Learning Hierarchical Multi-label Classification Trees from Network Data

Daniela Stojanova[1], Michelangelo Ceci[2], Donato Malerba[2], and Sašo Džeroski[1,3,4]

[1] Jožef Stefan Institute, Department of Knowledge Technologies, Ljubljana, Slovenia
[2] Dipartimento di Informatica, Università degli Studi di Bari, Bari, Italy
[3] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[4] COE for Integrated Approaches in Chemistry and Biology of Proteins, Slovenia
{daniela.stojanova,saso.dzeroski}@ijs.si, {ceci,malerba}@di.uniba.it

Abstract. We present an algorithm for hierarchical multi-label classification (HMC) in a network context. It is able to classify instances that may belong to multiple classes at the same time and consider the hierarchical organization of the classes. It assumes that the instances are placed in a network and uses information on the network connections during the learning of the predictive model. Many real world prediction problems have classes that are organized hierarchically and instances that can have pairwise connections. One example is web document classification, where topics (classes) are typically organized into a hierarchy and documents are connected by hyperlinks. Another example, which is considered in this paper, is gene/protein function prediction, where genes/proteins are connected and form protein-to-protein interaction (PPI) networks. Network datasets are characterized by a form of autocorrelation, where the value of a variable at a given node depends on the values of variables at the nodes it is connected with. Combining the hierarchical multi-label classification task with network prediction is thus not trivial and requires the introduction of the new concept of network autocorrelation for HMC. The proposed algorithm is able to profitably exploit network autocorrelation when learning a tree-based prediction model for HMC. The learned model is in the form of a Predictive Clustering Tree (PCT) and predicts multiple (hierarchically organized) labels at the leaves. Experiments show the effectiveness of the proposed approach for different problems of gene function prediction, considering different PPI networks. The results show that different networks introduce different benefits in different problems of gene function prediction.

1 Introduction

Hierarchical multi-label classification (HMC) is a variant of the classification task where instances may belong to multiple classes at the same time and classes are organized in a hierarchy. Due to the large number of applications (e.g., computational biology [31] and text categorization [7]) this particular classification setting has recently attracted the attention of many researchers.

J. Fürnkranz, E. Hüllermeier, and T. Higuchi (Eds.): DS 2013, LNAI 8140, pp. 233–248, 2013.

Another variant of the classification task which, due to the large number of applications, has attracted the attention of researchers over the last few years is that of learning predictive models from network data. This is due to the ubiquity of networks, which can be found in in several social, economical and scientific fields. As recently recognized [29], this problem is not trivial, mainly because the network introduces some form of autocorrelation which is a direct violation of the assumption that data are independently and identically distributed (i.i.d.). However, autocorrelation also offers a unique opportunity to improve the performance of predictive models on network data, as inferences about one entity can be used to improve inferences about related entities.

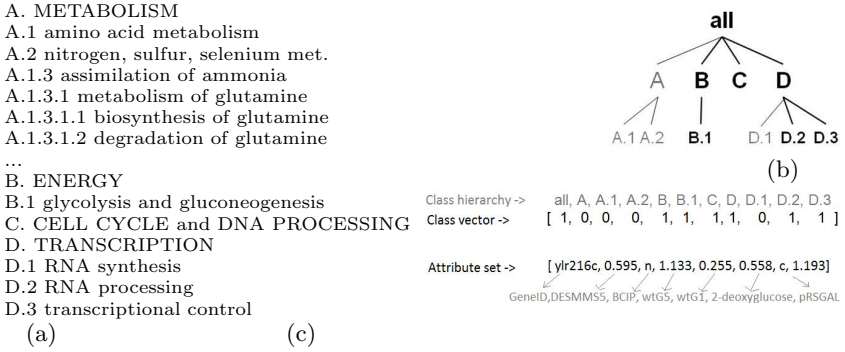

Fig. 1. An illustration of the task of HMC. (a) A part of the FUN hierarchy [21]. (b) An example of hierarchial labeling of instances (subset) of the hierarchy. (c) The class vector (top) and the attribute vector (bottom) of the instance.

The combination of these two research directions paves the way to solving prediction problems where instances may belong to multiple classes at the same time, classes are organized in a hierarchy and instances may be connected according to a network structure. One of the main cases where this combination turns out to be beneficial is in gene function prediction, where the goal is to discover the biological functions of the genes/proteins. In fact, in this case, ontologies and catalogs of functions of genes/proteins such as the Gene Ontology (GO) and MIPS-FUN assume that functional classes are organized hierarchically and that general functions include more specific functions.

Besides these relationships among classes, it is also possible to identify relationships among examples. An example is that of protein-protein interaction (PPI) networks which represent correlations between genes/proteins. Indeed, the topic of using protein-protein interaction (PPI) networks in the identification and prediction of protein functions has attracted increasing attention in recent years. The motivation for this stream of research is best summarized by the statement that "when two proteins are found to interact in a high throughput assay, we also tend to use this as evidence of functional linkage" [17].

This paper demonstrates the benefits (in terms of predictive accuracy) of considering network information in multi-label gene function prediction. In particular, during the learning process, we identify and take into account network autocorrelation, that is, the statistical relationships between the same variable (e.g., protein function) on different but related (dependent) objects (e.g., interacting proteins) [29]. We present a tree-based algorithm which extends and revises the system CLUS-HMC [31] that learns Predictive Clustering Trees (PCTs) by considering network autocorrelation in the setting of Hierarchical Multi-label Classification.

