1-1-2012

Hierarchical multi-label classification for protein function prediction going beyond traditional approaches

Noor Al aydie
Wayne State University,

HIERARCHICAL MULTI-LABEL CLASSIFICATION FOR PROTEIN FUNCTION PREDICTION GOING BEYOND TRADITIONAL APPROACHES

by

NOOR ALAYDIE

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2012

MAJOR: COMPUTER SCIENCE

Approved by:

Advisor Date

DEDICATION

To my beloved parents,

Naser Alaydie & Farah Naami

To my best friend and husband,

Omar Odibat

To my two little angels,

Sarah Odibat & Hala Odibat

ACKNOWLEDGMENTS

All thanks and praise are due to Allah for these favors from beginning to end. This dissertation would not have been possible without the support of many people, and it is a pleasure to thank all those who made it possible. I would like to express my gratitude to my advisor, Farshad Fotouhi, for all of his support, patience and guidance throughout my graduate studies and for making so much time for me in his busy schedule. I would also like to thank my co-advisor, Chandan Reddy, whose expertise and understanding, added considerably to my graduate experience. I appreciate their vast knowledge, patience and criticism while allowing me to work in my own way. Deepest gratitude is also given to the members of my committee, Zaki Malik, Shiyong Lu and Jeffery Loeb for the assistance they provided at all levels of the research project. All three of them contributed greatly to this research.

My time at Wayne State University was made enjoyable in large part due to the many friends and groups that became a part of my life. I am grateful for time spent with friends, for our memorable trips and for many other people and memories. Thanks to all my other friends whose friendship is very valuable for me. Not forgetting my best friends Sandra Shoukair and Manal Alziq for always being there for me.

I would also like to express my love and gratitude to my family for all their support and encouragement they provided me through my entire life. My wonderful parents deserve special mention for their inseparable encouragement and prayers. I am gratefully for the support I have received from my father Naser Alaydie and my mother Farah Naami, who sincerely were my first teachers, and supported me in all my pursuits. I owe my mom everything and wish I could show her just how much I love and appreciate her. Thanks to my brothers and sisters for being supportive and caring siblings.

Words fail me to express my appreciation to my husband and best friend, Omar Odibat, without whose love, support, encouragement, patience and editing assistance, I would not have finished this thesis. His faithful support and persistent confidence in me during the final stages

of this Ph.D. is so appreciated. I would also like to thank Omar's mother for raising him and for being so patient and supportive during the few past years. Special thanks go to my angels Sarah, who was there at the beginning of the project, and Hala, who joined us near its end, for being so patient while having their mom doing research and producing this piece of work.

Finally, I would like to thank everybody who was important to the successful realization of the thesis and apologize that I could not mention personally one by one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Classification is a challenging problem that encompasses several diverse domains, and is defined as the task of assigning data objects to one or several predefined classes. For instance, in biomedical domain, cells are classified as malignant or benign, based on the results of MRI scans. In text classification, documents are classified into a number of topics based on their textual context. In functional genomics, genes and proteins are associated with several functional labels simultaneously.

Given a set of training examples, characterized by the tuples $(x_i, y_i), i = 1, ..., n$, where n is the total number of training examples, x_i is the attribute set and y_i is the class label (also known as target attribute), the goal of classification is to learn a target function $f(x)$, that maps each attribute set, x_i, to its associated class label y_i. The target function is also known as a classification model. A classification model can be used as a descriptive model to distinguish between objects of different classes. It can also be used to predict the class labels of unknown examples.

This dissertation presents novel algorithms for hierarchical multi-label classification problem in the context of protein function prediction. Several issues have to be considered. First, each instance can be associated to more than one label at the same time (*multi-label classification*). Second, the labels are related to each other and their mutual relations are expressed by rooted trees or directed acyclic graphs (*hierarchical classification*). Third, the labels and the instances could have unrevealed correlations that are not expressed through the hierarchical structure (*label-label correlation and instance-instance correlation*). Fourth, the predictions found by a classification model should be consistent with the hierarchial relations between the labels (*hierarchy constraint*). Fifth, multiple heterogeneous data sources exist, that describe the instances differently (*heterogeneous multi-source integration*).

