A Novel Extended Hierarchical Dependence Network Method Based on non-Hierarchical Predictive Classes and Applications to Ageing-Related Data

Fabio Fabris and Alex A. Freitas
School of Computing, University of Kent, Kent, United Kingdom
{ff79,A.A.Freitas}@kent.ac.uk

***Abstract*—We propose a novel algorithm for hierarchical classification, the Hierarchical Dependence Network based on non-Hierarchical Predictive Classes (HDN-nHPC) algorithm. HDN-nHPC uses relationships among predictive classes that are not descendants or ancestors of each other to improve classification performance and, at the same time, provide insights to non-obvious predictive class relationships. To test our algorithm and baselines, we have used hierarchical ageing-related datasets where the classes are terms in the Gene Ontology. We have concluded, based on our experiments, that using non-hierarchical predictive class relationships improves the performance of the classification algorithm and that, considering one out of three accuracy measures, the HDN-nHPC is statistically significantly better than the other three algorithms that we have tested, while no statistical significant differences were found on the other two measures.**

I. Introduction

This work addresses the hierarchical classification task, where the classes are organized into a hierarchy. The goal is to build a classification model capable of assigning classes to new instances (with unknown classes) given a dataset containing instances with known classes.

This work's contribution is the proposal of a new hierarchical classification algorithm based on Hierarchical Dependence Networks (HDN). Our algorithm uses predictive class relationships among hierarchically unrelated classes (i.e., classes which are not ancestors or descendants of each other) to improve the performance of the hierarchical classifier.

We have explored HDNs in previous work [6], however, the current paper differs from [6] in significant ways, specifically: 1) the use of a data-driven approach to find predictive class relationships instead of relying solely on the pre-established structure of the class hierarchy; 2) the use of a dataset split approach to train the base classifiers, instead of extending instances with the classification of other classes; 3) potentially training several base classifiers to predict one class label given the prediction of other class labels. We show that these modifications improve the predictive performance and the interpretation potential of our approach.

Recent work [12] has shown that using hierarchical class relationships improves the performance of single hierarchical classifiers, while being unnecessary for classifier ensembles. In this work we go one step further: we show that using non-hierarchical relationships for single hierarchical classifiers improves the predictive performance compared to not using such non-hierarchical relationships.

In this work, we evaluate the proposed hierarchical classification algorithm, and three other algorithms, in 10 ageing related datasets. Our motivation to focus on ageing, as the target application domain, is as follows.

Ageing-related diseases are affecting an increasing portion of the population [4], and, consequently, the pressure to better understand the complex ageing process is increasing in several areas of study. Although our understanding about ageing has improved dramatically in the last decades, we are still far from the ultimate goal of slowing down or even reversing the ageing process in humans. Besides the increase in population age, another factor driving ageing research is the greater availability of datasets containing gene or protein sequence information (e.g. Universal Protein Resource [16]), as well as curated hierarchical ontologies used to annotated these genes and proteins (e.g. Gene Ontology (GO) [15]).

Studies have found several ageing-related genes in model organisms such as the mouse, the fruit fly, the worm, and the yeast. These genes, when turned *on* or *off*, considerably affect the lifespan of the organisms [4]. This suggests that the ageing process may be, to some extent, controllable through some intervention.

The development of hierarchical classification algorithms like the one proposed in this work has a greater potential to generate useful classification results for biologists than the more common approach of performing *flat* classification on the data, that is, using classical classification algorithms and ignoring the class hierarchy. Also, the predictive class relationships found by our algorithm may help biologists understanding biological aspects of ageing-related GO terms.

The remainder of this paper is organized as follows: Section II presents background and related work. Section III describes our proposed algorithm. Section IV explains the creation of the ageing datasets. Section V exemplifies the types of predictive class relationships found by our algorithm and reports the predictive accuracy results of the algorithms that we have tested. Section VI concludes our work.

II. Background and Related Work

Hierarchical classification problems are common in bioinformatics since, usually, genes' functional classes are specified by a hierarchical scheme like the GO [15]. In this scenario the set of class labels is organized into a hierarchy,

usually a tree or a DAG (Directed Acyclic Graph), where each node represents a class label and the edges represent generalization-specialization relationships among classes.

Hierarchical classification algorithms may be divided into two broad types [21]: global or local. Local Hierarchical Classification (LHC) algorithms build a set of local classification models (base classifiers) by training a traditional (flat) classification algorithm for each (typically small) part of the class hierarchy in the training phase. Global hierarchical classification algorithms build a single global classification model predicting classes in the full class hierarchy.

Broadly speaking, local hierarchical classification algorithms have the advantage of algorithmic simplicity, since they transform the original hierarchical classification problem into a set of simpler flat classification problems in the training phase. Conversely, global hierarchical classification algorithms have the advantage of producing a single coherent global classification model, which tends to be more easily interpreted than the large number of different classification models produced by the local approach.