The paper is organized as follows. In the next section, we present some related work. In Sections 3 and 4, we formally introduce the learning setting we intend to solve and present the proposed approach, namely NHMC (Network Hierarchical Multi-label Classification). In Section 5, we empirically evaluate the proposed algorithm on 12 yeast datasets using each of the MIPS-FUN and GO annotation schemes and exploiting 3 different PPI networks. Finally, we draw some conclusions and outline some directions for future work.

2 Related Work

Many machine learning approaches tackle the problem of multi-label classification on hierarchically-structured categories. A simple solution is to allow a classifier for a particular node to predict positive only if the classifier of its parent also predicts positive [3] [8]. A different approach is to construct a training set for each category such that it only consists of samples belonging to its parent [10]. Alternatively, large margin methods for structured output prediction can also be used [25]. In the work by Vens et al. [31] the algorithm CLUS-HMC is proposed. CLUS-HMC learns Predictive Clustering Trees (PCTs) for hierarchical multilabel classification. A similar approach is presented in Astikainen et al. [2], where a structured output kernel-based approach for enzyme function prediction is proposed. In [5], the authors propose to formulate the search for the optimal consistent multi-label as the finding (according to a greedy search) of the best subgraph in a tree/DAG.

As concerns the problem of learning predictive models from network data, numerous approaches have been designed for modeling a partially labeled network and providing accurate estimates of unknown labels associated with the unlabeled nodes. These approaches have been mainly studied in the research fields of collective inference. In collective inference, interrelated values are inferred simultaneously and estimates of neighboring labels influence one another [19,14,27]. In general, one of the major advantages of collective inference lies in its powerful ability to learn various kinds of dependency structures (e.g., different degrees of correlation [16]). However, as pointed out in [22], when the labeled data are very sparse, the performance of collective classification might be largely degraded due to the insufficient number of neighbors. This is overcome by incorporating informative "ghost edges" into the network to deal with sparsity issues [20,22]. An alternative solution to the sparsity of labels is provided by [6], where the

authors resort to an active learning approach in order to judiciously select nodes for which a manual labeling is required from the domain expert. Finally, in [29] the authors exploit the concept of network autocorrelation as a heuristic to be used when learning PCTs.

Unfortunately, there are only few initial works that combine these two research directions and most of them are developed in the context of protein function annotation and prediction. In this context, some studies use PPI networks as one of the data sources. For example, Valentini [30] developed the *true-path rule* ensemble learner for genome-wide gene function prediction. In these ensembles, positive (negative) probabilistic predictions for a node transitively influence the ancestors (descendants) of the node. An empirical evaluation of both this method and Hierarchical Bayes [24], which operates in the same way by approximating the Bayesian-optimal predictor with respect to the H-loss, proved that the usage of hierarchical prediction methods results in a consistently improved performance as compared to methods which do not consider the structure of the reference ontology on protein functions. Information from PPI networks considered in this work is limited to binary (input) attributes which express the generic interaction of a gene with others. Outside the context of protein function prediction, in [18], the authors propose a multi-label collective classifier, which, however, does not consider the hierarchical organization of class labels.

3 Autocorrelation in Network HMC Tasks

In this Section, we first define the task of hierarchical multi-label classification (HMC). We next discuss the network setting that we consider in this paper.

3.1 The Task of HMC

For the HMC task, the input is a dataset consisting of example pairs $(x_i, y_i) \in \mathbf{X} \times 2^C$, where $\mathbf{X} = X_1 \times X_2 \ldots \times X_m$ is the space spanned by m attributes or features, while 2^C is the power set of $C = \{c_1, \ldots, c_K\}$, the set of all possible class labels. C is hierarchically organized with respect to a partial order $\preceq$ which represents the superclass relationship. Note that each y_i satisfies the *hierarchical constraint*:

$$c \in y_i \Rightarrow \forall c' \preceq c : c' \in y_i. \tag{1}$$

3.2 Network HMC

Following Steinhaeuser et al. [28], we view a training set as a single network of labeled nodes. Formally, the network is defined as an undirected edge-weighted graph *G=(V,E)*, where V is the set of labeled *nodes*, while $E \subseteq \{\langle u, v, w\rangle | u, v \in V, w \in \mathbb{R}^+\}$ is the set of *edges*, such that to each edge $u \leftrightarrow v$ is assigned a non-negative real number w, called the *weight* of the edge. It can be represented by a symmetric adjacency matrix $\mathbf{W}$, whose entries are positive ($w_{ij} > 0$) if there is an edge connecting i to j in G, and zero ($w_{ij} = 0$) otherwise. Edge weights

can, for example express the strength of the interactions between proteins. Although the proposed method works with any non-negative weight values, we anticipate that in our experiments only binary (0/1) weights could be used, due to limitations of available data.

Each node of the network is associated with an example pair $(x_i, y_i) \in \mathbf{X} \times 2^C$, where $y_i = (y_{i_1}, y_{i_2}, ..., y_{i_q}), q \leq K$, is subject to the hierarchical constraint. Given a network $G = (V, E)$ and a function $\eta : V \longmapsto (\mathbf{X} \times 2^C)$ which associates each node with the corresponding example pair, we interpret the task of HMC as building a PCT which represents a multi-dimensional predictive function $f : \mathbf{X} \longmapsto 2^C$ whose prediction satisfies the hierarchical constraint (Formula (1)). When learning f, we take into account (e.g., maximize) the autocorrelation of the observed (hierarchically organized) classes y_i for the network G, and minimizes the predictive error f on the training data $\eta(V)$. It is noteworthy that, although f is learned by taking the network structure into account, it does not take as input the network structure. This is due to our network setting, which is very much different from existing approaches to network classification and regression, where typically the descriptive information is in a tight connection to the network structure. The connections (edges in the network) between the data in the training/testing set are predefined for a particular instance, and are used to generate the descriptive information associated to the nodes of the network (as in [28]). Therefore, in order to predict the value of the response variable(s), besides the descriptive information one needs the connections (edges in the network) to the related/similar entities.

