ \frac{1}{2}||w||_2^2 \ + \frac{C}{n} \ \sum_{i=1}^{n} \epsilon_i$$

subject to the following constraint:

$$f(x_i, y_i) \ - \max_{y \in \mathcal{Y}_\mathcal{C}} \ f(x_i, y_i) \geq 1 - \epsilon_i \qquad i = 1, \ldots, n \quad \epsilon \geq 0 \tag{3.1}$$

where w is the weight vector, C is a user-specified soft-margin constant, $\mathcal{Y}_\mathcal{C}$ is the set of candidate labels, ϵ_i are the slack variables which allow margin violations, and $|| \cdot ||_2$ is the L^2 norm. The Equation 3.1 ensures that the compatibility score for the actual label of a protein is higher than all other candidate labels, and the use of slack variables allow flexibility in satisfying this constraint.

Ben-Hur and coworkers, proposed a Structured Support Vector Machine (*SSVM*) approach for the prediction of *GO* [29] and *HPO* terms [24]. The authors named the methods respectively *GOstruct* and *PHENOstruct* just to emphasize the different problem domain. Both these approaches try to maximize the margin, or the difference between the compatibility value for the actual label and the compatibility for the next best candidate. In the structured-output setting, kernels correspond to dot products in the joint input-output feature space, and they are functions of both inputs and outputs. Both *GOstruct* and *PHENOstruct*, for the prediction of ontology terms, use a joint kernel that is a product of the input-space and the output-space kernels:

$$K((x_1, y_1), (x_2, y_2)) = K_\mathcal{X}(x_1, x_2) K_\mathcal{Y}(y_1, y_2)$$

The rationale of using a product kernel is that two input/output pairs are considered similar if they are similar in both their input feature space and their output label space. Recently, Kahanda et al. proposed an updated version of *GOstruct*, named *GOstrut* $v2.0$, which better handles the incompleteness of annotations [57].

Structured output maximum-margin algorithms, besides being applied to the *GO* [29, 57] and *HPO* term [24] prediction, have been applied to the tree-structured prediction of enzyme functions as well [90, 91]. The main limitation of these approaches is that they are computationally intensive and do not scale well when applied to complex multi-label bio-ontologies, as *GO* and *HPO*.

3.4 Hierarchical Ensemble Methods

Ensemble methods are one of the main topics of research in machine learning [92–95]. From a general standpoint, ensemble methods employ a set of learning machines to solve a classification problem. The advantage is that the combination of many classifiers improve the single one [96].

Protein function prediction (*PFP*) is a hierarchical multi-label classification (*HMC*) task, where a single biological object is associated with more than one label and all these labels are structured in a hierarchical fashion [45]. Depending of the domain problem, the hierarchical class structure can assume the form of a tree (e.g., FunCat) or a *DAG* (e.g., *GO*). *HMC* approaches can be divided into global or local [97]. Global approaches train a single classifier to cope with the whole classes hierarchy [98]. Local approaches employ reduction strategies to reduce the problem to smaller problems and then the local solutions are combined to solve the entire problem [45]. In others words these strategies associate to each node of the hierarchy a local learning machine. Then, the predictions provided by the trained classifier are assembled in a "consensus" decision by taking into account the hierarchical information among classes. There are advantages and drawbacks of using global or local approaches. Global approaches are usually cheaper than local ones (just one classifier is trained to cope with all the classes), but they neglect the local modularity in the label hierarchy, such as parent-child, ancestor-descendant and siblings relationships between different labels. Moreover, global approaches are unable of handling large scale datasets because the model becomes too complex and time-consuming [99]. Local approaches are more suitable for extracting information from regions of the class hierarchy, but they are computationally intensive since they are based on a cascade of classifiers. According to Silla et. al [38] a local classifier can be trained using three different strategies: one local classifier per node (*LCN*), one local classifier per parent node (*LCPN*), and one local classifier per hierarchical level (*LCL*). While *LCN* trains one binary classifier for each class [100], *LCPN* induces a multi-class classifier for each parent node in order to predict its child subclasses [101]. Finally, *LCL* trains one multi-class classifier for each level of the class hierarchy [102].

To properly position our True-Path-Rule ensemble algorithms (discussed in detail in Chapter 4) in the scenario of computational methods for *AFP*, below we present several cutting-edges *HMC* global-based and local-based approaches.

Obozinski and colleagues in [20] proposed several ensemble methods, that they named *reconciliation*, able to provide consistent predictions, that is predictions whose confidence (i.e., posterior probability) increases ascending from more specific to more general ontology terms. More precisely, they proposed eleven distinct reconciliation ensemble methods: three heuristic methods, four variants of a Bayesian network, an extension of logistic regression to the structured case and three novel projection methods, that is isotonic regression and two variants of a Kullback-Leibler projection method. These reconciliation approaches achieved outstanding results in the prediction of functions of mouse (*M. musculus*) proteins [20]. However, it is important to note the authors did not analyze the impact of the concurrent use of data integration and hierarchical

multi-label methods on the overall classification performances. Guan and coworkers in [26] proposed *HIER-MB* (hierarchical Bayesian combination involving nodes in the Markov Blanket). *HIER-MB* modifies the output of *SVMs* (used as base learners) by considering the Bayesian network constructed using the Markov blanket surrounding the node (i.e. ontology term) of interest. The Markov blanket of a node i is represented by its parents, its children and its children's other parents. The Bayesian network involving the Markov blanket of the node i is used to provide the ensemble prediction. *HIER-MB* does not take into account the overall topology of the network, thus its main drawback is represented by the locality of the hierarchical integration, limited only to the Markov blanket nodes. Alaydie et al. [103] designed *HiBLADE* (Hierarchical multi-label Boosting with LAbel DEpendency), an algorithm that takes advantage both of the predefined hierarchical structure of the labels and of the hidden correlation among the classes that is not shown through the hierarchy. In particular, the dependencies of the children for each label in the hierarchy are captured and analyzed using Bayesian method and instance-based similarity. Vens et al. [98] investigated three different methods based on Predictive Clustering Trees (*PCT*): *Clus-HMC*, a global-based approach which trains only one decision tree considering all the classes in the hierarchy, the local-based *Clus-SC*, which trains a binary decision tree for each class ignoring the relationships between classes, and the local-based *Clus-HSC*, which induces a separate decision tree for each class exploring the hierarchical relationships between them. In another study, Schietgat and coworkers [33] proposed a bagging strategy (*Clus-HMC-Ens*) for combining the decision trees induced by *Clus-HMC*, enhancing *Clus-HMC* performances. Triguero and Vens [104] studied alternatives to perform the final labeling in *HMC* problems. The authors evaluated the *Clus-HMC-Ens* method when it uses single and multiple thresholds to transform the continuous prediction scores into actual binary labels. The authors proposed two distinct strategies to select thresholds: optimizing on a given evaluation measure or simulating training set properties in the test set. They concluded that choosing thresholds for each class lead to an improved label-sets and faster execution time. Bi and Kwok [105] formulated the *HMC* as a graph problem of finding the best subgraph in a tree or *DAG*. The authors employed Kernel Dependency Estimation (*KDE*) [106] to reduce the original hierarchy of labels to a manageable number of single-label learning problems. To preserve the parent-child relationships among labels, they implemented a generalized Condensing Sort and Select Algorithm (*CSSA*) [107], which is used to find approximate subtrees. Cesa-Bianchi and Valentini [6] investigated the synergy between different local-based strategies related to gene function prediction (*GFP*) in FunCat taxonomy. The authors and coworkers integrated kernel-based data fusion tools and ensemble algorithms with cost-sensitive *HMC* methods [35, 100]. Synergy was defined as the improvement in the prediction accuracy, considering any evaluation measure, due to the use of concurrent learning strategies. Synergy is detected if the combination of two strategies achieves better correct classification rates than the average of the correct classification of the two strategies used individually [6]. Cerri et al. [45, 102, 108] proposed *HMC-LMLP*, Hierarchical Multi-label Classification with Local Multi-Layer Perceptrons, a local *HMC* approach based on

a chain of Multi-Layer Perceptrons (*MLPs*) with a single hidden layer. *HMC-LMLP* associates a *MLP* neural network to each hierarchical level. Then, the *MLP* predictions obtained at one level are used to augment the feature vectors of the instances employed to train the *MLP* associated with the next level. A gene product is annotated to a class if its corresponding output in the *MLP* is larger than a predefined threshold. Afterwards, in the second step of the hierarchical classification the inconsistent predictions (i.e. subclasses predicted without the predictions of their superclasses) are removed. Vateekul et al. [109] introduced a Hierarchical *R-SVM* system (*HR-SVM*) for gene function prediction. The threshold adjustment from *R-SVM* [110] is used to mitigate the problem of false negatives in *HMC*. Yu et al. [27] propose a method to predict protein function using incomplete hierarchical labels. The idea is to take the hierarchical and non-hierarchical similarities between functions and define a combined similarity between the labels. This similarity, together with the known labels, is used to estimate the missing functions of the proteins in the hierarchy. Subsequently, the approach predicts the protein functions by exploiting the information about their interactions. Sun et al. [111] proposed *PLS+OPP*, in which the classification task is formulated as a path selection problem. They used partial least squares (*PLS*) to transform the label prediction problem into an optimal-path prediction (*OPP*) strategy. Each multi-label prediction is a connected subgraph that includes a small number of paths and the final predictions are provided by merging optimal paths. Zhang et al. [51], proposed a cost-sensitive method. Similarly to [45], binary classifiers are trained for each node. Next, a weight matrix is created based on error minimization. Then, both predictions and weights are combined to provide final predictions. Since the weight matrix scales for a large amount of classes, parallel computing techniques may be required. Feng et al. [41] proposed a method to enhance gene function prediction performance. Firstly, to tackle the imbalanced data set problem, the authors applied a negative instances selecting policy associated with the *SMOTE* strategy. Secondly, they applied a nodes interaction method to combine the results of binary classifiers and to ensure the hierarchical constraints. Nakano et al. [36] proposed Stacking methods for protein function prediction and transposable elements classification. Stacking is an ensemble technique that employs many classifiers to achieve high generalization [112]. More specifically, Nakano and coworkers proposed three different stacking approaches: Augmented Stacking (*AS*), which extends Stacking by building a more robust meta-classifier; Cascade Sacking (*CS*) which employs a cascade of stacking where predictions made by previous classifiers are concatenated by considering the hierarchical constraints; and Cascade Augmented Stacking (*CAS*) that merely combines the benefits of the previous methods *AS* and *CS*. Caruna et al. [113, 114] proposed an iterative algorithm that starts with an empty ensemble and in each iteration adds a the base predictor that best improve the resultant ensemble's performance, partially due to the added predictor's complementary to the current ensemble. The process continues until the ensemble's performance does not improve anymore or starts decreasing. Wang et al. [115] assessed the ability of a variety of heterogeneous ensemble methods across a multitude of functional terms, proteins and organism. The experimental results shown that the ensemble methods, especially

Stacking using Logistic Regression, produce more accurate predictions for a variety of Gene Ontology terms differing in size and specificity. Wehrmann et al. [116] proposed an hybrid method named *HMCN*, Hierarchical Multi-label Classification Network, which is a multi-output deep neural network able to concurrently optimize both local and global loss functions. The authors proposed two distinct variants of *HMCN*: a more robust feed-forward version (*HMCN-F*) and a more efficient recurrent version (*HMCN-R*) inspired on Long Short Term Memory (*LSTM*) networks [117] for encoding hierarchical information. Both *HMCN* version can be applied to either tree or *DAG* structured hierarchies.

In the next chapter the True-Path-Rule-based ensemble algorithms are described in depth since they represent the "*core methods*" on which the experiments of this thesis (illustrated in Chapter 5 and 6) are based.

Chapter 4

Hierarchical Ensemble Methods for Directed Acyclic Graphs

In this chapter, we present several novel hierarchical ensemble methods able to provide theoretically guaranteed consistent predictions for any *DAG*-structured taxonomy and consequently also for any ontology structured according to a tree. The proposed ensemble learning strategies can be classified in two general categories. The first one includes the algorithms characterized by just one step, that are named Hierarchical Top-Down for *DAG* (*HTD-DAG*) [28] [1] and Generalized Pool-Adjacent-Violators (*GPAV*) [119]. The second one encompasses ensemble approaches characterized by a double flow of information, i.e. True Path Rule for *DAG* (*TPR-DAG*) [28] [2], DEScendant Classifier ENSemble (*DESCENS*) [121] and Isotonic True-Path-Rule (*ISO-TPR*) [122].

In their more general form, the proposed hierarchical ensemble methods adopt a two-step learning strategy [44]:

1. *Flat learning of the terms of the ontology.* In the first step each base classifier learns a specific ontology term. This yields a set of independent classification problems, where each base learning machine is trained to learn a specific class, independently of the other base learners.

2. *Hierarchical combination of the predictions.* In the second step the predictions provided by the trained classifiers are combined by considering the hierarchical relationships between the base classifiers modeled according to the hierarchy of the underlying bio-ontology.

Figure 4.1 illustrates the two learning steps of hierarchical ensemble methods. In the first step, a learning machine (represents by an ellipse object in Figure 4.1(a)) is applied to train the base classifiers (circles) associated with each class (represented with integer numbers from 1 to n). Then in the second step, the resulting base classifiers exploit the hierarchical relationships between classes to combine its predictions with those provided by the other base classifiers (Figure 4.1(b)). Note that a fake node R is added to obtain a rooted hierarchy.

This ensemble approach is highly modular: in principle any learning algorithm can be used to train the classifiers in the first step, and in the second step the hierarchical relationships

[1] a preliminary version of *HTD-DAG* was presented at the IWBBIO conference [118]

[2] a preliminary version of *TPR-DAG* was presented at the and MCS conference [120]

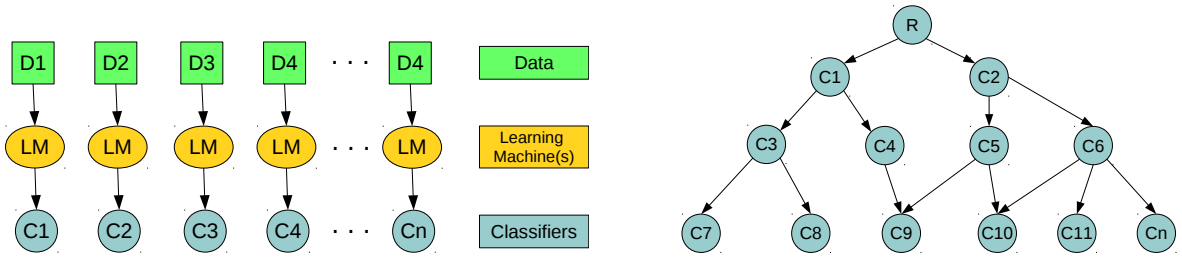


Figure 4.1: Schematic representation of the two main learning steps of hierarchical ensemble methods. (a) Training of the base classifiers; (b) Hierarchical combination of the base classifiers.

between the ontology terms are exploited to achieve the final ensemble predictions of the set of classes associated with a specific gene or gene product.

The main limitation of the flat learning of the ontology terms is that each class is separately learned without taking into account the relationships between classes. Other methods, such as the ensemble approach proposed in [33] can overcome this limitation by applying a multi-label classification "global" models to predict all the ontology terms at the same time, explicitly by considering the relationships between the classes just during the learning process. On the other hand, the hierarchical ensemble methods proposed here are able to correct the flat predictions, by splitting in two separate steps the process of flat learning of ontology terms and the process of evaluating the hierarchical relationships between classes.

Before explaining in detail the proposed hierarchical ensemble methods one-by-one, in the next section we will give some general and basic notations and definition of *DAGs*.

4.1 Basic Notations and Definitions

Let $G =< V, E >$ a Directed Acyclic Graph (*DAG*) with vertices $V = \{1, 2, \ldots, |V|\}$ and edges $e = (i, j) \in E, i, j \in V$. G represents a taxonomy structured as a *DAG*, whose nodes $i \in V$ represent classes (terms) of the ontology and a directed edge $(i, j) \in E$ the hierarchical relationships between i and j: i is the parent term and j is the child term. The set of children of a node i is denoted by $child(i)$, the set of its parents by $par(i)$, the set of its ancestors by $anc(i)$ and the set of its descendants by $desc(i)$. A "flat multi-label scoring" predictor $f : X \to \mathbb{Y}$ provides a score $f(x) = \hat{\boldsymbol{y}}$, where $\hat{\boldsymbol{y}} \in \mathbb{Y} = [0, 1]^{|V|}$ is the flat score for a given example $x \in X$, with X a suitable input space for the predictor f. In other words a flat predictor provides a score $\hat{y}_i \in [0, 1]$ that represents the likelihood that a given gene (or gene product) belongs to a given node/term $i \in V$ of the *DAG* G, and $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >$. We say that the multi-label scoring $\boldsymbol{y}$ is consistent if it obeys the *true path rule*:

$$\boldsymbol{y} \text{ is consistent } \iff \forall i \in V, j \in par(i) \Rightarrow y_j \geq y_i \tag{4.1}$$

It is straightforward to show that (4.1) holds even with flat classifiers that do not provide a score but a label $\hat{y}_i \in \{0, 1\}$ indicating that a given gene belongs ($\hat{y}_i = 1$) or does not ($\hat{y}_i = 0$)

to a given bio-ontology term i. From equations 4.1 descends that if we predict that a protein is annotated with a given term i then to provide consistent predictions the same protein must be annotated also with all the ancestor terms of i.

In real cases it is very unlikely that a flat multi-label scoring predictor satisfies the true path rule, since by definition the predictions are performed without considering the hierarchy of the classes. Nevertheless, by adding an additional topology-aware step we can modify the labeling and the scores of the flat predictors to obtain a hierarchical classifier that fairly satisfies the constraints imposed by the true path rule. More precisely, we can provide a prediction function $g(f(x)) : \mathbb{Y} \to \mathbb{Y}$ such that the *true path rule* (4.1) holds for all the predictions $g(f(x)) = \bar{\boldsymbol{y}}$: $\forall i \in V, j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$, where $\bar{y}$ is the score achieved by the hierarchical classifier.

4.2 Flat Learning of Ontology Terms

The ensemble algorithms first adopts a flat learning strategy by which each term $i \in V$ of the *GO* is independently learned through a term specific predictor $f_i : X \to [0, 1]$. Accordingly, the output of the flat classifier $f : X \to \mathbb{Y}$ on the instance $x \in X$ is $f(x) = \hat{\boldsymbol{y}}$:

$$f(x) =< f_1(x),\ f_2(x),\ \ldots,\ f_{|V|}(x) >=< \hat{y}_1, \hat{y}_2,\ \ldots,\ \hat{y}_{|V|} >$$

In other words a classifier f_i is associated to each *GO* class i in order to provide a flat evaluation of the membership of a specific example $x \in X$ to the class i.

To this end any supervised or semi-supervised base predictor can be applied, including also flat binary classifiers. Indeed both learners able to provide a probability or a score related to the likelihood that a gene product is annotated with a *GO* term (that is scores $\hat{y}_i \in [0, 1]$), and base binary classifiers that can directly provide a label (but not a score) about the protein-*GO* term association (that is a label $\hat{y}_i \in \{0, 1\}$) can be used to generate the flat predictions. Note that the training of per-class predictors $f_1,\ f_2,\ \ldots, f_{|V|}$ can be performed in parallel, and it is easy to achieve a linear speed-up in the number of the available processors by adopting simple parallel computational techniques.

4.3 Hierarchical Top-Down ensembles for DAG (HTD-DAG)

The main idea behind the Hierarchical top-down algorithm (*HTD-DAG*) consists in modifying the predictions of each base learner from "top to bottom", i.e. from the least to the most specific terms by exploiting at each step the predictions provided by the less specific predictors, e.g. predictors associated to parent terms. This is performed in a recursive way by transmitting the predictions from each node to the their children, and from the children to the children of the children through a propagation of the information towards the descendants of each node of the ontology. For instance in Figure 4.2 (a) the information can flow along the path traversing nodes $1, 5, 6, 7$ or $1, 3, 7$, and a prediction for e.g. the node 5 depends on the predictions performed

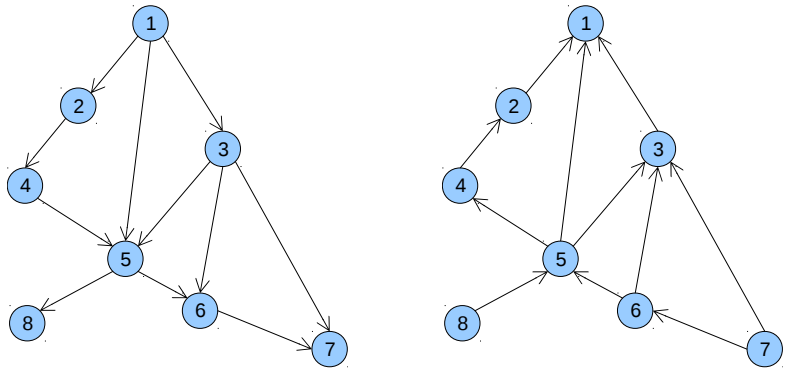


Figure 4.2: Flow of information in hierarchical ensembles. (a) Top-down flow (b) Bottom-up flow. See text for more explanations.

by the base learners for the parent nodes $4, 1$ and 3. This operating mode of the ensemble is performed in an ordered way from the top to the bottom nodes (Figure 4.2 (a)).

More precisely, the *HTD-DAG* algorithm modifies the flat scores according to the hierarchy of a *DAG* through a unique run across the nodes of the graph. For a given example $x \in X$, the flat predictions $f(x) = \hat{\boldsymbol{y}}$ are hierarchically corrected to $\bar{\boldsymbol{y}}$, by per-level visiting the nodes of the *DAG* from top to bottom, according to the following simple rule:

$$\bar{y}_i := \begin{cases} \hat{y}_i & \text{if} \quad i \in root(G) \\ \min_{j \in par(i)} \bar{y}_j & \text{if} \quad \min_{j \in par(i)} \bar{y}_j < \hat{y}_i \\ \hat{y}_i & \text{otherwise} \end{cases} \tag{4.2}$$

The node levels correspond to their maximum path length from the root. If $p(r, i)$ represents a path from the root node r and a node $i \in V$, $l\left(p\left(r, i\right)\right)$ the length of $p(r, i)$, $\mathcal{L} = \{0, 1, \ldots, \xi\}$ the set of observed levels, with ξ the maximum node level, then $\psi : V \longrightarrow \mathcal{L}$ is a level function which assigns each node $i \in V$ to its level $\psi(i)$:

$$\psi(i) = \max_{p(r,i)} \; l\left(p(r, i)\right) \tag{4.3}$$

Nodes $\{i|\psi(i) = 0\}$ correspond to the root nodes, $\{i|\psi(i) = 1\}$ is the set of nodes with a maximum path length from the root (distance) equal to 1, and $\{i|\psi(i) = \xi\}$ are nodes that lie at a maximum distance ξ from the root. The consistency of the predictions is guaranteed if and only if the levels are defined according to the maximum path length from the root. In the Section 4.8 we provide a formal proof of this fact.

The Figure 4.3 shows the pseudo-code of the second step of *HTD-DAG* algorithm, by which the flat predictions $\hat{\boldsymbol{y}}$ computed in the first step are combined and updated according to top-down per-level traversal of the *DAG*. The block A of the algorithm (rows $01 - 04$) computes the maximum distance of each node from the root; to this end the classical Bellman-Ford algorithm or the methods based on the Topological Sorting algorithm can be applied [123]. The block B of the algorithm implements a per-level top-down visit of the graph (rows $05 - 16$). Starting

Figure 4.3: Hierarchical Top-Down algorithm for DAGs (HTD-DAG)

```
Input:
- G =< V, E >
- ŷ =< ŷ_1, ŷ_2, ..., ŷ_|V| > (flat predictions)
begin algorithm
01:    A. Compute ∀i ∈ V the max distance from root(G):
02:       E' := {e'|e ∈ E, e' = −e}
03:       G' :=< V, E' >
04:       dist := Bellman.Ford(G', root(G'))
05:    B. Per-level top-down visit of G:
06:       ȳ_root(G) := ŷ_root(G)
07:       for each d from 1 to ξ do
08:          N_d := {i|dist(i) = d}
09:          for each i ∈ N_d do
10:             x := min_{j∈par(i)} ȳ_j
11:             if (x < ŷ_i)
12:                ȳ_i := x
13:             else
14:                ȳ_i := ŷ_i
15:          end for
16:       end for
end algorithm
Output:
- ȳ =< ȳ_1, ȳ_2, ..., ȳ_|V| >
```

from the children of the root (level 1) for each level of the graph the nodes are processed and the hierarchical top-down correction of the flat predictions $\hat{y}_i$, $i \in \{1, \ldots, |V|\}$ is performed according to equations 4.2, thus obtaining the *HTD-DAG* ensemble prediction $\bar{y}_i$. More precisely, the nested loops starting respectively at line 07 and 09 ensure that nodes are processed by level in an increasing order. Lines $10 - 14$ perform the hierarchical correction of the flat predictions $\hat{y}_i$, $i \in \{1, \ldots, |V|\}$. The algorithm ends when nodes at distance ξ from the root are processed (last iteration of the external loop within lines $07 - 16$) and it finally provides the hierarchically corrected predictions $\bar{\boldsymbol{y}}$.

The complexity of block A is $\mathcal{O}(|V| + |E|)$ (if the Topological Sort algorithm is used to implement *ComputeMaxDist*), while it is easy to see that the complexity of block B (rows $3 - 13$) is $\mathcal{O}(|V| + |E|)$. Hence the overall complexity of the top-down step of *HTD-DAG* is $\mathcal{O}(|V| + |E|))$, that is linear in the number of nodes of the corresponding *DAG*, considering that usually the bio-ontologies are sparse.

4.4 Hierarchical True Path Rule ensembles for DAG (TPR-DAG)

The *HTD-DAG* algorithm removes the constraints violations, by propagating towards the bottom of the hierarchy the negative predictions (i.e. instances predicted to be unannotated to an ontology term), as we can see from the equations 4.2. Consequently, in the worst case, might happen that the predictions at the leaves nodes (i.e. the most informative classes from a bio-medical standpoint) are all negatives. To overcome this limitation we introduce the *TPR-DAG* algorithm, in which we propagate from bottom to top the positive predictions (i.e. instances predicted to be annotated to an ontology term), before applying the top-down step. Indeed by considering the bottom-up flow of information we can construct the prediction of the ensemble by recursively propagating the predictions provided by the the most specific nodes toward their parents and ancestors. For instance in Figure 4.2 (b) a possible flow of information could be along the path $8, 5, 4, 2, 1$ or $7, 6, 5, 1$, and the prediction of the ensemble for e.g. node 3 depends on children nodes $5, 6$ and 7. The proposed *TPR-DAG* adopts this bottom-up flow of information, to take into account the predictions of the most specific ontology terms, but also the opposite flow from top to bottom to consider the predictions of the least specific terms. The Figure 4.4 provides a pictorial toy example of the operating mode of the *TPR-DAG* algorithm.

It is important to mention that the *TPR-DAG* algorithm presented in this thesis is related to the TPR algorithm for tree-structured taxonomies [8], but despite the similarity of their names, the TPR for trees cannot be applied to ontology as *HPO* and *GO*, since it provides inconsistent predictions when applied to *DAG*-structured taxonomies. One difference respect to the tree-version is that the the per-level traversal of the graph in the *DAG*-version is performed in two distinct and strictly separated steps: (1) the bottom-up per-level traversal of the *DAG* is followed by (2) a per-level top-down visit. The separation of the bottom-up and top-down steps is necessary to assure the true path rule consistency of the predictions in *DAG*-structured taxonomies (see Section 4.8). On the contrary in the tree-version the per-level traversal is performed in an "interleaved" fashion, i.e. the bottom-up and top-down traversal are alternated at each level, [8]. Another main difference consists in the way in which the levels are computed: in the new *DAG* version the levels are constructed according to the maximum distance from the root, since this guarantees that in the top- down step all the ancestor nodes have been processed before their descendants, assuring so the true path rule consistency of the predictions (see Section 4.8 for a formal proof of this fact).

The *TPR-DAG* algorithm provides a "consensus" ensemble predictions by integrating the flat predictions $\hat{y}_i$ through a per-level visit of the *DAG*:

$$\bar{y}_i := \frac{1}{1 + |\phi_i|}(\hat{y}_i + \sum_{j \in \phi_i} \bar{y}_j) \tag{4.4}$$

where ϕ_i are the "positive" children of i.

Note that only positive predictions of the children obey the true path rule. Indeed, according

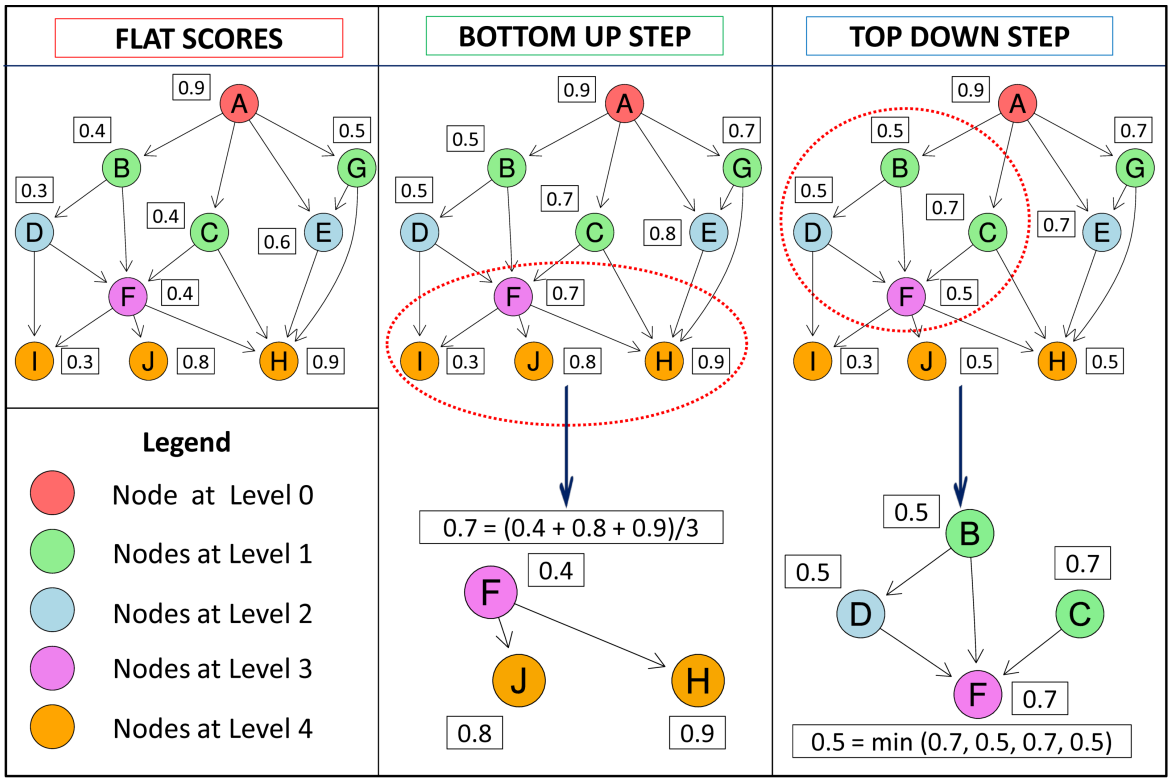


Figure 4.4: A toy example of the operating mode of the *TPR-DAG* method. Left: The nodes represent the terms of a bio-ontology and the numeric values the flat scores associated to each node of the graph. The different colors represent the levels, i.e. the maximum distance of the node from the root node A. Center: The bottom-up step introduces a correction of the flat scores by taking into account the scores of the children of each node. This procedure is methodically repeated from the bottom to the top nodes of the *DAG*; as an example, the bottom part shows that the correction for the F node is performed by averaging the flat score of the F parent node with those of the "positive" children, i.e. that children nodes having a value larger than that of the F parent node. Right: The Top-down step introduces a further correction by taking into account the scores of the parent nodes, by methodically parsing this time the *DAG* from the root node A down to descendant nodes; as an example, the bottom part of the figure shows that the score of the F node is set to the minimum of the bottom-up scores of F and that of its parents A, B and C.

to this rule, we may have a gene (or gene product) annotated to a term t, but not annotated to a terms $s \in child(t)$. Hence if we have a negative prediction for the terms s it is not meaningful to use this prediction to predict the term t. It is worth noting that we can combine children predictions using aggregation strategies other than the average. For instance using the maximum, we could likely improve the sensitivity, but with a plausible decrement of the precision. Different strategies to select the "positive" children ϕ_i can be applied, according to the usage of a specific threshold to separate positive from negative examples:

1. *Constant Threshold (T) strategy.* For each node the same threshold $\bar{t}$ is a priori selected: $t_j = \bar{t}, \quad \forall j \in V$. In this case $\forall i \in V$ we have:

$$\phi_i := \{j \in child(i) | \bar{y}_j > \bar{t}\} \tag{4.5}$$

   For instance if the predictions represent probabilities it could be meaningful to a priori

select $\bar{t} = 0.5$.

2. *Adaptive Threshold (AT) strategy.* The threshold is selected to maximize some performance metric $\mathcal{M}$ estimated on the training data, e.g. the F-score or the AUPRC. In other words the threshold is selected to maximize some measure of accuracy of the predictions $\mathcal{M}(j,t)$ on the training data for the class j with respect to the threshold t. The corresponding set of positives $\forall i \in V$ is:

$$\phi_i := \{j \in child(i) | \bar{y}_j > t_j^*, t_j^* = \arg\max_t \mathcal{M}(j,t)\} \tag{4.6}$$

For instance internal cross-validation can be used to select t_j^* from a set of $t \in (0,1)$.

3. *Threshold Free (TF) strategy.* This strategy does not require an a priori or experimentally selected threshold. We select as positive those children that increment the score of their parent node i:

$$\phi_i := \{j \in child(i) | \bar{y}_j > \hat{y}_i\} \tag{4.7}$$

Consequently, we can derive three different algorithmic variants from the "vanilla" TPR:

a) TPR-T: TPR with constant threshold, corresponding to strategy 1);

b) TPR-AT: TPR with adaptive thresholds, corresponding to strategy 2);

c) TPR-TF: TPR threshold-free, corresponding to strategy 3).

All the three variants of the *TPR-DAG* algorithms propagate the positive predictions towards the parents and recursively towards the ancestors of each node, moving in such way the predictions from bottom to top. The high-level pseudo-code of the *TPR-DAG* algorithm is shown in Figure 4.5. It is structured into three parts. The block A computes the maximum distances of each node V_i from the root via the Bellman-Ford algorithm (row 01). The block B (rows $02-09$) performs a bottom-up visit of the graph and updates the predictions $\bar{y}_i$ of the *TPR-DAG* ensemble according to the equation 4.4 and together with one of the three positive selection strategies described above. It is worth noting that this step assures the propagation of positive predictions, but it does not guarantee their consistency. This is accomplished by the block C (rows $10-21$) that simply performs a hierarchical top-down step, in the same way of the *HTD-DAG* algorithm.

The complexity of the *TPR-DAG* algorithm is quadratic in the number of nodes for the block A, but can be $\mathcal{O}(|V|+|E|)$ if the Topological Sort algorithm is used instead. It is easy to see that the complexity is $\mathcal{O}(|V|)$ for both the B and C blocks when graphs are sparse. Hence, by considering the sparseness of the bio-ontology, the algorithm is linear with respect to the number of the terms of the considered ontology.

To modulate the contribution to the ensemble prediction of the parent node and its children, we design a *TPR-DAG* variant similar to the weighted True Path Rule algorithms for

Figure 4.5: Hierarchical True Path Rule algorithm for DAGs (TPR-DAG)

```
Input:
- G =< V, E >
- V = {1, 2, ..., |V|}
- ŷ =< ŷ_1, ŷ_2, ..., ŷ_|V| >,   ŷ_i ∈ [0, 1]
begin algorithm
01:    A. dist := ComputeMaxDist (G, root(G))
02:    B. Per-level bottom-up visit of G:
03:       for each d from max(dist) to 0 do
04:          N_d := {i|dist(i) = d}
05:          for each i ∈ N_d do
06:             Select the set φ_i of "positive" children
07:             ȳ_i := 1/(1+|φ_i|) (ŷ_i + Σ_{j∈φ_i} ȳ_j)
08:          end for
09:       end for
10:    C. Per-level top-down visit of G:
11:       ŷ := ȳ
12:       for each d from 1 to max(dist) do
13:          N_d := {i|dist(i) = d}
14:          for each i ∈ N_d do
15:             x := min_{j∈par(i)} ȳ_j
16:             if (x < ŷ_i)
17:                ȳ_i := x
18:             else
19:                ȳ_i := ŷ_i
20:          end for
21:       end for
end algorithm
Output:
- ȳ =< ȳ_1, ȳ_2, ..., ȳ_|V| >
```

tree-structured taxonomies [6]. This variant, that we named TPR-Weighted (*TPR-W*), can be obtained simply by substituting the row 07 of the *TPR-DAG* algorithm with the following line of pseudo-code:

$$\bar{y}_i := w\hat{y}_i + \frac{(1-w)}{|\phi_i|} \sum_{j \in \phi_i} \bar{y}_j \tag{4.8}$$

In this approach a weight $w \in [0, 1]$ is added to balance between the contribution of the node i and that of its "positive" children. If $w = 1$ no weight is attributed to the children and the

TPR-DAG reduces to the *HTD-DAG* algorithm, since in this way only the prediction for node i is used in the bottom-up step of the algorithm. If $w = 0$ only the predictors associated to the children nodes "vote" to predict node i. In the intermediate cases for increasing values of w we attribute more importance to the predictor for the node i with respect to its children and for decreasing values of w we put more weight on children.

A different way to implement a weighting strategy could be also pursued not only considering balancing between the predictions on node i and nodes $j \in child(i)$, but including also weighting with respect to the estimated accuracy of each base learner, estimated e.g. by internal cross-validation.

4.5 Descendant Classifier Ensemble (DESCENS)

The novelty of *DESCENS* with respect to *TPR-DAG* algorithm consists in strongly considering the contribution of all the descendants of each node instead of only that of its children. Indeed, as shown in the *TPR* tree-version [8], the contribution of the descendants of a given node decays exponentially with their distance from the node itself, and it is easy to see that this is true also for the *TPR-DAG* algorithm. On the contrary, the *DESCENS* predictions are more influenced by the information embedded in the leaves nodes, that are the classes containing the most informative and meaningful information from a biological and medical standpoint. The pseudo-code of the *DESCENS* algorithm is shown in the Figure 4.6.

By looking at the pseudo-code of the *DESCENS* algorithm (Figure 4.6) it easy to see that the main difference respect to the *TPR-DAG* algorithm (Figure 4.5) consists in the selection of the set of "positive" descendants Δ_i of a node i instead of the set of "positive" children ϕ_i in the bottom-up step (block B rows $06 - 07$). For the choice of the "positive" descendants we can use the same strategies adopted for the selection of the "positive" children in the *TPR-DAG* algorithm (see 4.4).

Furthermore, we added a new variant specific only for *DESCENS* algorithm, that we named *DESCENS-τ*. *DESCENS-τ* balances the contribution between the "positive" children of a node i and that of its "positive" descendants excluding its children by adding a weight $\tau \in [0, 1]$:

$$\bar{y}_i := \frac{\tau}{1 + |\phi_i|}(\hat{y}_i + \sum_{j \in \phi_i} \bar{y}_j) + \frac{1 - \tau}{1 + |\delta_i|}(\hat{y}_i + \sum_{j \in \delta_i} \bar{y}_j) \tag{4.9}$$

where ϕ_i are the "positive" children of i and $\delta_i = \Delta_i \setminus \phi_i$ the descendants of i without its children. If $\tau = 1$ we consider only the contribution of the "positive" children of i, and if $\tau = 0$ only the descendants that are not children contribute to the score, while for intermediate values of τ we can balance the contribution of ϕ_i and δ_i positive nodes.