A. The Predictive Clustering Tree (PCT) algorithm

PCT (Predictive Clustering Tree) is a type of global hierarchical classification algorithms that builds a single decision tree by recursively finding a value for a predictive feature that splits the current set of instances in two clusters, maximizing the similarity of classes within each cluster and the dissimilarity of the classes across the two clusters. The algorithm recurses in each cluster that it forms and eventually stops if the split does not have a good quality (based on some quality measure) or the size of a cluster falls below a pre-established threshold. In the prediction phase, to classify an instance $\mathbf{x}$, a PCT algorithm first identifies the cluster associated with that instance and then assigns, to instance $\mathbf{x}$, classes whose probabilities in the class probability vector of that cluster are greater than a probability threshold. The threshold is varied when computing a Precision-Recall curve, as explained later.

The most well-known version of the PCT algorithm is the Clus-HMC algorithm [24]. There is also an ensemble version of Clus-HMC, called Clus-HMC-Ens [20]. In our experiments we do not use this ensemble version due to the interpretation difficulty that is inherent on the use of ensemble of classifiers [7].

The PCT algorithm has only one parameter to tune: the s-value that dictates how statistically significantly different two groups of instances in a tree node's split must be in order for the split to be accepted by a F-test. Larger s-values correspond to a more permissive test, and thus a larger decision tree. To tune this parameter we have applied an internal 10-fold cross-validation procedure (using only the training set) in each iteration of the main (external) cross-validation run. We have tested the default set of parameters suggested for the *Clus System* in [24]: $\{0.001, 0.005, 0.010, 0.050, 0.100, 0.125\}$, and chose the one with highest predictive accuracy.

B. The Hierarchical Dependence Network (HDN) algorithm

Dependence Networks (DNs) are a relatively under-explored type of probabilistic graphical model first described in [11]. Each node of a DN represents a random variable and encodes a probability distribution conditioned on the values of its parents, like in Bayesian Networks (BNs). However, DNs are more flexible than BNs since they allow for cycles in their graphical representation. In addition, in a DN the edges coming out of a node N_i connect N_i to the Markov blanket of N_i (N_i^{MB}), i.e. the minimal set of nodes that make N_i independent from all other nodes. In a conventional "flat", single-label, classification problem, the Markov blanket of a class corresponds to the set of predictive features that influences the value of the class variable.

Recently, DN classification algorithms were proposed for multi-label classification, where an instance can be assigned to multiple class labels [9, 13]. This type of problem still involves flat classification, since there is no hierarchy among class labels. In this paper we address the more difficult problem of *hiearchical* multi-label classification.

C. Methods for Exploring Flat Multi-label Dependencies

Classifier Chains [18] (CC) is a method that decomposes non-hierarchical multi-label classification problems into several single-label classification problems using the predictions of other classifiers in the chain as features, thus, potentially capturing important relationships among classes.

Our approach is only tangentially related to CC in the sense that it also uses the predictions of some classifiers as information to be passed out to other classifiers. However, we do not use predictions as features; there is no need to establish a coherent chain ordering (as the Gibbs Sampling algorithm allows for loops in the graph that represents classifier dependencies); and we deal with probabilistic predictions, instead of crisp ones.

In [3] authors propose ways to find class dependencies in flat multi-label datasets using classifiers and the χ^2 test. In our work we build the classifiers using dataset splits, the acceptance criterion is different and our algorithm is designed for multi-label hierarchical classification.

III. The Hierarchical Dependence Network Based on non-Hierarchical Predictive Classes

In this section we present our Hierarchical Dependence Network based on non-Hierarchical Predictive Classes (HDN-nHPC) classifier. We begin our discussion with some definitions and notations.

A DN for multi-label classification estimates $P(\mathbf{x}, C_1, C_2, \cdots, C_n)$ from $\prod_{C_i} P(C_i|C_i^{MB})$, where:

- $\mathbf{x}$ is a feature vector;

- C_i, $1 \leq i \leq n$, is a binary random variable (RV) representing the presence or absence of class label c_i;
- C_i^{MB} is an approximation for the markov blanket of RV C_i. In conventional, single-label classification problems, the set C_i^{MB} is the set of features that affects the prediction of the class C_i. In multi-label classification problems, the set C_i^{MB} consists of two parts:
  - The first part, C_i^{FB}, is the "Feature Blanket of C_i", a subset of $\mathbf{x}$, the minimal set of predictive features that affect the estimation of C_i. In this work, this set is estimated using the feature selection algorithm CFS [10].
  - The second part, C_i^{CB}, is the "Class Blanket of C_i", the minimal set of class variables that affect the estimation of C_i. In Section III-A we describe our approach to heuristically estimate C_i^{CB}.
- $P(C_i|C_i^{FB}, C_i^{CB})$ is estimated using a set of pairs of SVM classifiers, one *classifier pair* for each class in C_i^{CB}, as described in Section III-B.

A. Estimating the Class Blanket of each Class Variable

This work relies on the hypothesis that among the vast number of classes (typically hundreds) in typical hierarchical classification problems, there are class labels that, when present in an instance, change how the feature vector affects the prediction of some other class label.

Let $(C_i \leftharpoondown C_j)$ be an ordered pair of class variables such that the estimation given by $P(C_i|\mathbf{x}, C_j)$ is significantly different from $P(C_i|\mathbf{x})$. We call the set containing all such pairs of classes as set C.