Multi-label classification is a challenging problem in many real-world application domains. For instance, in text categorization, a document may be categorized into multiple categories: technology, science and computers. A medical patient may have more than one health condition: high blood pressure, obesity and diabetes. Semantic scene classification may result in several semantic classes: mountain, sky, ocean and sunset. A CNN news report can be tagged as people and economy at the same time. A movie can be categorized as action, adventure and thriller. A gene may be annotated with several functional labels in functional genomics. This dissertation was motivated by data in the fields of bioinformatics and functional genomics.

Hierarchical multi-label classification is attracting much research attention in recent years. There is an increasing need for hierarchical multi-label classification in several application domains such as text and music categorization, functional genomics, and image and video annotation. The particular application focused in this dissertation is protein function prediction: given an unseen instance (gene product), the goal is to predict its function label set based on a predefined class hierarchy such as FunCat (Functional Catalogue tree structure) taxonomy [101]. This is a central problem in modern biology since many proteins are still unannotated although a large number of them have been identified. For instance, 40% of the proteins encoded in eukaryotic genomes have not been functionally characterized [53].

1.1 Motivation

1.1.1 Multi-label Classification

Traditional classification tasks deal with assigning instances to a single label. In multi-label classification, the task is to associate the data object to a set of labels that data object can belong to rather than associating it with a single label. Traditional binary-class problem can be viewed as a special case of multi-label classification. Multi-label classification problems arise in several application domains such as text classification, image classification, and gene function prediction [31].

Many proposed methods tackle the multi-label classification problem by transferring the multi-label classification problem into a set of single-label classification problems and applying the traditional single-label classification algorithms [33, 15]. An obvious disadvantage of these methods is that they ignore the correlation between the labels. On the other hand, many methods suggest the adaptation of the conventional classification algorithms in order to deal directly with the multi-label problem [26, 104].

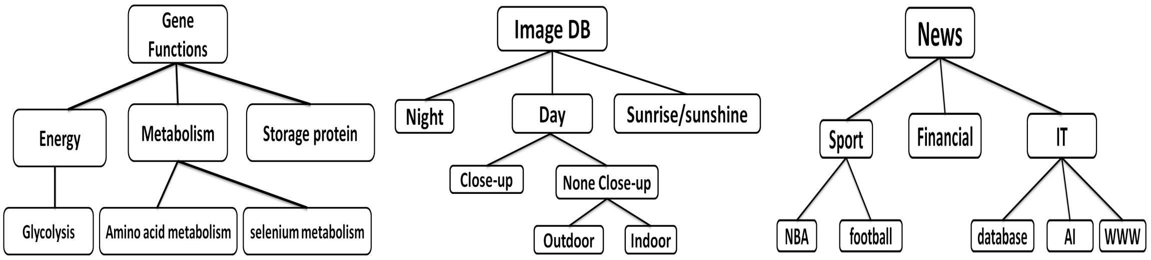
1.1.2 Hierarchical Classification

Hierarchical multi-label classification is a variant of traditional classification where the task is to assign the data objects to a set of labels where the labels are related through a hierarchical classification scheme. In other words, when an instance is labeled with a certain class, it should also be labeled with all of its superclasses. Organizing the classes in a hierarchical scheme is useful when the number of classes is large. For example, text documents, images and genes can be organized in conceptual hierarchies such as the one shown in Figure 1.1. Hierarchical classification can help in increasing the organization and the accessibility of databases in several real-world domains.

Hierarchical multi-label classification has received increasing attention in recent years due to its practical relevance in many domains. In hierarchical multi-label classification, the training set consists of instances, each of which is associated with a set of labels that are organized according to a predefined hierarchy (For example, GO or FunCat in the functional genomics setting) [6]. The goal is to predict the label sets of unseen instances by analyzing the training instances with known label sets.

1.1.3 Label-Label Correlation and Instance-Instance Correlation

Most of the state-of-the-art hierarchical multi-label classification methods are designed to focus mainly on taking into account the hierarchical structure of the classes by obeying the hierarchy constraint for the final predictions [11, 80, 100]. However, no attempts were made to explore the relations among the labels or the instances that are not expressed by the predefined class hierarchy. In other words, most of the existing approaches focus on learning models



(a) Gene functions (b) Images categorization (c) Text classification

Figure 1.1: Examples of Hierarchical classification in various domains where the hierarchical structure is a tree.

independently for each label. Additionally, the majority of the existing approaches for the protein function prediction problem (the application we focus in this dissertation) perform additional steps to correct inconsistent predictions.