Considering the sparseness of the bio-ontologies, it is easy to see that the overall computational complexity of *DESCENS* algorithm is $O(|V|)$.

Figure 4.6: DEScendant Classifier ENSemble for DAGs (DESCENS)

```
Input:
- G =< V, E >
- V = {1, 2, ..., |V|}
- ŷ =< ŷ_1, ŷ_2, ..., ŷ_|V| >,   ŷ_i ∈ [0, 1]
begin algorithm
01:    A. dist := ∀i ∈ V ComputeMaxDist (G, root(G))
02:    B. Per-level bottom-up visit of G:
03:       for each d from max(dist) to 0 do
04:          N_d := {i|dist(i) = d}
05:          for each i ∈ N_d do
06:             Select the set Δ_i of "positive" descendants
07:             ȳ_i := 1/(1+|Δ_i|) (ŷ_i + Σ_{j∈Δ_i} ȳ_j)
08:          end for
09:       end for
10:    C. Per-level top-down visit of G:
11:       ŷ := ȳ
12:       for each d from 1 to max(dist) do
13:          N_d := {i|dist(i) = d}
14:          for each i ∈ N_d do
15:             x := min_{j∈parents(i)} ȳ_j
16:             if (x < ŷ_i)
17:                ȳ_i := x
18:             else
19:                ȳ_i := ŷ_i
20:          end for
21:       end for
end algorithm
Output:
- ȳ =< ȳ_1, ȳ_2, ..., ȳ_|V| >
```

4.6 Generalized Pool-Adjacent-Violators (GPAV)

Another strategy to make flat predictions coherent with the underlying ontology predictions is to apply the *isotonic regression* or *monotonic regression* [124]. More precisely, the monotonic regression problem (MR) involves finding a weighted least-squares fit $x \in R^n$ to a vector $y \in R^n$ with weights vector $w \in R^n$ subject to a set of given constraints of the kind $x_i \leq x_j$. Such constraints define a partial or a total ordering and can be represented as a directed acyclic

graph $G(V, E)$, where $V = \{1, 2, 3 \dots |V|\}$ is the set of nodes and E is the set of pairs (i, j) $\forall (i, j) \in V$. Each node is associated with an observed value and each edge is associated with one monotonicity relationship. The *MR* problem can be formulated as follow. Given a vector of observed values $y \in R^n$, a strictly positive vector of weights $w \in R^n$ and a directed acyclic graph $G(V, E)$, we must find the vector of fitted values $x^\star \in R^n$ that solves the following problem:

$$\begin{array}{ll} min & \sum_{i \in V} w_i (x_i - y_i)^2 \\ s.t. & x_i \leq x_j \quad \forall (i, j) \in E \end{array} \tag{4.10}$$

In other words, the *MR* problem involves finding among all vectors $x \in R^n$ which preserve the monotonicity relationships, the one closest to the vector of observed values $y \in R^n$ in the least square sense. Since the problem 4.10 is a strictly convex quadratic programming problem, its solution $x^\star$ is unique.

It is easy to solve the problem 4.10 when the constraints are in the simple form:

$$x_1 \leq x_2 \leq \dots x_{|V|} \tag{4.11}$$

i.e., when the associated graph $G(V, E)$ is a path. For this special case of complete order, the most efficient and widely used algorithm is the Pool-Adjacent-Violators algorithm (*PAV*) [125, 126], which computational complexity is $\mathcal{O}(|V|)$ [127]. On the contrary, to solve the general *MR* problem 4.10, the conventional quadratic programming algorithms [128] can be used only when the number of observations is quite small (up to few hundred). In addition, the existing approximate *MR* algorithms [129, 130] are characterized either by a high computational complexity $\mathcal{O}(|V|^4)$ [131, 132] (which is prohibitive for large V) or by a too low accuracy of their solutions [119].

Burdakov and coworkers [133] proposed an approximate algorithm, that combines both low computational complexity and high accuracy, to figure out the problem 4.10. They named this algorithm Generalized Pool-Adjacent-Violators (*GPAV*), because it can be viewed as a generalization of the *PAV* algorithm. *GPAV* generates some splitting of the set of nodes V into disjoint block of nodes. The subset of nodes $B \subset V$ is called *block* if, for any $i, j \in B$, all the nodes in all the paths from i to j belong to B. For each nodes $i \in V$, we denote the set of its immediate predecessors $\{j \in V : (j, i) \in E\}$ by i^-. The block B_i is said to be *adjacent* to B_j (or an *immediate predecessor* for B_j), if there exist $k \in B_i$ and $l \in B_j$ such that $k \in l^-$. Let B_i^- denote the set of all blocks adjacent to block B_i. *GPAV* associates each block with one of its upper nodes, which is called *head* node. If i is the head node for some block, this block is denoted by B_i. The set of all head nodes is denoted by H. The set of blocks $\{B_i\}_{i \in H}$, where $H \subset V$ is called a block partitioning of V if

$$\bigcup_{i \in H} B_i = V$$

and

$$B_i \cap B_j = \emptyset, \quad \forall i \neq j, \quad i, j \in H.$$

For any point $x \in R_n$ feasible in 4.10, the set of nodes V can be uniquely partitioned into disjoint subsets using the following principles [119]:

- if $j \in i^-$ and $x_i = x_j$, then i and j belong to the same subset;
- if $x_i \neq x_j$ then i and j belong to different subsets;
- if there is no path connecting the node i and j, they belong to different subsets;

It is easy to see that these subsets are blocks and that for any block B_k

$$x_i = U_k \quad \forall i \in B_k$$

where U_k denotes the common value of the components x associated to the block B_k. U_k is computed as follow:

$$U_k = \frac{\sum_{i \in B_k} w_i y_i}{W_k} \tag{4.12}$$

where W_k

$$W_k = \sum_{i \in B_k} w_i$$

denote the weight of the block B_k.

The pseudo-code of the *GPAV* algorithm is shown in Figure 4.7 (adapted from [119]). *GPAV* assumes that the nodes V of the directed acyclic graph $G(V, E)$ have a topological order. *GPAV* starts by setting $B_i = i$ and $B_i^- = i^- \forall i \in V$ and then works with blocks. *GPAV* treats the blocks in the order consistent with the topological sort, i.e. $B_1, B_2, B_3 \ldots B_{|V|}$. For each block B_k its common value 4.12 is compared with those of its adjacent blocks. While exists an adjacent violator of the monotonicity, the block B_k *absorbs* the one with the most severe violation and inherits the list of blocks adjacent to the absorbed one. Therefore, the common value U_k is updated. The "absorption" operation is repeated until all the constraints involving a block B_k and its adjacent ones are satisfied. *GPAV* algorithm deals with a reduced acyclic graph of blocks, which is initially identical to $G(N, E)$. The graph shrinks whenever one block absorbs another one. This assures the low computational complexity of the *GPAV* algorithm, that is estimated to be $\mathcal{O}(|V|^2)$ [119].

Figure 4.7: Generalized Pool-Adjacent-Violators (GPAV)

Input:

- $G =< V, E >$
- $V = \{1, 2, \ldots, |V|\}$ *topologically ordered*;
- $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >, \quad \hat{y}_i \in [0, 1]$
- $\boldsymbol{w} =< w_1, w_2, \ldots, w_{|V|} >, \quad w_i \in [0, 1]$

```
begin algorithm
```

01: set $H = V$

02: $\forall i \in V$ set $B_i = \{i\}$; $B_i^- = i^-$; $U_i = \hat{y}_i$; $W_i = w_i$;

03: for each k from 1 to $|V|$ do

04: while exists $i \in B_k^-$ such that $U_i > U_k$ do

05: find $j \in B_k^-$ such that $U_j := max\{U_i : i \in B_k^-\}$

06: $H := H \setminus \{j\}$

07: $B_k^- := B_j^- \cup B_k^- \setminus \{j\}$

08: $U_k := (W_k U_K + W_j U_K)/(W_k + W_j)$

09: $B_k := B_k \cup B_j$

10: $W_k := W_k + W_j$

11: $\forall i \in H$ such that $j \in B_i^-$ set $B_i^- := B_i^- \cup \{k\} \setminus \{j\}$

12: end while

13 $\forall i \in B_k$ and $\forall k \in H$ set $\bar{y} := U_k$

14: end for

```
end algorithm
```

Output:

- $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$

4.7 Isotonic True-Path-Rule for DAG (ISO-TPR)

Simply by replacing the top-down step of the *TPR-DAG* algorithm (block C of the pseudo-code shows in Figure 4.5) with the *GPAV* approach (Figure 4.7) we design the *ISO-TPR* algorithm, whose pseudo-code is shown in Figure 4.8.

The most important feature of *ISO-TPR* is that it maintains the hierarchical constraints by construction and selects the closest solution (in the sense of the least squared error) to the flat predictions that obeys to the true path rule.

Variants of the *ISO-TPR* algorithm can be easily designed. For instance, if in the bottom-up step of the *ISO-TPR* pseudo-code (block B) we consider the "positive" descendants instead of the "positive" children we design the algorithm variant *ISO-DESCENS*. Obviously, the choice of the "positive" children (or descendants) can be done by adopting any of the strategies shown in Section 4.4. Finally, it easy to see the overall complexity of the *ISO-TPR* algorithm is $\mathcal{O}(|V|^2)$.

Figure 4.8: Isotonic True-Path-Rule for DAG (ISO-TPR)

Input:

- $G =< V, E >$
- $V = \{1, 2, \ldots, |V|\}$
- $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >, \quad \hat{y}_i \in [0, 1]$
- $\boldsymbol{w} =< w_1, w_2, \ldots, w_{|V|} >, \quad w_i \in [0, 1]$

begin algorithm

01: A. $dist := \forall i \in V$ ComputeMaxDist $(G, root(G))$

02: B. Per-level bottom-up visit of G:

03: for each d from $\max(dist)$ to 0 do

04: $N_d := \{i | dist(i) = d\}$

05: for each $i \in N_d$ do

06: Select the set ϕ_i of "positive" children

07: $\bar{y}_i := \frac{1}{1+|\phi_i|}(\hat{y}_i + \sum_{j \in \phi_i} \bar{y}_j)$

08: end for

09: end for

10: C. GPAV algorithm

12: $\hat{\boldsymbol{y}} := \bar{\boldsymbol{y}}$

14: $V = \{1, 2, \ldots, |V|\}$ *topologically ordered*;

14: $H := V$

15: $\forall i \in V$ set $B_i = \{i\}$; $B_i^- = i^-$; $U_i = \hat{y}_i$; $W_i = w_i$;

16: for each k from 1 to $|V|$ do

17: while exists $i \in B_k^-$ such that $U_i > U_k$ do

18: find $j \in B_k^-$ such that $U_j := max\{U_i : i \in B_k^-\}$

19: $H := H \setminus \{j\}$

20: $B_k^- := B_j^- \cup B_k^- \setminus \{j\}$

21: $U_k := (W_k U_K + W_j U_K)/(W_k + W_j)$

22: $B_k := B_k \cup B_j$

23: $W_k := W_k + W_j$

24: $\forall i \in B_k$ and $\forall k \in H$ set $\bar{y} := U_k$

25: end while

26: $\bar{y} := U_k \quad \forall i \in B_k$ and $\forall k \in H$

27: end for

end algorithm

Output:

- $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$

4.8 Correctness and Consistency of the Predictions

Hierarchical ensemble methods can improve flat predictions by reducing both the number of false positives (FP) and of false negatives (FN). For instance, the Figure 4.9 shows that hierarchical ensembles can correct FN flat predictions to TP for the gene *RGS9* (regulator of G-protein signaling 9) that encodes a member of the *RGS* family of GTPase whose mutations cause bradyopsia [134]. Instead by looking at the Figure 4.10 we can observe the ability of

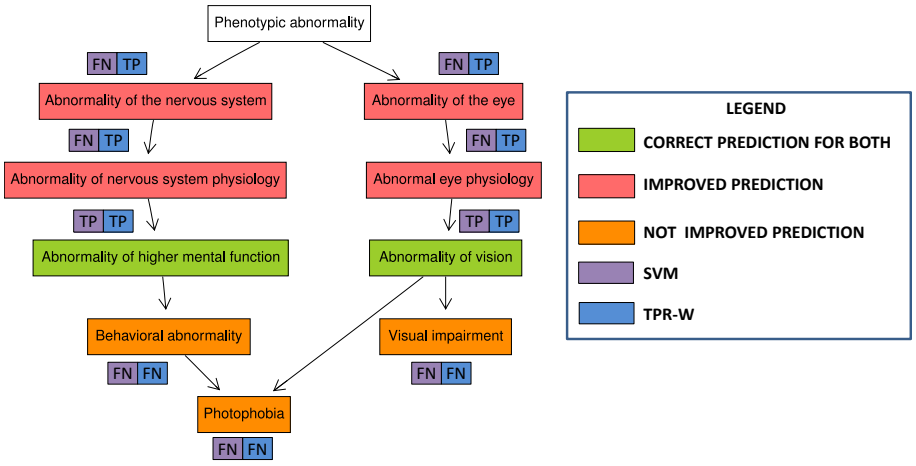


Figure 4.9: Flat and hierarchical (*TPR-W*) *HPO* predictions for the gene *RGS9*. The boxes close to each *HPO* term display the correct (TP or TN) or incorrect (FP or FN) predictions made respective by flat-*SVM* (purple rectangles) and by hierarchical-*TPR-W* (blue rectangles). Green rectangles represent correct predictions for both flat and hierarchical methods; light-red rectangles represent the predictions that the hierarchical method was able to correct respect to flat method ($FN \rightarrow TP$), while the orange rectangles represent the incorrect predictions that the hierarchical method was not able to correct respect to the flat method.

hierarchical ensembles of correcting FP to TN for the gene *ENAM* (enamelin) that encodes the largest protein in the enamel matrix whose deficiency is associated with *amelogenesis imperfecta type 1C* [135]. More precisely, the ensemble variant *TPR-W* was able to "recover" four TP and correct six FP to TN (red rectangles of Figure 4.9 and Figure 4.10). It is worth nothing that the hierarchical ensemble methods can improve but they cannot always guarantee the correctness of all the predictions. Indeed when the flat predicted scores are too "weak", the hierarchical ensembles algorithms may fail in improving the recovery of FP or FN. For example, in Figure 4.9 and Figure 4.10 we can observe how *TPR-W* fails in removing respectively three FN and FP (orange rectangles).

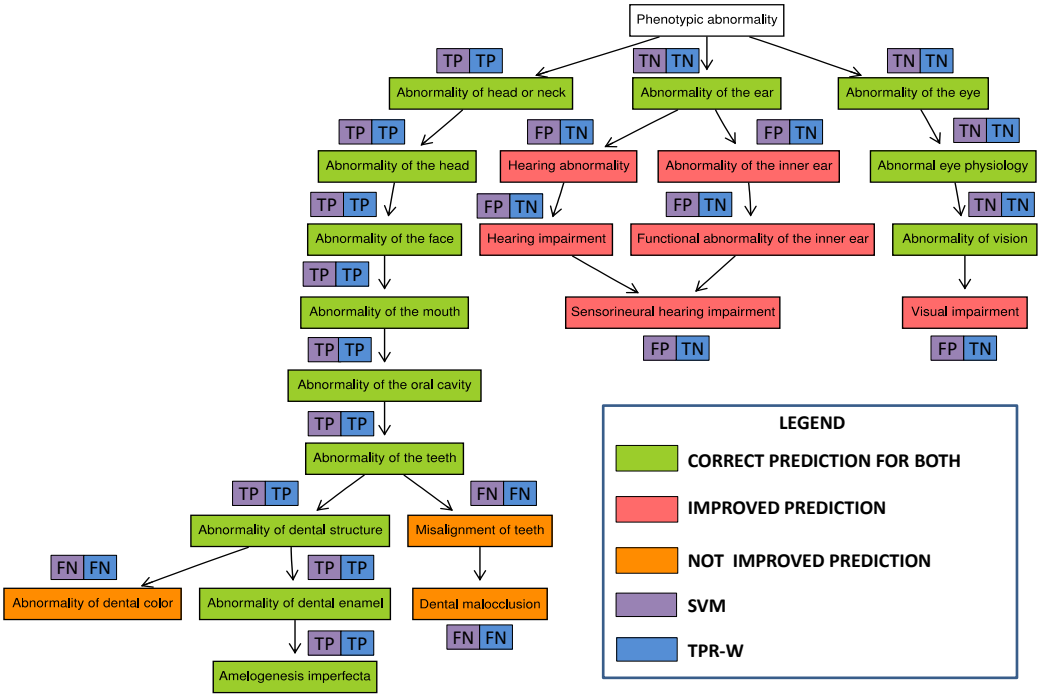


Figure 4.10: Flat and hierarchical (*TPR-W*) *HPO* predictions for the gene *ENAM*. The boxes close to each *HPO* term display the correct (TP or TN) or incorrect (FP or FN) predictions made respective by flat-*SVM* (purple rectangles) and by hierarchical-*TPR-W* (blue rectangles). Green rectangles represent correct predictions for both flat and hierarchical methods; light-red rectangles represent the predictions that the hierarchical method was able to correct respect to flat method ($FP \rightarrow TN$), while the orange rectangles represent the incorrect predictions that the hierarchical method was not able to correct respect to the flat method.

On the other hand, the hierarchical ensemble methods can guarantee the consistency of the predictions, i.e they provide predictions that always obey the true path rule. To this end we must visit the hierarchical taxonomy according to the maximum and not the minimum distance from the root. The Figure 4.11 shows an intuitive example of this fact. Indeed by looking at

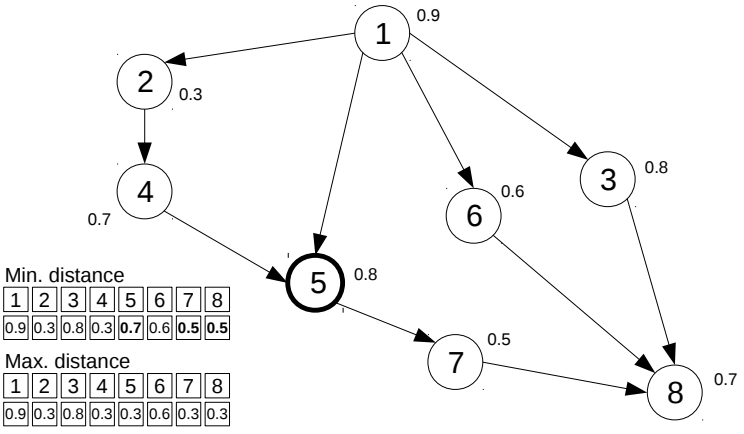


Figure 4.11: Levels defined in terms of the minimum distance from the root (node 1) lead to inconsistent predictions. The small numbers close to nodes correspond to the $\hat{y}_i$ scores of the flat predictions. The Hierarchical top-down scores obtained respectively by crossing the levels according to the minimum and the maximum distance from the root are shown in the bottom-left. Scores in boldface represent inconsistent predictions.

the *HTD-DAG* scores obtained respectively with the minimum and maximum distance from the root in the bottom-left corner of the Figure 4.11), we can see that only the maximum distance preserves the consistency of the predictions. For instance, focusing on node 5, by traversing the *DAG* levels according to the minimum distance from the root, we have that the level of node 5 is 1 ($\psi^{min}(5) = 1$) and in this case by applying the *HTD-DAG* rule (eq. 4.2) the flat score $\hat{y}_5 = 0.8$ is wrongly modified with the *HTD-DAG* ensemble score $\bar{y}_5 = 0.7$. If we instead traverse the *DAG* levels according to the maximum distance from the root, we have $\psi^{max}(5) = 3$ and the *HTD-DAG* ensemble score is correctly set to $\bar{y}_5 = 0.3$. In other words at the end of the *HTD-DAG*, by traversing the levels according to the minimum distance we have $\bar{y}_5 = 0.7 > \bar{y}_4 = 0.3$. Therefore a child node has a score larger the score of its parent and the true path rule is not preserved. On the contrary by traversing the levels according to the maximum distance we achieve $\bar{y}_5 = 0.3 \leq \bar{y}_4 = 0.3$ and the true path rule consistency is assured. This is due to the fact that by adopting the minimum distance when we visit node 5, node 4 has not just been visited, and hence the value 0.4 has not been transmitted by node 2 to node 4; on the contrary if we visit the DAG according to the maximum distance all the ancestors of node 5 (including node 4) have just been visited and the score 0.4 is correctly transmitted to node 5 along the path $2 \rightarrow 4 \rightarrow 5$.

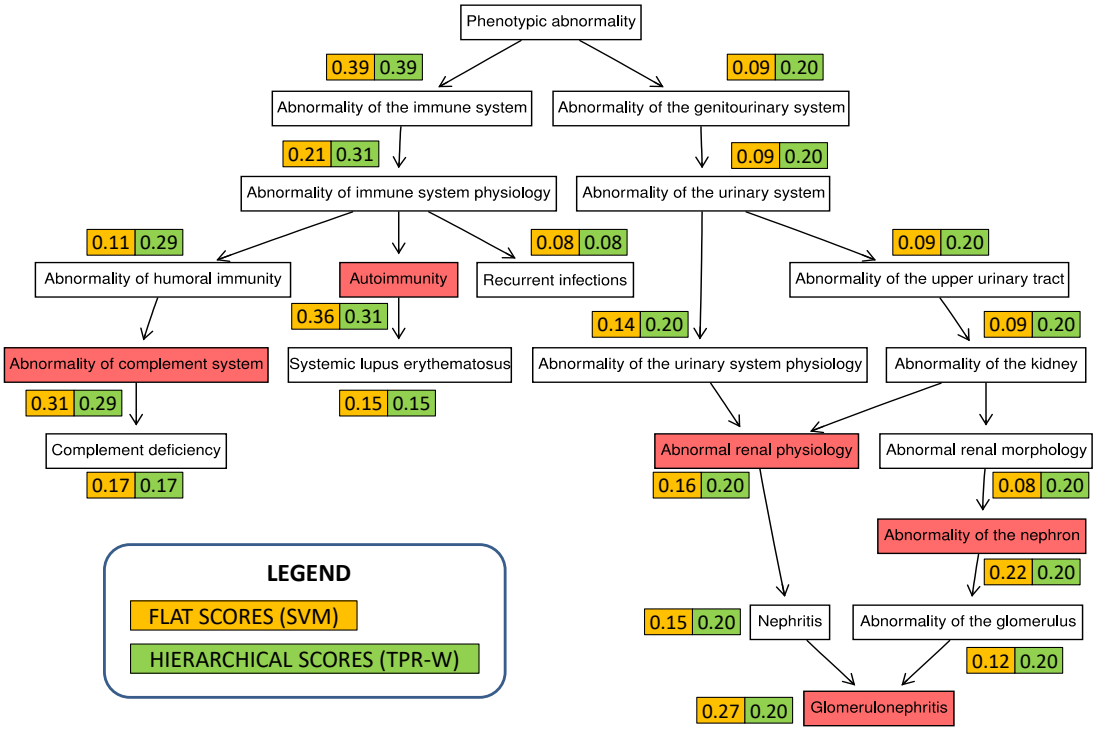


Figure 4.12: Flat and hierarchical (TPR-W) HPO predictions for the gene *C1QC*. The numbers close to each predicted HPO term represent flat (yellow rectangles) and hierarchically corrected (green) scores. The *TPR-W* predictions obey the true-path rule (the scores of the parent nodes are always larger or equal than that of their children nodes), while flat predictions are inconsistent for 5 HPO terms highlighted in light-red: Autoimmunity, Abnormality of complement system, Abnormal renal physiology, Abnormality of the nephron and Glomerulonephritis.

Instead Figure 4.12 shows a real example of the capability of obtaining hierarchically corrected consistent predictions starting from inconsistent flat predictions by considering the gene *C1QC* (complement C1q C chain), that encodes a 18 polypeptide chains protein whose deficiency is associated with lupus erythematosus and glomerulonephritis [136].

To prove that the hierarchical ensemble methods provide consistent prediction with the topology of the ontology, we need the following lemma:

Lemma 1. *Given a DAG* $G =< V, E >$*, a level function* ψ *that assigns to each node its maximum path length from the root (equation 4.3), it holds that* $\forall i \in V$*,* $\psi(j) < \psi(i)\ \forall j \in par(i)$*.*

Proof. The proof is based on the optimal-substructure property holding for the longest path problem in *DAGs*, that is a longest path between two vertices contains other longest path within it [137]. Indeed, let $\bar{p}(r, i)$ be the longest path from $r = root(G)$ to node $i \in V$, and suppose that there exists $j \in par(i)$ such that $\psi(j) \geq \psi(i)$. Let $\bar{p}(r, j)$ be the path between r and j whose length is $\psi(j)$ (that is the longest path between them). Note that the path $p(r, j)$ does not contain the node i, otherwise the DAG would contain a cycle. By adding the edge (j, i) to $\bar{p}(r, j)$, we obtain a path from r to i whose length is $\psi(j) + 1 > \psi(i)$, which contradicts the hypothesis that $\bar{p}(r, i)$ is the longest path between nodes r and i. □

By using Lemma 1, we can prove that the the top-down visit of the *DAG* obeys the true path rule:

Theorem 1. *Given a DAG* $G =< V, E >$*, a level function* ψ *that assigns to each node its maximum path length from the root and the set of HTD-DAG flat predictions* $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >$*, the top-down hierarchical correction of the HTD-DAG algorithm assures that the set of ensemble predictions* $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$ *satisfies the following property:*

$$\forall i \in V,\ j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$$

Proof. For an arbitrary node $i \in V$, when it is processed by the top-down step of *HTD-DAG* algorithm, we may have two basic cases:

1. $i \in root(G)$. By applying the *HTD-DAG* rule (equation 4.2) we set $\bar{y}_i := \hat{y}_i$ and the property $j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$ trivially holds, since $par(i) = \emptyset$.

2. $i \notin root(G)$. We may have two cases:

   (a) $\hat{y}_i \leq \min_{j \in par(i)} \hat{y}_j$. In this case the *HTD-DAG* rule (4.2) sets $\bar{y}_i := \hat{y}_i$ and hence it holds that $j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$.

   (b) $\hat{y}_i > \min_{j \in par(i)} \bar{y}_j$. In this case by applying (4.2) we have $\bar{y}_i := min_{j \in par(i)} \bar{y}_j$ and hence also in this case the property $j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$ holds.

Summarizing, in all cases we have that $j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$, after the node i has been processed. Moreover, we note that for the currently processed node i both $\bar{y}_i$ and $\bar{y}_j,\ j \in par(i)$ will not be further changed by the "per level" top-down visit of the *HTD-DAG* algorithm. Indeed, the score $\bar{y}_i$ is modified only once, since each node is visited exactly one time (each node belongs to one and only one level of the hierarchy); moreover, since the visit is top-down, Theorem 1 implies that parent nodes are processed before their children, and hence also the scores $\bar{y}_j$ of the nodes $j \in par(i)$ will not be further changed, since $j \in par(i)$ have just been visited and their scores $\bar{y}_j$ have just been set before visiting node i. As a consequence, once a node i is visited the property $j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$ will hold till to the end of the algorithm. Finally, since the top-down step of the algorithm visits each node exactly one time, at the end of this step the property $j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$ holds for each node $i \in V$. □

From Theorem 1 it is easy to prove that the consistency of the predictions holds for all the ancestors of a given node $i \in V$.

Corollary 1. *Given a DAG $G =< V, E >$, the level function ψ and the set of flat predictions $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >$, the HTD-DAG algorithm assures that for the set of ensemble predictions $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$ the following property holds: $\forall i \in V,\ j \in anc(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$.*

Proof. The corollary can be proven by "reductio ad absurdum" from Theorem 1. We suppose that for an arbitrary node i does exist a node $z \in anc(i)$ such that $\bar{y}_z < \bar{y}_i$. Let us consider all the edges (k, l) included in the path $\bar{p}(z, i)$ connecting node z with node i. Without loss of generality, we focus on a specific path, since we can repeat the same reasoning for any path connecting z with i. We claim that $\exists (k, l) \in \bar{p}(z, i)$ such that $\bar{y}_k < \bar{y}_l$, and we show this again by "reductio ad absurdum". By absurd we suppose that $\forall (k, l) \in \bar{p}(z, i)$ we have $\bar{y}_k \geq \bar{y}_l$. By transitivity along the path $\bar{p}(z, i)$, we obtain that $\bar{y}_z \geq \bar{y}_i$, but this contradicts our first hypothesis that $\bar{y}_z < \bar{y}_i$ and hence it does exist an edge $(k, l) \in \bar{p}(z, i)$ such that $\bar{y}_k < \bar{y}_l$. But for Theorem 1 it is not possible that $\bar{y}_k < \bar{y}_l$, since $k \in par(l)$. Since this contradiction comes from the assumption that does exist a node $z \in anc(i)$ such that $\bar{y}_z < \bar{y}_i$, it follows that $\forall i \in V,\ j \in anc(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$. □

Independently of the choice of the positive children, the following consistency theorem holds for *TPR-DAG* and its variants:

Theorem 2. *Given a DAG $G =< V, E >$, a set of flat predictions $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >$ for each class associated to each node $i \in \{1, \ldots, |V|\}$, the TPR-DAG algorithm assures that for the set of ensemble predictions $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$ the following property holds: $\forall i \in V,\ j \in anc(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$.*

The proof is substantially the same of Theorem 1 and is omitted for brevity.

It is worth noting that *HTD-DAG* and *TPR-DAG* algorithms hold the following properties:

Lemma 2. *Given a DAG $G =< V, E >$, a set of flat predictions $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >$ for each class associated to each node $i \in \{1, \ldots, |V|\}$, a set of ensemble predictions $\bar{\boldsymbol{y}} =<$*

$\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$ *for the HTD-DAG and a set of ensemble predictions* $\tilde{\boldsymbol{y}} =< \tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{|V|} >$ *for the TPR-DAG with "positive" children selected according to eq. 4.7, we have that* $\forall i \in V, \quad \tilde{y}_i \geq \bar{y}_i$.

The proof is based on the fact that the bottom-up step of *TPR-DAG* can only increment the scores $\tilde{y}_i$ with respect to the flat predictions $\hat{y}_i$. Hence the successive top-down step of the *TPR-DAG* starts from higher scores than that of the *HTD-DAG*, and the applied top-down procedure is the same for both algorithms.

A good property of *TPR-DAG* is that its sensitivity is always equal or better than that of *HTD-DAG*:

Theorem 3. *The TPR-DAG ensemble algorithm with "positive" children selected according to eq. 4.7, achieves always a sensitivity equal or higher than the HTD-DAG ensemble algorithm.*

Proof. From Lemma 2 we have that $\forall i \in V, \quad \tilde{y}_i \geq \bar{y}_i$. Hence the *TPR-DAG* ensemble algorithm, with respect to the *HTD-DAG* algorithm:

a) increments or maintains equal the number of true positives;

b) decreases or maintains equal the number of false negatives.

By definition of the sensitivity *TPR-DAG* achieves a sensitivity equal or higher than the *HTD-DAG*. □

Unfortunately there is no guarantee that the precision of *TPR-DAG* is always larger or equal than that of the *HTD-DAG* algorithm.

Finally, it is immediate to see that all the aforementioned theorems, corollaries and lemmas hold also for *GPAV* and *ISO-TPR* algorithm.

Chapter 5

Hierarchical Prediction of GO terms

The Automated Protein Function Prediction (*AFP*) is a complex multi-class and multi-label classification problem characterized by several features, such as the hierarchical organization of protein functions, the integration of several heterogeneous data source (e.g. genomic, proteomic, transcriptomic), the lack of annotated proteins for most biological functions with a consequently imbalance of functional classes, with rare positive instances and not uniquely defined negative instances. Furthermore, the hierarchical relationships between functional classes that characterizes the Gene Ontology (*GO*) [2], motivate the development of hierarchy-aware methods, since they capture the inherent structure of the annotation space. Nevertheless, although protein functions are unmistakably dependent (for instance, the *GO* terms *protein kinase activity* and *protein-tyrosine kinase activity* are clearly related, with the first the parent function of the latter), most of the approaches proposed in the literature, ranging from sequence-based methods [12–14] to network-based methods [15, 16, 138], predicted protein functions independently each from the other. However, two international challenges for the Critical Assessment of Function Annotation (*CAFA* [30] and *CAFA*2 [31]), organized to compare and evaluate computational methods that automatically assign to a protein its function, emphasized the need of using output-structured learning algorithms to predict a subgraph of *GO* terms, starting from a given protein. In this chapter we will exactly tackle this problem, putting a particular emphasis on the comparison between methods "hierarchical-unaware" and "hierarchical-aware".

5.1 Experimental Design

The experiments presented here aim at showing that the hierarchical ensemble methods proposed in Chapter 4 are able to provide consistent predictions with respect the underlying *GO* ontology and can improve upon flat predictions independently of the choice of the base learner. We compared the generalization performance of our hierarchical ensemble method versus several machine learning-based flat approaches by using a classical 5-fold cross-validation procedure. To reduce the computational burden we applied simple univariate feature selection methods. To assess the soundness of the hierarchical ensemble methods proposed in Chapter 4, we predicted the protein function of several species belonging to the *Animalia* kingdom, ranging from invertebrates to vertebrates. More precisely, for our experiments we used the following six model organisms: *C. elegans* (*CAEEL*), *G. Gallus* (*CHICK*), *D. rerio* (*DANRE*), *D. melanogaster* (*DROME*), *H.*

sapiens (*HUMAN*), *M. musculus* (*MOUSE*). From a computational standpoint, the considered task is intensive, since overall we considered over than 100 thousands of proteins (see Table 5.1) and more than 15 thousands of functional *GO* terms (by considering the *GO* terms having 10 or more annotations, see Table 5.2).

5.2 Data and Annotations

STRING is a database of known and predicted functional protein-protein interactions. The interactions stem from high-throughput experimental data, from the mining of databases and literature, from predictions based on genomic context analysis, from knowledge transfer between organisms and from interactions aggregated from other primary databases [139]. Consequently, the interaction network provided by *STRING* is built by integrating different heterogeneous sources of information.

The Gene Ontology (*GO*) [2] is the current standard for annotating gene products and proteins in a species independent manner. It is a structured vocabulary with thousands of terms which describes different aspects of the protein function using a hierarchy of keywords. It is composed of three independent subontologies for annotating the molecular functions (*MF*) of proteins, the biological processes (*BP*) they participate in, and the cellular components (*CC*) in which these occur. Examples of tuple of *GO* terms (*MF*, *BP*, *CC*) are for instance, (*RNA* binding, chromosome segregation, nucleus) or (citrate synthase activity, Krebs cycle, mitochondrion).

5.2.1 Protein-Protein Interaction Network Data

As datasets, for each considered model organism, we downloaded from the *STRING* website the most updated protein-protein interaction network (version 10.5) [140]. Each protein-protein interaction in the *STRING* database is annotated with a confidence score, ranging by default between zero and one-thousand and that we scaled between zero and one, representing the weight of the edges connecting a pair of proteins [141]. The *STRING* protein-protein interactions are derived from different data sources, such for instance co-expression, co-occurrence, gene-fusion and conserved genomic neighborhood. By combining these evidences types (named "evidence channels"), *STRING* provides, for each protein-protein interaction, a "combined score", that is the final measure used to weight the edges of the network. For a full and detailed description about the computation of the *STRING* edge score we refer the reader to [139]. Finally, it is worth noting that the *STRING* interactions have a gene-locus resolution. This means that for each gene *STRING* takes into account only the longest protein in terms of number of amino acids independently by the coverage of the exons set defined by the set of transcript encoded by gene. In the Table 5.1 are shown the features of the *STRING* protein-protein interaction network of the model organisms that we took into account in the experiments.

Organism	Proteins	Interactions
CAEEL (*C. elegans*)	15,752	2,889,134
CHICK (*G. Gallus*)	14,093	3,172,417
DANRE (*D. rerio*)	23,449	13,187,568
DROME (*D. melanogaster*)	13,702	3,981,727
HUMAN (*H. sapiens*)	19,576	5,676,528
MOUSE (*M. musculus*)	21,151	6,307,021

Table 5.1: The column **Organism** refers to the mnemonic name of the model organism such as specified in the UniProtKB, whereas in brackets is reported the Latin scientific name; the column **Proteins** and **Interactions** refer respectively to the number of Protein and Interactions in the *STRING* network of the considered organism.

5.2.2 GO DAG and Annotations

The experiments presented here are based on the December 2017 *GO* release by considering separately the three main *GO* domains: Biological Process (*BP*), Molecular Function (*MF*) and Cellular Component (*CC*). From the *GO* obo file we extracted both the is_a and the $part\_of$ relationships, since it is safe grouping annotations by using both these *GO* relationships. However is important to mention that whereas the is_a relations are disjoint (i.e. do not exist is_a relationships that operate between terms of different ontologies), the $part\_of$ relations may operate between terms of different ontologies. Because we considered the three *GO* subontologies separately each from the other, this aspect could lead to graphs with inconsistent nodes, i.e. graphs whose nodes are unreachable from the root. Hence we took the inconsistent nodes away by applying the Dijkstra's shortest paths algorithm. At the end we obtained a *BP* directed graph having $29,678$ nodes and $62,544$ edges, a *MF* directed graph with $12,147$ nodes and $14,854$ edges and a *CC* directed graph having $4,150$ nodes and $7,527$ edges.

For every organism, we downloaded from the Gene Ontology Annotation (*GOA*) website, the protein-*GO* term associations, released on the 20th of December 2017. We extracted just the experimentally supported annotations, i.e. the annotations that are directly supported by experimental evidences. The Experimental Evidence codes used to annotate the proteins are the following: (i) Inferred from Experiment (*EXP*), (ii) Inferred from Direct Assay (*IDA*), (iii) Inferred from Physical Interaction (*IPI*), (iv) Inferred from Mutant Phenotype (*IMP*), (v) Inferred from Genetic Interaction (*IGI*), (vi) Inferred from Expression Pattern (*IEP*).

The *GOA* database [142] provides high-quality *GO* annotations for proteins in the UniProt Knowledgebase (UniProtKB) [143]. Consequently, from the *GOA* website, we downloaded also the UniProtKB identifier mapping file, to map the UniProtKB proteins annotated with a *GO* term versus the *STRING* proteins, using as identifiers respectively the UniProtKB accession number (*UNIPROT-AC*) and the locus *STRING-ID*. Moreover, to limit as much as possible the number of the unmapped identifiers, we designed a precise pipeline whose procedure is schematically illustrated in Figure 5.1. For the detailed step-by-step explanation, please refer to the Appendix Section A.1.