3.3 Autocorrelation in NHMC Tasks

Network autocorrelation for HMC is a special case of network autocorrelation [13]. It can be defined as the statistical relationship between observations of a variable (e.g., protein function) on distinct but related (connected) nodes in a network (e.g., interacting proteins). Autocorrelation is typically defined for real-world variables. In HMC, domain values of the variable form a hierarchy, such as the GO hierarchy for protein functions. Therefore, it is possible to define network autocorrelation at various levels of the hierarchy. The network (e.g., PPI network) can provide useful (and diversified) information for different classes (e.g., different protein functions) at different levels of the hierarchy.

To better explain this concept, we refer to Fig. 2 which shows the DIP Yeast network. Fig. 2 (a) represents the network so that the examples that are not connected are randomly arranged along the ellipse border. To show the density of the PPI interactions, Fig. 2 (b) represents the same network so that all examples are arranged along the ellipse border. Fig. 2 (c) and Fig. 2 (d) provide us a different view of Fig. 2 (b), where examples are grouped according to the first (Fig. 2 (c)) and second level (Fig. 2 (d)) of the FUN hierarchy.

Keeping in mind that all these graphs represent the same number of edges, from the comparison of Fig. 2 (b) and Fig. 2 (c) we can see that the edges "move" from the center of the ellipse towards the border. This is clearly due to autocorrelation, since the number of interactions between genes of the same class

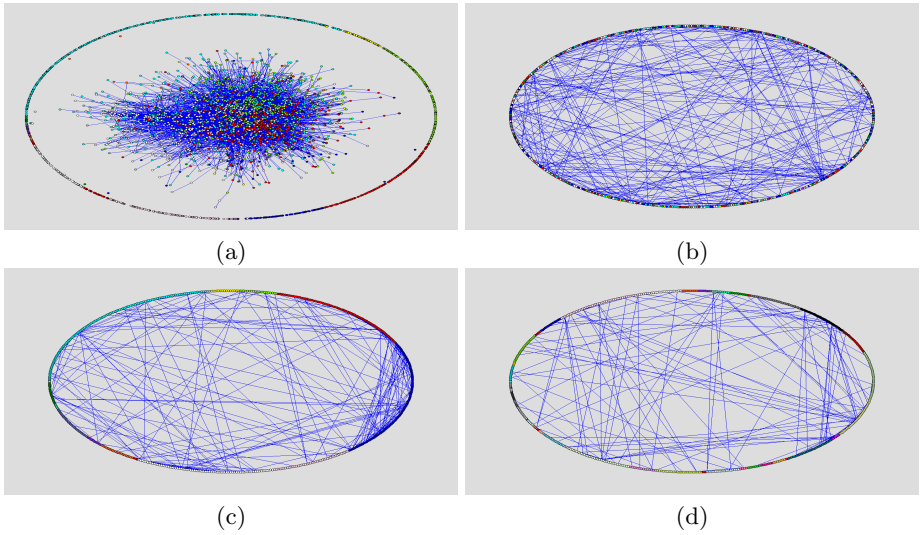

Fig. 2. DIP Yeast network. Different colors correspond to different classes of the FUN hierarchy. (a) Examples that are not connected are arranged along the ellipse's border; (b) Examples are arranged along the ellipse's border to show the density of the PPI interactions; (c) Examples are arranged along the ellipse's border and grouped according to the first level of the FUN hierarchy (not considering the root); d) Examples are arranged along the ellipse's border and grouped according to the second level of FUN. The networks are drawn by using the Pajek Software by Batagelj and Mrvar [4].

at the same level of the hierarchy is much larger than the number of interactions between genes of different classes. Moreover, by comparing Fig. 2 (c) and Fig. 2 (d) we notice that the autocorrelation effect is more localized at the second level of the hierarchy then at the first. Indeed, in Fig. 2 (d), we observe a reduction of the density of edges in the center of the ellipse (most of the edges overlap with (are hidden by) the examples arranged along the ellipse border).

4 NHMC

In this Section, we introduce the method NHMC (Network CLUS-HMC), which builds autocorrelation-aware HMC models. We shall start with a brief description of the algorithm CLUS-HMC which is at the base of NHMC. Moreover, before describing the NHMC method, we propose a new network autocorrelation measure for HMC tasks.

4.1 CLUS-HMC

The CLUS-HMC [31] algorithm builds HMC trees. These are very similar to classification trees, but each leaf predicts a hierarchy of class labels rather than

a single class label. CLUS-HMC builds the trees in a top-down fashion as in classical tree induction algorithms, with a key difference in the search heuristics.

To select the best test in an internal node of the tree, the algorithm scores the tests according to the reduction in variance (defined below) induced on the set U of examples associated to the node. In CLUS-HMC, the variance is defined as follows: $Var(U) = \frac{1}{|U|} \cdot \sum_{u_i \in U} d(L_i, \overline{L})^2$, where $d(\cdot, \cdot)$ is a distance function on vectors associated to class labels of examples in U. Class labels associated to an example (x_i, y_i) in U are represented as a binary vector L_i of size $|C|$, such that $L_{i,k} = 1$ if $c_k \in y_i$, $L_{i,k} = 0$ otherwise. Obviously, each L_i has to satisfy the hierarchical constraint (Formula (1)). Finally, $\overline{L}$ is the average vector of $\{L_i\}_i$.