To build set C and the Markov blanket of each RV $C_i \in C$, we created a method to find pairs of classes $(C_i \leftharpoondown C_j)$. These pairs of classes have the property that information about class C_j affects the probability distribution of class C_i. That is, knowing whether or not an instance is annotated with class C_j affects our predictions about C_i.

To achieve this, first we create a candidate set of predictive class relationships, C_{cand}, containing every pair of classes $(C_i \leftharpoondown C_j)$ that are not descendants (or ancestors) of each other. This condition is necessary since if classes are descendants (or ancestors) of each other, we would have deterministic relationships between c_i and c_j, that is, if c_i is an ancestor of c_j, $C_i = 0 \Rightarrow C_j = 0$ and $C_j = 1 \Rightarrow C_i = 1$, where $C_i = 0$ means that a particular class label c_j is not present in a instance, and $C_i = 1$ otherwise.

These deterministic relationships have been observed, in our preliminary experiments, to underfit classification models, i.e., models that rely too much on the values of classes that are descendants or ancestors of the current class being predicted, resulting in over-simplified models. Another condition that must be satisfied for a pair of classes to be in C_{cand} is that both classes, C_i and C_j, take the value "1" in at least 20 instances. We do this to avoid considering classes with too few instances, thus inducing unreliable classifiers. We used the 20 instances threshold based on exploratory experiments that indicate that this is a good tradeoff between classifier reliability and training set accuracy.

The next step is to induce two SVM models to classify C_i, for each pair $(C_i \leftharpoondown C_j) \in C_{cand}$. The first model, $M^0_{(i,j)}$, is trained using dataset $D^0_{(i,j)}$ and the second, $M^1_{(i,j)}$, using dataset $D^1_{(i,j)}$, where $D^0_{(i,j)} \cup D^1_{(i,j)} \equiv D_{learn}$, and D_{learn} is a random partition of 70% of the training set. Instances are assigned to $D^0_{(i,j)}$ if $C_j = 0$, conversely, assigned to dataset $D^1_{(i,j)}$ if $C_j = 1$. We use SVM because of its good predictive performance on preliminary experiments.

After training the two SVM models using the values of the class variable C_j, we need to test if the resulting pair of models (i.e., using $M^0_{(i,j)}$ and $M^1_{(i,j)}$ together) has better predictive performance than a single SVM classifier ($M_{(i,j)}$), induced using dataset D_{learn}, without splitting the data based on C_j values.

To do so, we use the training instances in D_{valid}, the complement of D_{learn}, as validation instances. Note that the classifiers did not have access to these instances earlier. To classify a new instance using the two classifiers, which we shall call a *classifier pair* from now on, we use the actual value of C_j to decide which classifier to use: if $C_j = 0$ we use $M^0_{(i,j)}$, if $C_j = 1$, we use $M^1_{(i,j)}$.

Intuitively, we would expect that if two given class variables are unrelated, i.e., the presence or absence of one class label does not change the distribution of another class label given the features ($P(C_i|\mathbf{x}, C_j) = P(C_i|\mathbf{x})$), the classifier pair would have similar predictive performance to a single classifier that uses the whole "learning set" D_{learn}. In fact, due to the reduced individual sizes of datasets $D^0_{(i,j)}$ and $D^1_{(i,j)}$, it is likely that the classifier pair ($M^0_{(i,j)}$, $M^1_{(i,j)}$) would perform worse on average than the single classifier ($M_{(i,j)}$) trained using the whole learning set D_{learn}. On the other hand, if the presence or absence of one class label affects the distribution of another class label given the predictive features, our approach would yield better classifiers than the single-classifier baseline if the classification models in the pair of classifiers can exploit such differences. Using this rationale, we select the pairs $(C_i \leftharpoondown C_j)$ whose classifiers achieved better predictive accuracy performance than the single classifier $M(i,j)$ on the validation set D_{valid} to construct the set of pairs of predictive classes C_{pred}, which is a subset of C_{cand}.

To estimate $P(C_i|C_i^{FB}, C_i^{CB})$ we need to define the class labels' class blankets, in other words, the sets C_i^{CB} for each C_i. This is done by inducing a directed graph G using set C_{pred}. For each pair $(C_i \leftharpoondown C_j) \in C_{pred}$ we create a directed edge from C_j to C_i, representing the fact that the value of C_j affects the value of C_i. Thus, the vertices that point to C_i comprise the set C_i^{CB}. In addition, recall that C_i^{FB} is estimated using the method CFS [10].

The procedures discussed thus far are presented in Algorithm 1 using the additional notation: $M^1_{(i,j)}(\mathbf{x})$, $M^0_{(i,j)}(\mathbf{x})$ and $M_{(i,j)}(\mathbf{x})$ denote the prediction of the corresponding models given the feature vector $\mathbf{x}$. The function $AUPRC(predictions)$ calculates the Area Under the Precision and Recall Curve for a single class, a measure of the predictive quality of the $predictions$, on the validation set D_{valid}.