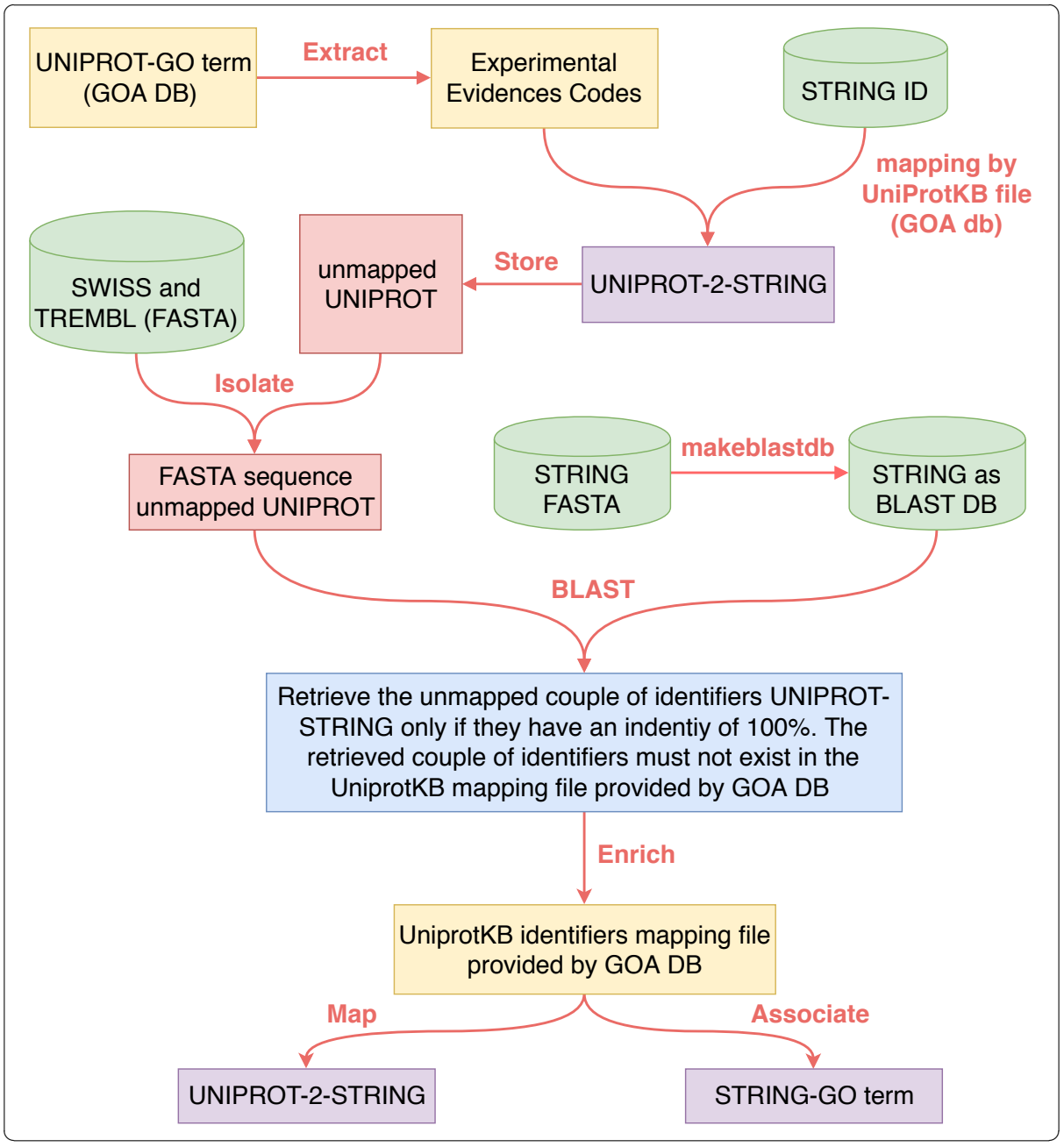


Figure 5.1: Pipeline for the data preparation for the experiments presented here. See Appendix Section A.1 for the detailed step-by-step explanation.

Furthermore, we performed the transitive closure of annotations, i.e. we transfered the most specific annotations from leaves to ancestors. Finally, to avoid the prediction of those *GO* terms having too few annotations for a reliable assessment and in order to consider the terms with a minimum amount of prior information we pruned all the *GO* terms having less than 5 and 10 annotations. In the Section 5.5.3 we will motivate the choice of the *GO* terms that we used in the experiments. The propriety of the graphs and of the annotations table, before and after applying these filters, are shown in Table 5.2. It is important to note that since the transitive closure of annotations was performed before applying the filters, the numbers of proteins annotated

with 5 ore more *GO* terms is equal to the numbers of proteins annotated with 10 ore more *GO* terms. For the sake of the simplicity, in Table 5.2 we just show the number of proteins having at least one experimental annotation when only *GO* terms with at least 5 annotations are taken into account. Finally, we chose negative examples (proteins) according to a "basic" selection strategy [20], this means that for positives we used all the instances annotated with a *GO* term, while for negatives we simply used all the not annotated instances. In theory other more refined strategies to select negatives may lead to improved performances [144, 145].

Organism	**Domain**	**Classes**	**Edges**	**Classes** ≥ 5	**Edges** ≥ 5	**Classes** ≥ 10	**Edges** ≥ 10	**Proteins** ≥ 5
CAEEL	BP	4068	8066	2013	3845	1335	2458	2597
	MF	1163	1567	341	434	186	231	1806
	CC	578	1082	321	585	221	392	1924
CHICK	BP	2309	4401	503	856	246	411	330
	MF	520	690	94	113	43	53	292
	CC	319	588	106	168	58	88	318
DANRE	BP	4441	8492	1804	3268	1182	2088	3351
	MF	865	1153	239	305	118	153	1000
	CC	291	528	134	231	89	145	550
DROME	BP	6029	11850	3104	5943	2244	4197	5238
	MF	1736	2295	530	701	327	419	2928
	CC	899	1720	515	943	348	614	3748
HUMAN	BP	11036	22292	5143	9903	3460	6492	8169
	MF	3405	4482	1209	1588	760	988	11203
	CC	1429	2665	788	1484	541	1001	9412
MOUSE	BP	11786	23820	5717	11046	3899	7372	8139
	MF	2522	3349	851	1120	511	662	7248
	CC	1137	2163	629	1191	445	818	7496

Table 5.2: Propriety of the graph and annotation table for each of the considered model organisms. The first column refer to the name of the model organism; the column **Domain** refers to one of the *GO* domains (*BP*, *MF*, *CC*); the column **Classes** refers to the number of *GO* classes/terms having at least one experimental annotations; the column **Edges** refers to whole edges number in the *GO* graph; the column **Classes** ≥ 5 refers to the number of *GO* terms having 5 or more experimental annotations; the column **Edges** ≥ 5 refers to the number of edges of the subgraph whose nodes having 5 or more experimental annotations and finally the column **Proteins** ≥ 5 refers to the number of protein having at least one experimental annotation when only terms with at least 5 annotations are considered.

5.3 Evaluation Metrics

We evaluated the generalization performance of the methods considering two different scenarios, i.e. (i) *term-centric* and (ii) *protein-centric* measure [31]. These two types of evaluations were chosen to address the following related questions: (i) what are the gene products associated with a specific functional *GO* term and (ii) what is the function of a particular protein. Then *term-centric* evaluation is more appropriate for the gene prioritization scenario, whereas the *protein-centric* evaluation is more related to the protein function prediction mode.

1. Term-centric metric. For each *GO* term we computed the classical Aurea Under the *ROC* Curve (*AUROC*) and the Area Under the Precision Recall Curve (*AUPRC*) to take into account the imbalance of annotated versus unannotated *GO* terms. Indeed the *AUPRC* is more informative on unbalanced dataset than the classical *AUROC* [146]. The *ROC* curve is computed by plotting the *recall* (or *sensitivity*) against the *false positive rate* (or $1 - specificity$) at different different threshold; whereas the precision-recall curve shows the trade-off between precision and recall at different cutoffs. For a particular functional term i, the recall (Rc), the specificity (Sp) and the precision (Pc) at a given score threshold $\tau \in [0, 1]$ are defined as follow:

$$Rc_i(\tau) = \frac{\sum_j \mathbb{1}(y_i^{g_j} > \tau \wedge g_j \in i)}{\sum_j \mathbb{1}(g_j \in i)} \tag{5.1}$$

$$Sp_i(\tau) = \frac{\sum_j \mathbb{1}(y_i^{g_j} \leq \tau \wedge g_j \notin i)}{\sum_j \mathbb{1}(g_j \notin i)} \tag{5.2}$$

$$Pc_i(\tau) = \frac{\sum_j \mathbb{1}(y_i^{g_j} > \tau \wedge g_j \in i)}{\sum_i \mathbb{1}(i \in P_{g_i}(\tau))} \tag{5.3}$$

where $y_i^{g_j} \in [0, 1]$ is the predicted score (probability) that gene product g_j is associated with the *GO* term i, $P_{y_i}(\tau)$ denotes the set of gene product that have a predicted scores greater than or equal to τ for a given *GO* term i, $\tau \in [0, 1]$ is the given threshold and $\mathbb{1}$ indicates the standard indicator function. Both the *AUROC* and the *AUPRC* can assume values in $[0 \dots 1]$. *AUROC* values close to 0.5 denote random predictions and *AUROC* values substantially larger than 0.5 denote good predictive ability. Instead a high area under the precision-recall curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate.

2. Gene-centric metric. Precision (Pr), Recall (Rc) and the maximum achievable $F - score$ (F_{max}) using thresholds $\tau \in [0, 1]$ are calculated as follows:

$$Pr(\tau) = \frac{1}{N} \sum_{j=1}^{N} \frac{\sum_i \mathbb{1}(i \in P_{g_j}(\tau) \wedge i \in T_{g_j})}{\sum_i \mathbb{1}(i \in P_{g_j}(\tau))} \tag{5.4}$$

$$Rc(\tau) = \frac{1}{N} \sum_{j=1}^{N} \frac{\sum_i \mathbb{1}(i \in P_{g_j}(\tau) \wedge i \in T_{g_j})}{\sum_i \mathbb{1}(i \in T_{g_j}))} \tag{5.5}$$

$$F_{max} = max_\tau \left\{ \frac{2 \cdot Pr(\tau) \cdot Rc(\tau)}{Pr(\tau) + Rc(\tau)} \right\} \tag{5.6}$$

where i denotes a protein function (*GO* term) in the gene ontology (excluding the root node), $P_{g_j}(\tau)$ denotes the set of terms that have a predicted scores greater than or equal to τ for a given protein g_j, T_{g_j} denotes the set of experimentally determined terms for a given gene product g_j, N the number of examples having at least one annotation with an *GO* term and $\mathbb{1}$ indicates a standard indicator function. In other words the F_{max} measure is the maximum hierarchical F-score achievable by "a posteriori" setting the optimal decision threshold [31].

We warn the reader that the hierarchical $F - score$ as defined in eq. 5.6 provides an over-optimistic assessment of the hierarchical score. Indeed the threshold τ in eq. 5.6 is by definition "a posteriori" selected, by choosing the optimal τ^* that optimizes the $F - score$ having a set of pre-computed scores. An unbiased evaluation should embed the selection of the optimal τ within the learning process. Nevertheless, we use this measure since it is the reference gene-centric metric within the computational biology community, as witnessed by its usage in the *CAFA*2 international challenge [31], and in the ongoing *CAFA*3 challenge.

For both measures, by averaging across proteins or terms, we can obtain an overall picture of the prediction performance of the methods.

5.4 Estimate of the Overall Time Complexity

Initially, we estimated the overall computational time needed to evaluate the generalization performance of each flat method through a classical 5-fold cross-validation procedure. It is worth noting that here we are not interested in predicting the protein function of a given model organism, but instead we would only like to estimate the overall time complexity required by each flat approach. With this goal in mind we assessed the empirical time complexity by calculating the computational time on a subset of 30 randomly chosen *GO* classes.

At the beginning, we evaluated the time complexity by using all the available functional protein-protein interactions of the *STRING* network of a given organism (Table 5.1). Nevertheless, this resulted unfeasible from a computational standpoint. Let us consider for instance as model organism *C. elegans* (one of the smallest among those considered, see Table 5.1) and as flat classifiers *C5.0 Decision Tree*, *Glmnet*, *Multi Layer Perceptron*, *Support Vector Machine* and *Extreme Gradient Boosting*, by setting their parameters to the default values (see Table 5.3). These algorithms would employ, in average and across the three *GO* subontologies, about 42 days to predict the protein function only on the subset of 30 randomly chosen *GO* terms using a workstation equipped with 12 cores (64bit, 2.60GHz) and 128GB of RAM. Consequently, we did not perform experiments considering other base learners (Table 5.3) and other model organisms (Table 5.1). The detailed results about the computational time using all the available functional protein-protein interactions for the organism *C. elegans* are shown in the Appendix Table A.2.

We used the weighted adjacency matrix representing the *STRING* network by considering the rows as examples (i.e. proteins) and the columns as features. In this way the features of each protein are represented by its functional interactions with all the other proteins of the network. In order to reduce the empirical temporal complexity, we applied simple univariate feature selection methods to choose the most informative features. More precisely, we selected the first 100 top-ranked features by applying the Pearson's correlation coefficient. It is important to note that in such a way we reduced the computational complexity of the algorithms without causing a decreasing in the performance, since we took the less informative features away (data not shown). For instance, by considering the first 100 top-ranked features of *C. elegans*, the same

set of flat classifiers used for the previous estimation, would employ about 7 hours to predict the protein function of the same subset of 30 randomly chosen *GO* terms. The detailed results about the evaluation of the overall time complexity by considering the first 100 top-ranked features are shown in Appendix Table A.3. In this table, for the sake of the visualization, for each flat approach we show only the two pairs model organism-ontology for which we estimated the maximum and the minimum empirical computational estimation. We warn the reader that since the evaluation of the overall computational time strictly depends on the number of *GO* classes and since we selected them randomly, this represents only a "raw" estimation of the empirical overall computational time. The step-by-step procedure used to assess the overall time complexity by selecting the first 100 top-ranked features, is explained in the Appendix Section A.3.

All experiments were performed on a workstation equipped with 12 processors Intel(R) Xeon(R) CPU E5-2630 2.60GHz, with 128GB of RAM and having Ubuntu Linux 14.04 as operating system.

5.5 Experimental Set-Up

The hierarchical ensemble algorithms are two step learning strategies. The first step consists in a flat learning of the ontology terms, while the second step hierarchically combines the predictions. Consequently we organized the experiments as follow: in Section 5.5.1, we introduce the flat classifiers, whereas in Section 5.5.2 we show the hierarchical ensemble methods used to reconcile the flat predictions according to the topology of the ontology. In Section 5.5.3 we explain the experiments setting.

5.5.1 Flat classifiers

As flat methods, used as base learners in the hierarchical ensemble methods proposed in Section 5.5.2, we applied the supervised machine learning methods shown in Table 5.3. Because of the high-dimensionality of our dataset (Table 5.1) and consequently due to the high running time of the flat classifiers, we did not tune the respective learning parameters, but we used their default configuration as specified in Table 5.3. For the sake of simplicity, in the rest of the thesis, we refer to the flat classifiers by using the short form shown in brackets in the column **Flat Classifier** of Table 5.3. To implement the flat classifier shown in Table 5.3 we made use of the **caret** R package (short for Classification And REgression Training) [147], since it provides an uniform interface to execute more than 250 predictive models. We warn the reader that the main goal of this chapter is to show that our hierarchical ensemble methods outperform flat approaches independently of the choice of the base learner. As a consequence we chose a range as broad as possible of flat classifier, ranging from linear classifiers, to probabilistic classifiers, to neural networks, to ensemble of learning machines and to gradient boosting algorithms (see Table 5.3).

Flat Classifier	Reference Paper	Learning Parameter	Set Value
C5.0 Decision Tree (C50)	[148]	mtry	2
Lasso and Elastic-Net Regularize Generalized Linear Models (glmnet)	[149]	alpha	1
		lambda	100
Linear Discriminant Analysis (lda)	[150]	none	none
Logit Boost (logit)	[151]	boosting iterations	10
Multi Layer Perceptron (mlp)	[152]	hidden neurons	5
		hidden layers	1
Naive Bayes (nb)	[153]	Laplace smoothing	0
Random Forest (rf)	[154]	mtry	10
		Trees Numbers	500
		splitting rule	gini
		maximum tree depth	1
Support Vector Machine (svm)	[155]	C	0.001
Bagging Ensemble of Decision Tree (treebag)	[156]	bootstrap replications	25
		mtry	2
		maximum tree depth	30
Extreme Gradient Boosting (xgboost)	[157]	boosting iterations	15
		learning rate	0.3
		L2 Regularization	0
		L1 Regularization	0

Table 5.3: Flat classifiers (in brackets the short form) and their parameter setting configuration used as base learners with the hierarchical ensemble methods.

5.5.2 Hierarchical Ensemble Methods

We applied upon flat scores, the non-parametric hierarchical ensemble methods shown in Table 5.4.

Hierarchical Ensemble Method	Reference Paper	Ensemble Variant	Ensemble Short Name
HTD-DAG	[28]	None	HTD
GPAV-DAG	None	None	GPAV
TPR-DAG	[28]	Threshold-Free (TF)	tprTF
DESCENS	[121]	threshold-Free (TF)	descensTF
ISO-TPR	None	Threshold-Free (TF)	ISOtprTF
ISO-DESCENS	None	Threshold-Free (TF)	ISOdescensTF

Table 5.4: Hierarchical ensemble variants adopted in the experiments presented here. None in the column **Reference Paper** means that the corresponding ensemble methods is presented for the first time in this thesis; None in the column **Ensemble Variant** means that the corresponding hierarchical algorithm does not have variants. Finally, **Ensemble Short Name** refers to the short form with which we call the ensemble variants in section 5.6.

For a detailed description about the hierarchical ensemble variants shown in Table 5.4, please refer to Chapter 4. To execute the hierarchical ensemble variants, we used the in-house developed R package **HEMDAG**, whose source code is publicly available on the CRAN repository under the GNU General Public License, version 3 (GPL-3.0).

5.5.3 Experimental Execution

We evaluated the generalization performance of the flat methods, cross-validating each model on the fourth-fifths of the data (training set) and evaluating the performance on the remaining one-fifths (test data). More precisely we created stratified-folds, that is folds containing the same amount of positives and negatives examples (proteins). In such way, by adopting a 5-stratified-fold cross-validation, we guarantee (in the worst scenario) to have at least 2 positive instances in each fold. In order to reduce the computational time complexity, we carried-out the experiments by using only those *GO* terms having 10 or more annotations (see Table 5.2). Moreover, as explained in the Section 5.4, during the training phase, we did not use all the features available, but instead we selected from the *STRING* network the first 100 top-ranked features by using the classical Pearson's correlation coefficient. The supervised feature selection method was cross-validated in an unbiased way, since we chose the top-ranked features during the training phase and then we used the selected features in the test phase [158]. However this entails to repeat the feature selection "on the fly" in each training fold of the cross-validation, with a consequent selection of diverse top-ranked features in every training set.

Due to the high running time of the flat classifiers, we did not tune their learning parameters, but we adopted the default parameter settings (Table 5.3). Furthermore, for a fair comparison with the flat methods, we used parameter-free hierarchical ensemble variants as well. In such a way, we avoid that the improvement yield by the hierarchical ensemble methods is due to the tuning of their hyperparameters.

Finally, since the supervised base learners reported in 5.3 return a probability or a score that a gene product belongs to a given functional class, we did not normalize the flat scores according to any procedure and we normalized the flat scores of each *GO* classes by dividing the score values for the maximum score of that class [28].

5.6 Experimental Results

The heatmaps illustrated in Figures 5.2, 5.3, 5.4 show that our hierarchical ensemble methods improve flat predictions independently of the choice of the base learner. Furthermore, the improvement occurs in all the considered model organisms and across all the three *GO* domains. Let us consider for instance the *AUPRC* as performance metric. The values of an heatmap cell of Figure 5.2 is computed as specified in formula 5.7:

$$HeatmapCell[i,j] = \frac{\overline{AUPRC}_{hier_i} - \overline{AUPRC}_{flat_j}}{max(\overline{AUPRC}_{hier_i}, \overline{AUPRC}_{flat_j})} \tag{5.7}$$

where $\overline{AUPRC}$ is the average *AUPRC*, i is one of the hierarchical ensemble methods shown in Table 5.4 and j is one of the flat classifiers shown in Table 5.3.

In other words, the value of an heatmap cell is computed by dividing the difference between the average *AUPRC* of a hierarchical algorithm with that of a flat approach, by the maximum

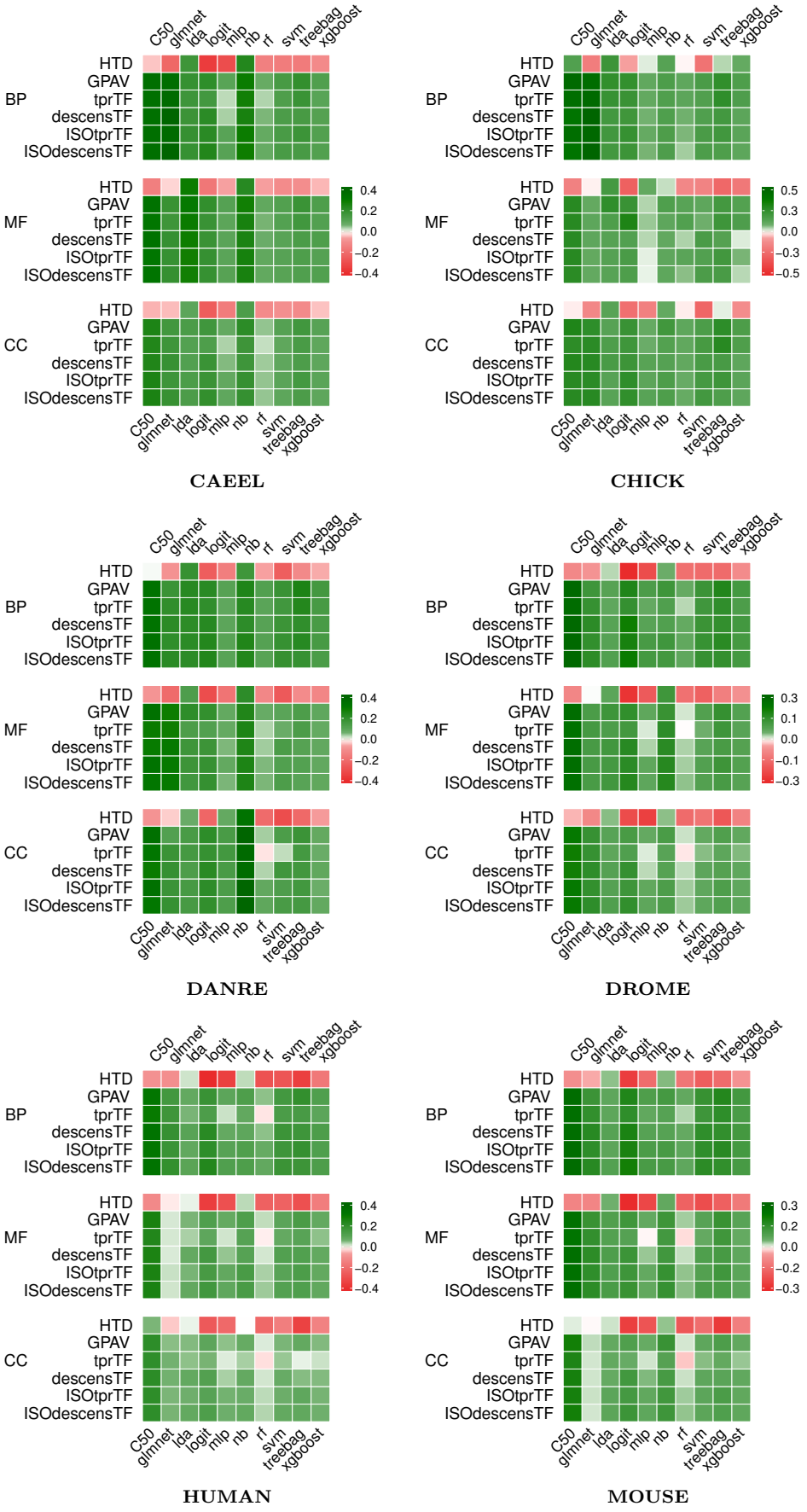


Figure 5.2: Heatmap of each model organism for the performance metric *AUPRC*. No normalization method was applied upon flat scores before the hierarchical algorithms. See text for further details.

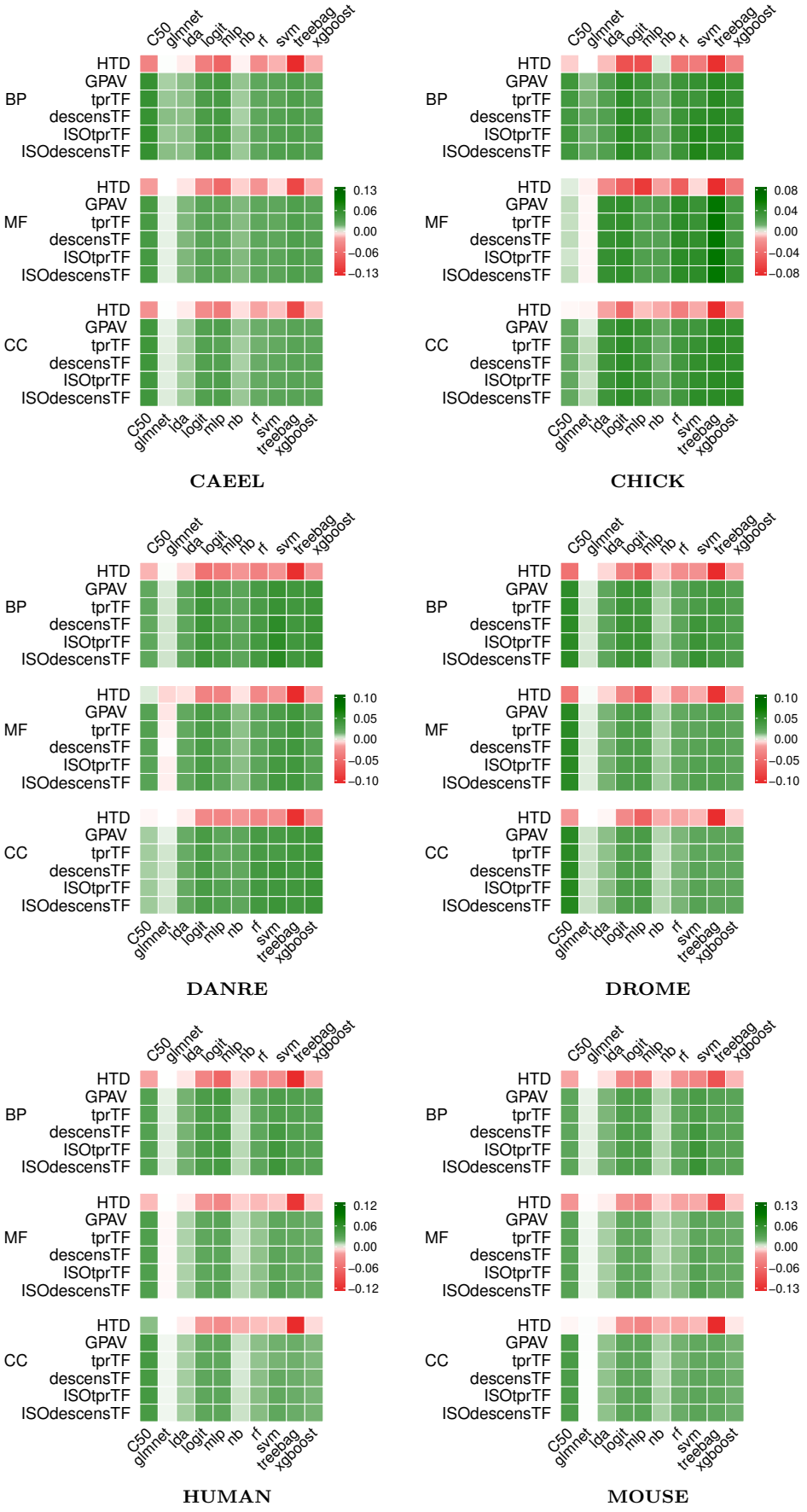


Figure 5.3: Heatmap of each model organism for the performance metric *AUROC*. No normalization method was applied upon flat scores before the hierarchical algorithms. See text for further details.

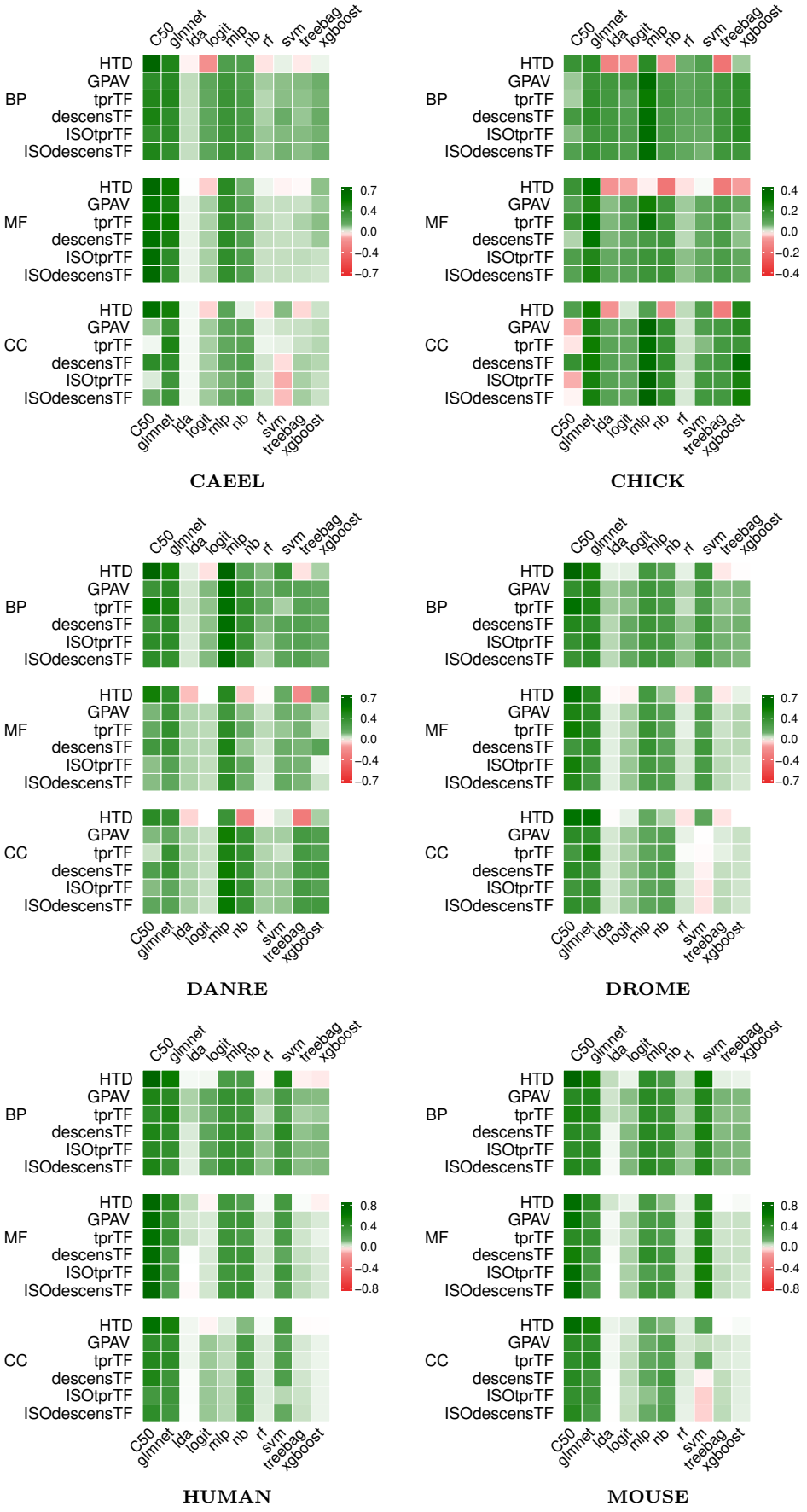


Figure 5.4: Heatmap of each model organism for the performance metric F_{max}. Flat scores was normalized in the sense of the maximum before applying hierarchical algorithms. Refer to text for further details.

value between these two values. In such a way, a positive value (associated with a green shade in the heatmap) indicates that the hierarchical method "wins" against the flat approach and vice versa a negative value (associated with a red shade in the heatmap) indicates that the flat approach "beats" the hierarchical one. The darker is the green shade and the higher is the difference (in positive) between the average *AUPRC* of a hierarchical algorithm versus that of a flat classifier. On the contrary, the darker is the red shade and the higher is the difference (in negative) between the average *AUPRC* of a hierarchical method versus that of flat approach. The heatmap in Figures 5.3 and 5.4 was produced by extending the formula 5.7 respectively to the metric *AUROC* and F_{max}. Observing all these figures both across performance and organisms, we can note in general that, except the *HTD-DAG* algorithm, all the other hierarchical methods win against the flat ones. This behavior of the *HTD-DAG* algorithm occurs almost always for the heatmap depicted in Figure 5.2 and 5.3, but almost never for the heatmap illustrated in Figure 5.4. This may due to the fact the F_{max} measure (defined in the equation 5.6) provides an over-optimistic evaluation of the hierarchical score. Indeed by definition, F_{max} measure is the maximum hierarchical F-score achievable by "a posteriori" setting the optimal threshold [31]. However, F_{max} is the benchmark *protein-centric* measure in the computational biology community, as witnessed by its usage in the *CAFA*2 international challenge [31], and in the ongoing *CAFA*3 challenge. Instead, for the heatmap represented in Figure 5.2 and 5.3, a reason for which *HTD-DAG* performs worse than the other hierarchical approaches is that, as pointed out in Section 4.3, this algorithm removes the constraints violations merely by reducing the flat scores. Consequently, it might happen that the predictions at the most specific nodes of the hierarchy (i.e., leaves nodes) are all negatives and and this may lead to a performances decreasing. Interestingly, the best results for the *term-centric* measures *AUPRC* and *AUROC* have been obtained without applying any normalization method before the hierarchical correction (Figures 5.2 and 5.3). Contrarily, the best results for the *protein-centric* measure F_{max} have been achieved by normalizing the flat scores in the sense of the maximum before executing any hierarchical ensemble methods. Furthermore, in Appendix Section A.4 we also show the heatmaps for *AUPRC* (Figure 5.2) and *AUROC* (Figure 5.3) obtained normalizing the flat scores in the sense of the maximum before the hierarchical correction, whereas in the Appendix Figure 5.4 we portray the heatmap for the metric F_{max} obtained without normalizing the flat scores before applying an ensemble algorithm.

To create the heatmaps shown in Figure 5.5 we firstly computed a two sided paired Wilcoxon rank sum test for each pair "*hierarchical algorithm - flat classifier*" and then we counted how many times a hierarchical algorithm "won" or "lost" against a flat classifier. We say that a hierarchical method "beats" a flat approach if the Wilcoxon test rejects the null hypothesis ($\alpha < 10^{-6}$) and if the average performance value of the hierarchical algorithm is greater than that of the flat classifier. Finally, we combined the results coming from all the model organisms and *GO* subontologies. Since we took into account six different model organisms and all the three *GO* ontologies, the value of each heatmap cell can range from a maximum of 18 (i.e., the

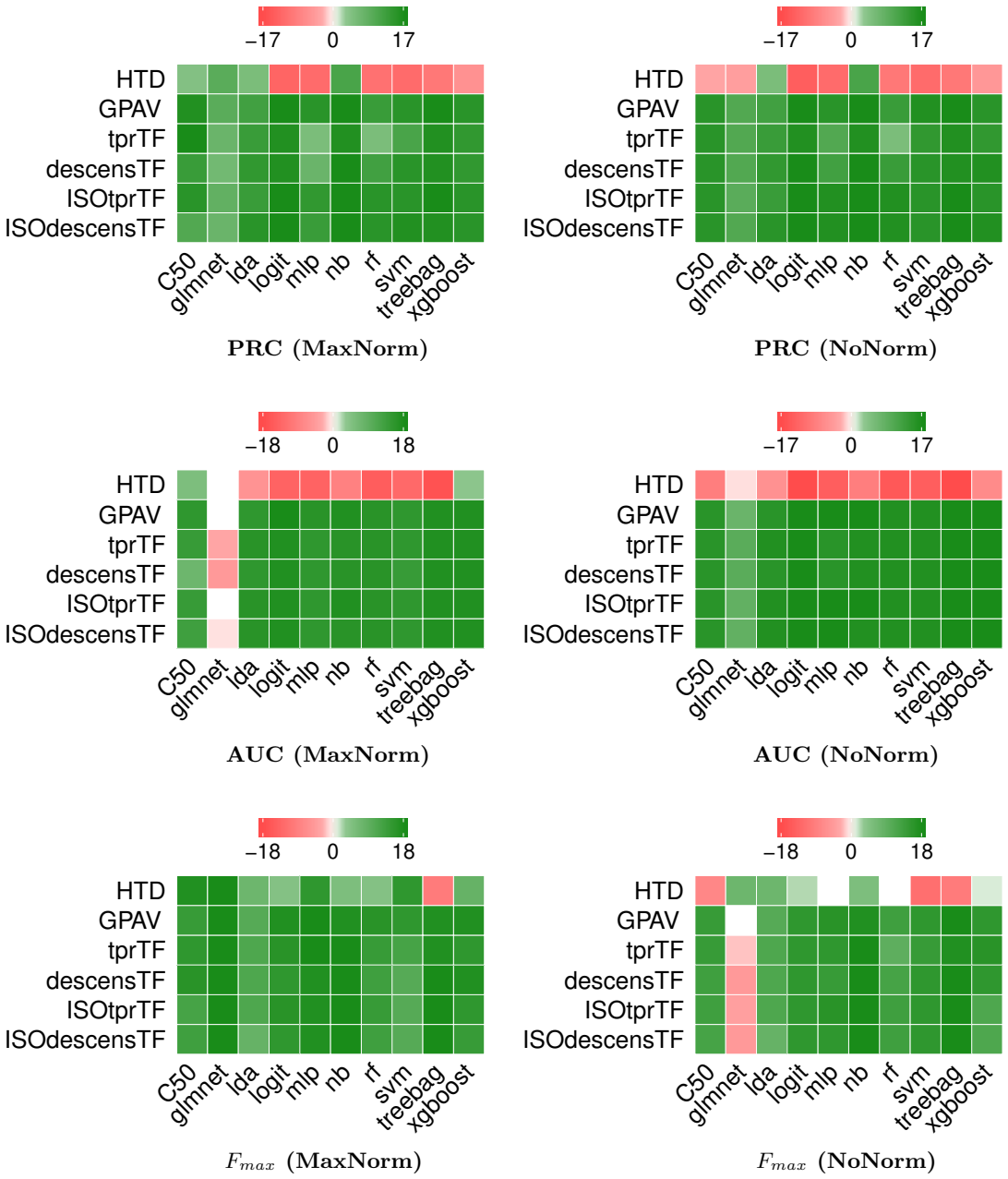


Figure 5.5: "Win-Loss" Heatmap. See text for further details.

hierarchical algorithm always wins against the flat approach) to a minimum of −18 (i.e., the hierarchical algorithm always "loses" against the flat approach). These procedure was repeated separately for each considered performance metric and normalization method. It is worth noting that the "win-loss" heatmaps depicted in Figure 5.5 support the "pattern" normalization in the sense of the maximum → best results in terms of *protein-centric* measure and no normalization → best results in terms of *term-centric* measure, previously discussed and shown in Figures 5.2, 5.3, 5.4. Finally, the "win-loss" heatmaps confirm also that the worst performing hierarchical algorithm is again *HTD-DAG*.