We attempt to make a fundamental advancement in this area by explicitly discovering and exploiting new relationships among the classes, that are not shown through the hierarchical taxonomy, and making use of such knowledge in the learning process. By leveraging on the hierarchical structure of the classification scheme, the classifiers can be more efficient [40].

1.1.4 Integration of Heterogenous Data Sources

Several types of data sources can be used for the prediction task. The recent advances in high-throughput technologies, such as the DNA microarray data, has enabled measuring the expression level of thousands of genes simultaneously. Protein-protein interaction (PPI) and genome sequences are two other important data sources [23]. In this work, various kinds of datasets will be used in evaluating the task of hierarchical gene function prediction.

Learning reliable classification models for several hierarchically structured functional classes from a single dataset is often hard due to several complex issues including noise in the data, low relevance of the dataset for some functional classes and an insufficient number of training examples for building accurate classification models [89, 77].

These issues, especially the last one, are expected to most severely affect the classification models for functional classes having few members, particularly classes located deep in a functional hierarchy and leaf classes.

However, it is surprising to see that only little has been done in terms of integration of multiple data sources when the classes are controlled by a hierarchical relations scheme. This observation motivates this dissertation. Integrating data from multiple sources can provide valuable knowledge that can be used to improve the prediction results. From the perspective of application domain, this dissertation will focus on "*protein function prediction*", an important application in the bioinformatics field.

1.2 Challenges and Research Objectives

The goal of this dissertation is the development of a hierarchical multi-label classification framework that improves the classification performance. In order to accomplish this objective, the work in this dissertation is divided into three main tasks.

The first task focuses on developing a hierarchical multi-label classification approach and compare it with the flat classification approach. Moreover, different testing strategies are examined in this task, namely, the top-down strategy and the bottom-up strategy. This task poses the following challenges:

1. **Multi-label classification:** an instance may belong to multiple class labels at the same time. There is a need for a prediction algorithm that is able to identify all the possible labels of a particular example.

2. **Hierarchical taxonomy:** classes are related to each other in a tree or a graph structure. The goal is to discover the specific unknown functions for an example rather than general functions. This challenge includes two more challenges:

   (a) ***Hierarchy constraint:*** the hierarchy constraint states that *annotating an example to a given class is automatically transferred to all of its ancestors*. The predictions for unknown instances must follow the hierarchy constraint.

   (b) ***Classes imbalance:*** as the specificity of the classes increases, fewer number of examples are annotated to the more specific classes.

(c) ***Non-mandatory leaf node prediction:*** the proposed prediction algorithm allows the predictions to stop at any level in the hierarchy.

The objective of this task is to develop an algorithm that predicts protein functions in a computationally efficient manner.

The second task is to extend the developed hierarchical multi-label classification algorithm to exploit the hidden correlations among the labels and the instances. A major objective here is to compare the performance of the hierarchical classification when utilizing the knowledge about the hierarchical relationships among the classes and the discovered new unrevealed relationships among other classes, that are not shown through the pre-established hierarchical structure of the classes, against flat classification and the local classifier approach. In addition to the above challenges, the following challenges are addressed:

1. ***The label-label correlation:*** the label-label correlation is modeled and utilized in building the classification model.

2. ***The instance-instance correlation:*** the instance-instance correlation is exploited and used in building the classification model.

3. ***Shared learning models:*** find, share and combine a number of models across the neighbor labels.

In the third task, the integration of multiple heterogenous data sources is explored. In addition to the challenges that are addressed in the first task, the following challenges are also being addressed:

1. ***Multi-source integration:*** functionally similar examples may have huge diversity in their measurements through various datasets. Each data source may provide information about the characteristics of genes to be classified from a different view. Moreover, one source can be useful to learn a specific functional class while being irrelevant to other classes.

2. ***Computational cost:*** the inter-relationships among the classes, shown through the hierarchical structure, can be exploited for reducing the computational cost for learning large amount of gene function classifiers, when multiple heterogeneous data sources exist.

To make this thesis complete, a weighting scheme of the integration framework of the heterogenous data sources in which different weights are associated with different data sources is also proposed. The weights are adaptable in the sense that they express the relevance of each data source to the label under-study.

Considering all the above issues, the problem of accurately predicting the gene functions from multiple sources is a challenging problem. The dissertation seeks to address the above challenges in a generalized framework. To the best of our knowledge, there is no existing work that handles all of the above challenges in a single framework.