Algorithm 1 Builds the set of predictive class relationships

procedure FIND THE SET OF PREDICTIVE CLASS RELATIONSHIPS C_{pred} AND THE GRAPH INDUCED BY C_{pred}.(Inputs: C_{cand}, D)
  $C_{pred} \leftarrow \emptyset$
  Randomly partition D into D_{learn} and D_{valid}.
  for Each $(C_i \leftharpoondown C_j) \in C_{cand}$ **do**
    $predsPair \leftarrow \emptyset$
    $predsSingle \leftarrow \emptyset$
    Generate $D^0_{(i,j)}$ and $D^1_{(i,j)}$ from D_{learn} using C_j to split the data and the CFS algorithm to select the most relevant features.
    Induce $M^0_{(i,j)}$ using the features selected from $D^0_{(i,j)}$.
    Induce $M^1_{(i,j)}$ using the features selected from $D^1_{(i,j)}$.
    Induce $M_{(i,j)}$ using the features selected from D_{learn}.
    for Each instance $(\mathbf{x}', C') \in D_{valid}$ **do**
      if $C'_j = 1$ **then**
        $predsPair \leftarrow predsPair \cup M^1_{(i,j)}(\mathbf{x}')$
      else
        $predsPair \leftarrow predsPair \cup M^0_{(i,j)}(\mathbf{x}')$
      end if
      $predsSingle \leftarrow predsSingle \cup M_{(i,j)}(\mathbf{x}')$
    end for
    if $AUPRC(predsPair) > AUPRC(predsSingle)$ **then**
      $C_{pred} \leftarrow C_{pred} \cup (C_i \leftharpoondown C_j)$
    end if
  end for
  Induce graph G, treating the pairs $(C_i \leftharpoondown C_j) \in C_{pred}$, as a directed edges from C_j to C_i.
  return G and C_{pred}.
end procedure

B. Estimating Class-label Probabilities

Once we have graph G, which represents the dependence network among class labels, we train the classifier pairs for the classes in C_{pred} using the whole training set, D_{all}, one classifier pair for each class in C^{CB}_i, for all i. The next step is to develop a way to query the classifiers of each class variable C_i and get a single prediction to estimate $P(C_i|C^{FB}_i, C^{CB}_i)$. As each class variable C_i may have several classifier pairs, one pair for each class in C^{CB}_i, to return a unified prediction, we calculate the average class-probability computed by the classifier pairs of each class variable C_i (one prediction for each element of C^{CB}_i) using the predictions of the classes in C^{CB}_i to choose which models in the classifier pairs to use. This procedure is presented in Algorithm 2.

Due to the fact that we may have cyclic dependencies - i.e. one class variable C_i may depend on some other class variable C_j that, directly or indirectly, depends on C_i - and because of the large number of possible class-label combinations, we must employ an strategy to query our Dependence Network. To accomplish this, we use the Gibbs sampling presented in [6] setting the number of iterations as 100 and the number of burn-in iterations as 50. Preliminary experiments showed that these numbers were sufficient to achieve a stationary sampling distribution.

Algorithm 2 Estimates $P(C_i|C^{FB}_i, C^{CB}_i)$.

procedure ESTIMATION OF $P(C_i|C^{FB}_i, C^{CB}_i)$(Inputs: $\mathbf{x}$: the instance's feature vector, C_i: Class variable to be predicted , G: Dependence Network, C_{pred}: Current predictions of classes other than C_i for the instance)
  $C^{CB}_i \leftarrow$ The class variables that point to C_i in G.
  $predictions \leftarrow \emptyset$
  for Each class variable $C_j \in C^{CB}_i$ **do**
    if $C_j = 1$ **then**
      $C^{FB}_j \leftarrow$ retrieve the features selected by CFS when $C_j = 1$.
      $predictions \leftarrow predictions \cup M^1_{(i,j)}(C^{FB}_i)$
    else
      $C^{FB}_j \leftarrow$ retrieve the features selected by CFS when $C_j = 0$.
      $predictions \leftarrow predictions \cup M^0_{(i,j)}(C^{FB}_i)$
    end if
  end for
  return the average class probability of the class labels in $predictions$
end procedure

If the set C^{CB}_i is empty for some class variable (either because there is no edge in G from some other variable to C_i, or because C_i is not in G), we use a standard SVM classifier model (denoted as $M_i(C^{FB}_i)$) to estimate $P(C_i|C^{FB}_i)$. In this case, there is no need to run the Gibbs sampling procedure for this particular class variable, as its class probability distribution does not depend on the presence of any other class variable.

After all posterior probabilities are calculated, we transverse the class hierarchy in a bottom up-fashion, checking if each class probability is greater than or equal to the class probability of its descendants. If this is not the case, we set the class probability of the offending class to be the maximum probability of its descendants [17].

We call the conjunction of algorithms 1, 2 and the Gibbs sampling algorithm, HDN-nHPC.

Now that we have precisely defined the HDN-nHPC algorithm, we can clarify the differences between the current work and the one proposed in [6], pointed out in the introduction. 1) The data-driven approach to find predictive class relationships is performed in Algorithm 1 by building the graph G. This graph represents important predictive class relationships. 2) The dataset split approach to induce classifiers is performed in Algorithm 1 by the creation of different classification models ($M^0_{(i,j)}$ and $M^1_{(i,j)}$) using different dataset splits ($D^0_{(i,j)}$ and $D^1_{(i,j)}$). 3) the use of several classification models to estimate probability of a class is done in Algorithm 2 by averaging the predictions of the classifiers in the Markov blanket of each class label.