Figures 5.6, 5.7, 5.8, show, for each model organism, the distribution of the *AUPRC* values

across the *GO* terms (*MF* ontology) having 10 or more annotations. In Appendix Section A.3 are also illustrated the distribution of the *AUPRC* values across the *GO* terms of both the *BP* and *CC* ontology, always separately for each model organism. The star symbol ($\star$) above each whisker plot, indicates the significance level between the flat approach and the hierarchical algorithm according to the two sided Wilcoxon sum rank test. More precisely, we adopted the following coding to map the p-value between the two compared approaches:

- if p value is $< 10^{-6}$ we used three asterisks ($\star\star\star$);
- if p value is $< 10^{-4}$ we used two asterisks ($\star\star$);
- if p value is $< 10^{-2}$ we used one asterisk ($\star$);
- if p value is $\geq 10^{-2}$ we said the difference is not statistically significant (ns);

Observing these box-and-whiskers plots over all the model organisms it easy to note, one more time, that the hierarchical algorithms achieve statistically significant better results than flat machine learning classifiers. These results also show that the performance of hierarchical ensembles mostly depends on that of the underlying flat base learners. This is not surprising since the improvement introduced by hierarchical methods is determined also by the capability of the base learners to provide correct and at least partially consistent predictions. Indeed when the flat classifier provides random or very noisy predictions, it is really unlikely that the hierarchical ensemble approaches are able to improve the flat predictions (see for instance the predictions returned by *glmnet* methods). On average non linear classifiers, as Random Forests and Gradient Boosting machines work better than linear ones, and from this standpoint the box-and-whiskers plots of Figures 5.6, 5.7, 5.8 show that these kind of classifiers lead to better results, both in the "flat" and the improved hierarchical ensemble predictions.

It is likely that, the performances of the flat classifiers *C5.0*, *glmnet* and *naive bayes* are very low, because we did not tune the learning parameter, but we use the default setting (specified in Table 5.3) due to the high computational complexity of the base learns. In addition it is important to mention that the improvement of hierarchical ensemble methods upon flat machine learning classifier is not due to the tuning of any hyper-parameters, since all the hierarchical algorithms used in these experiments are parameter-free.

About support vector machines (*SVMs*), there are different kernels which can be employed for this classifier, such as polynomial, gaussian and radial kernel. Probably these non-linear classifiers may lead to improved results. However, in these experiments we run just the linear kernel since the other kernels had a too high computational complexity for the task of predicting thousands of different *GO* terms in different model organisms.

Finally, our hierarchical ensemble methods are quicker than the flat classifiers. Indeed, the flat methods employ to end, in average and across all the organisms, about 21 hours, whereas the hierarchical algorithms about 39 minutes. Consequently, the hierarchical algorithms turned out to be (in average across all the organisms) 32 times faster than the flat approaches. On the other

hand, please note that the hierarchical ensemble methods require as first step flat predictions. All flat and hierarchical experiments were performed on Linux running machines with 12 cores (64bit, 2.60GHz) and 128GB of RAM. For the detailed results about the average running time of both the flat classifiers and the hierarchical algorithms, please see the Appendix Table A.4 and the Appendix Figure A.10.

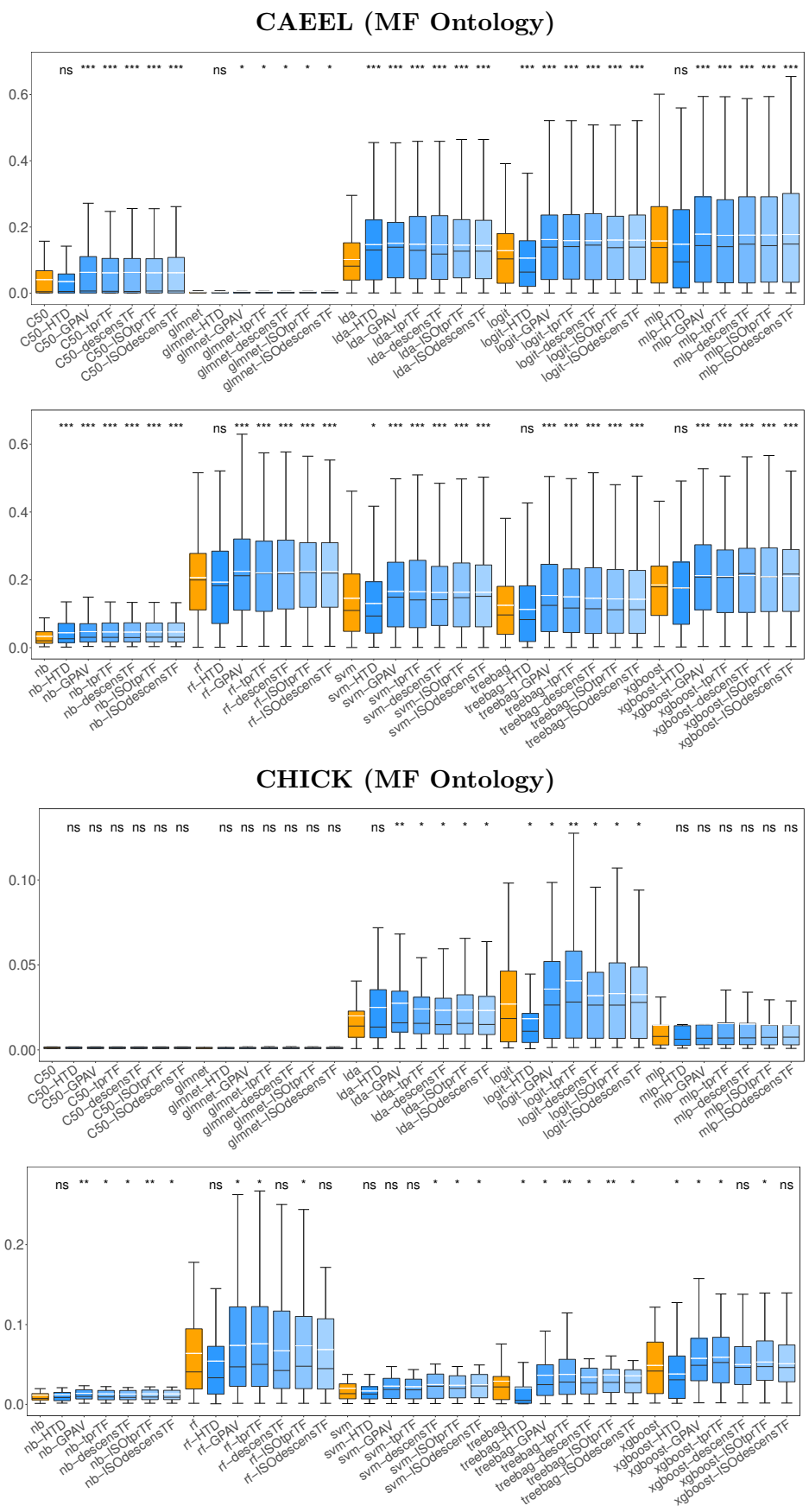


Figure 5.6: Distribution of *AUPRC* values across the GO terms (*MF* ontology) having 10 or more annotation respectively for the model organisms *C. elegans* (CAEEL) and *G. Gallus* (CHICK). No normalization was applied on the flat scores before the hierarchical correction.

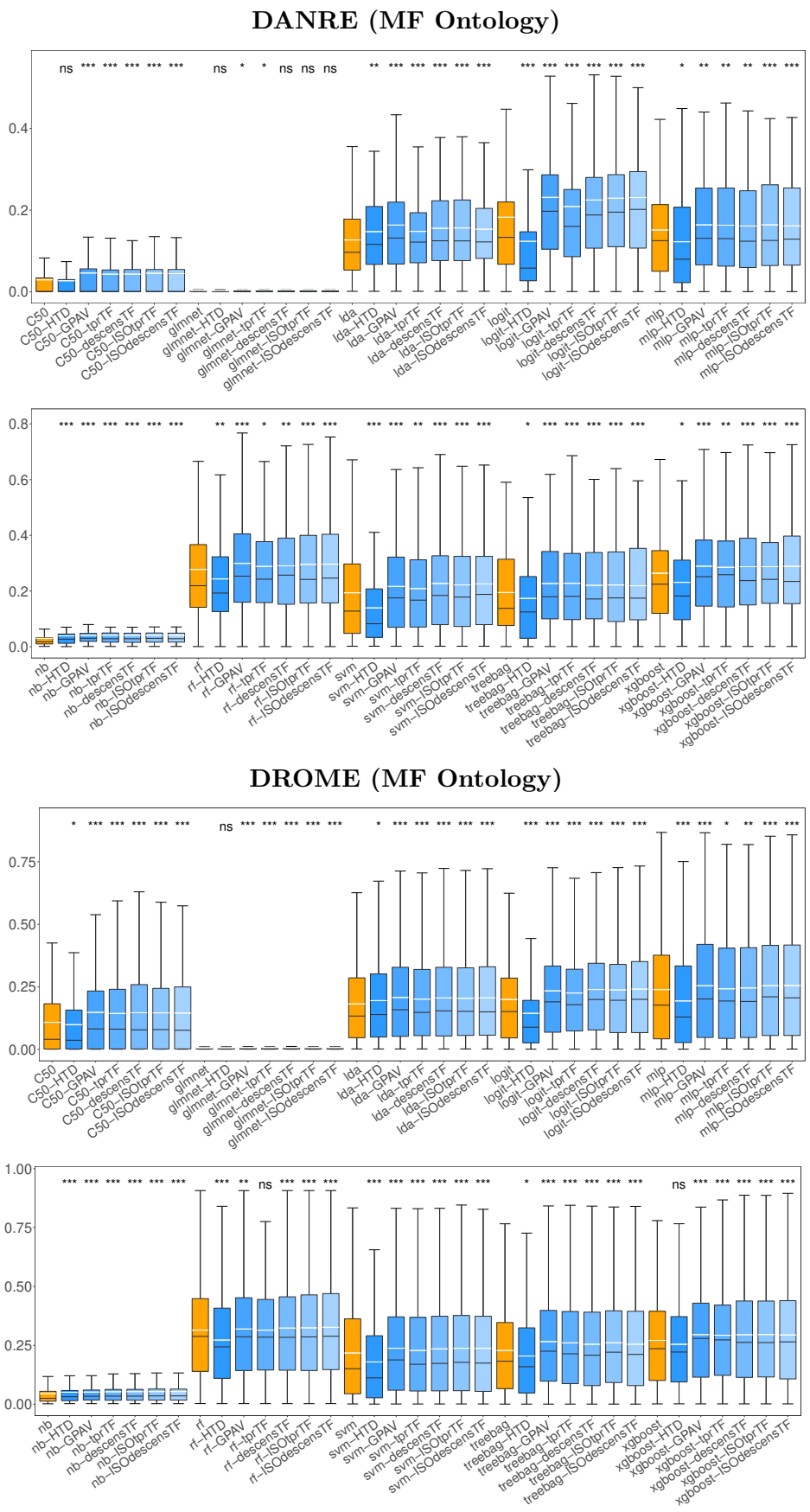


Figure 5.7: Distribution of *AUPRC* values across the GO terms (*MF* ontology) having 10 or more annotation respectively for the model organisms *D. rerio* (DANRE) and *D. melanogaster* (DROME). No normalization was applied on the flat scores before the hierarchical correction.

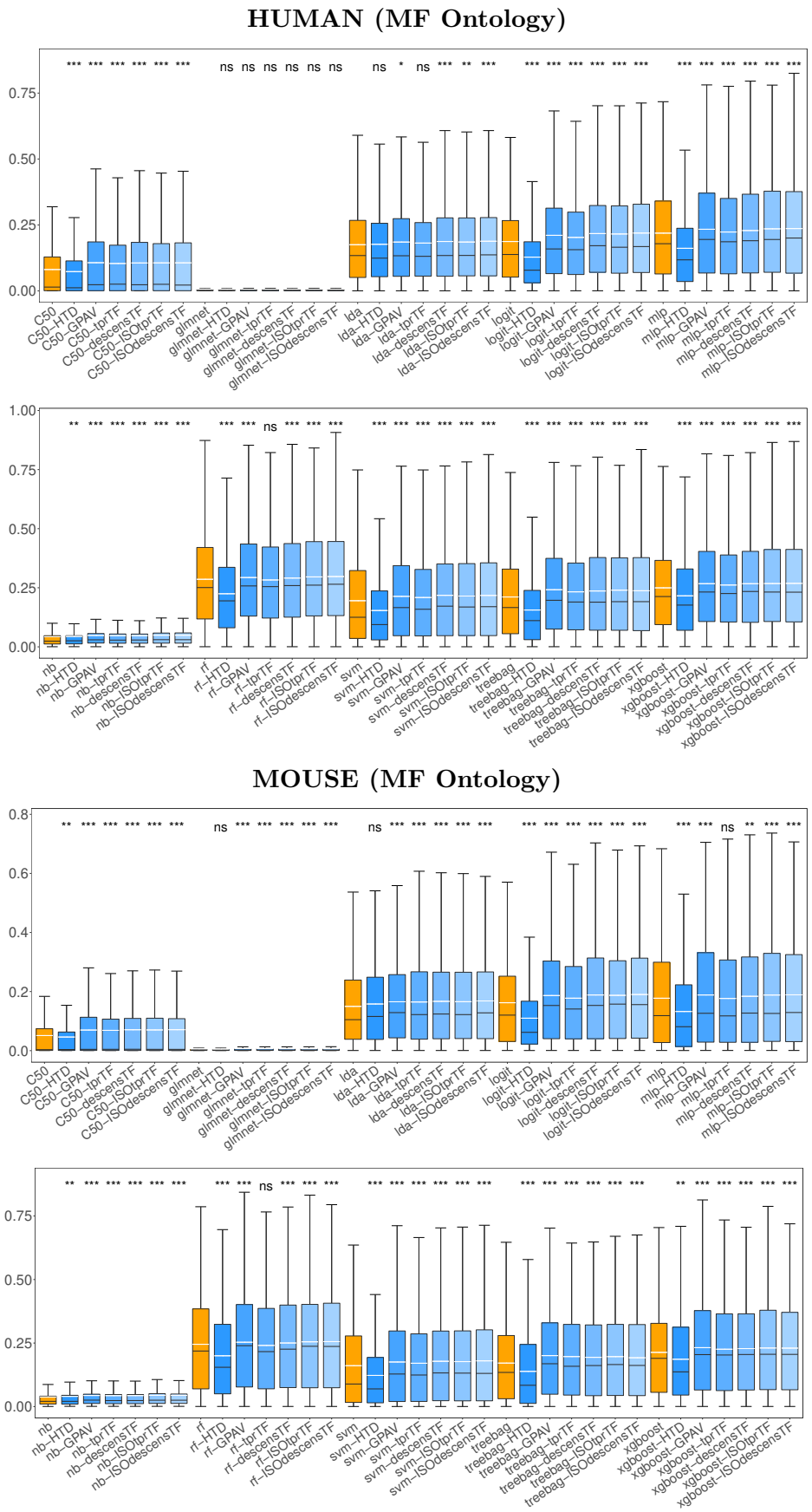


Figure 5.8: Distribution of *AUPRC* values across the GO terms (*MF* ontology) having 10 or more annotation respectively for the model organisms *H. sapiens* (HUMAN) and *M. musculus* (MOUSE). No normalization was applied on the flat scores before the hierarchical correction.

Chapter 6

Hierarchical Prediction of HPO Terms

DIFFERENTLY from its general meaning that usually refers to the traits or characteristics of an organism, in medical contexts, the word "*phenotype*" is defined as a deviation from normal morphology, physiology, or behavior [159]. The analysis of phenotype is essential for understanding the pathophysiology of cellular networks and plays a key role in medical research and in the mapping of disease genes [19, 160]. In this context, the Human Phenotype Ontology (*HPO*) project [3] provides a standardized vocabulary of human phenotypic abnormalities and of their semantic relationships. It is important to note that each *HPO* term does not denote a disease, but rather it describes a phenotypic abnormality that characterize a disease, such as *Atrial septal defect*. The *HPO* is currently developed using the medical literature, and OMIM [161], Orphanet [162] and DECIPHER [163] databases, and contains over $13,000$ terms and over $156,000$ annotations to hereditary diseases. *HPO* terms are arranged in a directed acyclic graph (DAG) and are connected by simple *is-a* (*subclass-of*) edges, such that a term represents a more specific or limited instance of its parent term(s). The relationships are transitive, meaning that they are inherited up all paths to the root.

While the problem of the prediction of gene–disease associations has been widely investigated [164], the related problem of gene– phenotypic feature (i.e., *HPO* term) associations has been considered in few studies [24], even if for most of human genes no *HPO* term associations are known and despite the quickly growing application of the *HPO* to relevant medical problems [165, 166]. The prediction of human gene–abnormal phenotype associations is a crucial step towards the discovery of novel genes associated with human disorders, especially considering that for about half of Mendelian disease the causative genes are unknown [167]. In addition, to the best of my knowledge, no methods based on hierarchical ensembles have been proposed in the context of structured output prediction of *HPO* terms associated with human genes. Indeed most of the hierarchical ensemble methods proposed in literature are conceived for tree-structured taxonomies [6], and the few ones specific for *DAG*s have been mainly applied to the prediction of the gene and protein functions [20, 26]. Nevertheless, several of these methods, mainly proposed in the context of *GO* term classification, could be in principle applied to the prediction of *HPO* terms. For a review of these approaches we refer the reader to [44].

6.1 Experimental Design

We performed two different sets of experiments to compare our proposed hierarchical ensemble approach with the state-of-the-art methods for the prediction of abnormal human phenotypes structured according to the *HPO*. In the first set of experiments (Section 6.2) we compared the hierarchical ensemble methods for *DAGs*, i.e. *HTD-DAG* and *TPR-DAG* and its ensemble variants with state-of-the-art methods using the same data and experimental set-up adopted by [24]. In the second set of experiments (Section 6.2) we evaluated the ability of our proposed hierarchical ensemble methods to predict newly annotated genes of the April 2016 *HPO* release, by using annotations of a previous release (January 2014). Finally we provided a list of currently unannotated genes that could be possible candidate genes for novel annotations, on the basis of the predictions performed by our proposed hierarchical ensemble methods.

6.2 Prediction of Human Phenotype Ontology Terms

We compared the hierarchical ensemble methods for DAGs, *HTD-DAG* (Section 4.3) and *TPR* and its variants (Section 4.4) against several state-of-the-art and baseline methods:

- *PHENOstruct*, a state-of-the art joint-kernel structured support vector machine approach [24]. This method uses the product of the input and output space kernel to construct the joint kernel. The rationale behind this approach is that two input/output pairs are considered similar if they are similar in both their input feature space and their output label space.

- *Clus-HMC-Ens*, a state-of-the art Hierarchical Multilabel classification (HMC) based on decision tree ensembles [33]. Differently from the proposed *HTD-DAG* and *TPR* methods, where each base learner solves a separate binary classification problem, each decision tree in the Clus-HMC-Ens ensemble is a "global" model built to predict all classes at the same time, thus allowing to explicitly take into account the relationships between the classes just at the level of each base learner.

- *SSVM* $\rightarrow$ *disease* $\rightarrow$ *HPO method*, an indirect two-step method that first predicts gene-disease associations and then maps them to *HPO* terms using the associations available on the HPO website [24].

- *PhenoPPIOrth*, a computational tool that can predict a set of OMIM diseases for given human genes using protein-protein interaction and orthologous proteins data and then maps the predicted OMIM terms to *HPO* terms by direct mapping [168].

- *Probabilistic support vector machines* (SVMs) [169]. This is a variant of the classical SVM algorithm, by which the output of the SVM is fitted to a sigmoid in order to provide an estimation of the probability that a given example belongs to the class to be predicted.

- *RANKS*, a semi-supervised method base on kernelized score functions [83], resulted one of the top-ranked methods in the recent *CAFA*2 challenge for the *HPO* term prediction [31].

To implement the base learners of the proposed hierarchical ensemble methods, we used both a semi-supervised (*RANKS* [83]) and a supervised (Support Vector Machines–SVM) machine learning method. The semi-supervised approach RAnking of Nodes with Kernalized Score functions (*RANKS*) is a very fast network-based method that combines local and global learning strategies to exploit both "local" similarities between nodes and "global" similarities embedded in the topology of the biomolecular network. From this standpoint *RANKS* can be considered a generalization of both guilt-by-association (*GBA*) methods [170], and kernel based algorithms for semi-supervised network analysis [171]. The *GBA* approach is generalized through efficient local learning strategies based on an extended notion of functional distance between nodes. Instead the global learning strategies are introduced by using kernel functions able to exploit the relationships and the overall topology of the underlying biological network. In principle, any valid kernel can be used (e.g. linear, polynomial, gaussian, Laplacian, Cauchy and inverse multiquadric kernels), but in the context of biomolecular networks it is often meaningful to use a *random walk kernel* [172] constructed from the weighted adjacency matrix of the graph under study. Indeed a *random walk kernel* can capture not only relationships coming from direct neighborhoods between nodes, similarly to *guilt by association* methods [170], but also relationships coming from shared and more in general indirect neighbors between nodes. In *RANKS* score functions are based on distance measures defined in a suitable Hilbert space, also called feature space. A distance measure in the Hilbert space can be introduced by using a proper kernel function and different score functions can be derived by choosing diverse distance measures. The score functions currently implemented in *RANKS* are the following: *Nearest Neighbours score*, *K-Nearest Neighbours score* and *Average score*. For further details about these kernelized score functions please refer to [83]. In this experiment we apply the average score function at 1, 2 and 3-step random walk kernels (that are respectively able to explore direct neighbors and genes at 2 or 3 step away in the network). *RANKS* was previously successfully applied in the prioritization of disease genes [173], the prediction of gene function [174] as well as for drug repositioning problems [175]. It is worth noting that *RANKS* returns a score and not a probability, that represents the likelihood that a gene belongs to a given class, but the "magnitude" of the scores may vary across different classes [174]. To make the scores comparable across classes, we considered two distinct normalization procedures:

1. Normalization in the sense of the maximum: the score of each class is normalized by dividing the score values for the maximum score of that class [28];

2. Quantile normalization: a method originally designed for the normalization of probe intensity levels for high density oligonucleotide microarray data across multiple experiments [176]. In our case we applied quantile normalization to make the scores comparable across different *HPO* terms.

SVMs were trained for each term using the R interface of the machine learning library *LiblineaR* [177] with default parameter settings. Because of the high running time of SVMs we implemented a *multicore* version of *LiblineaR* using *doParallel* and *foreach* R packages.

6.2.1 Experimental Set-Up

We used the same dataset and the same experimental set-up applied in [24] for a fair comparison with previously proposed methods [24, 33, 168]. Indeed, to our knowledge, [24] presented one of the largest comparative evaluation of different methods for the prediction of *HPO* terms at genome-wide level. Moreover *PHENOstruct*, described in the same paper, was one one of the top-ranked method in the recent CAFA2 challenge [31]. The generalization performance of the methods was evaluated through a classical 5-fold cross-validation procedure, according to [24]. It is worth noting in the training phase we selected the negative examples (genes) according to a "basic" selection strategy [6], i.e. for positives we used all the instances annotated with that *HPO* term, while for negatives we simply used all the not annotated instances. In principle other more refined strategies to select negatives may lead to improved performances [145, 178].

We used the same version of the the STRING (v. 9.1, [179]) and BioGRID (v. 3.2.106, [180]) databases used in [24]. More precisely we downloaded physical and genetic experimental interactions relative to 4970 proteins from BioGRID 3.2.106, and the integrated protein-protein interaction and functional association data for 18172 human proteins from STRING 9.1. From the same STRING website we also downloaded the protein aliases file to map proteins to genes, using as identifiers respectively the Locus STRING ID and the ENTREZ Gene ID. Moreover, starting from the Gene Ontology annotations for the three main sub-ontologies (Biological Process, Molecular Function and Cellular Component [2]) and from OMIM annotations [181], both represented as binary feature vectors, we constructed 4 more networks by using the classical Jaccard index to represent the edge weight (functional similarity) between the nodes (genes) of the resulting network. In our context the Jaccard index of two genes measures the ratio between the cardinality of their common annotations and the cardinality of the union of their annotations. The rationale behind the usage of this index is that two genes are similar if they share most of their annotations. All these annotations were obtained by parsing the raw text annotation files made available by UniProt knowledge-base considering only its SWISSprot component (release May 2013). Finally the resulting $n = 6$ networks were integrated by averaging the edge weights w_{ij}^d between the genes i and j of each network $d \in \{1, n\}$ after normalizing their weights in the same range of values $w_{ij}^d \in [0, 1]$ (*Unweighted Average* (UA) network integration, [173]):

$$\bar{w}_{ij} = \frac{1}{n} \sum_{d=1}^{n} w_{ij}^d \tag{6.1}$$

The weighted adjacency matrices representing the obtained networks have been directly used as the input of network-based transductive methods (e.g. *RANKS*), while the input for supervised feature-based inductive methods (e.g. SVM) has been constituted by the rows of the same

adjacency matrices. In other words for each gene its input features are represented by the interactions and functional similarities with all the other genes of the network.

Following the same experimental set-up of [24], we considered separately the three main sub-ontologies of the *HPO* (January 2014 release): *Organ Abnormality*, *Mode of Inheritance* and *Onset and Clinical Course*. *Organ Abnormality* is the main ontology and includes terms related to clinical abnormalities. *Mode of Inheritance* is a relatively small ontology and describes the inheritance pattern of the phenotypes. *Onset and Clinical Course* contains classes that describe typical modifiers of clinical symptoms, as the speed of progression, and the variability or the onset. For the sake of simplicity in the rest of the thesis we refer to these sub-ontologies respectively as *Organ*, *Inheritance* and *Onset*. Following the experimental set-up of [24], we pruned the *HPO* terms having less than 10 annotations in the January 2014 release, thus resulting in DAGs having respectively 2134 (*Organ*), 13 (*Inheritance*) and 23 (*Onset*) terms.

Since typically molecular biologists and physicians are interested in knowing both the set of genes associated with a certain *HPO* term and the phenotypic abnormalities associated with a particular human gene, we evaluated the results using two different performance metrics: (i) *term-centric* and (ii) *gene-centric*. These two types of evaluations were chosen to address the following related questions: (i) what are the genes associated with a specific abnormal phenotype? and (ii) what are the abnormal phenotypes associated with a particular gene?

For the experiments presented in this section, as *term-centric* measure we used the classical *AUROC*, wheres as *gene-centric* measure we used the hierarchical F-score (5.6). For further details on these two types of performance metrics, please refer to Section 5.3.

6.2.2 Experimental Results

The best results have been obtained with the *STRING* network and among the different variants of the *TPR-DAG* algorithm, *TPR-W* (eq. 4.8), with the *Threshold Free (TF)* strategy (equation 4.7) to select the set of "positive" children, achieved the best results. For this reason we firstly report the results obtained with *STRING* and the *TPR-W* ensemble, while the detailed results obtained with the other variants of the *TPR-DAG* algorithm as well as those obtained with the *UA* integrated network are respectively available in Appendix Table B.1 and B.2.

Table 6.1 summarizes the results achieved by the proposed hierarchical ensemble methods *HTD-DAG* and *TPR-DAG*, using as base learner *RANKS* (*TPR-W-RANKS* and *HTD-RANKS*) and a linear SVM (*TPR-W-SVM* and *HTD-SVM*). The results were compared with those achieved by state-of-the-art methods and the two flat methods used as base learner by the hierarchical ensembles (*RANKS* and *SVM*). *HTD-DAG* and *TPR-DAG* ensembles achieve statistically significant better results than state-of-the-art methods in terms of term-centric measures, independently of the base learner used: indeed, by applying the Wilcoxon rank sum test, the Bonferroni corrected p-value for multiple hypothesis testing resulted in a family wise error rate FWER $\leq 10^{-4}$. Considering the hierarchical multi-label score (F_{max}), *TPR-W-SVM* achieves significantly better results with respect to the other methods, with the only exception

of the smallest *HPO* sub-ontology (*Inheritance*) that includes only 13 *HPO* terms (Table 6.1). The best precision is obtained by *Clus-HMC-Ens* and the best recall by *PHENOstruct*, but the best compromise between these measures is obtained by *TPR-W-SVM*, thus resulting in the best overall F_{max} score. Interestingly enough, the hierarchical ensemble methods are always able to improve the results of the flat methods used as base learner; in particular we have a very large improvement of the F_{max} when *RANKS* is used as base learner, while the improvement

Organ sub-ontology				
	AUROC	F_{max}	Precision	Recall
TPR-W-RANKS	**0.89**	0.40	0.34	0.48
TPR-W-SVM	0.77	**0.44**	0.38	0.51
HTD-RANKS	0.88	0.37	0.30	0.49
HTD-SVM	0.75	0.43	0.37	0.49
PHENOstruct	0.73	0.42	0.35	**0.56**
Clus-HMC-Ens	0.65	0.41	**0.39**	0.43
PhenoPPIOrth	0.52	0.20	0.27	0.15
SSVM→Dis→HPO	0.49	0.23	0.16	0.41
RANKS	0.87	0.30	0.23	0.43
SVM	0.74	0.42	0.36	0.50
Inheritance sub-ontology				
	AUROC	F_{max}	Precision	Recall
TPR-W-RANKS	**0.91**	0.57	0.45	0.80
TPR-W-SVMs	0.82	0.69	0.59	0.82
HTD-RANKS	0.90	0.57	0.44	0.81
HTD-SVMs	0.81	0.69	0.59	0.82
PHENOstruct	0.74	**0.74**	**0.68**	0.81
Clus-HMC-Ens	0.73	0.73	0.64	**0.84**
PhenoPPIOrth	0.55	0.12	0.16	0.10
SSVM→Dis→HPO	0.46	0.11	0.07	0.25
RANKS	0.90	0.56	0.43	0.81
SVMs	0.82	0.69	0.59	0.82
Onset sub-ontology				
	AUROC	F_{max}	Precision	Recall
TPR-RANKS	**0.86**	0.44	0.33	**0.70**
TPR-SVMs	0.75	**0.48**	**0.38**	0.66
HTD-RANKS	**0.86**	0.42	0.30	0.69
HTD-SVMs	0.74	0.46	0.37	0.67
PHENOstruct	0.64	0.39	0.31	0.52
Clus-HMC-Ens	0.58	0.35	0.27	0.48
PhenoPPIOrth	0.53	0.25	0.25	0.24
SSVM→Dis→HPO	0.49	0.07	0.06	0.10
RANKS	0.83	0.41	0.30	0.67
SVMs	0.74	0.47	0.37	0.63

Table 6.1: Cross-validated prediction of genes associated with *HPO* terms of the *Organ*, *Inheritance* and *Onset* sub-ontology: average *AUROC* across terms and average F_{max}, Precision and Recall across genes of *HTD-DAG*, *TPR-W* ensembles and state-of-the-art methods. Best results for each metric are highlighted in bold.

is smaller with the *AUROC*, for which *RANKS* alone achieves relatively high values. These results also show that the performance of hierarchical ensembles largely depends on that of the flat base learners: for instance *HTD-RANKS* and *TPR-W-RANKS* achieve a significantly larger average *AUROC* than *HTD-SVM* and *TPR-W-SVM* (Table 6.1). This is not surprising since the improvement introduced by hierarchical methods depends also on the ability of the underlying flat base learners to provide correct and at least partially consistent predictions. However, not always flat scores can be improved by hierarchical ensemble methods, as shown for instance in Figure 4.9 and 4.10. Indeed when very noisy or incorrect flat scores are provided, it is unlikely that hierarchical ensemble methods can improve the predictions. This is also true in the opposite case, i.e. when flat scores are very close to optimal Bayes predictions, it is of course hard to improve performances for any method, including also *HTD-DAG* and *TPR-DAG*. With the *Onset* sub-ontology we obtain similar results, while with the smallest sub-ontology (*Inheritance*) including only 13 terms, *PHENOstruct* and *Clus-HMC-Ens* achieve the best performances (Table 6.1).

Overall, these results show that the proposed hierarchical ensemble methods are competitive with state-of-the-art methods such as *PHENOstruct* and *Clus-HMC-Ens* and moreover show that they can improve the results of different flat methods, such as the network-based semi-supervised *RANKS* algorithm and the supervised *SVM* classifier. Detailed results obtained with different variants of *TPR*, including *TPR-W*, *TPR-TF*, *TPR-T* (Section 4.4) and *DESCENS-T* (Section 4.5) are shown in Appendix Table B.1: *TPR-W* achieves most times the best results thanks to the tuning of the w parameter. The w parameter was selected through a classical double cross-validation procedure: the generalization performance was evaluated by the external cross-validation, while the "optimal" w parameter was chosen by internal cross-validation at each step of the external cross-validation. In this way we never accessed the examples of the test set for the selection of the w parameter. It is worth noting that the performance of other competing methods such as *PHENOstruct* or *Clus-HMC-Ens* could be enhanced by finely tuning their learning parameters. Nevertheless, *TPR-W* can further enhance its performance, by fine tuning the learning parameters of its base learners. We also observe that other *TPR* variants achieve competitive results without the need of tuning any parameter. For instance *TPR-TF-RANKS* shows an average F_{max} very close to that of *TPR-W-RANKS* in all the three *HPO* subontology.

In the Appendix Table B.2 we show the results attained through the *UA* integrated network. As is easy to note the results are in most cases worse than those obtained with *STRING* data alone: this is not so surprising since *STRING* just combines different sources of information to construct the integrated network.

6.3 HPO Prediction of Newly Annotated Genes

In this section we assess the capacity of our proposed hierarchical ensemble methods to predict novel *HPO* annotations for human genes. To this end we used annotations of an old *HPO*

release (January 2014) to predict the newly annotated genes of a recent *HPO* release (April 2016).

6.3.1 Experimental Set-Up

We compared the generalization performance of *HTD-DAG* and *TPR-DAG* hierarchical ensemble methods versus *PHENOstruct*, the best performing state-of-the-art method in the previous set of experiments (Section 6.2).

Let us denote with T the set of genes having at least 1 annotation with an *HPO* term of the "old" January 2014 *HPO* release (2804 genes) and with S the set of newly annotated genes, i.e. genes having at least one new annotation in the "new" April 2016 *HPO* release, but previously unannotated in the January 2014 HPO release (608 genes). Hence we have that $S \cap T = \emptyset$. We used the set T as training set and the set S as test set, and we applied a classical hold-out procedure to assess the capability of predicting newly annotated genes using only the annotations of the previous *HPO* release. In this way we predicted the newly annotated 608 genes of the test set, having on the average about 100 annotations per gene, distributed across 2445 *HPO* terms.

For the *HTD-DAG* and *TPR-DAG* methods we used the SVMs as base learners. To evaluate the performance of *PHENOstruct*, we downloaded and adapted the freely available C++ *PHENOstruct* code to perform the hold-out procedure described above.

As dataset we used the *STRING* 9.1 network, i.e one of the data sets used in the previous experiments. Indeed the previous experiments as well as the experiments performed by [24] revealed that *STRING* 9.1 was the most informative source of information for the prediction of *HPO* terms. We did not use the most recent release of the *STRING* database (v.10, [140]), since we might introduce an indirect bias in the prediction, considering that *STRING* 10 was not available when the January 2014 *HPO* version was released.

The experiments presented in this section are based on the January 2014 *HPO* release (10,320 terms and 13,549 between-term relationships) to predict the newly annotated genes of the April 2016 *HPO* release (11,673 terms and 15,459 between-term relationships). Since in different releases some terms could have been removed, others changed or become obsolete, we mapped the old *HPO* terms to the new ones by parsing the annotation file of the January 2014 *HPO* release using as key the alt-ID taken from the obo file of the April 2016 *HPO* release. From the same *HPO* releases we downloaded all the corresponding gene-term associations. Then we pruned *HPO* terms having less than 10 annotations obtaining a final *HPO DAG* composed of 2445 terms and 3059 between-terms relationships, to avoid the prediction of *HPO* terms having a too few annotations for a reliable assessment.

Unlike the previous experimental part (Section 6.2), in the experiments presented here we considered the whole *HPO*, without splitting it up in its main sub-ontologies and we used the same *gene-centric* and *term-centric* performance measures mentioned in Section 5.3. It is worth nothing that respect to the previous experiments (Section 6.2), in the experiments carried out here we considered also the *term-centric* measure Area Under the Precision Recall Curve

(*AUPRC*) to take into account the imbalance of annotated versus unannotated *HPO* terms [146].

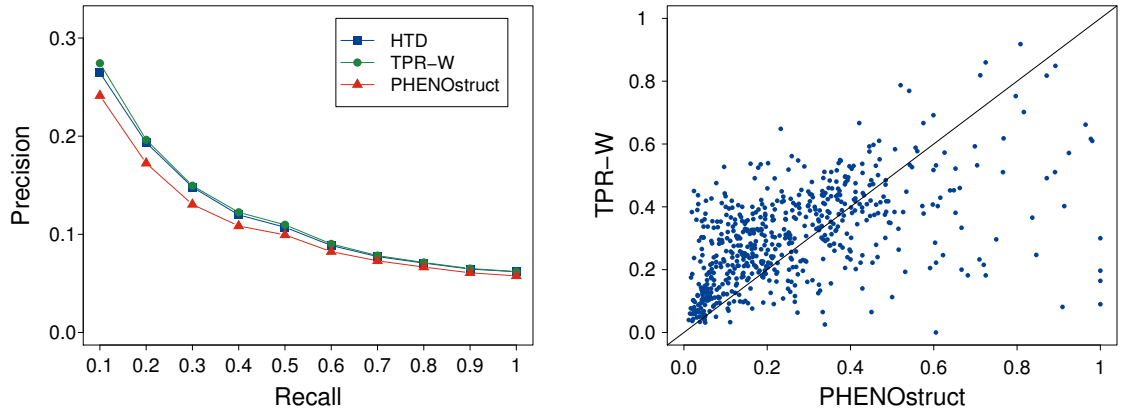
6.3.2 Experimental Results

Table 6.2 shows that *TPR-W* and *HTD* are able to predict newly annotated genes, even if with a certain decay in the overall performance, as expected, with respect to the cross-validated results of Table 6.1. Hierarchical ensemble methods attain significantly better results than

Method	AUROC	AUPRC	F_{max}	Precision	Recall
SVM	0.6506	0.1230	0.3774	0.3457	0.4155
HTD	0.6464	0.1207	0.3794	**0.3581**	0.4033
TPR-W	0.6512	0.1237	**0.3826**	0.3512	0.4202
PHENOstruct	**0.6661**	0.1089	0.3635	0.3040	**0.4519**

Table 6.2: Prediction of newly annotated human genes. Average *AUROC* and *AUPRC* across terms and average F_{max}, Precision and Recall across genes. Results significantly better than the others according to the Wilcoxon Rank Sum test ($\alpha = 10^{-9}$) are highlighted in bold.

PHENOstruct both in terms of average *AUPRC* and F_{max} (Wilcoxon paired rank sum test, p-value $< 10^{-9}$), while *PHENOstruct* achieves the best *AUROC* results. It is worth noting that the precision of *TPR-W* and *HTD* is higher than that of *PHENOstruct* at any recall level (Figure 6.1a), and these results are confirmed also by the "per-gene" hierarchical F_{max} score: *TPR-W* "wins" with 431 and "loses" with 177 human genes (Figure 6.1b). Results obtained



(a) (b)

Figure 6.1: (a) Compared precision at different recall levels averaged across 2444 *HPO* terms. (b) Scatter plot of F_{max} values. Each point represent one of the 608 genes of the test set. *PHENOstruct* values are in abscissa, *TPR-W* values in ordinate.

with other different *TPR-DAG* variants are comparable with those obtained by *TPR-W* (see Appendix Table B.3). We observe that on this task the *SVM* performance is close to that of the hierarchical ensemble methods. If on the one hand *TPR-W* achieves better results than the *SVM*, on the other hand not always the difference is statistically significant: more precisely

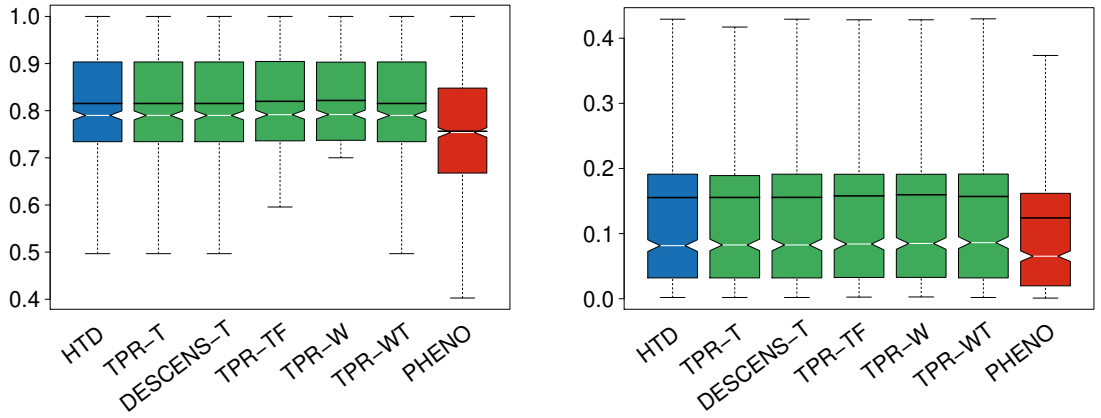
the difference is statistically significant with the F_{max} measure, while for the per-term metric *AUPRC* no statistical difference is detected. These results show that on this task is difficult to improve also on relatively simple baseline methods, and more research is needed to significantly enhance the performance.