In the HMC context, class labels at higher levels of the annotation hierarchy are more important than class labels at lower levels. This is reflected in the distance measure used in the above formula, which is a weighted Euclidean distance: $d(L_1, L_2) = \sqrt{\sum_{k=1}^{K} \omega(c_k) \cdot (L_{1,k} - L_{2,k})^2}$, where $L_{i,k}$ is the k-th component of the class vector L_i and the class weights $\omega(c_k)$ decrease with the depth of the class in the hierarchy. More precisely, $\omega(c) = \omega_0 \cdot \text{avg}_\text{j} \{\omega(p_j(c))\}$, where $p_j(c)$ denotes the j-th parent of class c and $0 < \omega_0 < 1$). This definition of the weights allows us to take the label hierarchy into account: The hierarchy can be either a tree or a DAG, where we can have multiple parents of a single label.

For instance, consider the small hierarchy in Fig. 1(b), and two examples (x_1, y_1) and (x_2, y_2), where[1]:

$$y_1 = \{all, B, B.1, C, D, D.2, D.3\} \text{ and } \; y_2 = \{all, A, D, D.2, D.3\}$$

The vectors for y_1 and y_2 are: $L_1 = [1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1]$ and $L_2 = [1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1]$. The distance between the two is:

$$d([1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1], [1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1]) = \sqrt{3 \cdot w_0^2 + 4 \cdot w_0^3} \quad (2)$$

At each node of the tree, the test that maximizes the variance reduction is selected. This hopefully results in maximizing cluster homogeneity with respect to the target variable, as well as in improving the predictive performance of the tree. If no test can be found that significantly reduces variance (as measured by a statistical F-test), then the algorithm creates a leaf and labels it with a prediction, which can consist of multiple hierarchically organized labels.

We can now proceed to describe the top-down induction algorithm for building Network HMC trees. The search space is exactly the same as for CLUS-HMC, while the heuristics is different. The network is considered as background knowledge to exploit only in the learning phase.

4.2 Outline of the NHMC Algorithm

The top-down induction algorithm for building PCTs from network data is given below (Algorithm 1). It takes as input the network $G = (V, E)$ and the

[1] "all" indicates the root of the hierarchy.

corresponding HMC dataset U defined by applying $\eta: V \mapsto \mathbf{X} \times 2^C$ to the vertices of the network. It then recursively partitions U until a stopping criterion is satisfied (Algorithm 1 line 2). Since the implementation of this algorithm is based on the implementation of the CLUS-HMC algorithm, we call this algorithm NHMC (Network CLUS-HMC).

Algorithm 1. Top-down induction of NHMC

1: **procedure** NHMC(G, U) **returns** tree
2: **if** stop(U) **then**
3: **return** leaf(Prototype(U))
4: **else**
5: $(t^*, h^*, \mathcal{P}^*) = (null, 0, \emptyset)$
6: **for each** possible Boolean test t according to the values of X in U **do**
7: $\mathcal{P} = \{U_1, U_2\}$ partition induced by t on U
8: $h = \alpha \cdot \left(\frac{|U_1| \cdot A_Y(U_1) + |U_2| \cdot A_Y(U_2)}{|U|}\right) + (1 - \alpha) \cdot \left(Var'(U) - \frac{|U_1| \cdot Var'(U_1) + |U_2| \cdot Var'(U_2)}{|U|}\right)$
9: **if** $(h > h^*)$ **then**
10: $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
11: **end if**
12: **end for**
13: $tree_1 = \text{NHMC}(G, U_1)$
14: $tree_2 = \text{NHMC}(G, U_2)$
15: **return** node$(t^*, tree_1, tree_2)$
16: **end if**

Geary's C for HMC. In order to measure the autocorrelation of the response variable Y in the network setting for HMC, we propose a new statistics, named $A_Y(U)$, whose definition draws inspiration from Global Geary's C [15], originally defined for spatial data analysis.

Let $(x_i, y_i) \in U \subseteq \mathbf{X} \times 2^C$ be an example pair in a training set U of N examples. Let K be the number of classes in C, possibly defining a hierarchy. Let $d(L_i, L_j)$ be a distance measure defined for two binary vectors associated to two example pairs (x_i, y_i), (x_j, y_j). It can be any distance which can take the class-label hierarchy into account. We will use the distance defined above.

The autocorrelation measure $A_Y(U)$ is defined as follows:

$$A_Y(U) = 1 - \frac{(N-1) \cdot \sum_i \sum_j w_{ij} \cdot d(L_i, L_j)^2}{4 \cdot \sum_i \sum_j w_{ij} \quad \cdot \quad \sum_p d(L_p, \overline{L})^2} \tag{3}$$

The constant 4 in the denominator is included for scaling purposes. The new autocorrelation measure $A_Y(U)$ takes values in the unit interval $[0, 1]$, where 1 (0) means strong positive (negative) autocorrelation and 0.5 means no autocorrelation.

Heuristics. The major difference between NHMC and CLUS-HMC is in the heuristics we use for the evaluation of each possible split. The variance reduction heuristics employed in CLUS-HMC aims at finding accurate models, since it considers the homogeneity in the values of the target variables and reduces the error on the training data. However, it does not consider the dependencies of the target variables values between related examples and therefore neglects the possible presence of autocorrelation in the training data. To address this issue, we introduced network autocorrelation in the search heuristic and combined it with the variance in a new heuristics.

More formally, the NHMC heuristics is a linear combination of the average autocorrelation measure $A_Y(\cdot)$ (first term) (Algorithm 1 line 8) and the variance reduction $Var(\cdot)$ (second term):

$$Var'(U) = \frac{Var(U) - \delta_{min}}{\delta_{max} - \delta_{min}}, \tag{4}$$

where $Var'(U)$ is the min-max normalization of $Var(U)$ required to keep the values of the linear combination in the unit interval $[0, 1]$, with δ_{max} and δ_{min} being the maximum and the minimum values of $Var(U)$ over all tests.