1.3 Summary of Contributions

Hierarchical multi-label classification problem can be viewed as an extension of the binary classification problem where an instance can be associated with multiple classes that are related through a hierarchical categorization scheme. Figure 1.2 shows an overview of the proposed framework. To address the challenges of the first part of this dissertation and to improve the hierarchical protein function prediction, we propose a novel hierarchical multi-label classification algorithm to efficiently predict protein functions. The developed algorithm is called "***HML-Boosting***" algorithm. The contributions of this algorithm can be summarized as follows:

1. Propose a class-membership consistency correction method in the model building process rather than having it as a separate step.

2. Study the effects of the different testing strategies, the top-down approach and the bottom-up approach, on the prediction results.

3. Compare the performance of the flat approach with the local approach of the hierarchical

multi-label classification.

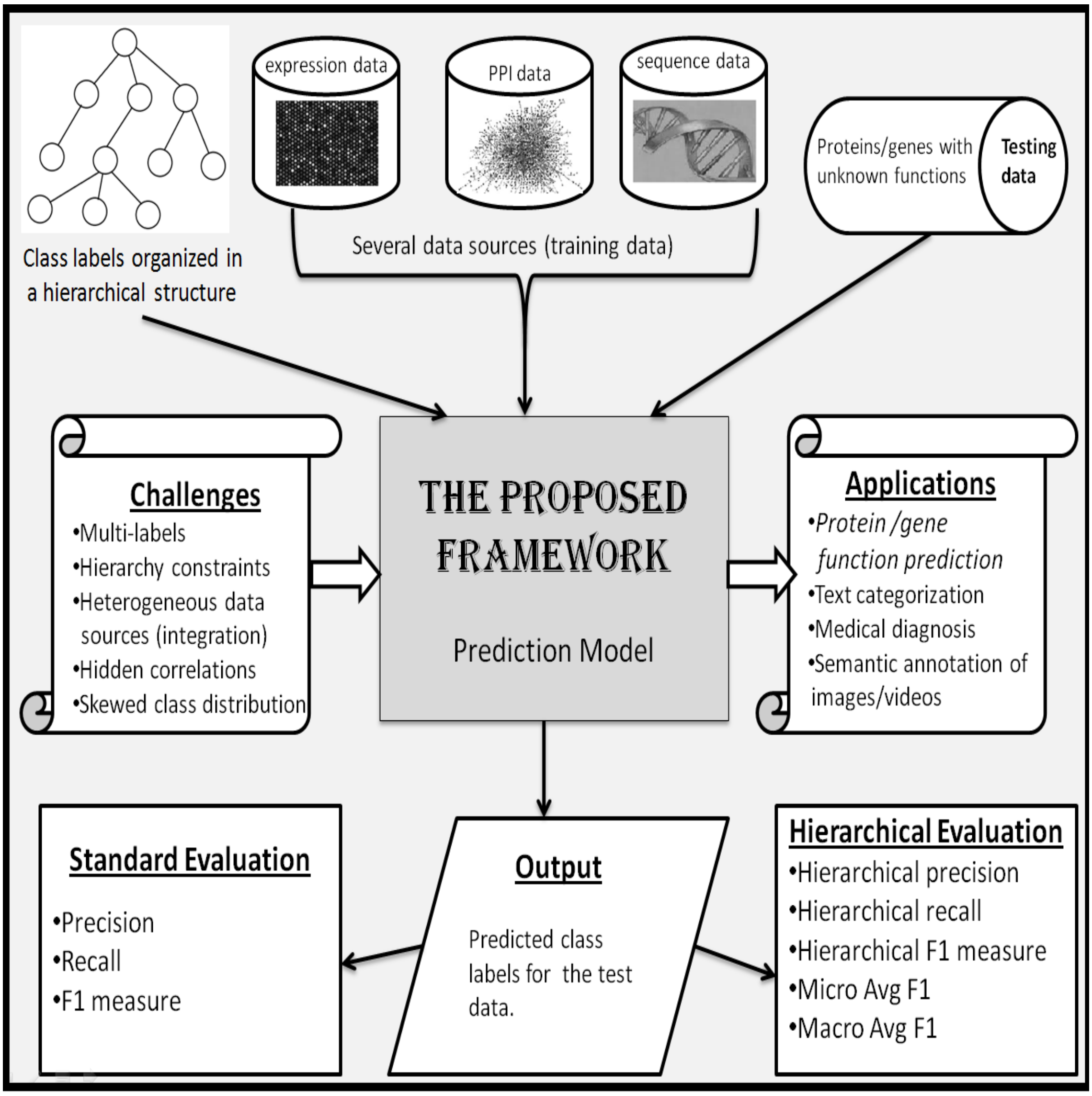


Figure 1.2: An overview of the proposed framework.