We noted that the best method (*TPR-W*) for about half of the newly annotated genes (296) obtained a reasonable accuracy, i.e. $F_{max} > 0.3$, as well as a relatively large area under the curve ($AUROC > 0.7$) for about 800 *HPO* terms. Results limited to these best predicted genes and terms are summarized in Table 6.3. Figure 6.2 shows the distribution of the best "per-

Method	AUROC	AUPRC	F_{max}	Precision	Recall
SVM	0.8208	0.1589	0.4727	0.4560	0.4908
HTD	0.8155	0.1551	0.4716	0.4429	0.5042
TPR-W	**0.8219**	0.1594	**0.4793**	**0.4572**	0.5037
PHENOstruct	0.7565	0.1241	0.4297	0.3583	**0.5366**

Table 6.3: Prediction of newly annotated human genes considering only the best predictions. Results significantly better than the others according to the Wilcoxon Rank Sum test ($\alpha = 10^{-9}$) are highlighted in bold.

term" *AUROC* and *AUPRC* results of *HTD-DAG* and different variants of *TPR-DAG*, and in the Appendix Table B.4 are shown their best results in terms of average *AUROC*, *AUPRC* and F_{max}. While the results of Figure 6.2 and those shown in Appendix Figure B.4 are biased in



(a) (b)

Figure 6.2: Distribution of the *AUROC* and *AUPRC* values across the best predicted terms (778 *HPO* terms). (a) AUROC (b) AUPRC. *HTD-DAG* and different *TPR* variants are compared with *PHENOstruct*

favor of *TPR-W* (only the genes and terms best predicted by *TPR-W* are included), Figure 6.3 shows that hierarchical ensemble methods achieve competitive results in terms of precision at any recall level independently if the best predicted *HPO* terms are selected with respect to *TPR-W* or *PHENOstruct* best predictions.

The empirical computational time of hierarchical ensemble methods is significantly lower than that of state-of-the-art joint kernel structured output methods. Indeed the overall training

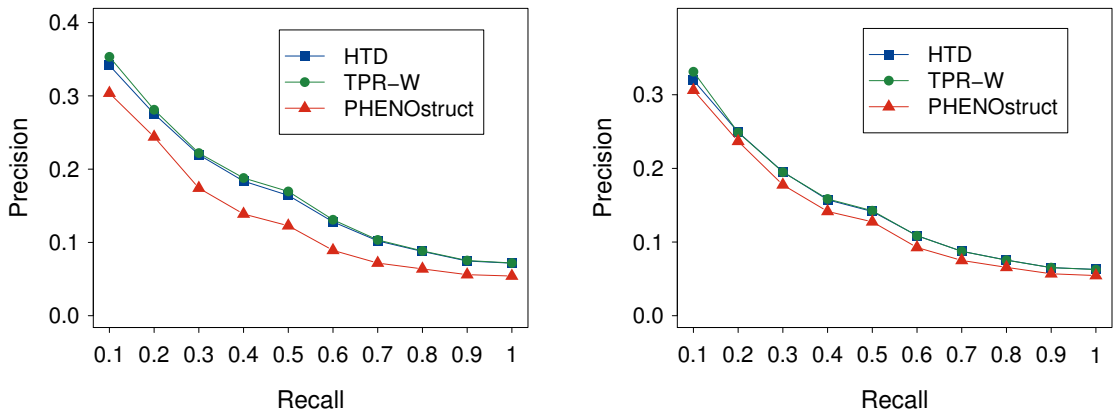


Figure 6.3: Precision at different recall levels of the newly annotated genes, considering only the best predicted terms. (a) Results considering only the *HPO* terms predicted with AUROC > 0.7 by *TPR-W* (778 terms);(b) results considering only the *HPO* terms predicted with AUROC > 0.7 by *PHENOstruct* (852 terms).

and test time for the hold-out processing is about 12 minutes with *HTD* and about 18 hours with *PHENOstruct* using an Intel Xeon CPU E5-2630 2.60GHz with 128GB of RAM. The overall computation time with *TPR-W* is significantly larger than *HTD* (about 3 hours), due to the tuning of the w parameter of the algorithm, but in any case significantly lower than *PHENOstruct*. It is worth noting that the overall computational time of the hierarchical ensemble methods depend on the training time of the base learner, and may of course vary with the complexity of the base learner used. We can also observe that *TPR-W* usually achieves slightly better results than *HTD* (Table 6.1 and 6.3), but if the computational time is an issue, we can safely use *HTD* or other *TPR* variants with lower computational complexity, such as *TPR-TF* or *TPR-D*, with only a small decay in performance.

6.4 Prediction of "Candidate" Genes for Novel Annotations

Table 6.4 provides a list of currently unannotated genes that could be possible candidate for novel annotations. To this purpose we selected a list of the *HPO* terms best predicted by the hierarchical ensemble methods, and then we selected among these *HPO* terms, those currently unannotated genes that achieved a hierarchical score larger than those of the annotated genes. More precisely, by exploiting the scores computed by *TPR-W* in the hold-out experiments, we selected genes not annotated in the April 2016 release of *HPO*, but predicted to be annotated by our method to specific *HPO* terms, thus resulting to be possible candidate genes for being annotated with that term.

First of all, by considering the hierarchical ensemble *TPR-W-SVM*, that achieved the best results in the detection of newly annotated genes, we selected the *HPO* terms best predicted by our method, i.e. terms predicted with *AUROC* larger than a given threshold. To this end

Gene Symbol	HPO term	AUROC	Depth	Distance from Leaves	Evidence (DOI/HPO release)
XRCC2	Clubbing of Toes	1	9	0	HPO (March 2017)
MSH3	Breast Carcinoma	0.9723	5	0	10.1371/journal.pone.0125571
COX10	Abnormal Mitochondria in Muscle Tissue	0.9967	6	0	10.1001/jamaneurol.2013.3242
CFB	Systemic Lupus Erythematosus	0.9967	5	0	10.1016/j.imbio.2015.08.001
CAD	Abnormality of Pyrimidine Metabolism	0.9951	4	0	10.1093/hmg/ddv057
LIPE	Insulin-Resistance Diabetes Mellitus	0.9934	6	0	HPO (March 2017)
TGFBR3	Emphysema	0.9785	5	0	10.1165/rcmb.2008-0427OC
IGF2	Neoplasm of the Adrenal Gland	0.9781	5	0	HPO (March 2017)
ECHS1	Abnormality of Fatty-Acid Metabolism	0.9753	4	0	10.1002/humu.22730
BARD1	Nephroblastoma aka Wilms Tumor	0.9615	8	0	10.1016/j.ebiom.2017.01.038

Table 6.4: List of novel gene-abnormal phenotype associations predicted by our algorithms using *HPO*April 2016 and *STRING* $v9.1$. Confirmed by literature or by the *HPO* March 2017 release. See text for further detail about each confirmed associations. Depth stands for the maximum distance of a given term from the root node (in the considered *HPO* graph the longest path from a node to a root is 14). Distance from leaves indicates the minimum distance of a given node from one of the leaves of the *HPO-DAG*. A value equal to 0 is assigned to the leaves, 1 to nodes with distance 1 from a leaf and so on.

we considered the 130 *HPO* terms that obtained an *AUROC* value higher than 0.95 (Appendix Table B.5). From this set of *HPO* terms we selected the most specific nodes of the hierarchy, that is those that correspond to the leaves of the *HPO DAG* (65 *HPO* terms). Finally we selected for these most specific and best predicted terms, the genes candidate to be annotated for that term. To this end, for each of the selected 65 *HPO* terms we adopted the following procedure:

1. Sort in descending order all the genes on the basis of the *TPR-W-SVM* scores;
2. Select the first top 5 ranked genes (set S);
3. Select the top ranked genes in S annotated for the *HPO* term (set $A \subseteq S$);
4. Select the maximum score $\bar{s}$ among the annotated genes belonging to A;
5. The candidate genes are those unannotated genes in S (the top ranked 5 genes) having a score larger or equal than $\bar{s}$, or all the genes in S if $A = \emptyset$.

As a result, for each *HPO* term we selected the top ranked unannotated genes having a *TPR-W-SVM* score higher or equal than the highest score achieved by the genes annotated for that term. In other words the procedure selects those unannotated genes that *TPR-W* strongly thinks to be annotated for that term. The procedure limits the analysis to the top 5 ranked genes for each best predicted and most specific term, in order to provide a list of genes well-characterized about their abnormal phenotype and possibly reliable, according to the performance of the *TPR-W-SVM* ensemble. It is worth noting that following the above procedure for selecting the candidate genes, not necessarily for each *HPO* leaf term we have always five candidate genes. Indeed it may happen that for a given *HPO* term an annotated gene fills, e.g., the third position in the rank, and hence for that term we will have only two candidate genes. The full list of the the genes candidate to be annotated for the best predicted and most specific 65 *HPO* terms are available in the Appendix Table B.6. We note that some

of these newly predicted gene-*HPO* term associations have been confirmed in the most recent literature and in the newer *HPO* release (March 2017). Table 6.4 summarized these findings.

For instance, the predicted association of the gene *XRCC2* with *HPO* term *HP:0100760* (Clubbing of toes) has been confirmed in the March 2017 *HPO* release, as well as the association of the gene *LIPE* (that encodes the protein *LIPE E*) with the *HPO* term *HP:0000831* (Insulin-resistant diabetes mellitus). Interestingly enough *LIPE* is correlated also with the disease *LIPE*-related familial partial Lipodystrophy (*ORPHA:435660*). The March 2017 *HPO* release also confirmed the predicted association of gene *IGF2*, that encodes a member of the insulin family of polypeptide growth factor, with the phenotype *HP:0100631* (Neoplasm of the adrenal gland), and moreover recent works postulated the association of *IGF2* with adrenal tumors ed in particular with adrenocortical carcinomas and pheochromocytomas [182].

Besides the aforementioned evidence of associations, the novel predicted gene-*HPO* term pairs show a clear relationship with specific diseases, according to the most recent literature. For example the human gene *ECHS1*, that encodes the enzyme that catalyzes the second step of the mitochondrial fatty acid β-oxidation (*FAO*) pathway, is correlated with the phenotype *HP:0004359* (Abnormality of fatty-acid metabolism) and from literature is also known being associated with *FAO* disorders and in particular with the Leigh syndrome [183]. Moreover the predicted association between the Complement factor *B* (*CFB*) and the term *HP:0002725* (Systemic lupus erythematosus) is supported by recent literature, since *CFB* is an important activator of the alternative complement pathway and increasing evidence supports reducing factor *B* as a potential novel therapy to lupus nephritis [184].

Another strong evidence of associations between the gene–*HPO* term pair predicted by our ensemble method and the corresponding disease is given by the work of Hersh et al. [185]. In this work the authors demonstrated the importance of *TGFBR3* gene (encoding the transforming growth factor (*TGF*)-beta type III receptor) in the Chronic Obstructive Pulmonary Disease (*COPD*), confirming the prediction made by our method which associated *TGFBR3* gene with the *HPO* term Emphysema (*HP:0002097*). Furthermore the predicted association between the gene *BARD1* and Nephroblastoma, also known as Wilms' tumor (*HP:0002667*), is supported in the very recent work of Fu et al. [186], in which the authors claim that the *BARD1* gene polymorphism is significantly associated with increased nephroblastoma risk. As stated in [187], in-frame and frameshift mutations in the *MSH3* gene play an important role in the development of breast cancer (*HP:0003002*), supporting in this way the selfsame gene–*HPO* term association predicted by our methods. The predicted association between the *CAD* gene (encoding for a tri-functional enzyme involved in the pyrimidine biosynthesis) and the phenotype *HP:0004353* (Abnormality of pyrimidine metabolism) is confirmed by the fact that a mutation in *CAD* impairs *de novo* pyrimidine biosynthesis [188]. Another interesting example is represented by the predicted association between the *COX10* gene (Cytochrome c Oxidase) and the phenotype Abnormal mitochondria in muscle tissue (*HP:0008316*), supported by literature evidence that correlates the *COX10* dysfunction with mitochondrial disease [189]. Overall, the novel annota-

tions in the recent (March 2017) *HPO* release as well as recent bio-medical literature support the novel predictions obtained with our hierarchical ensemble methods.

Chapter 7

Discussion and Conclusions

Several real-world problems ranging from web page categorization, to transposable element classification and to gene or protein function prediction are characterized by hierarchical multi-label classification tasks. In this context flat methods may provide inconsistent predictions and more in general are not able to exploit the hierarchical constraints between classes. Indeed, many computational methods that exploit "omics" data can be successfully applied to predict the biomolecular function of a protein (Chapter 6) or to rank a gene with respect to an abnormal phenotype of a rare Mendelian disease [28], but usually their predictions are inconsistent, in the sense that do not necessarily obey the parent-child relationships between ontology terms (i.e. a gene or a protein may achieve a score for a child term larger than that of its parent class). In this PhD thesis work we presented efficient computational methods (Chapter 4) that guarantee biologically meaningful predictions that obey to the *true path rule* which governs both the *GO* and the *HPO*. In Section 4.8 we also showed a formally proof of this fact.

The proposed hierarchical learning algorithms have a highly modular structure: the first step consists in a "flat" learning of the ontology terms, while the second step combines the predictions to make them consistent with the underling ontology. On the one hand, the experimental results achieved by using the Gene Ontology (Chapter 5), show that our proposed hierarchical ensemble methods significantly outperform flat approaches, by considering six different model organisms and independently of the base learner used. On the other hand, the experimental results obtained by using the Human Phenotype Ontology (Chapter 6), show that our hierarchical ensemble methods are able to predict novel associations between genes and abnormal phenotypes with results competitive with state-of-the-art algorithms. From this standpoint, our "true-path-rule"-based hierarchical learning algorithms can be conceptualized as a flexible tool that can be applied to any off-the-shelf flat classifier to improve its predictions for any *DAG*-structured taxonomy (e.g. *GO* and *HPO*), and consequently also for any tree-structured taxonomies (e.g. FunCat), since obviously trees are *DAGs*. Our experiments results (Section 5.6 and 6.3.2) confirmed this aspect, by showing that the proposed hierarchical algorithms are able to improve the predictions of both semi-supervised flat methods, such as the *RANKS* algorithm, that resulted one of the top ranked method in the recent *CAFA*2 challenge for *HPO* term prediction [31], and of supervised methods such as *SVM*, *MLP* and decision trees.

We provided also a list of unannotated genes that could be possible candidate for novel *HPO* annotations, and we showed that several predicted gene - *HPO* term associations have

been confirmed in the March 2017 *HPO* release and in the most recent bio-medial literature. Since for several disorders no disease genes have been discovered (e.g. for about half of Mendelian diseases no causative genes are known [167]), our methods can contribute to the discovery of such genes, and to unravel the full spectrum of phenotypes associated with them.

It is worth pointing out that the complexity of our hierarchical methods is linear in the number of the ontology classes ($\mathcal{O}(|V|)$) for *HTD-DAG* and *TPR*, whereas is quadratic in the number of the ontology classes for *GPAV* and *ISO-TPR* ($\mathcal{O}(|V|^2)$). The empirical computational time of the hierarchical algorithms is significantly lower both than several flat classifiers (Section 5.6) and state-of-the-art joint kernel structured output approach (Section 6.3.2). Therefore, the proposed approaches scale nicely with large data sets and ontologies.

Last but not least, all the ensemble methods described in the Chapter 4 of this PhD thesis, were packaged in the R library called **HEMDAG**, which is publicly available both from CRAN and BIOCONDA repository under the GNU General Public License, version 3 (GPL-3.0). In addition, in order to simplify the integration of our approaches within other works, all the *HEMDAG* methods are fully documented alongside a comprehensive step-by-step tutorial, available at the following link.

For future research work, different bottom-up and top-down variants of the proposed hierarchical learning algorithms could be investigated. Considering bottom-up learning strategies, we could design algorithms characterized by contributions of "positive" descendants that decay linearly with their distance from the root. An opposite strategy could consist in an increment of the weights from bottom to top, to put more weights on predictions made on the most specific terms. Otherwise, in the bottom-up step instead of taking into account the children or descendants, we can consider the "*Markov blanket*" as proposed in [26]. Alternative top-down learning strategies to be explored could be based for instance on the Kullback-Leibler divergence [20]. In addition, by exploiting the relationships between classes induced by the hierarchy of the ontology we could also apply multi-task learning strategies [190] to take into account the relationships between classes just during the learning phase, in order to establish a functional connection among the learning processes. In this way, just during the training of the base learners, the learning process will be dependent on each other, making possible a "mutual learning" between related classes in the taxonomy. It is worth noting that multi-task learning strategies alone does not assure the consistency of the predictions, but it can be used to enhance the faithfulness of the predictions during the learning phase before applying a topology-aware step. Indeed, the improvement introduced by hierarchical methods depends also on the ability of the base learners to provide correct predictions. Finally, in order to further test the soundness and the robustness of our hierarchical ensemble algorithms, we could compare them with the most recent literature works, such as *HPO2GO* [191] or the top-ranked methods of the international *CAFA*3 challenge, that at the time of writing this thesis, is still ongoing.

Appendix

A Prediction of GO Terms: Supplementary Material

Here are collected all the supplementary materials on experiments on the *GO* terms prediction (Chapter 5).

A.1 Data Preparation

Separately for each model organism listed in 5.1, we extracted the annotations supported by one of the following Experimental Evidences codes: (i) Inferred from Experiment (*EXP*), (ii) Inferred from Direct Assay (*IDA*), (iii) Inferred from Physical Interaction (*IPI*), (iv) Inferred from Mutant Phenotype (*IMP*), (v) Inferred from Genetic Interaction (*IGI*), (vi) Inferred from Expression Pattern (*IEP*). Successively, we used the UniProtKB identifier mapping file to map the UniProtKB proteins annotated with a *GO* term versus the *STRING* proteins, using as identifiers respectively the UniProtKB accession number (*UNIPROT-AC*) and the locus *STRING-ID*. As key we used the *UNIPROT-AC*. In a separated file we stored all the unmatched *UNIPROT-AC* identifiers. Afterwards, we downloaded the whole SWISS (82MB, zipped) and TREMBL (22GB, zipped) databases, and we isolated the protein *FASTA* sequence of the *UNIPROT-AC* for which we did not find a match towards a *STRING* identifiers. Hence, we downloaded and transformed the *FASTA* file of the *STRING* proteins as *BLAST* database by using the command **makeblastdb** of the standalone *BLAST+* package (version 2.7.1) of the *NCBI* C++ toolkit. Therefore we blasted the protein *FASTA* sequences of the unmapped *UNIPROT-AC* identifiers against the *FASTA* sequences of the *STRING* proteins, in order to retrieve the mapping between the two identifiers whenever their corresponding protein sequences show a percentage of identity of 100%. To blast the protein *FASTA* sequences we used the command **blastp** of the standalone *BLAST+* package. The goal is to reduce as much as possible the number of the *UNIPROT-AC* identifiers that did not have a corresponding *STRING* identifiers in the UniProtKB mapping file. It could happen that the map between the couple of identifiers retrieved by *BLAST* (i.e. *BLAST* hit) already exists in the mapping file provided by UniprotKB, but the two identifiers are mapped differently. In other words, it could occur that a same *UNIPROT-AC* identifier is mapped with different *STRING* proteins. This kind of "mismatch", is due to the recovery of a post-translationally modified isoforms, since we blasted *FASTA* protein sequences. To avoid this issue we removed from the *BLAST* output file all the *BLAST* hits with a 100% of identity and whose *STRING* identifier already exists in the UniProtKB mapping file. Therefore we enriched the original UniProtKB identifier mapping file by adding in it all the retrieved *BLAST* hits having an identity of 100% and whose *STRING* identifier does not exist in the UniProtKB

identifier mapping file provided by *GOA* database. At the end of this process, we remapped the *UNIPROT-AC* proteins annotated with a *GO* term versus the *STRING* proteins, by using this time the enriched UniprotKB mapping file. The soundness of the above procedure is shown in Table A.1 where, for each organism, are shown some statistics before and after the enrichment of the UniProtKB mapping file. The first entry of the column **Stats Entry** of the Tables A.1

Organism	Stats Entry	At the Beginning	After Enrichment
CAEEL	UNIPROT-AC associated with a GO term	3449	3449
	GO term associated with a UNIPROT-AC	3083	3083
	STRING-ID mapped with a UNIPROT-AC	3338	3405
	STRING-ID unmapped with a UNIPROT-AC	111	44
	GO term associated with a STRING-ID	3052	3061
	GO terms not associated with a STRING-ID	31	22
CHICK	UNIPROT-AC associated with a *GO* term	740	740
	GO term associated with a UNIPROT-AC	1204	1204
	STRING-ID mapped with a UNIPROT-AC	521	544
	STRING-ID unmapped with a UNIPROT-AC	219	196
	GO term associated with a STRING-ID	1018	1048
	GO terms not associated with a STRING-ID	186	156
DANRE	UNIPROT-AC associated with a GO term	4227	4227
	GO term associated with a UNIPROT-AC	3051	3051
	STRING-ID mapped with a UNIPROT-AC	3566	3736
	STRING-ID unmapped with a UNIPROT-AC	661	491
	GO term associated with a STRING-ID	2840	2880
	GO terms not associated with a STRING-ID	211	171
DROME	UNIPROT-AC associated with a GO term	6179	6179
	GO term associated with a UNIPROT-AC	5510	5510
	STRING-ID mapped with a UNIPROT-AC	5675	6025
	STRING-ID unmapped with a UNIPROT-AC	504	154
	GO term associated with a STRING-ID	5393	5444
	GO terms not associated with a STRING-ID	117	66
HUMAN	UNIPROT-AC associated with a GO term	13603	13603
	GO term associated with a UNIPROT-AC	11134	11134
	STRING-ID mapped with a UNIPROT-AC	12795	13160
	STRING-ID unmapped with a UNIPROT-AC	808	443
	GO term associated with a STRING-ID	11006	11042
	GO terms not associated with a STRING-ID	128	92
MOUSE	UNIPROT-AC associated with a GO term	11265	11265
	GO term associated with a UNIPROT-AC	11101	11101
	STRING-ID mapped with a UNIPROT-AC	10592	10948
	STRING-ID unmapped with a UNIPROT-AC	673	317
	GO term associated with a STRING-ID	10959	11029
	GO terms not associated with a STRING-ID	142	72

Table A.1: Mapping statistics before and after the enrichment of the UniProtKB mapping file. Refer to text for further details about the entries of the column **Stats Entry**.

points out the total number of *UNIPROT-AC* associated with a *GO* terms, whereas the second entry indicates the total unique *GO* terms associated to an *UNIPROT-AC* identifier. These two entries are identical before and after the enrichment because the UniProtKB identifiers mapping file provided by *GOA* database contains annotations towards *UNIPROT-AC* proteins. The third and fourth entry show respectively the number of mapped and unmapped *STRING-ID* to *UNIPROT-AC* before and after the enrichment of the UniProtKB mapping file according to the pipeline illustrated above. Finally, the fifth and the sixth entry points out respectively to the total number of *GO* terms associated with a *STRING-ID* and to the total number of *GO* terms not associated with a *STRING-ID* before and after the enrichment of the UniProtKB mapping file. Looking at the Table A.1 and considering all the organisms, using the above pipeline we were able to rescue in average the 3% of the *UNIPROT-AC* identifiers that did not have a corresponding *STRING-ID* in the UniProtKB mapping file provided by the *GOA* database. Consequently, we rescued also in average the 1% of *GO* terms associated to a *STRING* proteins. Nevertheless, it is worth noting that just using the mapping file provided by *GOA* database the mapping between the couple of identifiers *UNIPROT-AC-STRING-ID* is around the 89% and the associations *STRING-ID-GO* terms are covered at 96%.

A drawback of the above pipeline is that some of the retrieved *STRING* proteins might be lost during the construction of the annotations matrix, since these proteins might be "singleton" in the corresponding *STRING* network. Nevertheless, this risk should be low since we rescued an high number of *STRING* proteins that were initially "unseen" in the UniProtKB mapping file provided by the *GO* database. The most time-consuming step of the above pipeline, is the part in which we isolated the *FASTA* sequence of the unmapped *UNIPROT-AC* identifiers from the *TREMBL* database (22Gb zipped). This step requires about a couple of hours for each organism (using an Intel Xeon CPU E5-2630 2.60GHz with 128GB of RAM), but it must be repeated just once. Finally, the pipeline was repeated separately for each model organism and then we did not retrieve orthologues proteins (i.e. protein sequence belongs to different organisms) during the *BLAST* phase.

A.2 Estimate of the Overall Time Complexity

In Table A.2 are shown the execution times required by some flat classifiers (with the default parameters configuration) to predict the protein function of the model organism *CAEEL* by considering all the available interactions (Table 5.1). Due to the high running time we did not perform experiments considering other base learners (Table 5.3) and other model organisms (Table 5.1).

Flat Classifier	Domain	**Days**	**Days Estimate**
C5.0 Decision Tree	BP	2.54	113.03
	MF	2.15	13.33
	CC	2.13	15.69
Glmnet	BP	1.5	66.75
	MF	1.46	9.05
	CC	1.56	11.49
Multi Layer Perceptron	BP	4.07	181.12
	MF	3.55	22.01
	CC	3.24	23.87
Support Vector Machine	BP	1.43	63.63
	MF	1.65	10.23
	CC	1.94	14.29
Extreme Gradient Boosting	BP	1.43	63.63
	MF	1.4	8.68
	CC	1.4	10.31

Table A.2: Evaluation of the empirical time complexity using all the protein-protein interactions of the model organism *CAEEL*. The flat algorithms was run by using the default parameter configuration shown in Table 5.3. The column **Flat Classifier** refers to the name of the flat classifiers; the column **Domain** refers to one of the *GO* subontology (*BP*, *MF*or *CC*); the column **Days** refer to the total amount of time in days required by each classifier to make flat predictions on the 30 randomly chosen *GO* terms and finally the column **Days Estimate** indicates an estimation of the total amount of time in days required by each flat approach to compute the predictions considering all the GO classes having 10 or more annotations (column **Classes** ≥ 10 of the Table 5.2).

In Table A.3 are shown the results of the evaluation of the overall computational time (in terms of hours and days of computing) by considering the first 100 top-ranked features. For the sake of the visualization, for each flat approach, are shown just the two pairs model organism-ontology for which the maximum and the minimum empirical computational estimation were estimated. Looking at the Table A.3 it easy to see that, over the organisms, the most computational time-consuming flat approaches are *Bagging Ensemble of Decision Tree* and *Glmnet*.

Flat Classifier	Organism	Domain	Hours	Days	Hours Estimate	Days Estimate
C5.0 Decision Trees	MOUSE	BP	0.51	0.02	66.28	2.60
	CHICK	MF	0.27	0.01	0.39	0.01
Glmnet	MOUSE	BP	1.01	0.04	131.27	5.20
	CHICK	MF	0.78	0.03	1.12	0.04
Linear Discriminant Analysis	MOUSE	BP	0.38	0.02	49.39	2.60
	CHICK	MF	0.18	0.01	0.26	0.01
Logit Boost	MOUSE	BP	0.38	0.02	49.39	2.60
	CHICK	MF	0.19	0.01	0.27	0.01
Multi Layer Perceptron	MOUSE	BP	0.74	0.03	96.18	3.90
	CHICK	MF	0.49	0.02	0.70	0.03
Naive Bayes	MOUSE	BP	0.36	0.02	46.79	2.60
	CHICK	MF	0.17	0.01	0.24	0.01
Random Forest	MOUSE	BP	0.46	0.02	59.78	2.60
	CHICK	MF	0.21	0.01	0.30	0.01
Support Vector Machine	MOUSE	BP	0.75	0.03	97.48	3.90
	CHICK	MF	0.26	0.01	0.37	0.01
Bagging Ensemble of Decision Tree	MOUSE	BP	1.08	0.04	140.36	5.20
	CHICK	MF	0.63	0.03	0.90	0.04
Extreme Gradient Boosting	MOUSE	BP	0.50	0.02	64.98	2.60
	CHICK	MF	0.26	0.01	0.37	0.01

Table A.3: The column **Flat Classifier** refers to the name of the flat classifiers; the column **Organism** refers to the name of the model organism; the column **Domain** refers to one of the GO subontology (BP, MF or CC); the columns **Hours** and **Days** refer to the total amount of time, respectively in hours and days, required by each classifier to make flat predictions on the 30 *GO* terms randomly chosen and finally the columns **Hours Estimate** and **Days Estimate** refers to the total amount of time, respectively in hours and days, required by each flat approach to compute the predictions considering all the GO classes having 10 or more annotations, i.e the number of terms shown in the column **Classes** ≥ 10 of the Table 5.2.

A.3 Empirical Time Complexity Pipeline

For each of the considered model organisms, we adopted the following procedure to assess the empirical time complexity of each flat method, by using the default parameter setting reported in Table 5.3:

1. during the training phase we selected from the *STRING* network the first 100 top-ranked features (i.e. columns of the weighted adjacency matrix) using the classical Pearson's correlation coefficient;

2. during the test phase we computed the performance metrics (AUPRC and AUROC) and we stored the flat predictions, using the most informative features selected in 1;

3. we took note of the computational time required to do the step 1 and 2, that is the time to compute a single fold;

4. we repeated the step 3 for all the k fold of the cross-validation, with $k = 5$, achieving in this way the computational time to compute a single *GO* class;

5. we repeated the step 4 for 30 randomly chosen *GO* classes;

6. we calculated the average time of a single *GO* class, using the total elapsed time obtained in 5;

7. we multiplied the average time calculated in 6 for all the *GO* classes listed in the column **Classes** ≥ 10 of the Table 5.2;

A.4 Experimental Results

In Figure A.1 and A.2 are respectively illustrated the heatmap for the metric *AUPRC* and *AUROC*. The flat scores was normalized in the sense of the maximum before applying the hierarchical correction. Instead, in Figure A.3 are depicted the results for the metric F_{max} obtained without normalizing the flat scores before applying the ensemble algorithms.

In Figures A.4, A.5, A.6 are shown the distribution of the *AUPRC* values across the *GO* terms (*BP* ontology) having 10 or more annotations. Instead in Figures A.7, A.8, A.9 are depicted the distribution of the *AUPRC* values across the *GO* terms (*CC* ontology) having 10 or more annotations.

In Table A.4 we compare the average execution time (in minutes) of the flat classifiers against that of the hierarchical ensemble methods. The columns **Flat Timing** and **Hierarchical Timing** refer respectively to the average computational time achieved by the flat classifier over the three domain of *GO* (*BP*, *MF*, *CC*) and to the average running time (across *GO* subontologies) of all the six non-parametric hierarchical ensemble methods (Table 5.4) to correct the flat scores. Finally the third column shows the speed-up of the hierarchical ensemble methods respect the flat ones. The large speed-up of the hierarchical algorithms is mainly due to the fact that the elapsed time of the flat classifier includes both the time for the training and the test phase of the 5-fold cross-validation. Instead the ensemble approaches shown in Table 5.4 are parametric-free and do not require any cross-validated technique for the tuning of the hyper-parameters. Looking at the bar plots of Figure A.10 it is easy to note that the maximum speed-up is achieved in the *CHICK* organism with the *glmnet* method, whereas the minimum speed-up is obtained in *HUMAN* with the flat classifier *lda*.

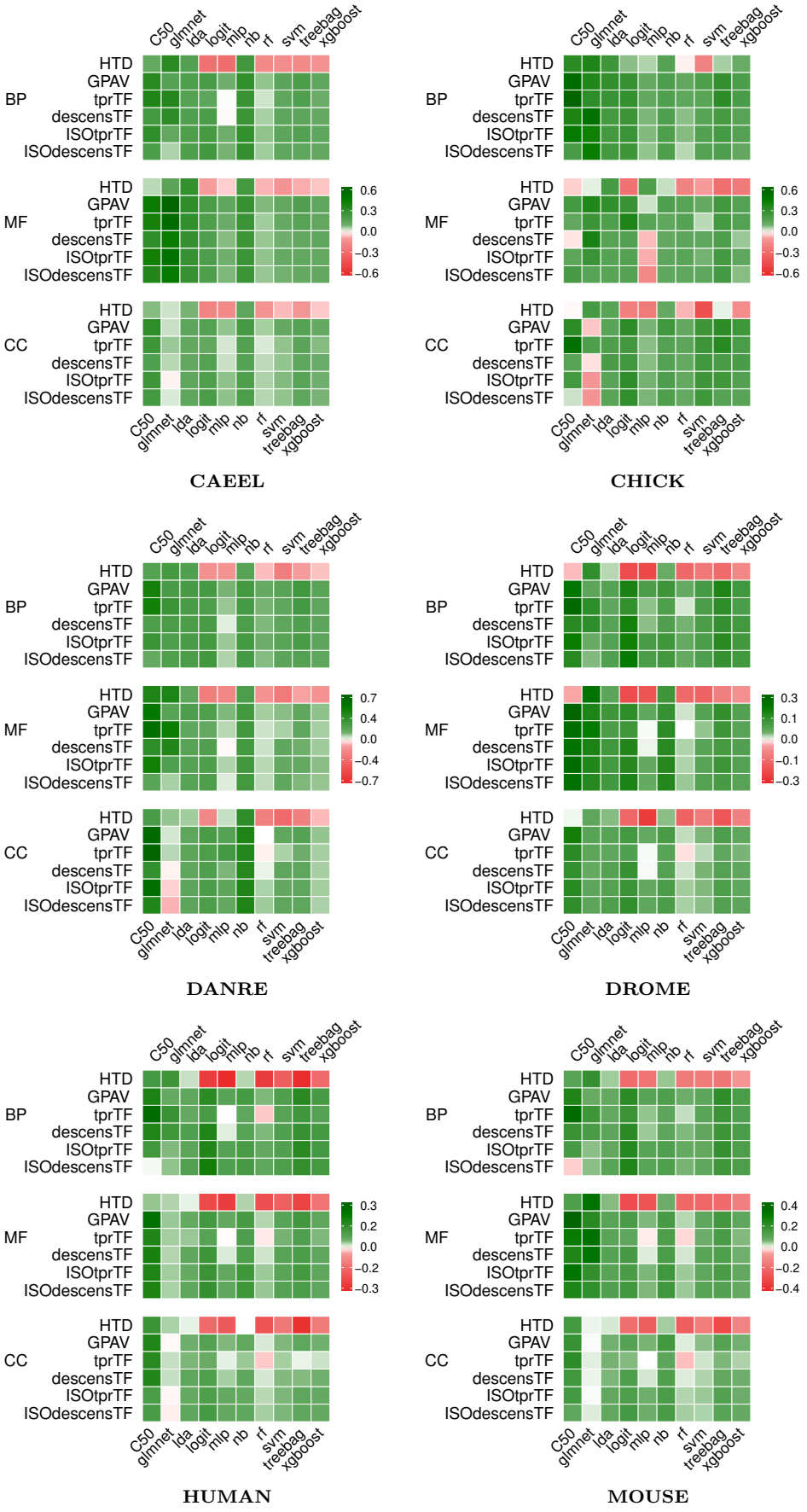


Figure A.1: Heatmap of each model organism for the performance metric *AUPRC*. Flat scores was normalized in the sense of the maximum before applying the hierarchical correction. Refer to text for further details.

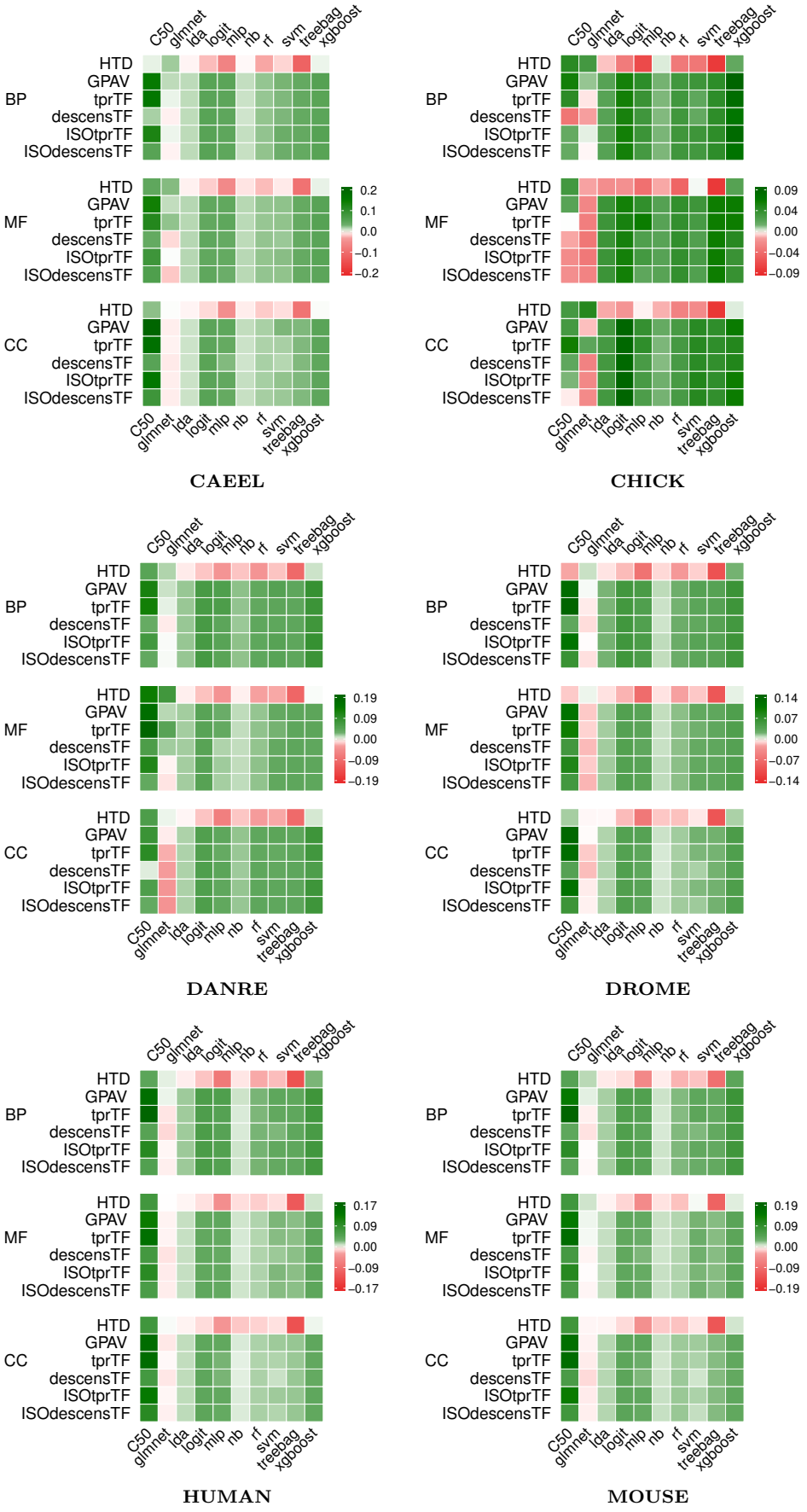


Figure A.2: Heatmap of each model organism for the performance metric *AUROC*. Flat scores was normalized in the sense of the maximum before applying the hierarchical correction. Refer to text for further details.