IV. CREATION OF THE DATASETS OF AGEING-RELATED GENES

To study the biological aspects of ageing/longevity using our hierarchical classification algorithm, we have built 10

datasets containing features extracted from the proteins encoded by the genes in the Ageing Gene Database (GenAge) [23]. GenAge is a catalog of ageing-related genes coming from several species, including human and model organisms such as *S. cerevisiae* (yeast) and *M. musculus* (mouse).

Salama and Freitas [19] have already compiled an ageing-related dataset for the hierarchical classification of ageing-related proteins. We build upon their work by updating and expanding the dataset to contain more species and the features used in [22], which focused on the hierarchical classification of generic (not specifically ageing-related) proteins functions. In our datasets, each instance represents an ageing-related gene, and the hierarchical classes to be predicted are Gene Ontology (GO) terms. In the following sections we describe the steps that we followed to build the ageing-related datasets.

All genes were collected from the GenAge database, build 17. This version contains 298 human ageing-related genes and 1,825 genes from model organisms, related to both ageing and longevity.

The human gene dataset contains a comprehensive list of genes potentially associated with human ageing. The model organism datasets contain genes associated with ageing in non-human organisms.

After removing the species with less than 50 genes, we are left with five species: *S. cerevisiae* (yeast), *C. elegans* (worm), *H. sapiens* (human), *D. melanogaster* (fly), *M. musculus* (mouse). For each species we derive two datasets with numeric alignment independent features and protein motif features, leaving us with 10 ageing-related datasets.

The GenAge database contains the *"Entrez Gene Id"* as an external gene identifier; we use it to retrieve the *"UniprotKB AC ID"* protein identifier using the UniProt ID Mapping Tool. Because more than one protein may be associated with a single gene, 2,855 UniProt identifiers were retrieved from the 2,123 genes. However, from the 2,855 proteins, we discard 1,243 whose functions were not manually reviewed by experts or whose species is one of the four that were discarded. After this step, we end up with 1,612 proteins (instances), distributed among organisms as presented in the last column of Table I.

Finally, we downloaded the amino acid sequence of each protein from the UniProt-SwissProt database, using build 2014_02 [1]. In the next sections we briefly describe how we created the features for our datasets.

The hierarchical classes were created for each model organism by first retrieving the GO terms associated with each protein sequence using the UniProt-SwissProt database. Next, we used the DAVID tool[2] to retrieve the over-expressed GO terms of each model organism, considering only these GO terms in our final dataset. We call these over-expressed GO terms ageing-related GO terms, as they occur significantly more often than statistically expected by chance in our datasets of ageing-related proteins.

Table I: Number of features and instances for each organism (dataset type).

Species	Number of features		Number of instances
	Numeric	Motifs	
Caenorhabditis elegans	59	112	263
Drosophila melanogaster	59	55	79
Homo sapiens	60	284	301
Mus musculus	59	40	107
Saccharomyces cerevisiae	59	296	762

Numeric Alignment-Independent Features: We extracted the following numeric features described in [19, 22]: "Amino Acid Composition", "Composition", "Transition", "Distribution", and "Z-Values". Furthermore, all datasets (with all types of features) have two features: "Sequence Length" (the amino acid sequence length), and "Molecular Weight" (the molecular weight of the protein). These features are called alignment-independent, as they do not require any alignment procedure such as "BLAST" to be performed on the sequences prior to their calculation. In addition, following [8], we extend each of the human ageing datasets with the D_n/D_s ratio, which measures the degree of conservation between two gene sequences. Using the D_n/D_s ratios from the BioMart tool[3] we extracted 288 D_n/D_s ratios from the human/rhesus genes.

Protein Motif Features: The binary motif features represent the presence or absence of a given motif in the amino acid sequence of the protein. A motif is a template describing similar sequences of amino acids that occur recurrently in proteins. Motifs are a high-level representation of a protein and it is expected that proteins sharing some specific motifs share similar functions. We used the same four motif datasets investigated in [22]: Interpro, Pfam, Prosite and PRINTS. We have only considered motifs occurring in at least three proteins (instances) in the dataset, to avoid overfiting.

Table I shows the number of features and instances of each dataset type and model organism.

V. Experimental Evaluation

In this section we present the experimental evaluation of the HDN-nHPC algorithm in the ageing-related datasets. In Section V-A we present the strongest predictive class relationships found by the HDN-nHPC algorithm. In Section V-B we show the predictive accuracy results of our algorithm in several ageing-related datasets and the corresponding statistical analysis.

[1] ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/

[2] http://david.abcc.ncifcrf.gov

[3] http://www.ensembl.org/biomart/

Table II: Mean, maximum and minimum AUPRC differences (across the folds of the 10-fold cross validation process) between the pair-wise classifiers ($M^0_{(i,j)}$ and $M^1_{(i,j)}$) and the single classifier ($M_{(i,j)}$). From this table it is clear that the SVM classifier predicting class C_1 greatly benefits from the dataset split based on C_2 values.