The second developed system is called “***HiBLADE***”, a hierarchical multi-label classification framework for incorporating information about the hierarchical relationships among the labels as well as the label and instance correlations. The experimental results showed that the proposed algorithm outperforms the flat classification method and the local approach that builds an independent classifier for each class without considering the correlation between the classes. The main contributions of this algorithm are summarized as follows:

1. The underlying pre-defined taxonomy of the labels is explicitly expressed by adding the ancestor labels to be part of the feature vector, which allows us to gain further insights into the learning problem.

2. It is capable of addressing the correlations and the mutual interdependencies among the children of a particular label using the Bayesian framework and instance-based similarity.

3. The use of a shared boosting model for the children labels for each label in the hierarchy using the found correlations leads to efficient and effective results.

More importantly, the integration of multiple heterogeneous data sources in the context of hierarchical multi-label classification is presented in the third part of the dissertation. In essence, a novel algorithm for heterogeneous multi-source integration for hierarchical multi-label classification is proposed to efficiently exploit different data sources in the learning of the classification models. This algorithm is called the "***HiBiN***" algorithm. The main contributions of the HiBiN algorithm are summarized as follows:

1. To predict multi-labels for genes, the HiBiN allows the prediction of more than one functional class.

2. To maintain the hierarchy constraint, the parent-child mutual inter-relationships are exploited during the training and the testing phases.

3. To handle the imbalance class problem, the positive and negative examples for each classifier are chosen based on the hierarchical taxonomy of the classes.

4. To handle the problem of source diversity, the proposed framework integrates multiple heterogeneous data sources, such as protein-protein interaction and gene expression data, to characterize the gene functions effectively.

5. To minimize the computational cost for training the classifiers, HiBiN filters out unsuitable genes from percolating to lower levels in the hierarchy.

Our experiments showed that the integration can help improve the performance of the standard classification-based protein function prediction algorithms. This inspired us to study this approach and to propose an adaptable weighting scheme for the different data sources. The extended weighting scheme-based algorithm of the HiBiN algorithm (***wHiBiN***, where *w* stands for weighting scheme-based algorithm) is presented lastly in the dissertation.

1.4 Organization

The organization of this dissertation is as follows: Chapter 2 presents a survey of the related work and the performance criteria. Chapter 3 briefly explains the motivation for the work presented in this dissertation and describes the problem of protein/gene function prediction. Chapter 4 presents the HML-Boosting algorithm which handles the hierarchical multi-label classification problem from a single data source. Chapter 5 presents our second algorithm, HiBLADE algorithm, which handles the problem of capturing label-label and instance-instance correlations. Chapter 6 presents our third algorithm, HiBiN algorithm, which handles hierarchical multi-label classification from heterogenous data sources. Chapter 7 presents the extended version of HiBiN, wHiBiN algorithm, which includes a novel adaptable weighting scheme for integrating the results from several data sources. Finally, Chapter 8 concludes the dissertation and presents possible directions for future research work.

CHAPTER 2

PROTEIN FUNCTION PREDICTION

Functional classification of genes/proteins using diverse bio-molecular data obtained from high-throughput technologies is a fundamental problem in bioinformatics and functional genomics [95, 81]. The study of gene/protein functions in functional genomics is performed on a large scale by conducting parallel analysis of gene expression for a large number of genes [55, 73]. An individual gene or protein may be involved in more than one biological activity, and hence, there is a need for a prediction algorithm that is able to identify all the possible functions of a particular protein. Recently, diverse high-throughput functional genomic data have become broadly available such as protein-protein interaction, gene expression microarrays and genome sequences [53, 75]. For example, DNA microarrays allow researchers to measure the expression levels of thousands of different genes simultaneously, producing overwhelming amounts of data [109]. Since the functions of significant number of genes are still unknown, such high throughput data can play a key role in assigning accurate functional annotations on a large-scale through computational prediction models.