We point out that the heuristics in NHMC combines information on both the network structure, which affects $A_Y(\cdot)$, and the hierarchical structure of the class, which is embedded in the computation of the distance $d(\cdot, \cdot)$ used in Formula (3) and Algorithm 1 line 8). We also note that the tree structure of the NHMC model makes it possible to consider different effects of the autocorrelation phenomenon at different levels of the tree model, as well as at different levels of the hierarchy (non-stationary autocorrelation). In fact, the effect of the class weights $\omega(c_j)$ is that higher levels of the tree will likely capture the regularities at higher levels of the hierarchy.

In NHMC, the time complexity of selecting a splitting test represents the main cost of the algorithm. The cost is $O(m{\cdot}(N{\cdot}logN{+}N{\cdot}s){\cdot}K){+}O(m{\cdot}d{\cdot}(N{+}N{\cdot}s){\cdot}K)$, that is $O(m \cdot N \cdot (logN + d \cdot s) \cdot K)$, where N is the number of examples in the training set, m is the number of descriptive variables, s is the average number of edges for each node in the network and K is the number of classes. This complexity is similar to that of CLUS-HMC, except for the s factor which equals N in the worst case, although such worst-case is unlikely.

The relative influence of the two parts of the linear combination in Algorithm 1, line 8 is determined by a user-defined coefficient α that falls in the interval [0,1]. When $\alpha = 0$, NHMC uses only autocorrelation and when $\alpha = 0.5$, it weights equally variance reduction and autocorrelation. When $\alpha = 1$ NHMC just works as the original CLUS-HMC algorithm.

If autocorrelation is present, examples with high autocorrelation will fall in the same cluster and will have similar values of the response variable. In this way, we are able to keep together connected examples without forcing splits on the network structure (which can result in losing generality of the induced models).

Finally, note that the linear combination that we use in this article (Algorithm 1, line 8) was selected as a results of our previous work on learning from autocorrelated data [29]. The variance and autocorrelation can also be combined in some

other way (e.g., as a cross-product). Investigating different ways of combining them is one of the directions for our future work.

5 Empirical Evaluation

In this Section, we present the evaluation of the system NHMC on several datasets related to predicting gene function in yeast. Before we proceed to presenting the empirical results, we provide a description of the datasets used and the experimental settings.

5.1 Data

We use 12 yeast (*Saccharomyces cerevisiae*) datasets as considered by Clare and King [11], but with new and updated class labels [31]. The datasets describe different aspects of the genes in the yeast genome. They include five types of bioinformatics data: sequence statistics, phenotype, secondary structure, homology and expression. The different sources of data highlight different aspects of gene function.

We construct two versions of each dataset. The values of the descriptive attributes are identical in both versions, but the classes are taken from two different classification schemes. In the first version, they are from FUN[2], a scheme for classifying the functions of gene products, developed by MIPS [26]. FUN is a tree-structured class hierarchy; a small part is shown in Fig. 1(a). In the second version of the data sets, the genes are annotated with terms from the Gene Ontology (GO) [1][3], which forms a directed acyclic graph instead of a tree: Each term can have multiple parents (we use GO's "is-a" relationship between terms). Only annotations from the first six GO levels are taken. Note that GO has an order of magnitude more classes than FUN for our datasets. The 24 resulting datasets can be found at the webpage[4].

In addition, we use several protein-protein interaction networks (PPIs) for yeast genes as in Rahmani et al. [23]. In particular, the networks DIP [12], VM [32] and MIPS [21] are used, which contain 51233, 65982 and 38845 interactions among 7716, 2399 and 40374 proteins, respectively. DIP (Database of Interacting Proteins) stores and organizes information on binary protein-protein interactions that are retrieved from individual research articles. VM stores protein-protein interactions that are retrieved from numerous sources, including experimental data, computational prediction methods and public text collections. Finally, MIPS represents interactions between proteins determined on the basis of their signal transduction.

The basic properties of the datasets and of the networks are given in Table 1. Columns 6-8 (% of connected genes) show the percentage of proteins that are covered by each of the PPI networks. On average, only a half of the proteins

[2] `http://www.helmholtz-muenchen.de/en/mips/projects/funcat`
[3] `http://www.geneontology.org`
[4] `http://kt.ijs.si/daniela_stojanova/NHMC/`

Table 1. Basic properties of the datasets and the PPI networks when predicting gene function in yeast. We use 12 yeast (*Saccharomyces cerevisiae*) datasets (as considered by [11]) grouped by their functional (FUN and GO) annotation and 3 different PPI networks (DIP [12], VM [32] and MIPS [21]). In addition, the percentage of connected genes and the percentage of function related genes are presented for each of the networks.