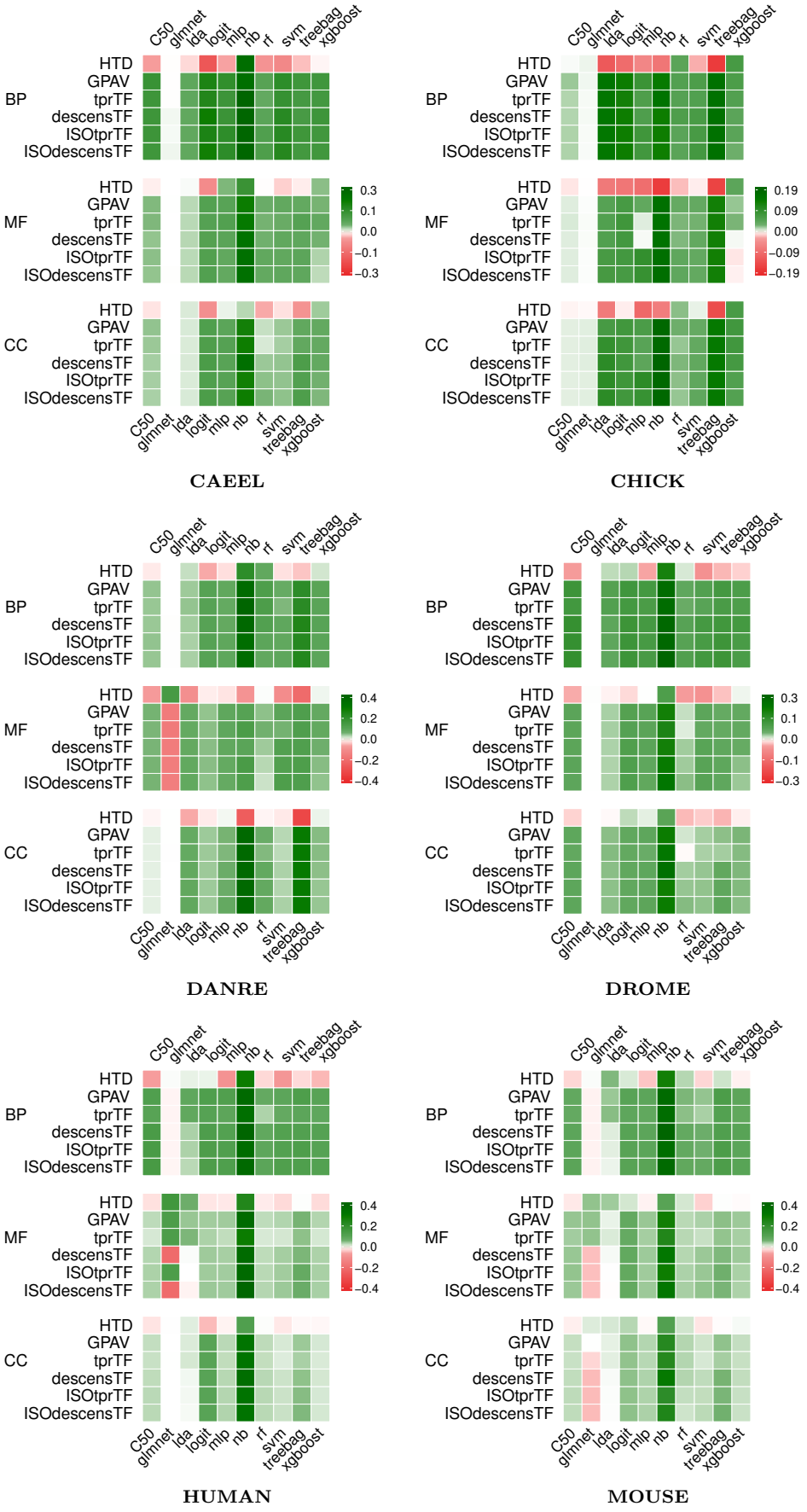


Figure A.3: Heatmap of each model organism for the performance metric F_{max}. None normalization method was applied upon flat scores before applying the hierarchical correction. Refer to text for further details.

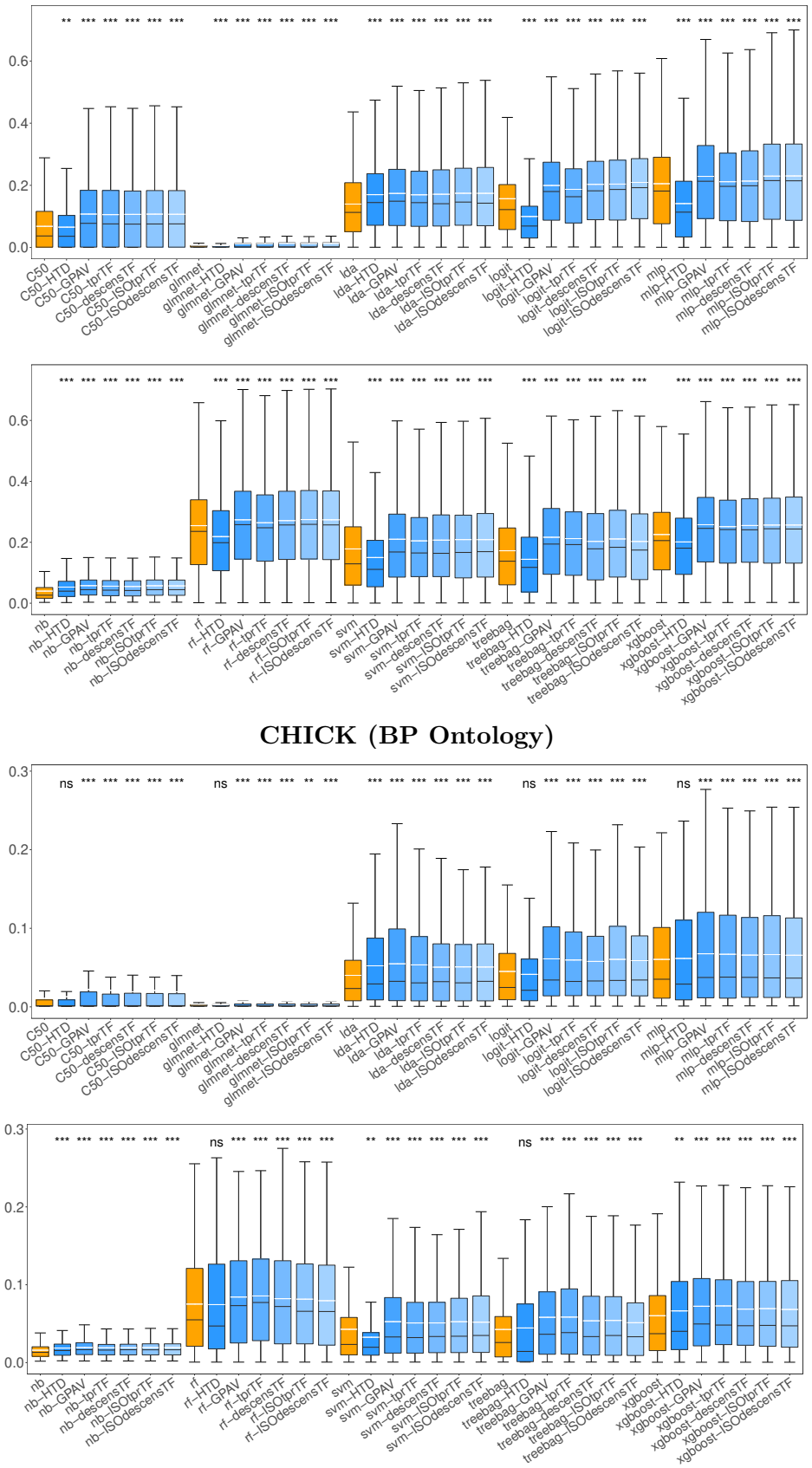


Figure A.4: Distribution of *AUPRC* values across the *GO* terms (*BP* ontology) having 10 or more annotation respectively for the model organisms *C. elegans* (CAEEL) and *G. Gallus* (CHICK). None normalization was applied on the flat scores before the hierarchical correction.

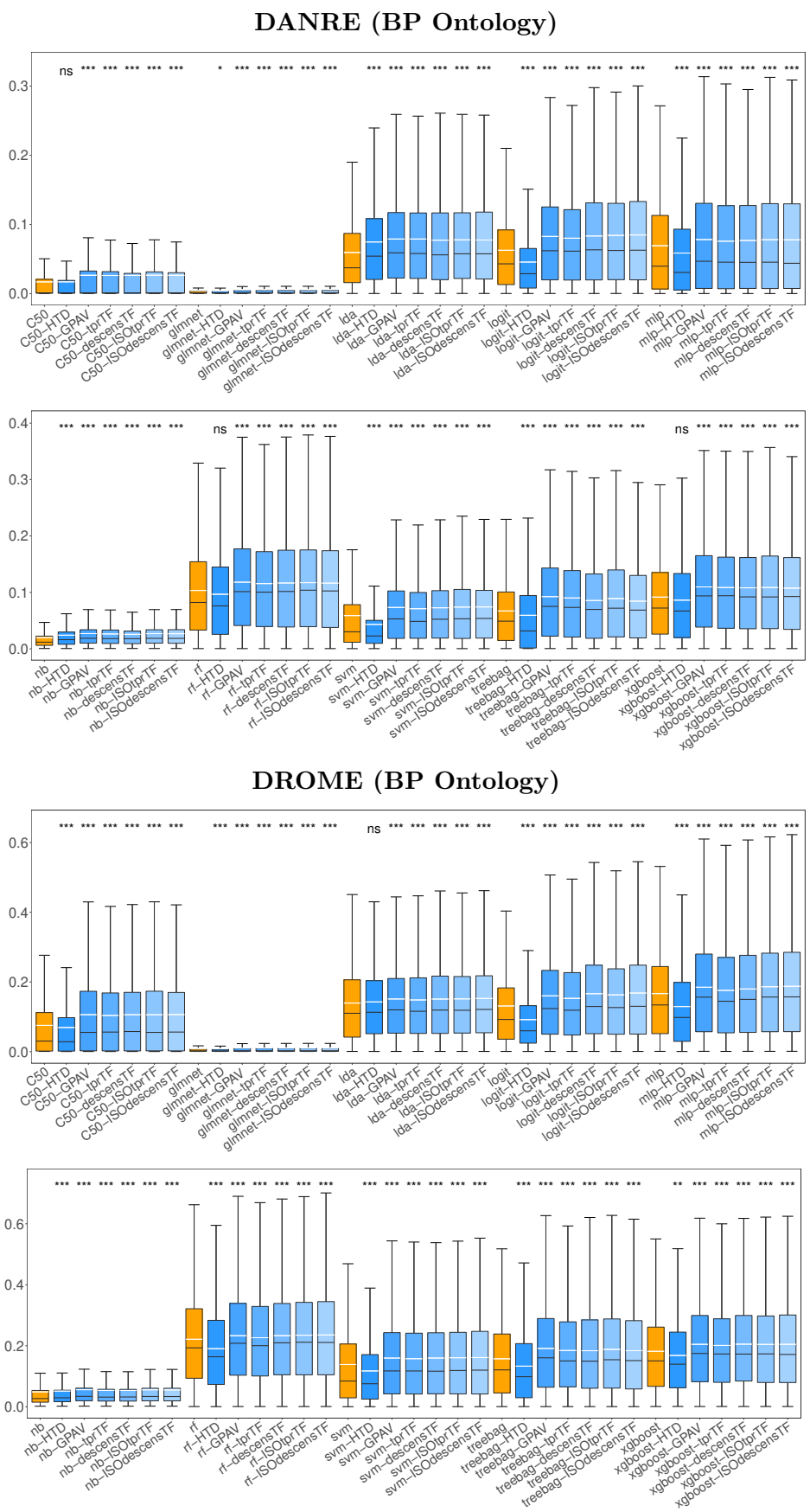


Figure A.5: Distribution of *AUPRC* values across the *GO* terms (*BP* ontology) having 10 or more annotation respectively for the model organisms *D. rerio* (DANRE) and *D. melanogaster* (DROME). None normalization was applied on the flat scores before the hierarchical correction.

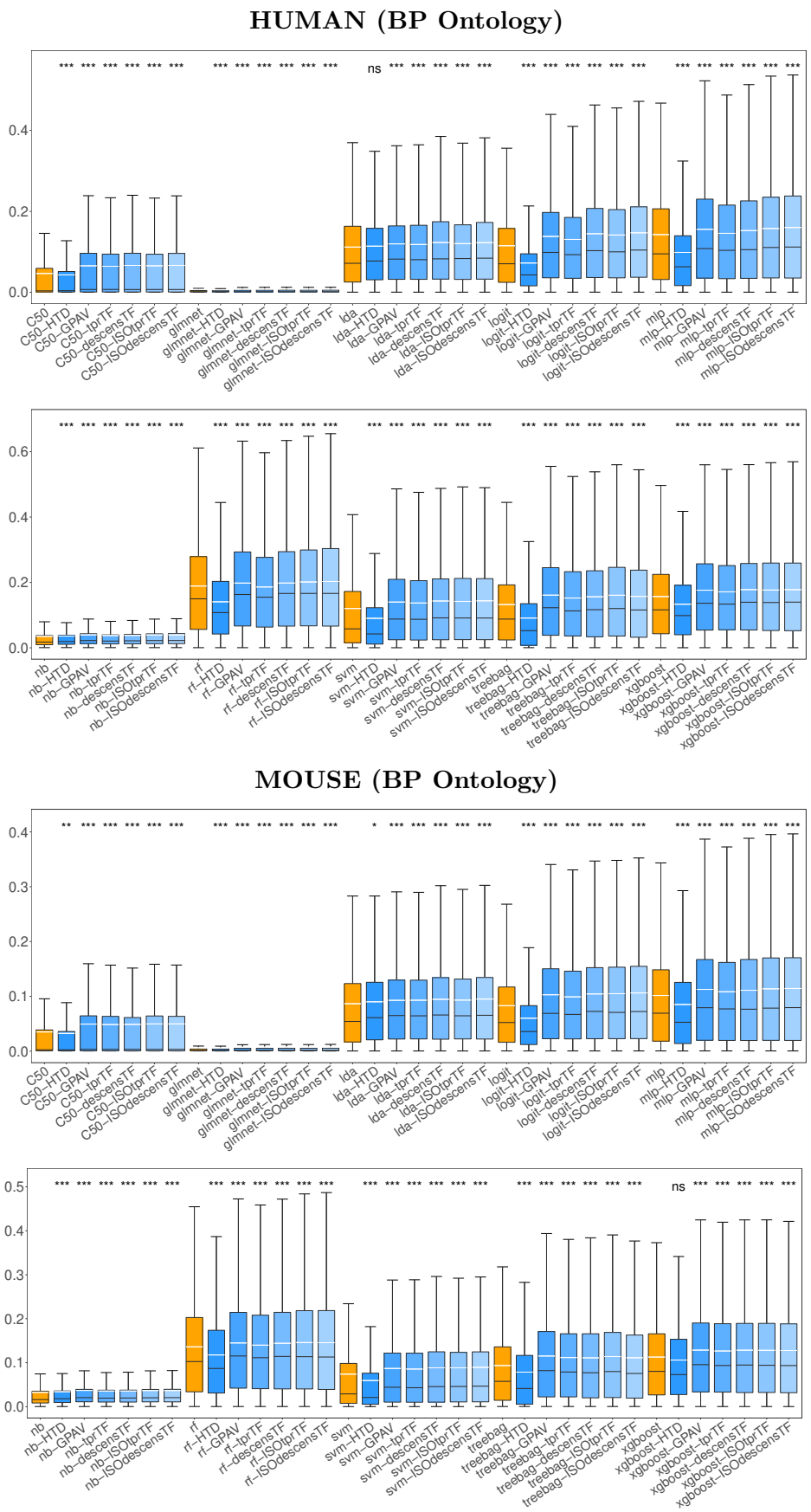


Figure A.6: Distribution of *AUPRC* values across the *GO* terms (*BP* ontology) having 10 or more annotation respectively for the model organisms *H. sapiens* (HUMAN) and *M. musculus* (MOUSE). None normalization was applied on the flat scores before the hierarchical correction.

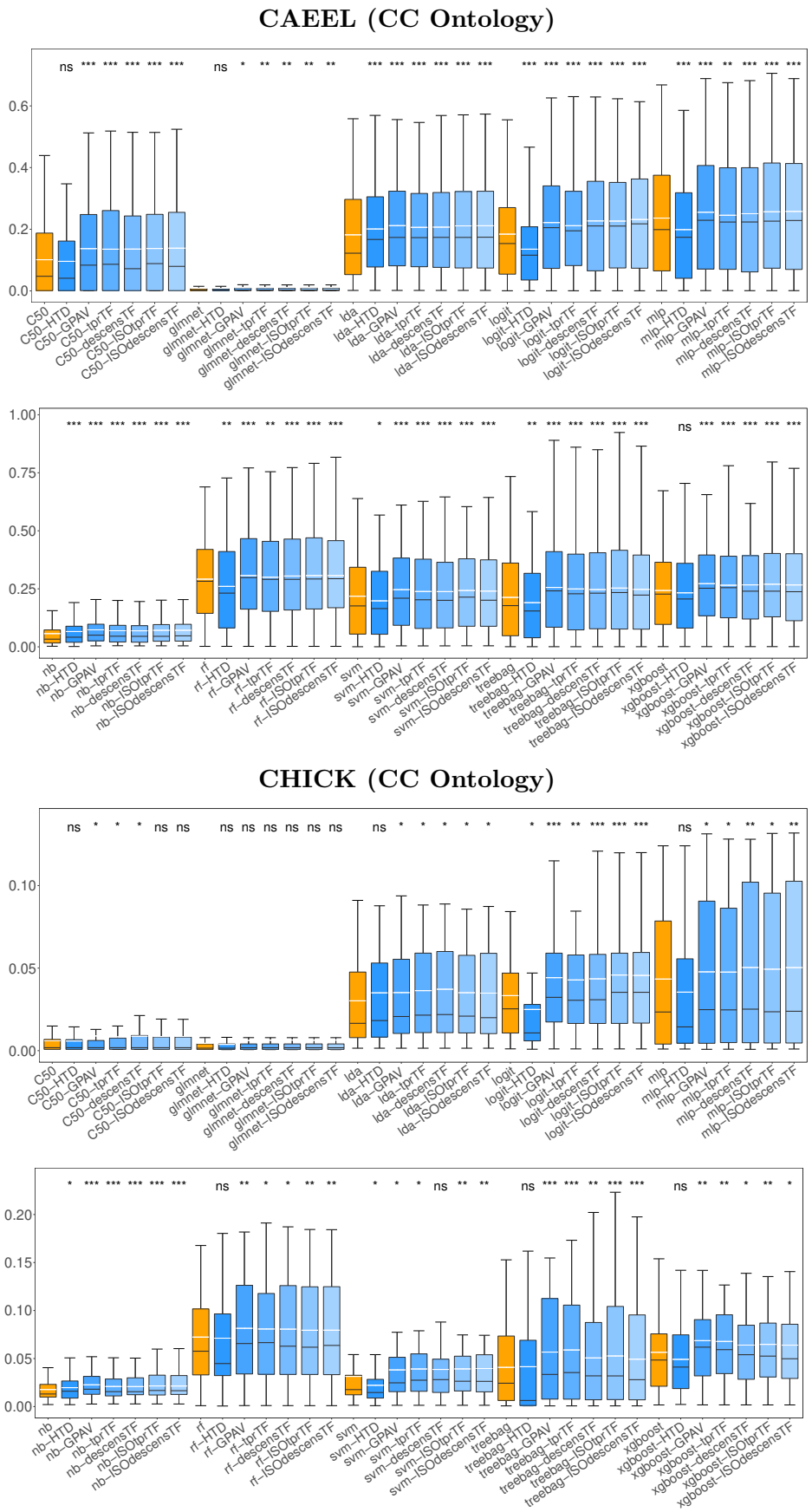


Figure A.7: Distribution of *AUPRC* values across the *GO* terms (*CC* ontology) having 10 or more annotation respectively for the model organisms *C. elegans* (CAEEL) and *G. Gallus* (CHICK). None normalization was applied on the flat scores before the hierarchical correction.

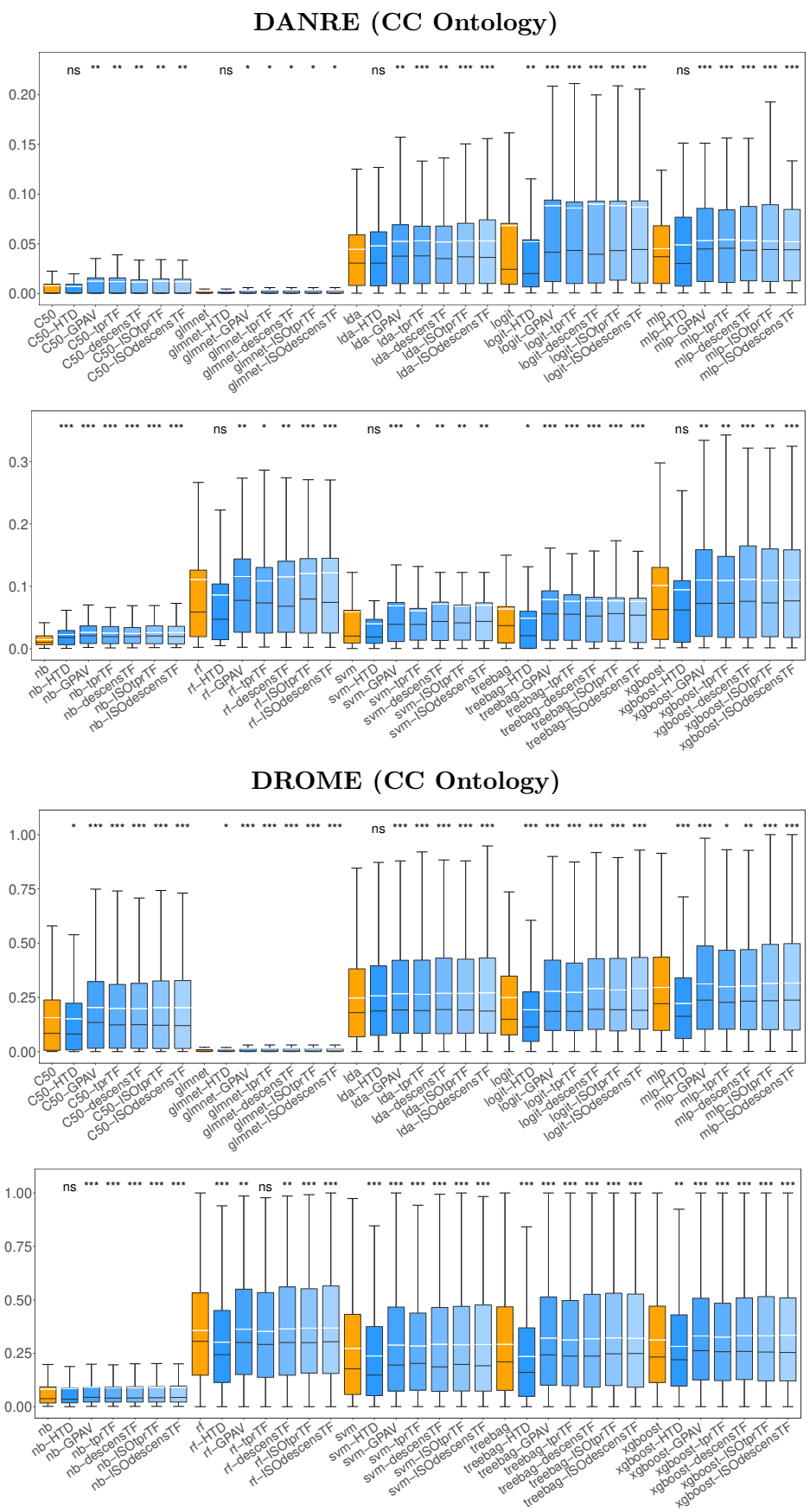


Figure A.8: Distribution of *AUPRC* values across the *GO* terms (*CC* ontology) having 10 or more annotation respectively for the model organisms *D. rerio* (DANRE) and *D. melanogaster* (DROME). None normalization was applied on the flat scores before the hierarchical correction.

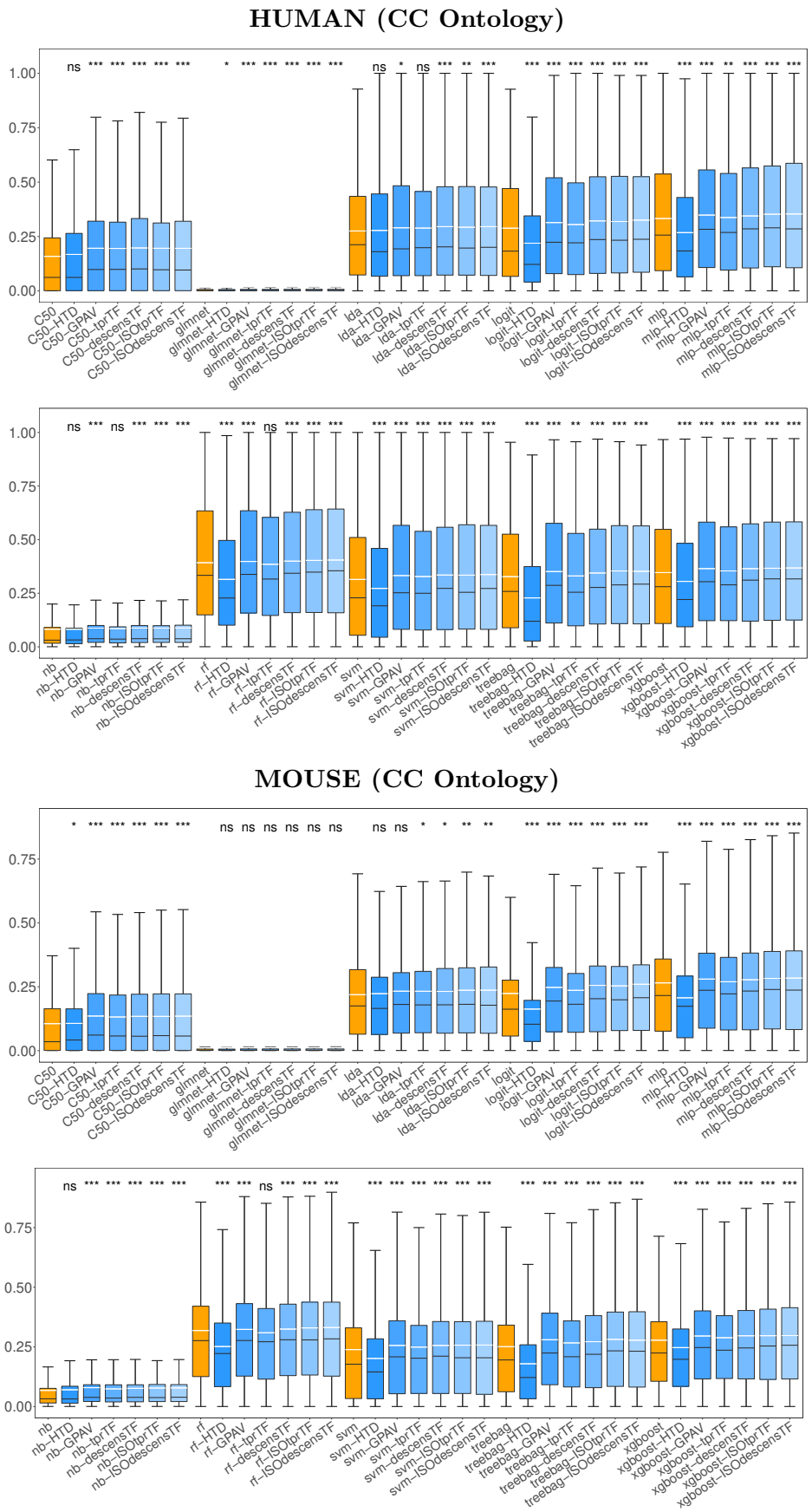


Figure A.9: Distribution of *AUPRC* values across the *GO* terms (*CC* ontology) having 10 or more annotation respectively for the model organisms *H. sapiens* (HUMAN) and *M. musculus* (MOUSE). None normalization was applied on the flat scores before the hierarchical correction.

Table A.4: Computational complexity (in minutes) of flat classifiers vs. hierarchical ensemble methods. See text for further details about each column entry.

Organism	Flat Classifier	Flat Timing	Hierarchical Timing	Speed-Up
CAEEL	C5.0 Decision Trees	321.593	11.987	26.828
	Glmnet	1085.77	6.436	168.703
	Linear Discriminant Analysis	230.817	12.44	18.554
	Logit Boost	235.25	6.782	34.687
	Multi Layer Perceptron	517.723	12.431	41.648
	Naive Bayes	196.623	12.02	16.358
	Random Forest	413.367	11.51	35.914
	Support Vector Machine	413.43	11.467	36.054
	Bagging Ensemble of Decision Tree	788.097	6.543	120.449
	Extreme Gradient Boosting	310.243	11.623	26.692
CHICK	C5.0 Decision Trees	104.453	0.777	134.431
	Glmnet	243.71	0.725	336.152
	Linear Discriminant Analysis	91.463	0.808	113.197
	Logit Boost	91.17	0.711	128.228
	Multi Layer Perceptron	140.06	0.863	162.294
	Naive Bayes	85.46	0.762	112.152
	Random Forest	105.037	0.848	123.864
	Support Vector Machine	111.473	0.745	149.628
	Bagging Ensemble of Decision Tree	184.003	0.711	258.795
	Extreme Gradient Boosting	117.197	0.842	139.189
DANRE	C5.0 Decision Trees	1579.36	14.106	111.964
	Glmnet	2518.833	7.546	333.797
	Linear Discriminant Analysis	1583.543	16.912	93.634
	Logit Boost	1177.283	9.323	126.277
	Multi Layer Perceptron	1689.153	15.388	109.771
	Naive Bayes	1264.457	16.072	78.675
	Random Forest	1077.747	15.687	68.703
	Support Vector Machine	1658.31	14.378	115.337
	Bagging Ensemble of Decision Tree	1112.157	9.239	120.376
	Extreme Gradient Boosting	1073.753	14.858	72.268
DROME	C5.0 Decision Trees	2307.08	41.487	55.61
	Glmnet	3021.547	12.067	250.398
	Linear Discriminant Analysis	2046.9	42.589	48.062
	Logit Boost	1796.327	19.828	90.595
	Multi Layer Perceptron	2227.793	41.988	53.058
	Naive Bayes	1796.743	41.731	43.055
	Random Forest	818.277	40.264	20.323
	Support Vector Machine	2072.273	39.839	52.016
	Bagging Ensemble of Decision Tree	1878.28	19.741	95.146
	Extreme Gradient Boosting	414.327	40.342	10.27
HUMAN	C5.0 Decision Trees	1294.683	115.926	11.168
	Glmnet	3093.427	33.742	91.679
	Linear Discriminant Analysis	875.75	121.143	7.229
	Logit Boost	915.453	54.801	16.705
	Multi Layer Perceptron	1937.6	121.695	15.922
	Naive Bayes	874.7	119.586	7.314
	Random Forest	1116.37	115.752	9.644
	Support Vector Machine	2185.65	119.202	18.336
	Bagging Ensemble of Decision Tree	3267.49	59.825	54.617
	Extreme Gradient Boosting	1108.483	124.471	8.906
MOUSE	C5.0 Decision Trees	1793.42	94.616	18.955
	Glmnet	3771.763	29.058	129.801
	Linear Discriminant Analysis	1175.71	94.081	12.497
	Logit Boost	1241.787	47.001	26.42
	Multi Layer Perceptron	2446.127	93.453	26.175
	Naive Bayes	1386.397	99.17	13.98
	Random Forest	1644.5	94.538	17.395
	Support Vector Machine	2585.91	96.612	26.766
	Bagging Ensemble of Decision Tree	3661.213	51.174	71.544
	Extreme Gradient Boosting	1340.663	97.34	13.773

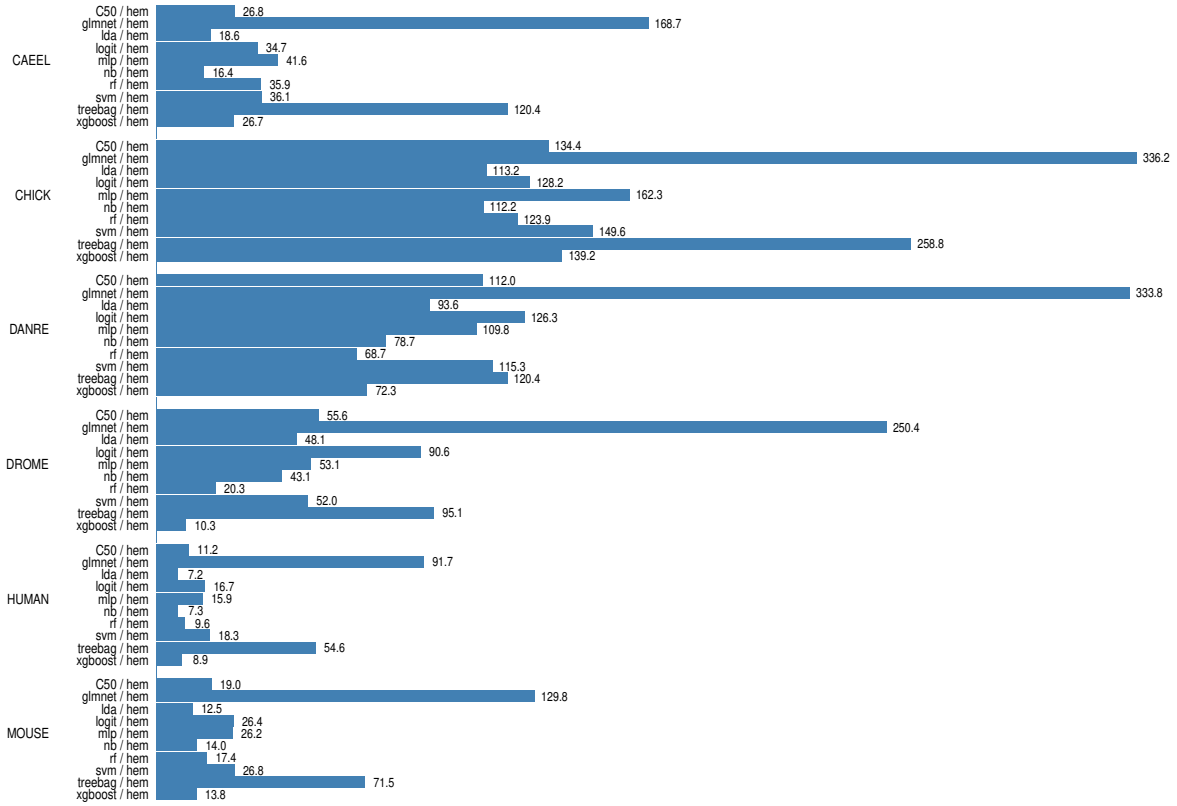


Figure A.10: Speed-up of the hierarchical ensemble algorithms respect to the flat approaches

B Prediction of HPO Terms: Supplementary Material

Here are collected all the supplementary materials of experiments on the *HPO* terms prediction (Chapter 6).

Table B.1: Average *AUROC* across terms and average F_{max}, Precision and Recall across genes of *HTD-DAG* and *TPR-DAG* ensemble variants using *RANKS* and *SVMs* as base learner and the *STRING* network. Results of "flat" *RANKS* and *SVMs* are also reported. Results are estimated through 5-fold cross-validation. For each sub-ontology and each metric best results are highlighted in bold. Results significantly better than all the others methods according to the Wilcoxon Rank Sum test ($\alpha = 10^{-6}$) are underlined.

Subontology	Method	AUROC	F_{max}	Precision	Recall
Organ	*RANKS*	0.8540	0.3048	0.2349	0.4338
	SVM	0.7440	0.4188	0.3598	0.5008
	HTD-RANKS	0.8812	0.3743	0.3041	0.4865
	HTD-SVM	0.7475	0.4249	0.3739	0.4919
	TPR-T-RANKS	0.8604	0.3775	0.3048	0.4973
	TPR-T-SVM	0.7602	0.4318	0.3850	0.4931
	DESCENS-T-RANKS	0.8598	0.3986	0.3409	0.4798
	DESCENS-T-SVM	0.7723	0.4341	**0.3883**	0.4933
	TPR-TF-RANKS	0.8609	0.3752	0.3056	0.4858
	TPR-TF-SVM	0.7584	0.4330	0.3696	**0.5227**
	TPR-W-RANKS	**0.8857**	0.3999	0.3429	0.4805
	TPR-W-SVM	0.7713	**0.4354**	0.3782	0.5107
Inheritance	*RANKS*	0.8983	0.5601	0.4292	0.8061
	SVM	0.8164	0.6835	0.5876	0.8167
	HTD-RANKS	0.9005	0.5682	0.4392	0.8045
	HTD-SVM	0.8101	0.6869	0.5892	0.8234
	TPR-T-RANKS	0.9043	0.5718	0.4469	0.7950
	TPR-T-SVM	0.8187	0.6894	0.5939	0.8219
	DESCENS-T-RANKS	0.9044	0.5718	0.4469	0.7950
	DESCENS-T-SVM	0.8187	0.6894	0.5939	0.8219
	TPR-TF-RANKS	**0.9150**	0.5385	0.4024	0.8139
	TPR-TF-SVM	0.8254	0.6848	0.5854	**0.8248**
	TPR-W-RANKS	0.9147	0.5718	0.4470	0.7950
	TPR-W-SVM	0.8187	**0.6898**	**0.5951**	0.8208
Onset	*RANKS*	0.8325	0.4143	0.3025	0.6568
	SVM	0.7365	0.4656	0.3689	0.6309
	HTD-RANKS	**0.8605**	0.4174	0.2999	0.6861
	HTD-SVM	0.7433	0.4584	0.3651	0.6156
	TPR-T-RANKS	0.8575	0.4184	0.2926	**0.7432**
	TPR-T-SVM	0.7432	0.4616	0.3691	0.6171
	DESCENS-T-RANKS	0.8563	0.4372	0.3214	0.6868
	DESCENS-T-SVM	0.7434	0.4618	0.3594	0.6528
	TPR-TF-RANKS	0.8580	0.4218	0.3046	0.6853
	TPR-TF-SVM	0.7463	0.4668	0.3670	0.6410
	TPR-W-RANKS	0.8573	0.4401	0.3264	0.7004
	TPR-W-SVM	0.7442	**0.4770**	**0.3741**	0.6644

Table B.2: Average *AUROC* across terms and average F_{max}, Precision and Recall across genes of *HTD-DAG* and *TPR-DAG* ensemble variants using *RANKS* and *SVMs* as base learner and the *UA* integrated network. Results are estimated through 5-fold cross-validation. For each sub-ontology and each metric best results are highlighted in bold. Results significantly better than all the others methods according to the Wilcoxon Rank Sum test ($\alpha = 10^{-6}$) are underlined.