C_1	C_2	Mean	Max.	Min.
GO:0046483	GO:0034641	0.56	0.50	0.59
GO:0034641	GO:0006725	0.55	0.51	0.62
GO:0006725	GO:0046483	0.54	0.51	0.58
GO:0006807	GO:0046483	0.53	0.52	0.53
GO:0034641	GO:1901360	0.51	0.41	0.58
GO:0006725	GO:0034641	0.51	0.49	0.54
GO:1901360	GO:0006807	0.49	0.43	0.60
GO:0046483	GO:0006807	0.49	0.46	0.53
GO:0034641	GO:0046483	0.49	0.35	0.57
GO:1901360	GO:0034641	0.47	0.42	0.52

A. *Analysis of the Predictive Class Relationships*

As an example of the benefit of detecting predictive class relationships and how to use them to improve predictive accuracy, Table II presents the 10 strongest predictive class relationships for the numeric yeast dataset, the biggest dataset with respect to the number of instances. Relationships are ranked in decreasing order of the mean $AUPRC$ accuracy measure difference, calculated as the $AUPRC$ of the pair of classifiers ($M^0_{(i,j)}$ and $M^1_{(i,j)}$) minus the $AUPRC$ of the single classifier ($M_{(i,j)}$), over the 10 folds of the cross-validation procedure.

It is clear that there are strong relationships in the yeast dataset that may be exploited by our algorithm. Table III presents additional information about the GO terms from Table II. This table contains in its columns the GO term Id, its name, the average depth of the term and the average height of the term. The average depth of the term is computed by calculating the average length of all paths from the root node of the hierarchy to the GO term being analysed. Similarly, the average height of the GO term is computed by calculating the average length of all paths from the GO term being analysed to the reachable leaves of the term hierarchy.

The average depth and average height of the GO term are used to inform us how specific a GO term is. More specific GO terms tend to provide more information to users of the classification system. Although average depth is a more common measure of term specificity, we also use the average height because sometimes a relatively shallower node may be more specific than a deeper one.

Table III shows that the selected GO terms are located, in general, in the middle of the GO, as far as depth and height are concerned. In other words, there is a compromise between specificity and generality, probably because deeper GO terms are easier to differentiate from the other nodes but, at the same time, contain less labeled instances.

Figure 1 shows the ancestors of the GO terms present in Table III, with solid edges representing the original GO relationships and dashed edges representing the predictive class

Table III: Information about the GO terms in table II

Go term	Name	Avg. Depth	Avg. Height
GO:0046483	heterocycle metabolic process	4.0	4.5
GO:0034641	cellular nitrogen compound metabolic process	4.0	4.0
GO:0006725	cellular aromatic compound metabolic process	4.0	4.8
GO:0006807	nitrogen compound metabolic process	3.0	4.8
GO:1901360	organic cyclic compound metabolic process	4.0	4.5

relationships detected by the algorithm. It is clear that the relationships found by the algorithm connect classes that are semantically related; all of them are related to metabolism and are relatively close in the original hierarchy. This is a sign that these predictive class relationships are in fact meaningful. Note that the class dependencies represented by the dashed edges (i.e., detected by our method) would be ignored by a conventional local hierarchical classification algorithm, which would consider only parent-child relationships in the original class hierarchy.

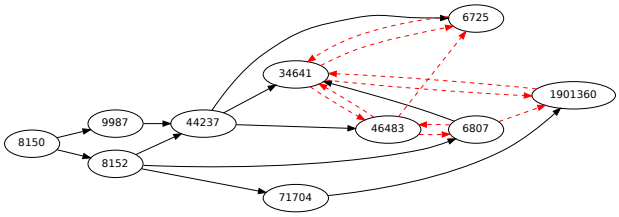

Figure 1: GO terms in Table III and their ancestors. Solid edges represent the original GO term relationships. Dashed edges represent the class dependencies present in Table II.

B. *Predictive Accuracy Results and Statistical Analysis*

We have used three measures of predictive accuracy: $AU(\overline{PRC})$, $\overline{AUPRC}_w$, and $\overline{AUPRC}$ [24]. These measures are variations of the hierarchical version of the $AUPRC$ measure, which we used as the criteria for choosing the predictive class relationships. Notice that the $AUPRC$ measure is used for individual classes, while the other measures work in the class hierarchy as a whole. For each class and for each instance, we construct a PR curve (a plot of the classifier's precision as a function of its recall) by thresholding the output (class probability) of the classifier using values in the interval $[0, 1]$. Each threshold is associated with a value of precision and recall, corresponding to a point in the PR space. To obtain a single performance measure from the curve, we calculate the area under the curve using a trapezoidal approximation [2]. A perfect classifier would have an $AUPRC$ of 1.0.

To calculate $AU(\overline{PRC})$, we use the hierarchical versions of precision and recall for a fixed threshold, defined as:

$hP \equiv \frac{\sum_j |P_j \cap T_j|}{\sum_j |P_j|}$ and $hR \equiv \frac{\sum_j |P_j \cap T_j|}{\sum_j |T_j|}$.

Where P_j is the set of predicted classes of the j-th instance and T_j is the set of true classes of the j-th instance.