Annotation	Dataset	#Instances	#Attributes	#Classes	% of connected genes			% of function related genes		
					DIP	VM	MIPS	DIP	VM	MIPS
FUN	seq	3932	476	499	46	43	46	8	11	7
	pheno	1592	67	455	46	34	46	6	6	7
	struc	3838	19629	499	13	43	13	7	10	6
	hom	3848	47035	499	45	43	45	7	10	6
	cellcycle	3757	77	499	72	44	72	2	10	6
	church	3779	550	499	46	44	46	15	9	5
	derisi	2424	63	499	72	69	72	7	10	6
	eisen	3725	79	461	35	31	35	9	10	7
	gasch1	3764	172	499	47	44	47	9	10	6
	gasch2	3779	51	499	47	44	47	7	10	6
	spo	3703	79	499	48	44	48	3	4	3
	exp	3782	550	499	46	44	46	15	9	5
GO	seq	3900	476	4133	46	43	46	15	22	13
	pheno	1587	67	3127	46	34	46	16	18	3
	struc	3822	19629	4132	59	42	59	14	19	12
	hom	3567	47035	4126	48	47	48	14	22	12
	cellcycle	3751	77	4125	47	44	47	17	26	14
	church	3774	550	4131	46	44	46	13	23	13
	derisi	2418	63	3573	73	69	73	11	18	9
	eisen	3719	79	4119	35	31	35	19	25	15
	gasch1	3758	172	4125	47	44	47	19	26	14
	gasch2	3758	51	4131	47	44	47	17	26	14
	spo	3698	79	4119	48	44	48	17	25	14
	exp	3773	550	4131	46	44	46	39	23	13

are known to interact with other proteins. DIP covers the highest percentage of proteins. However, this percentage is not much different from that of the other two networks, especially from MIPS.

In addition, Table 1 shows the percentage of function-relevant interactions. An interaction is considered to be function-relevant if the two proteins involved in the interaction have at least one function in common (with respect to a given hierarchy). As it is possible to see, 6%-23% observed interactions are relevant. However, a closer look at the statistics reveals that the connections are more function-relevant with respect to GO annotations (for all networks) than with respect to FUN annotations. This is expected, as GO contains a much larger number of functions.

5.2 Experimental Setup

In order to evaluate the performance of the proposed NHMC algorithm, we compare it to CLUS-HMC (NHMC works just as CLUS-HMC when $\alpha = 1$). Moreover, we report the results of NHMC with $\alpha = 0$, when it uses only autocorrelation, and with $\alpha = 0.5$, when it equally weights variance reduction and autocorrelation. The performance of the methods is investigated across a range of experimenters.

In the experiments, we deal with several dimensions: different descriptions of the genes, different descriptions of gene functions, and different gene interaction networks. We have 12 different descriptions of the genes from the Clare and King' datasets [11] and 2 class hierarchies (FUN and GO), resulting in 24 datasets with several hundreds of classes each. Furthermore, we use 3 different PPI networks (DIP, VM and MIPS) for each of those.

As suggested by Vens et al. [31], we build models trained on 2/3 of each data set and test on the remaining 1/3. We use the same splitting in order to allow a direct comparison with their work. To prevent over-fitting, we use two pre-pruning methods: minimal number of examples in a leaf (set to 5) and F-test pruning. The latter uses the F-test to check whether the variance reduction is statistically significant at a given level (0.001, 0.005, 0.01, 0.05, 0.1, 0.125).

Following Vens et al. [31], we evaluate the proposed algorithm by using the Average Area Under the Precision-Recall Curve ($\overline{AUPRC}$), i.e., the (weighted) average of the areas under the individual (per class) Precision-Recall (PR) curves, where all weights are set to $1/|C|$, with C the set of classes. The closer the $\overline{AUPRC}$ is to 1.0, the better the model is. In the considered datasets, the positive examples for a given class are rare as compared to the negative ones. A PR curve plots the precision of a classifier as a function of its recall. The points in the PR space are obtained by varying the value for a threshold τ. In the case of NHMC, the threshold ranges from 0 to 1 with a step of 0.02. The evaluation by using PR curves (and the area under them), is the most suitable in this context, because we are more interested in correctly predicting the positive instances (i.e., that a gene has a given function), rather than correctly predicting the negative ones.

5.3 Results

For each of the datasets, the $\overline{AUPRC}$ of CLUS-HMC (which does not consider network information) and NHMC, which uses the DIP, VM and MIPS PPI networks is shown in Table 2. For each algorithm and dataset, both FUN and GO annotations are considered.

The best results are obtained by using CLUS-HMC for FUN annotations and NHMC with $\alpha = 0.5$ for GO annotations (In the latter case, the average good results are mainly due to the results obtained on the cellcycle dataset). This can be explained by the fact that only a half of the genes have at least one connection to other genes in the PPI networks and this is not enough to improve the predictive accuracy of the global predictive HMC model that is constructed using NHMC (over the model constructed by using CLUS-HMC).

NHMC shows competitive results with respect to CLUS-HMC when using FUN annotations. The situation is different in the case of GO annotations, where NHMC outperforms (in almost all the cases) CLUS-HMC. This is mainly due to the the larger percentage of functionally relevant connections (see Table 1)

Table 2. The $\overline{AUPRC}$ of CLUS-HMC ($\alpha = 1$) and NHMC ($\alpha = 0.5$ and $\alpha = 0$) when predicting gene function in yeast. We use 12 yeast (*Saccharomyces cerevisiae*) datasets (as considered by [11]) with 2 different functional annotation schemes (FUN and GO) and 3 networks (DIP, VM and MIPS).