Subontology	Method	AUROC	F_{max}	Precision	Recall
Organ	*RANKS*	0.8493	0.3106	0.2407	0.4377
	SVM	0.7128	0.1668	0.1688	0.1648
	HTD-RANKS	0.8446	0.3411	0.2717	0.4583
	HTD-SVM	0.8328	0.3155	0.2370	0.4718
	TPR-T-RANKS	**0.8646**	0.3575	0.2933	0.4586
	TPR-T-SVM	0.8240	0.3155	0.2370	0.4718
	DESCENS-T-RANKS	0.8408	**0.3773**	**0.3231**	0.4538
	DESCENS-T-SVM	0.7883	0.3209	0.2800	0.3782
	TPR-TF-RANKS	0.8420	0.3547	0.2880	0.4615
	TPR-TF-SVM	0.7060	0.2359	0.2074	0.2736
	TPR-W-RANKS	0.8377	0.3620	0.3025	0.4515
	TPR-W-SVM	0.8238	0.3171	0.2343	**0.4916**
Inheritance	*RANKS*	0.8715	0.5986	0.4709	0.8215
	SVM	0.7637	0.1668	0.1688	0.1648
	HTD-RANKS	**0.8846**	0.6034	0.4709	0.8396
	HTD-SVM	0.7991	0.5429	0.3885	0.9008
	TPR-T-RANKS	0.8708	0.6066	**0.4795**	0.8276
	TPR-T-SVM	0.7909	0.5444	0.3892	0.9063
	DESCENS-T-RANKS	0.8659	0.6064	0.4793	0.8278
	DESCENS-T-SVM	0.7919	0.5441	0.3889	0.9063
	TPR-TF-RANKS	0.8675	0.6029	0.4757	0.8228
	TPR-TF-SVM	0.7546	0.4480	0.2972	**0.9099**
	TPR-W-RANKS	0.8636	**0.6070**	0.4769	0.8355
	TPR-W-SVM	0.7900	0.5448	0.3897	0.9059
Onset	*RANKS*	0.8159	0.4494	0.3821	0.5455
	SVM	0.7025	0.3978	0.2717	0.7421
	HTD-RANKS	**0.8253**	0.4466	0.3563	0.5981
	HTD-SVM	0.7781	0.4492	0.3374	0.6719
	TPR-T-RANKS	0.8249	0.4631	0.3791	0.6021
	TPR-T-SVM	0.7511	0.4517	0.3385	0.6813
	DESCENS-T-RANKS	0.8170	**0.4684**	0.3839	0.6077
	DESCENS-T-SVM	0.7513	0.4519	0.3394	0.6780
	TPR-TF-RANKS	0.8148	0.4553	0.3782	0.5718
	TPR-TF-SVM	0.6788	0.3983	0.2714	**0.7483**
	TPR-W-RANKS	0.7901	0.4682	**0.3914**	0.5856
	TPR-W-SVM	0.7496	0.4522	0.3404	0.6757

Table B.3: Prediction of newly annotated genes. Average *AUROC* and *AUPRC* across 2444 *HPO* terms and average F_{max}, Precision and Recall of *HTD-DAG* and *TPR-DAG* ensemble variants across the newly annotated 608 genes. Best results for each metric are highlighted in bold.

Meas. / Methods	AUROC	AUPRC	F_{max}	Precision	Recall
HTD	0.6464	0.1207	0.3794	0.3581	0.4033
TPR-T	0.6466	0.1209	0.3795	0.3580	0.4036
DESCENS-T	0.6465	0.1211	0.3794	**0.3584**	0.4030
TPR-TF	0.6498	0.1224	0.3812	0.3560	0.4101
TPR-W	**0.6512**	**0.1237**	**0.3826**	0.3512	**0.4202**

Table B.4: Comparison among *HTD-DAG* and *TPR-DAG* ensemble variants considering only the best predictions for the newly annotated genes. Average AUROC and AUPRC across terms and average F_{max}, Precision and Recall across genes considering only HPO terms with AUROC > 0.7 (778 terms) and $F_{max} > 0.3$ (296 genes). Best results for each metric are highlighted in bold.

Meas. / Methods	AUROC	AUPRC	F_{max}	Precision	Recall
HTD	0.8155	0.1551	0.4716	0.4429	0.5042
TPR-T	0.8155	0.1553	0.4714	0.4443	0.5021
DESCENS-T	0.8155	0.1554	0.4714	0.4436	0.5030
TPR-TF	0.8200	0.1577	0.4765	0.4419	**0.5171**
TPR-W	**0.8219**	**0.1594**	**0.4793**	**0.4572**	0.5037

Table B.5: The best predicted *HPO* terms sorted in descending order on the basis of *AUROC*. Depth stands for the maximum distance of a given term from the root node (in the considered *HPO* graph the longest path from a node to a root is 14). Distance from leaves indicates the minimum distance of a given node from one of the leaves of the *HPO-DAG*. A value equal to 0 is assigned to the leaves, 1 to nodes with distance 1 from a leaf and so on.

HPO ID	HPO term	AUROC	Depth	Distance from Leaves
HP:0004942	Aortic aneurysm	1.000	7	0
HP:0100760	Clubbing of toes	1.000	9	0
HP:0000114	Proximal tubulopathy	0.9984	9	0
HP:0003557	Increased variability in muscle fiber diameter	0.9984	6	0
HP:0002040	Esophageal varix	0.9967	7	0
HP:0002913	Myoglobinuria	0.9967	6	0
HP:0003642	Type I transferrin isoform profile	0.9967	8	0
HP:0006477	Abnormality of the alveolar ridges	0.9967	7	1
HP:0006846	Acute encephalopathy	0.9967	5	1
HP:0006965	Acute necrotizing encephalopathy	0.9967	6	0
HP:0008316	Abnormal mitochondria in muscle tissue	0.9967	6	0
HP:0004339	Abnormality of sulfur amino acid metabolism	0.9967	5	0
HP:0004944	Cerebral aneurysm	0.9967	8	0
HP:0002725	Systemic lupus erythematosus	0.9967	5	0
HP:0001659	Aortic regurgitation	0.9951	7	0
HP:0004353	Abnormality of pyrimidine metabolism	0.9951	4	0
HP:0000831	Insulin-resistant diabetes mellitus	0.9934	6	0
HP:0001019	Erythroderma	0.9934	7	0
HP:0002223	Absent eyebrow	0.9934	9	0
HP:0003160	Abnormal isoelectric focusing of serum transferrin	0.9934	7	1
HP:0004481	Progressive macrocephaly	0.9934	9	0
HP:0010459	True hermaphroditism	0.9934	7	0
HP:0012345	Abnormal glycosylation	0.9934	4	4
HP:0012346	Abnormal protein glycosylation	0.9934	5	3
HP:0012347	Abnormal protein N-linked glycosylation	0.9934	6	2
HP:0003521	Disproportionate short-trunk short stature	0.9926	6	0
HP:0010996	Abnormality of monocarboxylic acid metabolism	0.9918	4	0
HP:0000991	Xanthomatosis	0.9901	6	0
HP:0003645	Prolonged partial thromboplastin time	0.9901	4	0
HP:0009161	Aplasia/Hypoplasia of the middle phalanx of the 5th finger	0.9885	11	0
HP:0010932	Abnormality of nucleobase metabolism	0.9885	3	1
HP:0003076	Glycosuria	0.9869	7	0
HP:0011016	Abnormality of urine glucose concentration	0.9869	6	1
HP:0002304	Akinesia	0.9868	6	0
HP:0003953	Absent forearm bone	0.9868	9	1
HP:0003974	Absent radius	0.9868	10	0
HP:0009822	Aplasia involving forearm bones	0.9868	9	1
HP:0003215	Dicarboxylic aciduria	0.9868	8	0
HP:0010995	Abnormality of dicarboxylic acid metabolism	0.9868	4	1
HP:0004219	Abnormality of the middle phalanx of the 5th finger	0.9852	10	1
HP:0002085	Occipital encephalocele	0.9835	8	0
HP:0000677	Oligodontia	0.9819	10	0

HPO ID	HPO term	AUROC	Depth	Distance from Leaves
HP:0001218	Autoamputation	0.9819	3	0
HP:0003254	Abnormality of DNA repair	0.9819	4	0
HP:0100864	Short femoral neck	0.9819	10	0
HP:0009027	Foot dorsiflexor weakness	0.9810	7	0
HP:0009108	Foot dorsiflexor weakness	0.9802	9	1
HP:0002905	Hyperphosphatemia	0.9802	5	0
HP:0001436	Abnormality of the foot musculature	0.9794	5	1
HP:0002839	Urinary bladder sphincter dysfunction	0.9786	7	0
HP:0009617	Abnormality of the distal phalanx of the thumb	0.9786	11	0
HP:0002097	Emphysema	0.9785	5	0
HP:0100631	Neoplasm of the adrenal gland	0.9781	5	0
HP:0000073	Ureteral duplication	0.9769	6	0
HP:0002181	Cerebral edema	0.9769	8	0
HP:0002557	Hypoplastic nipples	0.9753	5	0
HP:0004359	Abnormality of fatty-acid metabolism	0.9753	4	0
HP:0009720	Adenoma sebaceum	0.9736	8	0
HP:0010615	Angiofibromas	0.9736	7	1
HP:0003233	Hypoalphalipoproteinemia	0.9730	7	0
HP:0010980	Hyperlipoproteinemia	0.9727	6	0
HP:0003002	Breast carcinoma	0.9723	5	0
HP:0001480	Freckling	0.9720	6	0
HP:0005293	Venous insufficiency	0.9720	5	0
HP:0005613	Aplasia/hypoplasia of the femur	0.9720	8	2
HP:0006443	Patellar aplasia	0.9719	9	0
HP:0002025	Anal stenosis	0.9711	8	0
HP:0002298	Absent hair	0.9711	6	1
HP:0007431	Congenital ichthyosiform erythroderma	0.9703	8	1
HP:0001992	Organic aciduria	0.9703	7	1
HP:0006498	Aplasia/Hypoplasia of the patella	0.9691	8	1
HP:0003310	Abnormality of the odontoid process	0.9686	6	1
HP:0009888	Abnormality of secondary sexual hair	0.9678	5	0
HP:0003311	Hypoplasia of the odontoid process	0.9671	7	0
HP:0010979	Abnormality of the level of lipoprotein cholesterol	0.9661	5	1
HP:0002898	Embryonal neoplasm	0.9655	4	1
HP:0004311	Abnormality of macrophages	0.9653	6	0
HP:0001974	Leukocytosis	0.9638	6	0
HP:0100578	Lipoatrophy	0.9638	5	0
HP:0000625	Cleft eyelid	0.9631	8	0
HP:0007361	Abnormality of the pons	0.9625	8	0
HP:0010981	Hypolipoproteinemia	0.9625	6	1
HP:0002667	Nephroblastoma (Wilms tumor)	0.9615	8	0
HP:0002612	Congenital hepatic fibrosis	0.9576	6	0
HP:0012144	Abnormality of cells of the monocyte/macrophage lineage	0.9570	4	1
HP:0011794	Embryonal renal neoplasm	0.9569	7	1
HP:0009733	Glioma	0.9555	7	1
HP:0002967	Cubitus valgus	0.9541	7	0
HP:0012072	Aciduria	0.9539	6	2
HP:0010286	Abnormality of the salivary glands	0.9539	7	0
HP:0010675	Abnormal foot bone ossification	0.9539	7	0

HPO ID	HPO term	AUROC	Depth	Distance from Leaves
HP:0010899	Abnormality of aspartate family amino acid metabolism	0.9539	5	0
HP:0002922	Increased CSF protein	0.9522	6	0
HP:0000855	Insulin resistance	0.9516	5	1

Table B.6: Top 5 candidate genes for each of the 65 leaf nodes showed in the Appendix Table B.5. Refer to Section 6.4 for more details about the selection of the candidate genes for each *HPO* leaf term.

HPO ID	HPO term	Entrez Gene ID	Gene Symbol
HP:0004942	Aortic aneurysm	72	ACTG2
HP:0100760	Clubbing of toes	7516	XRCC2
HP:0000114	Proximal tubulopathy	1892	ECHS1
HP:0003557	Increased variability in muscle fiber diameter	8516	ITGA8
		3694	ITGB6
HP:0002040	Esophageal varix	85440	DOCK7
		7049	TGFBR3
HP:0002913	Myoglobinuria	1962	EHHADH
		1892	ECHS1
HP:0003642	Type I transferrin isoform profile	201595	STT3B
		1109	AKR1C4
		199857	ALG14
HP:0006965	Acute necrotizing encephalopathy	4697	NDUFA4
		4539	MT-ND4L
HP:0008316	Abnormal mitochondria in muscle tissue	4697	NDUFA4
		51204	TACO1
		1352	COX10
		4539	MT-ND4L
		4538	MT-ND4
HP:0004339	Abnormality of sulfur amino acid metabolism	1757	SARDH
		55811	ADCY10
		1137	CHRNA4
		271	AMPD2
		272	AMPD3
HP:0004944	Cerebral aneurysm	1757	SARDH
		55811	ADCY10
		91137	SLC25A46
		271	AMPD2
		272	AMPD3
HP:0002725	Systemic lupus erythematosus	629	CFB
HP:0001659	Aortic regurgitation	27229	TUBGCP4
		2239	GPC4
		3340	NDST1
		57822	GRHL3
		5378	PMS1
HP:0004353	Abnormality of pyrimidine metabolism	790	CAD
		7516	XRCC2
		580	BARD1
		10747	MASP2
		58484	NLRC4
HP:0000831	Insulin-resistant diabetes mellitus	3991	LIPE
		3764	KCNJ8
		5260	PHKG1

HPO ID	HPO term	Entrez Gene ID	Gene Symbol
HP:0001019	Erythroderma	3149	HMGB3
		7516	XRCC2
		5657	PRTN3
		28952	CCDC22
		2224	FDPS
HP:0002223	Absent eyebrow	9463	PICK1
		389549	FEZF1
HP:0004481	Progressive macrocephaly	4697	NDUFA4
		4538	MT-ND4
		54539	NDUFB11
		4539	MT-ND4L
HP:0010459	True hermaphroditism	54585	LZTFL1
		9786	KIAA0586
		11020	IFT27
		57822	GRHL3
		26281	FGF20
HP:0003521	Disproportionate short-trunk short stature	54567	DLL4
		3371	TNC
		9096	TBX18
HP:0010996	Abnormality of monocarboxylic acid metabolism	790	CAD
HP:0000991	Xanthomatosis	9971	NR1H4
		4047	LSS
		3991	LIPE
		80347	COASY
		4744	NEFH
HP:0003645	Prolonged partial thromboplastin time	2266	FGG
		3026	HABP2
		10747	MASP2
		733	C8G
		199857	ALG14
HP:0009161	Aplasia/Hypoplasia of the middle phalanx of the 5th finger	26585	GREM1
		2535	FZD2
		57216	VANGL2
HP:0003076	Glycosuria	140628	GATA5
		388753	COA6
HP:0002304	Akinesia	1139	CHRNA7
		9037	SEMA5A
		54567	DLL4
		8506	CNTNAP1
		3763	KCNJ6
HP:0003974	Absent radius	6909	TBX2
		26585	GREM1
		7468	WHSC1
		23129	PLXND1
		6997	TDGF1
HP:0003215	Dicarboxylic aciduria	1962	EHHADH
		1892	ECHS1
		3417	IDH1
		4329	ALDH6A1

HPO ID	HPO term	Entrez Gene ID	Gene Symbol
		126129	CPT1C
HP:0002085	Occipital encephalocele	284217	LAMA1
		1109	AKR1C4
		85301	COL27A1
		8516	ITGA8
		3694	ITGB6
HP:0000677	Oligodontia	25885	POLR1A
		2535	FZD2
		7161	TP73
		7049	TGFBR3
		28952	CCDC22
HP:0001218	Autoamputation	4916	NTRK3
		51164	DCTN4
		10715	CERS1
		127833	SYT2
		10087	COL4A3BP
HP:0003254	Abnormality of DNA repair	7516	XRCC2
		4437	MSH3
		7469	NELFA
		580	BARD1
		79991	STN1
HP:0100864	Short femoral neck	11020	IFT27
		2239	GPC4
		283375	SLC39A5
		2817	GPC1
		22978	NT5C2
HP:0009027	Foot dorsiflexor weakness	56776	FMN2
		4744	NEFH
		5859	QARS
HP:0002905	Hyperphosphatemia	844	CASQ1
		26281	FGF20
		6809	STX3
		80055	PGAP1
HP:0002839	Urinary bladder sphincter dysfunction	25894	PLEKHG4
		81570	CLPB
		6327	SCN2B
		23025	UNC13A
		27445	PCLO
HP:0009617	Abnormality of the distal phalanx of the thumb	2619	GAS1
		26585	GREM1
		57216	VANGL2
		22978	NT5C2
		84976	DISP1
HP:0002097	Emphysema	5450	POU2AF1
		3105	HLA-A
		7049	TGFBR3
HP:0100631	Neoplasm of the adrenal gland	1021	CDK6
		7468	WHSC1
		3481	IGF2

HPO ID	HPO term	Entrez Gene ID	Gene Symbol
HP:0000073	Ureteral duplication	9091	PIGQ
		7161	TP73
		126129	CPT1C
		1962	EHHADH
		284098	PIGW
HP:0002181	Cerebral edema	790	CAD
		4697	NDUFA4
		4538	MT-ND4
		4539	MT-ND4L
		1892	ECHS1
HP:0002557	Hypoplastic nipples	9091	PIGQ
		6909	TBX2
		4040	LRP6
		284098	PIGW
		120	ADD3
HP:0004359	Abnormality of fatty-acid metabolism	1962	EHHADH
		1892	ECHS1
		126129	CPT1C
		4329	CPT1C
		57468	SLC12A5
HP:0009720	Adenoma sebaceum	2475	MTOR
		4437	MSH3
		2272	FHIT
		5293	PIK3CD
		5378	PMS1
HP:0003233	Hypoalphalipoproteinemia	9971	NR1H4
		3991	LIPE
		5260	PHKG1
		27229	TUBGCP4
		5446	PON3
HP:0010980	Hyperlipoproteinemia	9971	NR1H4
		5446	PON3
		8694	DGAT1
		5445	PON2
HP:0003002	Breast carcinoma	4437	MSH3
HP:0001480	Freckling	4437	MSH3
		5378	PMS1
		7161	TP73
		1021	CDK6
		1031	CDKN2C
HP:0005293	Venous insufficiency	54567	DLL4
		1303	COL12A1
		1399	CRKL
		3371	TNC
		6678	SPARC
HP:0006443	Patellar aplasia	7994	KAT6A
HP:0002025	Anal stenosis	6909	TBX2
		80055	PGAP1

HPO ID	HPO term	Entrez Gene ID	Gene Symbol
HP:0009888	Abnormality of secondary sexual hair	6909	TBX2
		7161	TP73
		4744	NEFH
		4047	LSS
		6862	T
HP:0003311	Hypoplasia of the odontoid process	3371	TNC
		4774	NFIA
		22926	ATF6
		8239	USP9X
		26137	ZBTB20
HP:0004311	Abnormality of macrophages	2214	FCGR3A
		330	BIRC3
		4671	NAIP
		58484	NLRC4
		6809	STX3
HP:0001974	Leukocytosis	3551	IKBKB
		3932	LCK
		5657	PRTN3
		58484	NLRC4
		330	BIRC3
HP:0100578	Lipoatrophy	3991	LIPE
		23022	PALLD
		844	CASQ1
		29119	CTNNA3
		6006	RHCE
HP:0000625	Cleft eyelid	23129	PLXND1
		246243	RNASEH1
		9463	PICK1
		25885	POLR1A
		10512	SEMA3C
HP:0007361	Abnormality of the pons	5297	PI4KA
		9091	PIGQ
		284098	PIGW
		80055	PGAP1
		163786	SASS6
HP:0002667	Nephroblastoma (Wilms tumor)	580	BARD1
HP:0002612	Congenital hepatic fibrosis	11020	CYP4B1
		54585	LZTFL1
		85440	85440
		51164	DCTN4
		55690	PACS1
HP:0002967	Cubitus valgus	26585	GREM1
		26281	FGF20
HP:0010286	Abnormality of the salivary glands	4437	MSH3
		7161	TP73
		5378	PMS1
		2272	FHIT
		3417	IDH1

HPO ID	HPO term	Entrez Gene ID	Gene Symbol
HP:0010675	Abnormal foot bone ossification	4047	LSS
		3149	HMGB3
		440275	EIF2AK4
		389549	FEZF1
		22926	ATF6
HP:0010899	Abnormality of aspartate family amino acid metabolism	1757	SARDH
		55811	ADCY10
		271	AMPD2
		272	AMPD3
		91137	SLC25A46
HP:0002922	Increased CSF protein	57468	SLC12A5
		3106	HLA-B
		3105	HLA-A
		4179	CD46
		283375	SLC39A5

C HEMDAG: Tutorial

Here we show some examples of how to perform experiments using the *HEMDAG*, the in-house R software library implementing the hierarchical ensemble methods proposed in Chapter 4, to the *GO* (Chapter 5) and *HPO* (Chapter 6) term prediction. For the full and detailed step-by-step tutorial, please visit the following link. The documentation were created using *SPHINX* (link). The *HEMDAG* library is publicly available both from CRAN and BIOCONDA repository under the GNU General Public License, version 3 (GPL-3.0).

C.1 Application to the Hierarchical Prediction of GO terms

Let us illustrate now a step-by-step application of *HEMDAG* package to the hierarchical prediction of protein functions of the model organism *DROME* (D. melanogaster), one of the six model organisms used in the experiments illustrated in Chapter 5. The data used for the experiments shown below are available at: `https://github.com/marconotaro/HEMDAG/tree/master/docs/data/`.

Firstly, we must load the input data (i.e. the flat scores matrix S, the graph g and the annotation table *ann*) and we store them in the directory *data*. The output data (i.e. the hierarchical scores matrix and the performances) will be store in the folder *results*:

```
# loading input data
link <- "https://raw.githubusercontent.com/marconotaro/HEMDAG/master/docs/data/";
load(url(paste0(link,"7227_DROME_GO_MF_DAG_STRING_v10.5_20DEC17.rda")));
load(url(paste0(link,"7227_DROME_GO_MF_ANN_STRING_v10.5_20DEC17.rda")));
load(url(paste0(link,"Scores.7227.DROME.GO.MF.pearson.100.feature.LogitBoost.5fcv.rda")));

if(!dir.exists("data"))
    dir.create("data");

if(!dir.exists("results"))
    dir.create("results");

# storing data
save(g,file="data/7227_DROME_GO_MF_DAG_STRING_v10.5_20DEC17.rda");
save(ann,file="data/7227_DROME_GO_MF_ANN_STRING_v10.5_20DEC17.rda");
save(S,file="data/Scores.7227.DROME.GO.MF.pearson.100.feature.LogitBoost.5fcv");
```

Here we will perform experiments by executing the "true-path-rule"-based hierarchical learning algorithms *GPAV* and *ISO-TPR*. In the experiments shown below, we considered the flat scores matrix achieved by using as base learner *Logit Boost* with its default parameter setting (see Table 5.3). We normalized the flat scores matrix in the sense of the maximum, i.e. the score of each *GO* term was normalized by dividing the score values for the maximum score of that class (variable $norm.type = MaxNorm$).

Now we are ready to run the *GPAV* high-level function:

```
Do.GPAV( norm=FALSE, norm.type= "MaxNorm", W=NULL, parallel=TRUE, ncores=7, folds=NULL,
         seed=NULL, n.round=3, f.criterion ="F", recall.levels=seq(from=0.1, to=1, by=0.1),
         flat.file=flat.file, ann.file=ann.file, dag.file=dag.file, flat.dir=flat.dir,
         ann.dir=ann.dir, dag.dir=dag.dir, hierScore.dir=hierScore.dir,
         perf.dir=perf.dir, compute.performance=TRUE);
```

By looking at the results it easy to see that *GPAV* outperforms *LogitBoost*:

```
## loading results
basefolder <- "results/";
perf <- "PerfMeas.MaxNorm.Scores"
orgonto <- "7227.DROME.GO.MF";
baselearner <- "pearson.100.feature.LogitBoost.5fcv";
hiermeth <- "hierScores.GPAV";
file <- paste0(basefolder, perf, ".", orgonto, ".", baselearner, ".", hiermeth, ".rda");
load(file);

## AUC performance: flat vs hierarchical
AUC.flat$average
[1] 0.8211
AUC.hier$average
[1] 0.8552

## PRC performance: flat vs hierarchical
PRC.flat$average
[1] 0.1995
PRC.hier$average
[1] 0.2352

## F-score performance: flat vs hierarchical
FMM.flat$average
   P      R      S      F      avF    A      T
0.4255 0.5515 0.9781 0.4803 0.4055 0.9684 0.1190
FMM.hier$average
   P      R      S      F      avF    A      T
0.4837 0.5582 0.9830 0.5183 0.4398 0.9735 0.1080

## Precision at different recall levels: flat vs hierarchical
PXR.flat$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.4053 0.3349 0.2795 0.2304 0.1839 0.1349 0.0911 0.0597 0.0314 0.0105
PXR.hier$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.4687 0.3896 0.3356 0.2924 0.2352 0.1778 0.1223 0.0797 0.0401 0.0119
```

Similarly as done for the experiment above, we can execute *ISO-TPR* algorithm (a *TPR-DAG* variant), where in the top-down step instead of applying the *HTD-DAG* strategy we use the *GPAV* algorithm.

1. *ISO-TPR-TF*: *ISO-TPR* Threshold-Free variant with "positive" children

```
Do.TPR.DAG( threshold=0, weight=0, kk=NULL, folds=NULL, seed=NULL, norm=FALSE,
            norm.type="MaxNorm", positive="children", bottomup="threshold.free",
            topdown="GPAV", W=NULL, parallel=TRUE, ncores=7, n.round=3, f.criterion="F",
            metric=NULL, recall.levels=seq(from=0.1, to=1, by=0.1), flat.file=flat.file,
            ann.file=ann.file, dag.file=dag.file, flat.dir=flat.dir, ann.dir=ann.dir,
            dag.dir=dag.dir, hierScore.dir=hierScore.dir, perf.dir=perf.dir,
            compute.performance=TRUE);
```

2. *ISO-DESCENS-TF*: *ISO-TPR* Threshold-Free variant with "positive" descendants

```
Do.TPR.DAG( threshold=0, weight=0, kk=NULL, folds=NULL, seed=NULL, norm=FALSE,
          norm.type="MaxNorm", positive="descendants", bottomup="threshold.free",
          topdown="GPAV", W=NULL, parallel=TRUE, ncores=7, n.round=3, f.criterion="F",
          metric=NULL, recall.levels=seq(from=0.1, to=1, by=0.1), flat.file=flat.file,
          ann.file=ann.file, dag.file=dag.file, flat.dir=flat.dir, ann.dir=ann.dir,
          dag.dir=dag.dir, hierScore.dir=hierScore.dir, perf.dir=perf.dir,
          compute.performance=TRUE);
```

By loading the results of the *ISO-TPR-TF* variant, we can see that the ensemble method improves upon *LogitBoost* performances:

```
## loading results
basefolder <- "results/";
perf <- "PerfMeas.MaxNorm.Scores"
orgonto <- "7227.DROME.GO.MF";
baselearner <- "pearson.100.feature.LogitBoost.5fcv";
hiermeth <- "hierScores.ISOtprTF";
file <- paste0(basefolder, perf, ".", orgonto, ".", baselearner, ".", hiermeth, ".rda");
load(file);

## AUC performance: flat vs hierarchical
AUC.flat$average
[1] 0.8211
(AUC.hier$average
[1] 0.8544

## PRC performance: flat vs hierarchical
PRC.flat$average
[1] 0.1995
PRC.hier$average
[1] 0.2397

## F-score performance: flat vs hierarchical
FMM.flat$average
   P      R      S      F      avF    A      T
0.4255 0.5515 0.9781 0.4803 0.4055 0.9684 0.1190
FMM.hier$average
   P      R      S      F      avF    A      T
0.4820 0.5652 0.9822 0.5203 0.4413 0.9729 0.1200

## Precision at different recall levels: flat vs hierarchical
PXR.flat$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.4053 0.3349 0.2795 0.2304 0.1839 0.1349 0.0911 0.0597 0.0314 0.0105
PXR.hier$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.4764 0.4025 0.3444 0.2938 0.2383 0.1773 0.1226 0.0794 0.0402 0.0119
```

By looking at the results we can see that *ISO-DESCENS-TF* outperforms *LogitBoost*:

```
## loading results
basefolder <- "results/";
perf <- "PerfMeas.MaxNorm.Scores"
orgonto <- "7227.DROME.GO.MF";
baselearner <- "pearson.100.feature.LogitBoost.5fcv";
hiermeth <- "hierScores.ISOdescensTF";
file <- paste0(basefolder, perf, ".", orgonto, ".", baselearner, ".", hiermeth, ".rda");
load(file);

## AUC performance: flat vs hierarchical
AUC.flat$average
[1] 0.8211
AUC.hier$average
[1] 0.8549

## PRC performance: flat vs hierarchical
PRC.flat$average
[1] 0.1995
PRC.hier$average
[1] 0.2449

## F-score performance: flat vs hierarchical
FMM.flat$average
   P      R      S      F      avF    A      T
0.4255 0.5515 0.9781 0.4803 0.4055 0.9684 0.1190
FMM.hier$average
   P      R      S      F      avF    A      T
0.4798 0.5683 0.9817 0.5203 0.4406 0.9725 0.1200

## Precision at different recall levels: flat vs hierarchical
PXR.flat$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.4053 0.3349 0.2795 0.2304 0.1839 0.1349 0.0911 0.0597 0.0314 0.0105
PXR.hier$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.5023 0.4109 0.3528 0.3027 0.2427 0.1785 0.1226 0.0781 0.0400 0.0118
```

C.2 Application to the Hierarchical Prediction of HPO terms

For the sake of simplicity, in the examples shown below we make use of the pre-built dataset available in the *HEMDAG* library. The *HEMDAG* library provides high-level functions for batch experiments, where input and output data must be stored in compressed **rda** files. For the following experiments we store the input data (i.e. the flat scores matrix **S**, the graph **g** and the annotation table **L**) in the directory **data** and the output data (i.e. the hierarchical scores matrix and the performances) in the folder **results**:

```
# loading data
data(graph);
data(scores);
data(labels);

if(!dir.exists("data"))
    dir.create("data");

if(!dir.exists("results"))
    dir.create("results");

# storing data
save(g,file="data/graph.rda");
save(L,file="data/labels.rda");
save(S,file="data/scores.rda");
```

Now, we can experiment with different ensemble variants by properly changing the arguments of the high-level functions. Below we show how to perform experiments by using *HTD-DAG* and some of the *TPR-DAG* ensemble variants.

1. *HTD-DAG* algorithm

```
Do.HTD( norm=FALSE, norm.type="MaxNorm", folds=3, seed=1, n.round=3, f.criterion="F",
        recall.levels=seq(from=0.1, to=1, by=0.1), flat.file="scores", ann.file="labels",
        dag.file="graph", flat.dir="data/", ann.dir="data/", dag.dir="data/",
        hierScore.dir="results/", perf.dir="results/", compute.performance=TRUE);
```

2. *TPR-TF*: *TPR-DAG* Threshold-Free variant with "positive" children

```
Do.TPR.DAG( threshold=0, weight=0, kk=NULL, folds=3, seed=1, norm=FALSE,
            norm.type="MaxNorm", positive="children", bottomup="threshold.free",
            topdown="HTD", n.round=3, f.criterion="F", metric=NULL, flat.file="scores",
            recall.levels=seq(from=0.1, to=1, by=0.1), ann.file="labels", dag.file="graph",
            flat.dir="data/", ann.dir="data/", dag.dir="data/", hierScore.dir="results/",
            perf.dir="results/", compute.performance=TRUE);
```

3. *DESCENS-TF*: *DESCENS* Threshold-Free variant with "positive" descendants

```
Do.TPR.DAG( threshold=0, weight=0, kk=NULL, folds=3, seed=1, norm=FALSE,
            norm.type="MaxNorm", positive="descendants", bottomup="threshold.free",
            topdown="HTD", n.round=3, f.criterion="F", metric=NULL, flat.file="scores",
            recall.levels=seq(from=0.1, to=1, by=0.1), ann.file="labels", dag.file="graph",
            flat.dir="data/", ann.dir="data/", dag.dir="data/", hierScore.dir="results/",
            perf.dir="results/", compute.performance=TRUE);
```

4. *TPR-T*: *TPR-DAG* Threshold variant with "positive" children, maximizing the threshold on the *F-score* (*FMAX*)

```
Do.TPR.DAG( threshold=seq(0.1,0.9,0.1), weight=0, kk=5, folds=3, seed=1, norm=FALSE,
            norm.type="Qnorm", positive="children", bottomup="threshold", topdown="HTD",
            n.round=3, f.criterion="F", metric="FMAX",
            recall.levels=seq(from=0.1, to=1, by=0.1), flat.file="scores",
            ann.file="labels", dag.file="graph", flat.dir="data/",
            ann.dir="data/", dag.dir="data/", hierScore.dir="results/",
            perf.dir="results/", compute.performance=TRUE);
```

5. *DESCENS-$\mathcal{T}$*: *TPR-DAG* variant with "positive" descendants, maximizing the threshold on the *F-score* (*FMAX*)

```
Do.TPR.DAG( threshold=seq(0.1,0.9,0.1), weight=0, kk=5, folds=3, seed=1, norm=FALSE,
            norm.type="Qnorm", positive="descendants", bottomup="tau", topdown="HTD",
            n.round=3, f.criterion="F", metric="FMAX",
            recall.levels=seq(from=0.1, to=1, by=0.1), flat.file="scores",
            ann.file="labels", dag.file="graph", flat.dir="data/",
            ann.dir="data/", dag.dir="data/", hierScore.dir="results/",
            perf.dir="results/", compute.performance=TRUE);
```

By loading, for instances, the results of *HTD-DAG* and *TPR-TF* variant, we can see that both the ensemble methods outperform the flat approach *RANKS*.

Results achieved by *HTD-DAG* algorithm:

```
## loading results
load("results/PerfMeas.MaxNorm.scores.hierScores.HTD.rda");

## AUC performance: flat vs hierarchical
AUC.flat$average
[1] 0.8263
AUC.hier$average
[1] 0.8312

## PRC performance: flat vs hierarchical
PRC.flat$average
[1] 0.4373
PRC.hier$average
[1] 0.4827

## F-score performance: flat vs hierarchical
FMM.flat$average
   P      R      S      F      avF    A      T
0.7071 0.6443 0.6853 0.6743 0.5768 0.7314 0.7020
FMM.hier$average
   P      R      S      F      avF    A      T
0.5087 0.9394 0.4430 0.6600 0.5922 0.6570 0.4457

## Precision at different recall levels: flat vs hierarchical
PXR.flat$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.5872 0.5872 0.5872 0.5715 0.5715 0.4487 0.4361 0.4361 0.4361 0.4361
PXR.hier$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.6465 0.6465 0.6465 0.6227 0.6227 0.4996 0.4897 0.4897 0.4897 0.4897
```

Results achieved by *TPR-T* algorithm:

```
## loading results
load("results/PerfMeas.MaxNorm.scores.hierScores.tprTF.rda");

## AUC performance: flat vs hierarchical
AUC.flat$average
[1] 0.8263
AUC.hier$average
[1] 0.8424

## PRC performance: flat vs hierarchical
PRC.flat$average
[1] 0.4373
PRC.hier$average
[1] 0.5042

## F-score performance: flat vs hierarchical
FMM.flat$average
   P      R      S      F      avF    A      T
0.7071 0.6443 0.6853 0.6743 0.5768 0.7314 0.7020
FMM.hier$average
   P      R      S      F      avF    A      T
0.6174 0.7624 0.6045 0.6823 0.5824 0.7355 0.6113

## Precision at different recall levels: flat vs hierarchical
PXR.flat$avgPXR
  0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9     1
0.5872 0.5872 0.5872 0.5715 0.5715 0.4487 0.4361 0.4361 0.4361 0.4361
PXR.hier$avgPXR
  0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9     1
0.6740 0.6740 0.6740 0.6556 0.6556 0.5142 0.4951 0.4951 0.4951 0.4951
```

HEMDAG library allows to do also classical hold-out experiments. Respect to the cross-validated experiments performed above, we only need to load the indices of the examples to be used in the test set:

```
data(test.index);
save(test.index, file="data/test.index.rda");
```

Now we can perform hold-out experiments. Below we perform experiments by using respectively the hold-out version of *HTD-DAG* and *TPR-DAG* algorithm (*DESCENS-T* variant).

1. *HTD-DAG*

```
Do.HTD.holdout( norm=FALSE, norm.type="MaxNorm", n.round=3, f.criterion ="F", folds=NULL,
                seed=NULL, recall.levels=seq(from=0.1, to=1, by=0.1), flat.file="scores",
                ann.file="labels", dag.file="graph", flat.dir="data/", ann.dir="data/",
                dag.dir="data/", ind.test.set="test.index", ind.dir="data/",
                hierScore.dir="results_ho/", perf.dir="results_ho/",
                compute.performance=TRUE);
```

2. *DESCENS-T*: *DESCENS* Threshold variants with "positive" descendants

```
Do.TPR.DAG.holdout( threshold=seq(0.1,0.9,0.1), weight=0, kk=5, folds=NULL, seed=1, norm=FALSE,
                    norm.type="MaxNorm", positive="descendants", bottomup="threshold",
                    topdown="HTD", recall.levels=seq(from=0.1, to=1, by=0.1), n.round=3,
                    f.criterion="F", metric="PRC", flat.file="scores", ann.file="labels",
                    dag.file="graph", flat.dir="data/", ann.dir="data/", dag.dir="data/",
                    ind.test.set="test.index", ind.dir="data/", hierScore.dir="results_ho/",
                    perf.dir="results_ho/", compute.performance=TRUE);
```

By loading the results of *HTD-DAG* and *DESCENS-T* we can see that both the ensemble approaches improve upon "RANKS" performances:

Results achieved by *HTD-DAG* algorithm:

```
## loading results
load("results_ho/PerfMeas.MaxNorm.scores.hierScores.HTD.rda");

## AUC performance: flat vs hierarchical
AUC.flat$average
[1] 0.8621
AUC.hier$average
[1] 0.8997

## PRC performance: flat vs hierarchical
PRC.flat$average
[1] 0.2789
PRC.hier$average
[1] 0.4504

## F-score performance: flat vs hierarchical
FMM.flat$average
   P      R      S      F      avF    A      T
0.5952 0.8182 0.4190 0.6891 0.6404 0.7424 0.3770
FMM.hier$average
   P      R      S      F      avF    A      T
0.5589 0.9444 0.2824 0.7023 0.6506 0.6818 0.3590

## Precision at different recall levels: flat vs hierarchical
PXR.flat$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.4424 0.4424 0.4424 0.4379 0.4379 0.3708 0.3621 0.3621 0.3621 0.3621
PXR.hier$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.6629 0.6629 0.6629 0.6174 0.6174 0.4698 0.4547 0.4547 0.4547 0.4547
```

Results obtained by *DESCENS-T* algorithm:

```
## loading results
load("results_ho/PerfMeas.MaxNorm.scores.hierScores.descensT.rda");

## AUC performance: flat vs hierarchical
AUC.flat$average
[1] 0.8621
AUC.hier$average
[1] 0.8789

## PRC performance: flat vs hierarchical
PRC.flat$average
[1] 0.2789
PRC.hier$average
[1] 0.5482

## F-score performance: flat vs hierarchical
FMM.flat$average
   P      R      S      F      avF    A      T
0.5952 0.8182 0.4190 0.6891 0.6404 0.7424 0.3770
FMM.hier$average
   P      R      S      F      avF    A      T
0.7481 0.8889 0.5532 0.8125 0.7809 0.8788 0.5510

## Precision at different recall levels: flat vs hierarchical
PXR.flat$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.4424 0.4424 0.4424 0.4379 0.4379 0.3708 0.3621 0.3621 0.3621 0.3621
PXR.hier$avgPXR
   0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.7538 0.7538 0.7538 0.6932 0.6932 0.4851 0.4796 0.4796 0.4796 0.4796
```

Bibliography

[1] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Guldener, G. Mannhaupt, M. Munsterkotter, and H. Mewes, "The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes," *Nucleic Acids Research*, vol. 32, no. 18, pp. 5539–5545, 2004.