To calculate $\overline{AUPRC}$ we simply average all the class-wise $AUPRC$ performances. Similarly, to calculate $\overline{AUPRC}_w$, we calculate the $AUPRC$ of each class and then the average over all classes weighted by the number of instances belonging to each class, that is, $\overline{AUPRC_w} \equiv \frac{\sum_i AUPRC_i \times S_i}{\sum_i S_i}$; where S_i is the number of instances in the i-th class.

Table IV shows the predictive accuracy results of our algorithm (HDN-nHPC) in comparison with three baseline algorithms: 1) the PCT algorithm, 2) the original HDN algorithm and 3) a LHC classifier using one classifier per node, SVM as a base classifier and CFS to select the features for each class node. The LHC classifier for class C_i uses as positive instances the ones belonging to class C_i, and as negative instances the ones that do not belong to C_i. The PCT algorithm is a strong baseline because it is considered a state-of-the-art interpretable hierarchical classifier with good predictive accuracy. The HDN is a natural baseline since it also builds a Dependence Network to estimate class-probabilities and also uses Gibbs sampling to query the network. Comparing HDN-nHPC with the LHC algorithm is important for checking if using non-hierarchical predictive class relationships enhances the performance of the base classifiers, since LHC does not use predictive class relationships.

We also tested a version of HDN-nHPC where predictive class relationships are found by a much faster statistical χ^2 test, instead of SVMs. This approach (results not shown due to lack of space) yields, on average, worst predictive performance than our HDN-nHPC algorithm, indicating that the use of SVMs to find predictive class relationships is more effective than the simpler χ^2 statistical test.

In Table IV, the best result for each dataset and each $AUPRC$ measure is shown in boldface. The best (lowest) average rank over the 10 datasets and 3 accuracy measures was obtained by HDN-nHPC with the $AU(\overline{PRC})$.

We have carried out a statistical analysis of the results in Table IV using the non-parametric Friedman test (based on the algorithms' ranks) to detect if there are statistically significant differences among classifiers, followed by the *Hochberg's* post-hoc test to detect which classifiers were statistically significant worse than the best performing classifier (the control). This procedure is explained in detail in [5]. We have performed three individual tests, one for each accuracy measure, using the significance level of 0.05.

The Friedman test has detected differences among classifiers for all three AUPRC measures. According to the post-hoc test, our new algorithm, HDN-nHPC, was the best considering the $AU(\overline{PRC})$ measure, with statistically significant better results than all three other algorithms. HDN-nHPC was also the second best classifier considering the other two measures, with no statistically significant differences with respect to the best algorithm (PCT).

As $AU(\overline{PRC})$ is a micro-average measure, it tends to assign greater weight to classes containing more instances, whereas the measures $\overline{AUPRC}_w$ and $\overline{AUPRC}$ are variations of a macro-average measure, which tends to weight classes uniformly [14]. Hence, the fact that our approach achieves significantly better performance on classes with more instances seems to be partly explained by the fact that our approach does not consider predictive class relationships occurring in classes with few instances (less than 20). Recall that for those classes we use standard SVM classifiers, which have a bias to overfit when dealing with very imbalanced class distributions [1].

VI. Conclusion

We have proposed a novel hierarchical classification algorithm, the Hierarchical Dependence Network based on non-Hierarchical Predictive Classes (HDN-nHPC). We have tested our method on 10 hierarchical classification datasets where the classes are ageing-related Gene Ontology terms.

We have shown that our algorithm is capable of finding strong predictive relationships among hierarchically unrelated classes in our datasets. Furthermore, the HDN-nHPC algorithm achieved statistically significantly better predictive accuracy than three other hierarchical classifiers in one of the three measures that we used to test the algorithms, with no statistically significant differences between our algorithm and the PCT algorithm according to the other two predictive accuracy measures. Also, the proposed HDN-nHPC algorithm obtained the best mean rank out of 12 combinations of 4 algorithms times 3 accuracy measures.

Considering that a version of the HDN-nHPC algorithm without using predictive class relationships would degenerate to the Local Hierarchical Classification (LHC) algorithm used in our experiments, we can also conclude that using predictive relationships among hierarchically unrelated classes improved the performance of the classification system, as the HDN-nHPC algorithm had better or equal mean rank than the LHC algorithm in all three accuracy measures that we have used.

Acknowledgment: The first author is supported by CAPES (Brazilian government) proc. no. 0653/13-6.

References

[1] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Mach. Learn.: ECML 04*, pages 39–50. Springer, 2004.

[2] K. Boyd, K. H. Eng, and C. D. Page. Area under the precision-recall curve: Point estimates and confidence intervals. In *Machine Learning and Knowledge Discovery in Databases*, volume 8190 of *Lecture Notes in Comp. Sci.*, pages 451–466. Springer Berlin, 2013.

Table IV: $AU(\overline{PRC})$, $\overline{AUPRC}_w$ and $\overline{AUPRC}_w$, measured by 10 fold cross-validation for the 4 hierarchical classifiers.