Dataset	FUN annotated datasets							GO annotated datasets						
		DIP		VM		MIPS			DIP		VM		MIPS	
	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$
seq	**.059**	.054	.053	.054	.054	.043	.043	.023	.032	.030	.028	.028	**.033**	**.033**
pheno	**.036**	.035	.028	**.036**	**.036**	.035	.035	.019	.016	.016	**.083**	**.083**	.051	.051
struc	**.030**	.020	.020	.020	.020	**.030**	**.030**	.018	.012	.012	**.066**	**.066**	**.066**	**.066**
homo	**.073**	.020	.023	.020	.020	**.073**	**.073**	.040	.013	.013	.041	.041	**.049**	**.049**
cellcycle	.032	.030	**.037**	.032	.032	.032	.032	.019	**.287**	**.288**	.036	.036	.027	.027
church	**.029**	.020	.020	**.029**	**.029**	.027	.027	.014	.015	.012	**.036**	**.036**	.030	.030
derisi	.027	**.028**	.025	**.028**	**.028**	.027	.027	.017	.015	.017	**.041**	**.041**	.030	.030
eisen	**.047**	.042	.025	.036	.036	.037	.037	.030	.024	.024	**.052**	**.052**	.046	.046
gasch1	.036	.040	.032	**.047**	**.047**	.030	.045	.024	.018	.019	.031	.031	**.040**	**.040**
gasch2	**.034**	**.034**	.027	.029	.029	.028	.030	.020	.021	.021	**.050**	**.050**	.031	.031
spo	**.030**	.029	.025	**.030**	**.030**	.028	.029	.019	.018	.015	.031	.031	**.041**	**.041**
exp	.040	.030	.025	.033	.033	**.045**	**.045**	.023	.017	.016	**.065**	**.065**	.041	.041
Average:	.039	.032	.028	.033	.033	.036	.038	.022	.041	.040	.047	.047	.041	.041

which indicates that DIP; VM and MIPS networks present some form of autocorrelation on the GO labels. Moreover, the results obtained on GO annotated datasets by using NHMC with $\alpha = 0.5$ and $\alpha = 0$ are similar, indicating that the autocorrelation in some cases, dominates the heuristics.

Comparing the results obtained with the use of different PPI networks, it can be seen that there is no clear indication that one network is better than others. In particular, the best network to be included seems to be related to the specific dataset. This means that some networks provide more information for the classification of examples in some datasets than for the classification of other datasets. In this perspective, NHMC can be used in the evaluation of the contribution of single networks for a given classification problem.

We also compare the results of NHMC to the results of recent bio-inspired strategies, which work in the HMC setting, but do not consider network information. These include Artificial Neural Networks (HMC-LMLP), Ant Colony Optimization (hmAnt-Miner), as well as a genetic algorithm for HMC (HMC-GA) [9]. While the first algorithm is a 1-vs-all (it solves several binary classification problems) method based on artificial neural networks trained with the Backpropagation algorithm, the latter two are methods that discover HMC rules.

The algorithms are evaluated on 7 yeast FUN annotated datasets [11] using the same experimental setup as for CLUS-HMC and NHMC. In Table 3, we present the $AU\overline{PRC}$ results obtained by using HMC-GA, HMC-LMLP, hmAnt-Miner and NHMC ($\alpha = 0.5$) on several FUN annotated datasets. NHMC outperforms all other methods by a great margin. An exception is only the church dataset, for which NHMC performs worse than hmAnt-Miner. As in Table 2, CLUS-HMC and NHMC results are comparable (we remind that these results are obtained with the FUN annotation schema and not with the GO annotation schema, where NHMC ouperforms CLUS-HMC of a great margin). Note that $AU\overline{PRC}$ [9] is similar to $\overline{AUPRC}$, but uses weights that consider the number of examples in each class - $AU\overline{PRC}$ is used here to make results easily comparable to results obtained with the other competitive methods.

Table 3. Comparison with other methods.The $AU\overline{PRC}$ of HMC-GA, HMC-LMLP, hmAnt-Miner and NHMC (using $\alpha = 0.5$ and the DIP PPI network), for 7 FUN annotated yeast datasets, as used in Cerri at al. [9].

Dataset	HMC-GA	HMC-LMLP	hmAnt-Miner	CLUS-HMC	NHMC_05
pheno	0.148	0.085	0.162	0.239	0.241
cellcycle	0.150	0.144	0.154	0.172	0.173
church	0.149	0.140	0.168	0.171	0.152
derisi	0.152	0.138	0.161	0.175	0.172
eisen	0.165	0.173	0.180	0.205	0.196
gasch2	0.151	0.132	0.163	0.195	0.186
spo	0.151	0.139	0.174	0.186	0.181

6 Conclusion

In this work, we tackle the problem of multi-label prediction when relationships among the classes (instances may belong to multiple classes and classes are organized into a hierarchy), as well as relationships among the instances (instances may be connected in network data) exist. The use of the latter relationships between the instances introduces autocorrelation and lead to violate the assumption that instances are independently and identically distributed (i.i.d.), which underlines most machine learning algorithms. The main contribution of this work is in the consideration of network autocorrelation in hierarchical multi-label classification (HMC). Specifically, in this work, we presented a definition of network autocorrelation in the HMC setting, introduced an appropriate autocorrelation measure for autocorrelation in such setting and developed the method NHMC for hierarchical multi-label classification from network data.

NHMC has been evaluated in the context of hierarchical gene function prediction in a PPI network context. Given a set of genes with known functions, NHMC learns to predict multiple gene functions when gene classes are hierarchically organized (and, possibly, in the form of DAGs), according to a hierarchical classification scheme (such as the MIPS-FUN and the Gene Ontology) and exploiting a given PPI network (such as DIP, VM and MIPS).

Due to the tree structure of the learned models, NHMC is able to consider the non-stationary effect of autocorrelation i.e., different effects of network autocorrelation at different levels of granularity. However, NHMC does not need the PPI network in the prediction phase, which is beneficial, especially in cases where the prediction needs to be made for new examples (genes) for which connections/interactions to other examples (genes) are not known or still need to be confirmed.

Empirical evidence shows that explicitly taking network autocorrelation into account increases the predictive capability of the models, especially when network interactions express information on the class labels. In future work, we intend to evaluate our approach by using additional datasets and networks, possibly considering new evaluation measures. Moreover, we intend to study a different mechanism to balance variance reduction and autocorrelation. Finally, we intend to evaluate how the sparseness of network information affects predictive capabilities of NHMC.