2.1 Overview

The need to understand the functional roles of genes is central in modern biology for several reasons ranging from general knowledge to the development of targeted medicine diagnostics [57]. Moreover, assigning functions to genes and proteins is essential to understand the molecular and biochemical processes that sustain health or cause disease [83]. In a gene function prediction setting, a gene may be associated with multiple biological functions. Recently, the advancements in genome sequencing has lead to the generation of a number of novel putative genes and hypothetical proteins [83]. The goal of functional classification of genes is to predict the biological functions of these genes and proteins. Moreover, discovering the unknown functions for a given protein is critical in understanding several diseases and in identifying and validating novel drug targets [83].

Providing accurate predictions can advance experimental studies by providing specific hypothesis for targeted experimental testing [117, 39]. A popular method used for predicting gene function from biological data is classification. Each gene is represented by a set of features, such as the set of proteins that the gene product interacts with or its gene expression profile. This is often achieved by an automated prediction process that interacts with laboratory experiments [11, 37]. Several approaches apply machine learning techniques to predict gene functions from a predefined set of functions have been proposed in the literature [11, 100, 121, 128]. Predictions with the highest confidence are taken to the lab for testing [105]. Other approaches predict functions of unannotated genes using the known functions of genes that are nearby in a functional association network or protein-protein interaction network [117, 115, 82, 64, 63, 20, 24].

In the context of gene function prediction, with the availability of data from different biological sources, assigning biological functions to genes is a challenging task in functional genomics [22]. First, there are many functional classes which may be related to each other in a tree such as Funcat or a graph structure such as Gene Ontology(GO). Second, a gene may have multiple class labels. Biologically, a gene may be involved in more than one biological activity. Third, learning reliable classification models for several hierarchically structured functional classes from a single dataset is often hard due to several complex issues including noise in the data, low relevance of the dataset for some functional classes and an insufficient number of training examples for building accurate classification models [89].

A number of algorithms have addressed the first two points above [11, 106, 57, 121]. However, most of these methods predict gene functions in a “flat” fashion without employing the ontological structures among the classes. Moreover, these methods require an additional step to correct inconsistent predictions, rather than producing them directly in a coherent way. Furthermore, the integration of multiple data sources while supporting the hierarchical multi-label classification is not being considered in the existing literature.

2.2 Hierarchical Taxonomies of Protein Functions

There are two main types of class hierarchy structures, namely, a rooted-tree structure and a directed acyclic graph (DAG) structure. A typical example of a class hierarchy for gene function prediction that is structured as a DAG is the Gene Ontology (GO) [28], and a typical example of a class hierarchy that is structured as a rooted tree is the MIPS's (The Munich Information Center for Protein Sequences) FunCat (Functional Catalogue tree structure) taxonomy [101].

Functional classification schemes, such as Gene Ontology (GO) and MIPS's FunCat taxonomy, are the source of gold standard information for computational efforts at supervised gene function prediction [89]. The definitions of the functional classes come from biochemical and genetic studies of gene function [74]. The arrangement of the classes in a directed acyclic graph (DAG) representation allows the capturing of rich inter-relationships between different gene functions [89]. In GO, each node represents a functional group [132]. In the context of FunCat, where the existence of a hierarchical structure of classes is expressed as a tree, each class is subdivided into more specific classes, and these, in turn, are subdivided again and again until the most specific functions are reached [80]. According to the "true path rule" [80, 120] that governs the annotation of both GO and FunCat taxonomies, annotating a gene to a given class is automatically transferred to all of its ancestors to maintain the hierarchy constraint. In this dissertation, we focus on FunCat as the hierarchical scheme used for protein function prediction.

As an example, Table 2.1 shows a small portion of the FunCat taxonomy. In this Table, *Metabolism, Energy and Cell Cycle and DNA Processing* are considered as general functional classes, while *regulation of glycolysis and gluconeogenesis* is a more specific class. Figure 2.1(a) shows the hierarchical tree structure of the sample FunCat classes that are shown in Table 2.1. As will be discussed later, flat classification approach ignores any predefined inter-relationships between the classes. In these schemes, the nodes are the labels (classes) which are connected to other nodes through parent-child edges, that impose hierarchical inter-relationships between them. The nodes contain member genes that have been annotated with the corresponding functional class.

Table 2.1: A small sample of FunCat hierarchical classification scheme.