		$AU(\overline{PRC})$				$\overline{AUPRC}_w$				$\overline{AUPRC}$			
Feat. Type	Dataset	PCT	HDN	HDN-nHPC	LHC	PCT	HDN	HDN-nHPC	LHC	PCT	HDN	HDN-nHPC	LHC
Num.	worm	0.411	0.392	**0.431**	0.418	**0.319**	0.302	0.317	0.311	**0.096**	0.085	0.089	0.087
	fly	0.421	0.436	**0.441**	0.437	**0.398**	0.384	0.386	0.384	**0.133**	0.128	0.128	0.128
	human	0.455	0.410	**0.469**	0.465	**0.360**	0.332	0.350	0.344	**0.094**	0.085	0.088	0.087
	mouse	0.466	0.445	**0.471**	0.464	**0.376**	0.362	0.368	0.362	**0.132**	0.126	0.127	0.126
	yeast	0.426	0.347	**0.463**	0.460	0.322	0.300	**0.347**	0.344	0.080	0.068	**0.083**	0.082
Motifs	worm	**0.485**	0.406	0.445	0.433	**0.346**	0.318	0.330	0.334	**0.106**	0.089	0.092	0.093
	fly	**0.458**	0.438	0.441	0.440	**0.404**	0.383	0.384	0.386	**0.134**	0.128	0.128	0.128
	human	0.472	0.423	**0.496**	0.486	0.386	0.356	0.388	**0.398**	**0.103**	0.092	0.098	0.101
	mouse	0.466	0.447	**0.469**	0.465	0.378	0.366	0.369	**0.381**	**0.133**	0.127	0.128	0.130
	yeast	0.424	0.321	**0.453**	0.443	0.316	0.302	0.334	**0.341**	0.079	0.072	0.081	**0.087**
Avg. Rank		2.5	3.9	**1.2**	2.4	1.7	3.9	2.2	2.2	1.4	3.7	2.4	2.5

[3] L. Chekina, D. Gutfreund, A. Kontorovich, L. Rokach, and B. Shapira. Exploiting label dependencies for improved sample complexity. *Mac. Learn.*, 91(1):1–42, 2013.

[4] J. P. de Magalhães. The Biology of Ageing: A Primer. In *An Introduction to Gerontology*, chapter 2, pages 22–47. Cambridge Uni. Press, Cambridge, UK, 2011.

[5] J. Demsar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[6] F. Fabris and A. A. Freitas. Dependency Network Methods for Hierarchical Multi-label Classification of Gene Functions. *Proc. of the 2014 IEEE Comp. Intel. and Data Mining*, pages 241–248, December 2014.

[7] A. A. Freitas, D. C. Wieser, and R. Apweiler. On the importance of comprehensible classification models for protein function prediction. *IEEE/ACM trans. on comp. bio. and bioinf.*, 7(1):172–82, 2010.

[8] A. A. Freitas, O. Vasieva, and J. P. de Magalhães. A data mining approach for classifying DNA repair genes into ageing-related or non-ageing-related. *BMC genomics*, 12(1):27, January 2011.

[9] Y. Guo and S. Gu. Multi-Label Classification Using Conditional Dependency Networks. In *Proc. of the 22nd International Joint Conference on Artificial Intelligence*, volume 2, pages 1300–1305, 2011.

[10] M. A. Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, New Zealand, 1999.

[11] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *The Journal of Machine Learning Res.*, 1:49–75, 2001.

[12] J. Levatić, D. Kocev, and S. Džeroski. The importance of the label hierarchy in hierarchical multi-label classification. *Journal of Intelligent Information Systems*, December 2014. ISSN 0925-9902.

[13] L. Li, L. Zhang, G. Li, and H. Wang. Probabilistic classifier chain inference via gibbs sampling. In *Proc. of the 23rd ACM Inter. Conf. on Infor. and Know. Mana. (CIKM '14)*, pages 1855–1858, 2014.

[14] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge University Press, 2008.

[15] The Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic acids research*, 32:D258–61, 2004.

[16] The Uniprot Consortium. The Universal Protein Resource (UniProt) in 2010. *Nucleic acids research*, 38: D142–8, January 2010.

[17] G. Obozinski, G. Lanckriet, and C. Grant. Consistent probabilistic outputs for protein function prediction. *Genome Biology*, 9(S6.1-S6.19):1–19, 2008.

[18] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier Chains for Multi-label Classification. *Machine Learning*, 85(3):333–359, June 2011.

[19] K. M. Salama and A. A. Freitas. ACO-Based Bayesian network ensembles for the hierarchical classification of ageing-related proteins. volume 7833 of *Lecture Notes in Computer Science*, pages 80–91, 2013.

[20] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, and S. Dzeroski. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC bioinformatics*, 11:2, January 2010.

[21] C. N. Jr Silla and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowl. Disc.*, 44(1-2):31–72, 2011.

[22] C. N. Jr Silla and A. A. Freitas. Selecting different protein representations and classification algorithms in hierarchical protein function prediction. *Intelligent Data Analysis*, 15(6):979–999, 2011.

[23] R. Tacutu, T. Craig, A. Budovsky, D. Wuttke, G. Lehmann, et al. Human Ageing Genomic Resources: integrated databases and tools for the biology and genetics of ageing. *Nucleic acids research*, 41: D1027–33, January 2013.

[24] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. *Mach. Learning*, 73(2):185–214, 2008.