References

1. Ashburner, M., et al.: Gene ontology: tool for the unification of biology. The gene ontology consortium. Nature Genetics 25, 25–29 (2000)
2. Astikainen, K., Pitkänen, E., Rousu, J., Holm, L., Szedmák, S.: Reaction kernels - structured output prediction approaches for novel enzyme function. Bioinformatics, 48–55 (2010)
3. Barutcuoglu, Z., Schapire, R.E., Troyanskaya, O.G.: Hierarchical multi-label prediction of gene function. Bioinformatics 22(7), 830–836 (2006)
4. Batagelj, V., Mrvar, A.: PAJEK – Program for large network analysis (1998)
5. Bi, W., Kwok, J.T.: Multilabel classification on tree- and dag-structured hierarchies. In: Getoor, L., Scheffer, T. (eds.) ICML, pp. 17–24. Omnipress (2011)
6. Bilgic, M., Getoor, L.: Effective label acquisition for collective classification. In: Proc. 14th ACM SIGKDD Intl. Conf on Knowledge Discovery and Data Mining, pp. 43–51 (2008)
7. Ceci, M.: Hierarchical text categorization in a transductive setting. In: ICDM Workshops, pp. 184–191 (2008)
8. Ceci, M., Malerba, D.: Classifying web documents in a hierarchy of categories: a comprehensive study. J. Intell. Inf. Syst. 28(1), 37–78 (2007)
9. Cerri, R., Barros, R.C., de Carvalho, A.C.P.L.F.: A genetic algorithm for hierarchical multi-label classification. In: Proc. of the 27th Annual ACM Symposium on Applied Computing, SAC 2012, pp. 250–255. ACM (2012)
10. Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Incremental algorithms for hierarchical classification. J. Mach. Learn. Res. 7, 31–54 (2006)
11. Clare, A., King, R.D.: Predicting gene function in s. cerevisiae. In: Proc. Eur. Conf. on Computational Biology, pp. 42–49 (2003)
12. Deane, C.M., Salwiński, Ł., Xenarios, I., Eisenberg, D.: Protein interactions. Molecular & Cellular Proteomics: MCP 1(5), 349–356 (2002)
13. Doreian, P.: Network Autocorrelation Models: Problems and Prospects. In: Spatial Statistics: Past, Present, and Future. Monograph, vol. 12. Ann Arbor Institute of Mathematical Geography (1990)
14. Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C.: Using ghost edges for classification in sparsely labeled networks. In: Proc. 14th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining, pp. 256–264 (2008)
15. Jensen, D., Neville, J.: Linkage and autocorrelation cause feature selection bias in relational learning. In: Proc. 9th Intl. Conf. on Machine Learning, pp. 259–266. Morgan Kaufmann (2002)
16. Jensen, D., Neville, J., Gallagher, B.: Why collective inference improves relational classification. In: Proc. 10th Intl. Conf. on Knowledge Discovery and Data Mining, pp. 593–598 (2004)
17. Jiang, X., Nariai, N., Steffen, M., Kasif, S., Kolaczyk, E.: Integration of relational and hierarchical network information for protein function prediction. BMC Bioinformatics 9(1) (2008)
18. Kong, X., Shi, X., Yu, P.S.: Multi-label collective classification. In: SDM, pp. 618–629. SIAM/Omnipress (2011)
19. Macskassy, S., Provost, F.: Classification in networked data: a toolkit and a univariate case study. Machine Learning 8, 935–983 (2007)
20. Macskassy, S.A.: Improving learning in networked data by combining explicit and mined links. In: Proc. 22nd Intl. Conf. on Artificial Intelligence, pp. 590–595 (2007)

21. Mewes, H.W., Heumann, K., Kaps, A., Mayer, K., Pfeiffer, F., Stocker, S., Frishman, D.: Mips: A database for protein sequences and complete genomes. Nucl. Acids Res. 27, 44–48 (1999)
22. Neville, J., Jensen, D.: Relational dependency networks. Journal of Machine Learning Research 8, 653–692 (2007)
23. Rahmani, H., Blockeel, H., Bender, A.: Predicting the functions of proteins in protein-protein interaction networks from global information. Journal of Machine Learning Research 8, 82–97 (2010)
24. Re, M., Valentini, G.: An experimental comparison of hierarchical bayes and true path rule ensembles for protein function prediction. In: El Gayar, N., Kittler, J., Roli, F. (eds.) MCS 2010. LNCS, vol. 5997, pp. 294–303. Springer, Heidelberg (2010)
25. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Kernel-based learning of hierarchical multilabel classification models. J. Mach. Learn. Res. 7, 1601–1626 (2006)
26. Ruepp, et al.: The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucleic Acids Research 32(18), 5539–5545 (2004)
27. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 3, 93–106 (2008)
28. Steinhaeuser, K., Chawla, N.V., Ganguly, A.R.: Complex networks as a unified framework for descriptive analysis and predictive modeling in climate science. Statistical Analysis and Data Mining 4(5), 497–511 (2011)
29. Stojanova, D., Ceci, M., Appice, A., Džeroski, S.: Network regression with predictive clustering trees. Data Mining and Knowledge Discovery 14 (2012)
30. Valentini, G.: True path rule hierarchical ensembles for genome-wide gene function prediction. IEEE ACM Transactions on Computational Biology and Bioinformatics 8(3), 832–847 (2010)
31. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning 73(2), 185–214 (2008)
32. von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S.G., Fields, S., Bork, P.: Comparative assessment of large-scale data sets of protein-protein interactions. Nature 417(6887), 399–403 (2002)