[2] The Gene Ontology Consortium, "Gene ontology: tool for the unification of biology," *Nature Genet.*, vol. 25, pp. 25–29, 2000.

[3] P. Robinson, S. Kohler, S. Bauer, D. Seelow, D. Horn, and S. Mundlos, "The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease.," *Am. J. Hum. Genet.*, vol. 83, pp. 610–615, 2008.

[4] S. Mostafavi and Q. Morris, "Fast integration of heterogeneous data sources for predicting gene function with limited annotation.," *Bioinformatics (Oxford, England)*, vol. 26, pp. 1759–65, Jul 2010.

[5] A. Sokolov, C. S. Funk, K. Graim, K. Verspoor, and A. Ben-Hur, "Combining heterogeneous data sources for accurate functional annotation of proteins," *BMC Bioinformatics*, vol. 14, no. S-3, p. S10, 2013.

[6] N. Cesa-Bianchi, M. Re, and G. Valentini, "Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference," *Machine Learning*, vol. 88, pp. 209–241, 2012.

[7] M. Frasca, A. Bertoni, and G. Valentini, "Unipred: Unbalance-aware network integration and prediction of protein functions," *Journal of Computational Biology*, vol. 22, no. 12, pp. 1057–1074, 2015.

[8] G. Valentini, "True Path Rule hierarchical ensembles for genome-wide gene function prediction," *IEEE ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, pp. 832–847, 2011.

[9] N. Youngs, D. Penfold-Brown, K. Drew, D. Shasha, and R. Bonneau, "Parametric bayesian priors and better choice of negative examples improve protein function prediction," *Bioinformatics*, vol. 29, no. 9, pp. 1190–1198, 2013.

[10] M. Frasca and D. Malchiodi, "Selection of negative examples for node label prediction through fuzzy clustering techniques," in *Advances in Neural Networks* (S. Bassis, A. Esposito, F. C. Morabito, and E. Pasero, eds.), (Cham), pp. 67–76, Springer International Publishing, 2016.

[11] S. Mostafavi and Q. Morris, "Using the gene ontology hierarchy when predicting gene function," *CoRR*, vol. abs/1205.2622, 2012.

[12] A. S. Juncker, L. J. Jensen, A. Pierleoni, A. Bernsel, M. L. Tress, P. Bork, G. von Heijne, A. Valencia, C. A. Ouzounis, R. Casadio, and S. Brunak, "Sequence-based feature prediction and annotation of proteins.," *Genome Biology*, vol. 10, no. 2, p. 206, 2009.

[13] D. M. A. Martin, M. Berriman, and G. J. Barton, "Gotcha: a new method for prediction of protein function assessed by the annotation of seven genomes.," *BMC bioinformatics*, vol. 5, p. 178, Nov 2004.

[14] T. Hawkins, M. Chitale, S. Luban, and D. Kihara, "Pfp: Automated prediction of gene ontology functional annotations with confidence scores using protein sequence data.," *Proteins*, vol. 74, pp. 566–82, Feb 2009.

[15] R. Sharan, I. Ulitsky, and R. Shamir, "Network-based prediction of protein function.," *Molecular Systems Biology*, vol. 3, p. 88, 2007.

[16] A. Vazquez, A. Flammini, A. Maritan, and A. Vespignani, "Global protein function prediction from protein-protein interaction networks.," *Nature Biotechnology*, vol. 21, pp. 697–700, Jun 2003.

[17] S. Oliver, "Guilt-by-association goes global.," *Nature*, vol. 403, pp. 601–3, Feb 2000.

[18] T. G. O. Consortium, "Creating the gene ontology resource: Design and implementation," *Genome Research*, vol. 11, pp. 1425–1433, aug 2001.

[19] P. Robinson, P. Krawitz, and S. Mundlos, "Strategies for exome and genome sequence data analysis in disease-gene discovery projects.," *Cin. Genet.*, vol. 80, pp. 127 – 132, 2011.

[20] G. Obozinski, G. Lanckriet, C. Grant, J. M., and W. Noble, "Consistent probabilistic output for protein function prediction," *Genome Biology*, vol. 9, pp. 135–142, 2008.

[21] I. Tsochantaridis, T. Joachims, T. Hoffman, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of Machine Learning Research*, vol. 6, pp. 1453–1484, 2005.

[22] K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, and J. Rousu, "Structured output prediction of novel enzyme function with reaction kernels," in *Biomedical Engineering Systems and Technologies. BIOSTEC 2010* (A. Fred, J. Filipe, and H. Gamboa, eds.), pp. 367–379, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[23] C. Lampert and M. Blaschko, "Structured prediction by joint kerenl support estimation," *Machine Learning*, vol. 77, pp. 249–269, 2009.

[24] I. Kahanda, C. Funk, K. Verspoor, and A. Ben-Hur, "PHENOstruct: Prediction of human phenotype ontology terms using heterogeneous data sources.," *F1000Research*, vol. 4, p. 259, 2015.

[25] Z. Barutcuoglu, R. Schapire, and O. Troyanskaya, "Hierarchical multi-label prediction of gene function," *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.

[26] Y. Guan, C. Myers, D. Hess, Z. Barutcuoglu, A. Caudy, and O. Troyanskaya, "Predicting gene function in a hierarchical context with an ensemble of classifiers," *Genome Biology*, vol. 9, p. S3, 2008.

[27] G. Yu, H. Zhu, and C. Domeniconi, "Predicting protein functions using incomplete hierarchical labels," *BMC Bioinformatics*, vol. 16, no. 1, 2015.

[28] M. Notaro, M. Schubach, P. N. Robinson, and G. Valentini, "Prediction of human phenotype ontology terms by means of hierarchical ensemble methods," *BMC Bioinformatics*, vol. 18, p. 449, Oct 2017.

[29] A. Sokolov and A. Ben-Hur, "Hierarchical classification of gene ontology terms using the gostruct method," *Journal of Bioinformatics and Computational Biology*, vol. 8, pp. 357–376, 2010.

[30] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur, *et al.*, "A large-scale evaluation of computational protein function prediction," *Nature methods*, 2013.

[31] Y. Jiang *et al.*, "An expanded evaluation of protein function prediction methods shows an improvement in accuracy," *Genome Biology*, vol. 17, p. 184, 2016.

[32] D. Cozzetto, D. Buchan, K. Bryson, and D. Jones, "Protein function prediction by massive integration of evolutionary analyses and multiple data sources," *BMC Bioinformatics*, vol. 14, no. Suppl 3:S1, 2013.

[33] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, and S. Dzeroski, "Predicting gene function using hierarchical multi-label decision tree ensembles," *BMC Bioinformatics*, vol. 11, p. 2, 2010.

[34] R. Cerri and A. de Carvalho, "New top-down methods using SVMs for hierarchical multilabel classification problems," in *IJCNN 2010*, pp. 1–8, IEEE Computer Society, 2010.

[35] N. Cesa-Bianchi and G. Valentini, "Hierarchical cost-sensitive algorithms for genome-wide gene function prediction," *Journal of Machine Learning Research, W&C Proceedings, Machine Learning in Systems Biology*, vol. 8, pp. 14–29, 2010.

[36] F. K. Nakano, S. M. Mastelini, S. Barbon, and R. Cerri, "Stacking methods for hierarchical classification," in *16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, Cancun, Mexico, December 18-21, 2017*, pp. 289–296, 2017.

[37] G. Armano, "Modelling Progressive Filtering," *Fundamenta Informaticae*, vol. 138, no. 3, pp. 385–320, 2015.

[38] C. Silla and A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, pp. 31–72, 2011.

[39] D. Kocev, C. Vens, J. Struyf, and S. Dzeroski, "Tree ensembles for predicting structured outputs," *Pattern Recognition*, vol. 46, no. 3, pp. 817–833, 2013.

[40] L. Cheng, H. Lin, Y. Hu, J. Wang, and Z. Yang, "Gene function prediction based on the gene ontology hierarchical structure," *PLOS ONE*, vol. 9, pp. 1–7, 09 2014.

[41] S. Feng, P. Fu, and W. Zheng, "A hierarchical multi-label classification algorithm for gene function prediction," *Algorithms*, vol. 10, no. 4, p. 138, 2017.

[42] Y. Zhao, G. Fu, J. Wang, M. Guo, and G. Yu, "Gene function prediction based on gene ontology hierarchy preserving hashing," *Genomics*, 2018.

[43] X. Jiang, N. Nariai, M. Steffen, S. Kasif, and E. Kolaczyk, "Integration of relational and hierarchical network information for protein function prediction," *BMC Bioinformatics*, vol. 9, no. 350, 2008.

[44] G. Valentini, "Hierarchical ensemble methods for protein function prediction," *ISRN Bioinformatics*, vol. 2014, p. 34, 2014.

[45] R. Cerri, R. C. Barros, A. C. P. L. F. de Carvalho, and Y. Jin, "Reduction strategies for hierarchical multi-label classification in protein function prediction," *BMC Bioinformatics*, vol. 17, p. 373, 2016.

[46] A. Mohammed and C. Guda, "Application of a hierarchical enzyme classification method reveals the role of gut microbiome in human metabolism," *BMC Genomics*, vol. 16, p. S16, Jun 2015.

[47] C. Cortes, V. Kuznetsov, and M. Mohri, "Ensemble methods for structured prediction," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1134–1142, 2014.

[48] S. Kang, Y. Ko, and J. Seo, "Hierarchical speech-act classification for discourse analysis," *Pattern Recognition Letters*, vol. 34, pp. 1119–1124, 2013.

[49] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," *CoRR*, vol. abs/1709.08267, 2017.

[50] N. Pappas and A. Popescu-Belis, "Multilingual hierarchical attention networks for document classification," *CoRR*, vol. abs/1707.00896, 2017.

[51] L. Zhang, S. K. Shah, and I. A. Kakadiaris, "Hierarchical multi-label classification using fully associative ensemble learning," *Pattern Recognition*, vol. 70, pp. 89–103, 2017.

[52] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Dzeroski, "Hierarchical annotation of medical images," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2436–2449, 2011.

[53] A. Ghosal, R. Chakraborty, B. C. Dhara, and S. K. Saha, "A hierarchical approach for speech-instrumental-song classification," *SpringerPlus*, vol. 2, p. 526, Oct 2013.

[54] A. Tsaptsinos, "Lyrics-based music genre classification using a hierarchical attention network," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pp. 694–701, 2017.

[55] E. Mendi, "Hierarchical representation of video sequences for annotation," *Computers & Electrical Engineering*, vol. 40, no. 8, pp. 247–256, 2014.

[56] G. S. Chambers, S. Venkatesh, G. A. W. West, and H. H. Bui, "Hierarchical recognition of intentional human gestures for sports video annotation," in *16th International Conference on Pattern Recognition, ICPR 2002, Quebec, Canada, August 11-15, 2002.*, pp. 1082–1085, 2002.

[57] I. Kahanda and A. Ben-Hur, "Gostruct 2.0: Automated protein function prediction for annotated proteins," in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB 2017, Boston, MA, USA, August 20-23, 2017*, pp. 60–66, 2017.

[58] D. J. Lipman and W. R. Pearson, "Rapid and sensitive protein similarity searches.," *Science (New York, N.Y.)*, vol. 227, pp. 1435–41, Mar 1985.

[59] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, "Basic local alignment search tool.," *Journal of Molecular Biology*, vol. 215, 1990.

[60] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[61] Y. Loewenstein, D. Raimondo, O. C. Redfern, J. Watson, D. Frishman, M. Linial, C. Orengo, J. Thornton, and A. Tramontano, "Protein function annotation by homology-based inference.," *Genome biology*, vol. 10, no. 2, p. 207, 2009.

[62] M. Falda, S. Toppo, A. Pescarolo, E. Lavezzo, D. B., A. Facchinetti, E. Cilia, R. Velasco, and P. Fontana, "Argot2: a large scale function prediction tool relying on semantic similarity of weighted Gene Ontology terms," *BMC Bioinformatics*, vol. 13, no. Suppl 4:S14, 2012.

[63] P. Törönen, A. Medlar, and L. Holm, "Pannzer2: a rapid functional annotation web server," *Nucleic Acids Research*, vol. 46, no. W1, pp. W84–W88, 2018.

[64] T. Hamp *et al.*, "Homology-based inference sets the bar high for protein function prediction," *BMC Bioinformatics*, vol. 14, no. Suppl 3:S7, 2013.

[65] A. Conesa, S. Gotz, J. Garcia-Gomez, J. Terol, T. M., and M. Robles, "Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research," *Bioinformatics*, vol. 21, no. 18, pp. 3674–3676, 2005.

[66] E. Marcotte, M. Pellegrini, M. Thompson, T. Yeates, and D. Eisenberg, "A combined algorithm for genome-wide prediction of protein function," *Nature*, vol. 402, pp. 83–86, 1999.

[67] B. Schwikowski, P. Uetz, and S. Fields, "A network of protein-protein interactions in yeast.," *Nature biotechnology*, vol. 18, pp. 1257–61, Dec 2000.

[68] P. Bogdanov and A. K. Singh, "Molecular function prediction using neighborhood features," *IEEE/ACM Trans. Comput. Biology Bioinform.*, vol. 7, no. 2, pp. 208–217, 2010.

[69] Y. Li and J. C. Patra, "Integration of multiple data sources to prioritize candidate genes using discounted rating system," *BMC Bioinformatics*, vol. 11, p. S20, Jan 2010.

[70] J. Wang, R. Du, R. Payattakil, P. Yu, and C. Chen, "Improving GO semantic similarity measures by exploring the ontology beneath the terms and modelling uncertainty," *Bioinformatics*, vol. 23, no. 10, pp. 1274–1281, 2007.

[71] H. Yang, T. Nepusz, and A. Paccanaro, "Improving GO semantic similarity measures by exploring the ontology beneath the terms and modelling uncertainty," *Bioinformatics*, vol. 28, no. 10, pp. 1383–1389, 2012.

[72] H. Yu, L. Gao, K. Tu, and Z. Guo, "Broadly predicting specific gene functions with expression similarity and taxonomy similarity," *Gene*, vol. 352, pp. 75–81, 2005.

[73] Y. Tao, L. Sam, J. Li, C. Friedman, and Y. Lussier, "Information theory applied to the sparse Gene Ontology annotation network to predict novel gene function," *Bioinformatics*, vol. 23, no. 13, pp. i529–i538, 2007.

[74] G. Pandev, C. Myers, and V. Kumar, "Incorporating functional inter-relationships into protein function prediction algorithms," *BMC Bioinformatics*, vol. 10, no. 1-142, 2009.

[75] X. Zhang and D. Dai, "A framework for incorporating functional interrelationships into protein function prediction algorithms," *IEEE ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 3, pp. 740–753, 2012.

[76] Z. Hui and H. Trevor, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2007.

[77] X. Zhu *et al.*, "Semi-supervised learning with gaussian fields and harmonic functions," in *Proc. of the 20th Int. Conf. on Machine Learning*, (Washintgton DC, USA), 2003.

[78] M. Szummer and T. S. Jaakkola, "Partially labeled classification with markov random walks," in *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pp. 945–952, 2001.

[79] A. Azran, "The rendezvous algorithm: multiclass semi-supervised learning with markov random walks," in *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pp. 49–56, 2007.

[80] M. Frasca, S. Bassis, and G. Valentini, "Learning node labels with multi-category hopfield networks," *Neural Computing and Applications*, vol. 27, no. 6, pp. 1677–1692, 2016.

[81] A. Bertoni, M. Frasca, and G. Valentini, "Cosnet: A cost sensitive neural network for semi-supervised learning in graphs," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011. Proceedings, Part I*, pp. 219–234, 2011.

[82] E. Nabieva, K. Jim, A. Agarwal, B. Chazelle, and M. Singh, "Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps," in *Proceedings Thirteenth International Conference on Intelligent Systems for Molecular Biology 2005, Detroit, MI, USA, 25-29 June 2005*, pp. 302–310, 2005.

[83] G. Valentini, G. Armano, M. Frasca, J. Lin, M. Mesiti, and M. Re, "RANKS: a flexible tool for node label ranking and classification in biological networks," *Bioinformatics*, vol. 32, p. 2872, 2016.

[84] M. Deng, T. Chen, and F. Sun, "An integrated probabilistic model for functional prediction of proteins," *Journal of Computational Biology*, vol. 11, no. 2/3, pp. 463–475, 2004.

[85] K. Tsuda, H. Shin, and B. Schölkopf, "Fast protein classification with multiple networks," in *ECCB/JBI'05 Proceedings, Fourth European Conference on Computational Biology/Sixth Meeting of the Spanish Bioinformatics Network (Jornadas de BioInformática), Palacio de Congresos, Madrid, Spain, September 28 - October 1, 2005*, p. 65, 2005.

[86] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, and Q. Morris, "Genemania: a real-time multiple association network integration algorithm for predicting gene function," *Genome Biology*, vol. 9, p. S4, Jun 2008.

[87] Y. Bengio, O. Delalleau, and N. Le Roux, "Label Propagation and Quadratic Criterion," in *Semi-Supervised Learning* (O. Chapelle, B. Scholkopf, and A. Zien, eds.), pp. 193–216, MIT Press, 2006.

[88] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Boston, MA: PWS Publishing Company, 1996.

[89] N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella, "Random spanning trees and the prediction of weighted graphs," in *Proceedings of the 27th International Conference on Machine Learning*, (Haifa, Israel), 2010.

[90] K. Astikainen, L. Holm, E. Pitkanen, S. Szedmak, and J. Rousu, "Towards structured output prediction of enzyme function," *BMC Proceedings*, vol. 2, p. S2, 2008.

[91] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification models," *Journal of Machine Learning Research*, vol. 7, pp. 1601–1626, 2006.

[92] G. Valentini and F. Masulli, "Ensembles of learning machines," in *Neural Nets WIRN-02*, vol. 2486 of *Lecture Notes in Computer Science*, pp. 3–19, Springer, 2002.

[93] T. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy* (J. Kittler and F. Roli, eds.), vol. 1857 of *Lecture Notes in Computer Science*, pp. 1–15, Springer-Verlag, 2000.

[94] O. Okun, G. Valentini, and M. Re, *Ensembles in Machine Learning Applications*, vol. 373 of *Studies in Computational Intelligence*. Berlin: Springer, 2011.

[95] M. Re and G. Valentini, "Ensemble methods: a review," in *Advances in Machine Learning and Data Mining for Astronomy*, Data Mining and Knowledge Discovery, pp. 563–594, Chapman & Hall, 2012.

[96] A. Jurek, Y. Bi, S. Wu, and C. D. Nugent, "A survey of commonly used ensemble-based classification techniques," *Knowledge Eng. Review*, vol. 29, no. 5, pp. 551–581, 2014.

[97] A. M. Santos and A. M. P. Canuto, "Applying semi-supervised learning in hierarchical multi-label classification," *Expert Syst. Appl.*, vol. 41, no. 14, pp. 6075–6085, 2014.

[98] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, vol. 73, p. 185, Aug 2008.

[99] M. Ramírez-Corona, L. E. Sucar, and E. F. Morales, "Multi-label classification for tree and directed acyclic graphs hierarchies," in *Probabilistic Graphical Models - 7th European Workshop, PGM 2014, Utrecht, The Netherlands, September 17-19, 2014. Proceedings*, pp. 409–425, 2014.

[100] G. Valentini, "True path rule hierarchical ensembles for genome-wide gene function prediction," *IEEE/ACM Trans. Comput. Biology Bioinform.*, vol. 8, no. 3, pp. 832–847, 2011.

[101] "Kiritchenko s, matwin s, famili af. hierarchical text categorization as a tool of associating genes with gene ontology codes. in: European workshop on data mining and text mining in bioinformatics: 2004. p. 30–4."

[102] R. Cerri, R. C. Barros, and A. C. P. L. F. de Carvalho, "Hierarchical multi-label classification using local neural networks," *J. Comput. Syst. Sci.*, vol. 80, no. 1, pp. 39–56, 2014.

[103] N. Alaydie, C. K. Reddy, and F. Fotouhi, "Exploiting label dependency for hierarchical multi-label classification," in *Advances in Knowledge Discovery and Data Mining - 16th Pacific-Asia Conference, PAKDD 2012, Kuala Lumpur, Malaysia, May 29-June 1, 2012, Proceedings, Part I*, pp. 294–305, 2012.

[104] I. Triguero and C. Vens, "Labelling strategies for hierarchical multi-label classification techniques," *Pattern Recognition*, vol. 56, pp. 170–183, 2016.

[105] W. Bi and J. T. Kwok, "Multilabel classification on tree- and dag-structured hierarchies.," in *ICML* (L. Getoor and T. Scheffer, eds.), pp. 17–24, Omnipress, 2011.

[106] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik, "Kernel dependency estimation," in *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pp. 873–880, 2002.

[107] R. G. Baraniuk and D. L. Jones, "A signal-dependent time-frequency representation: fast algorithm for optimal kernel design," *IEEE Trans. Signal Processing*, vol. 42, no. 1, pp. 134–146, 1994.

[108] R. Cerri, R. C. Barros, and A. C. P. L. F. de Carvalho, "Hierarchical multi-label classification for protein function prediction: A local approach based on neural networks," in *11th International Conference on Intelligent Systems Design and Applications, ISDA 2011, Córdoba, Spain, November 22-24, 2011*, pp. 337–343, 2011.

[109] P. Vateekul, M. Kubat, and K. Sarinnapakorn, "Hierarchical multi-label classification with svms: A case study in gene function prediction," *Intell. Data Anal.*, vol. 18, no. 4, pp. 717–738, 2014.

[110] P. Vateekul, S. Dendamrongvit, and M. Kubat, "Improving SVM performance in multi-label domains: Threshold adjustment," *International Journal on Artificial Intelligence Tools*, vol. 22, no. 1, 2013.

[111] Z. Sun, Y. Zhao, D. Cao, and H. Hao, "Hierarchical multilabel classification with optimal path prediction," *Neural Processing Letters*, vol. 45, no. 1, pp. 263–277, 2017.

[112] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[113] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, 2004.

[114] R. Caruana, A. Munson, and A. Niculescu-Mizil, "Getting the most out of ensemble selection," in *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pp. 828–833, 2006.

[115] L. Wang, J. Law, S. Kale, T. Murali, and G. Pandey, "Large-scale protein function prediction using heterogeneous ensembles," *F1000Research*, vol. 7, no. 1577, 2018.

[116] J. Wehrmann, R. Cerri, and R. Barros, "Hierarchical multi-label classification networks," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 5225–5234, PMLR, 10–15 Jul 2018.

[117] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[118] G. Valentini, S. Köhler, M. Re, M. Notaro, and P. N. Robinson, "Prediction of human gene - phenotype associations by exploiting the hierarchical structure of the human phenotype ontology," in *Bioinformatics and Biomedical Engineering. IWBBIO 2015* (F. Ortuño and I. Rojas, eds.), vol. 9043 of *Lecture Notes in Computer Science*, pp. 66–77, Cham: Springer International Publishing, 2015.

[119] O. Sysoev, A. Grimvall, and O. Burdakov, "Data preordering in generalized pav algorithm for monotonic regression," *Journal of Computational Mathematics*, vol. 24, no. 6, pp. 771–790, 2006.

[120] P. N. Robinson, M. Frasca, S. Köhler, M. Notaro, M. Re, and G. Valentini, "A hierarchical ensemble method for dag-structured taxonomies," in *Multiple Classifier Systems. MCS 2015* (F. Schwenker, F. Roli, and J. Kittler, eds.), Lecture Notes in Computer Science, pp. 15–26, Cham: Springer International Publishing, 2015.

[121] M. Notaro, M. Schubach, P. N. Robinson, and G. Valentini, "Ensembling descendant term classifiers to improve gene - abnormal phenotype predictions," in *CIBB 2017: the 14th International Conference on Bioinformatics and Biostatistics, 7-9 September Cagliari, Italy, (Proceedings)*, 2018.

[122] G. Valentini, "Notes on hierarchical ensemble methods for dag-structured taxonomies," *CoRR*, vol. abs/1406.4472, 2014.

[123] T. Cormen, C. Leiserson, R. Rivest, and S. RL, *Introduction to Algorithms*. Boston: MIT Press, 2009.

[124] R. Barlow and H. Brunk, "The isotonic regression problem and its dual," *Journal of the American Statistical Association*, vol. 67, pp. 140–147, 1972.

[125] M. Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and E. Silverman, "An empirical distribution function for sampling with incomplete information," *Ann. Math. Statist.*, vol. 26, pp. 641–647, 12 1955.

[126] R. Barlow, *Statistical Inference Under Order Restrictions: The Theory and Application of Isotonic Regression*. Out-of-print Books on demand, J. Wiley, 1972.

[127] S. J. Grotzinger and C. Witzgall, "Projections onto order simplexes," *Applied Mathematics and Optimization*, vol. 12, pp. 247–270, Oct 1984.

[128] J. Nocedal and S. J. Wright, *Numerical optimization*. Springerverlang, USA, 1999.

[129] M. J. Best and N. Chakravarti, "Active set algorithms for isotonic regression; a unifying framework," *Mathematical Programming*, vol. 47, pp. 425–439, May 1990.

[130] H. Mukarjee and S. Stern, "Feasible nonparametric estimation of multiargument monotone functions," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 77–80, 1994.

[131] W. L. Maxwell and J. A. Muckstadt, "Establishing consistent and realistic reorder intervals in production-distribution systems," *Operations Research*, vol. 33, no. 6, pp. 1316–1341, 1985.

[132] R. Roundy, "A 98%-effective lot-sizing rule for a multi-product, multi-stage production / inventory system," *Mathematics of Operations Research*, vol. 11, no. 4, pp. 699–727, 1986.

[133] O. Burdakov, O. Sysoev, A. Grimvall, and M. Hussian, "An $\mathcal{O}(n^2)$ algorithm for isotonic regression," in *Large-Scale Nonlinear Optimization*, no. 83 in Nonconvex Optimization and Its Applications, pp. 25–33, Springer-Verlag, 2006.

[134] M. Michaelides, Z. Li, N. A. Rana, E. C. Richardson, P. G. Hykin, A. T. Moore, G. E. Holder, and A. R. Webster, "Novel mutations and electrophysiologic findings in rgs9- and r9ap-associated retinal dysfunction (bradyopsia)," *Ophthalmology*, vol. 117, p. 120—127.e1, 2010.

[135] M. Rajpar, K. Harley, C. Laing, R. Davies, and M. Dixon, "Mutation of the gene encoding the enamel-specific protein, enamelin, causes autosomal-dominant amelogenesis imperfecta," *Human molecular genetics*, vol. 10, p. 1673—1677, 2001.

[136] A. Lopez-Lera, J. Torres-Canizales, S. Garrido, A. Morales, and M. Lopez-Trascasa, "Thomson syndrome and glomerulonephritis in a homozygous C1q-deficient patient due to a Gly164Ser C1qC mutation," *Journal of Investigative Dermatology*, vol. 134, pp. 1152–1154, 2014.

[137] S. Dasgupta, C. Papadimitriou, and U. Vazirani, *Algorithms*. Boston: McGraw Hill, 2008.

[138] M. Frasca, "Automated gene function prediction through gene multifunctionality in biological networks," *Neurocomputing*, vol. 162, pp. 48–56, 2015.

[139] C. von Mering, L. J. Jensen, B. Snel, S. D. Hooper, M. Krupp, M. Foglierini, N. Jouffre, M. A. Huynen, and P. Bork, "String: known and predicted protein-protein associations, integrated and transferred across organisms.," *Nucleic acids research*, vol. 33, pp. D433–7, Jan 2005.

[140] D. Szklarczyk, A. Franceschini, S. Wyder, K. Forslund, D. Heller, J. Huerta-Cepas, M. Simonovic, A. Roth, A. Santos, K. Tsafou, M. Kuhn, P. Bork, L. J. Jensen, and C. von Mering, "STRING v10: protein-protein interaction networks, integrated over the tree of life.," *Nucleic Acids Research*, vol. 43, pp. 447–452, 2015.

[141] D. Szklarczyk, J. H. Morris, H. Cook, M. Kuhn, S. Wyder, M. Simonovic, A. Santos, N. T. Doncheva, A. Roth, P. Bork, L. J. Jensen, and C. von Mering, "The STRING database in 2017: quality-controlled protein-protein association networks, made broadly accessible," *Nucleic Acids Research*, vol. 45, no. Database-Issue, pp. D362–D368, 2017.

[142] D. Barrell, E. Dimmer, R. P. Huntley, D. Binns, C. O'Donovan, and R. Apweiler, "The goa database in 2009–an integrated gene ontology annotation resource.," *Nucleic acids research*, vol. 37, pp. D396–403, Jan 2009.

[143] R. Apweiler, A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi, and L.-S. L. Yeh, "Uniprot: the universal protein knowledgebase.," *Nucleic acids research*, vol. 32, pp. D115–9, Jan 2004.

[144] G. Fu, J. Wang, B. Yang, and G. Yu, "Neggoa: negative go annotations selection using ontology structure," *Bioinformatics*, vol. 32, no. 19, pp. 2996–3004, 2016.

[145] M. Frasca and D. Malchiodi, "Exploiting negative sample selection for prioritizing candidate disease genes," *Genomics and Computational Biology*, vol. 3, no. 3, p. e47, 2017.

[146] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PLOS ONE*, vol. 10, pp. 1–21, 03 2015.

[147] M. Kuhn, "Building predictive models in r using the caret package," *Journal of Statistical Software*, vol. 28, no. 5, pp. 1–26, 2008.

[148] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[149] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, vol. 33, no. 1, 2010.

[150] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*. New York: Springer, fourth ed., 2002.

[151] M. Dettling and P. Bühlmann, "Boosting for tumor classification with gene expression data.," *Bioinformatics (Oxford, England)*, vol. 19, pp. 1061–9, Jun 2003.

[152] D. Rumelhart, G. Hintont, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[153] H. Zhang, "The optimality of naïve bayes," in *In FLAIRS2004 conference*, 2004.

[154] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.

[155] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[156] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, Aug 1996.

[157] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016.

[158] C. Ambroise and G. J. McLachlan, "Selection bias in gene extraction on the basis of microarray gene-expression data.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, pp. 6562–6, May 2002.

[159] P. Robinson, "Deep phenotyping for precision medicine.," *Human Mutations*, vol. 33, pp. 777–780, 2012.

[160] S. Köhler, N. A. Vasilevsky, M. Engelstad, Foster, *et al.*, "The human phenotype ontology in 2017," *Nucleic Acids Research*, vol. 45, p. D865, 2017.

[161] A. Hamosh, A. F. Scott, J. S. Amberger, C. A. Bocchini, D. Valle, and V. A. McKusick, "Online mendelian inheritance in man (omim), a knowledgebase of human genes and genetic disorders.," *Nucleic Acids Research*, vol. 30, pp. 52–55, 2002.

[162] S. Aymé and J. Schmidtke, "Networking for rare diseases: a necessity for europe.," *Cin. Genet.*, vol. 50, pp. 1477 – 1483, 2007.

[163] E. Bragin, E. A. Chatzimichali, C. F. Wright, M. E. Hurles, H. V. Firth, A. P. Bevan, and G. J. Swaminathan, "DECIPHER: database for the interpretation of phenotype-linked plausibly pathogenic sequence and copy-number variation.," *Nucleic Acids Research*, vol. 42, pp. 993–1000, 2014.

[164] Y. Moreau and L. Tranchevent, "Computational tools for prioritizing candidate genes: boosting disease gene discovery.," *Nature Rev. Genet.*, vol. 13, pp. 523–536, 2012.

[165] T. Zemojtel, S. Köhler, L. Mackenroth, M. Jäger, J. Hecht, P. Krawitz, L. Graul-Neumann, S. Doelken, N. Ehmke, M. Spielmann, N. C. Oien, M. R. Schweiger, U. Krüger, G. Frommer, B. Fischer, U. Kornak, R. Flöttmann, A. Ardeshirdavani, Y. Moreau, S. E. Lewis, M. Haendel, D. Smedley, D. Horn, S. Mundlos, and P. N. Robinson, "Effective diagnosis of genetic disease by computational phenotype analysis of the disease-associated genome.," *Sci Transl Med*, vol. 6, p. 252ra123, 2014.

[166] D. Smedley, M. Schubach, J. O. Jacobsen, S. Köhler, T. Zemojtel, M. Spielmann, M. Jäger, H. Hochheiser, N. L. Washington, J. A. McMurry, M. A. Haendel, C. J. Mungall, S. E. Lewis, T. Groza, G. Valentini, and P. N. Robinson, "A whole-genome analysis framework for effective identification of pathogenic regulatory variants in mendelian disease.," *The American Journal of Human Genetics*, vol. 99, pp. 595–606, 2016.

[167] J. Chong *et al.*, "The genetic basis of mendelian phenotypes: Discoveries, challenges, and opportunities.," *Am J Hum Genet*, vol. 97, pp. 199–215, 2015.

[168] P. Wang *et al.*, "Inference of gene-phenotype associations via protein-protein interaction and orthology.," *PLoS ONE*, vol. 8, pp. 1–8, 2013.

[169] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers* (A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, eds.), pp. 61–74, Cambridge, MA: MIT Press, 1999.

[170] M. Mayer and P. Hieter, "Protein networks - guilt by association," *Nature Biotechnology*, vol. 18, no. 12, pp. 1242–1243, 2000.

[171] H. Kashima, K. Tsuda, and A. Inokuchi, *Kernels for Graphs*, pp. 150–170. MIT Press, 2004.

[172] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines* (B. Schölkopf and M. K. Warmuth, eds.), Lecture Notes in Computer Science, pp. 144–158, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[173] G. Valentini, A. Paccanaro, H. Caniza, A. Romero, and M. Re, "An extensive analysis of disease-gene associations using network integration and fast kernel-based gene prioritization methods," *Artificial Intelligence in Medicine*, vol. 61, pp. 63–78, 2014.

[174] M. Re, M. Mesiti, and G. Valentini, "A fast ranking algorithm for predicting gene functions in biomolecular networks," *IEEE ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, pp. 1812–1818, 2012.

[175] M. Re and G. Valentini, "Network-based Drug Ranking and Repositioning with respect to DrugBank Therapeutic Categories," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, pp. 1359–1371, 2013.

[176] B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. P. Speed, "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias," *Bioinformatics*, vol. 19, pp. 185–193, 2003.

[177] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification.," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[178] G. Fu, J. Wang, B. Yang, and G. Yu, "Neggoa: negative go annotations selection using ontology structure," *Bioinformatics*, vol. 32, no. 19, pp. 2996–3004, 2016.

[179] A. Franceschini, D. Szklarczyk, S. Frankild, M. Kuhn, M. Simonovic, A. Roth, J. Lin, P. Minguez, P. Bork, C. von Mering, and L. J. Jensen, "STRING v9.1: protein-protein interaction networks, with increased coverage and integration," *Nucleic Acids Research*, vol. 41, pp. 808–815, 2013.

[180] A. Chatr-Aryamontri, B.-J. Breitkreutz, S. Heinicke, L. Boucher, A. G. Winter, C. Stark, J. Nixon, L. Ramage, N. Kolas, L. O'Donnell, T. Reguly, A. Breitkreutz, A. Sellam, D. Chen, C. Chang, J. M. Rust, M. S. Livstone, R. Oughtred, K. Dolinski, and M. Tyers, "The BioGRID interaction database: 2013 update.," *Nucleic Acids Research*, vol. 41, pp. 816–823, 2013.

[181] J. Amberger, C. Bocchini, and A. Amosh, "A new face and new challenges for online mendelian inheritance in man (OMIM)," *Hum. Mutat.*, vol. 32, pp. 564–7, 2011.

[182] H. Nielsen, A. How-Kit, C. Guerin, F. Castinetti, H. Vollan, *et al.*, "Copy number variations alter methylation and parallel IGF2 overexpression in adrenal tumors," *Endocrine-Related Cancer*, vol. 22, pp. 953–967, 2015.

[183] S. Chika, Y. Seijin, S. Masayuki, M. Yusaku, M. Yuichi, and G. Yu-ichi, "ECHS1 mutations cause combined respiratory chain deficiency resulting in leigh syndrome," *Human Mutation*, vol. 36, pp. 232–239, 2015.

[184] T. Grossman, L. Hettrick, R. Johnson, G. Hung, R. Peralta, A. Watt, S. Henry, P. Adamson, B. Monia, and M. McCaleb, "Inhibition of the alternative complement pathway by antisense oligonucleotides targeting complement factor b improves lupus nephritis in mice," *Immunobiology*, vol. 221, pp. 701–708, 2016.

[185] C. P. Hersh, N. N. Hansel, K. C. Barnes, D. A. Lomas, S. G. Pillai, H. O. Coxson, R. A. Mathias, N. M. Rafaels, R. A. Wise, J. E. Connett, B. J. Klanderman, F. L. Jacobson, R. Gill, A. A. Litonjua, D. Sparrow, J. J. Reilly, E. K. Silverman, , and the ICGN Investigators, "Transforming growth factor-β receptor-3 is associated with pulmonary emphysema," *American Journal of Respiratory Cell and Molecular Biology*, vol. 41, pp. 324–331, 2009.

[186] W. Fua, J. Zhuc, S.-W. Xiongd, W. Jiaa, Z. Zhaoa, S.-B. Zhua, J.-H. Hua, F.-H. Wanga, H. Xiaa, J. Hea, and G.-C. Liua, "BARD1 gene polymorphisms confer nephroblastoma susceptibility," *American Journal of Respiratory Cell and Molecular Biology*, vol. 16, pp. 101–105, 2017.

[187] X. Yang, J. Wu, J. Lu, G. Liu, G. Di, C. Chen, Y. Hou, M. Sun, W. Yang, X. Xu, Y. Zhao, X. Hu, D. Li, Z. Cao, X. Zhou, X. Huang, Z. Liu, H. Chen, Y. Gu, Y. Chi, X. Yan, Q. Han, Z. Shen, Z. Shao, and Z. Hu, "Identification of a comprehensive spectrum of genetic factors for hereditary breast cancer in a chinese population by next-generation sequencing," *PLOS ONE*, vol. 10, pp. 1–20, 04 2015.

[188] G. Bobby, L. A. Wolfe, M. Ichikawa, T. Markello, M. He, C. J. Tifft, W. A. Gahl, and H. H. Freeze, "Biallelic mutations in CAD, impair de novo pyrimidine biosynthesis and decrease glycosylation precursors," *Human Molecular Genetics*, vol. 24, pp. 3050–3057, 2015.

[189] R. Pitceathly, J. Taanman, S. Rahman, and et al, "COX10 mutations resulting in complex multisystem mitochondrial disease that remains stable into adulthood," *JAMA Neurology*, vol. 70, pp. 1556–1561, 2013.

[190] C. Widmer and G. Ratsch, "Multitask Learning in Computational Biology," *Journal of Machine Learning Research, W&P*, vol. 27, pp. 207–216, 2012.

[191] T. Doğan, "HPO2GO: prediction of human phenotype ontology term associations using cross ontology annotation co-occurrences," *PeerJ Preprints 6:e26663v2*, 2018.