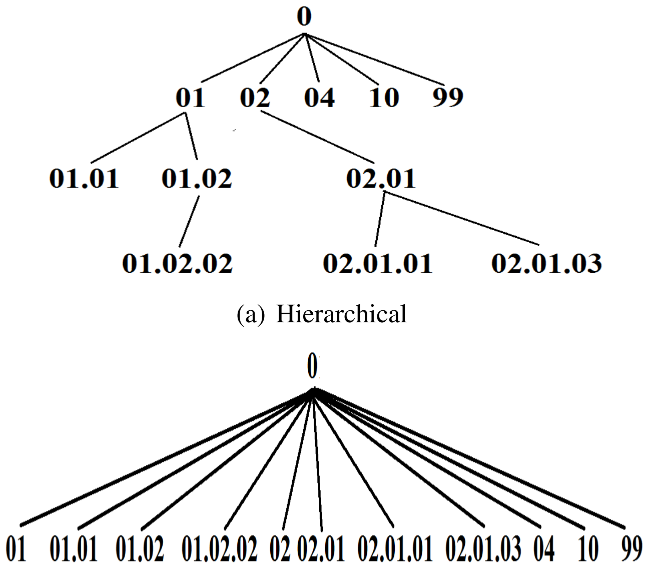
01 METABOLISM
01.01 amino acid metabolism
01.02 nitrogen, sulfur and selenium metabolism
01.02.02 nitrogen metabolism
02 ENERGY
02.01 glycolysis and gluconeogenesis
02.01.01 glycolysis methylglyoxal bypass
02.01.03 regulation of glycolysis and gluconeogenesis
04 STORAGE PROTEIN
10 CELL CYCLE AND DNA PROCESSING
...
99 UNCLASSIFIED PROTEINS

Several types of data sources can be used for the prediction task. The recent advances in high-throughput technologies, such as the DNA microarray data, has enabled measuring the expression level of thousands of genes simultaneously. Protein-protein interaction (PPI) is another important data source. In this work, various kinds of datasets will be used in evaluating the task of hierarchical gene function prediction. For example, we used two types of protein domain data, gene expression data, predicted and experimentally supported protein-protein interaction data and pairwise sequence similarity data.

2.3 Motivating Challenges

The functional classification of proteins is an important and challenging problem in the field of functional genomics due to the following issues:

1. **Multi-label:** a gene may have multiple class labels. Biologically, a gene may be involved



(a) Hierarchical

(b) Flat

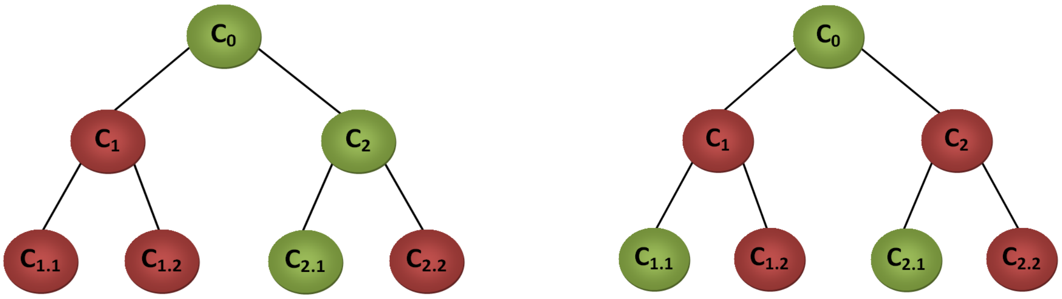
Figure 2.1: (a) A tree structure representation for the classes in Table 2.1 (b) A flat representation for the same set of classes.

in more than one biological activity. Therefore, there is a need for a prediction algorithm that is able to identify all of the possible functions of a particular gene.

2. **Hierarchical taxonomy:** functional classes are related to each other in a tree or a graph structure (Gene Ontology(GO) [28] or MIPS's FunCat taxonomy [101]). The goal is to discover the specific unknown functions for a gene rather than the general functions.

   (a) ***Hierarchy constraint:*** the hierarchy constraint, also known as true path rule, governs the annotation of both GO and FunCat taxonomies. It states that *annotating a gene to a given class is automatically transferred to all of its ancestors.* This constraint should be maintained such that a gene can be annotated to a class only if it is annotated with the parent class as shown in Figure 2.2. As shown in the figure 2.2(b), a gene is annotated with classes $C_{1.1}$ and $C_{2.1}$ but not with classes C_1 and C_2, though the latter classes are parent classes of the former ones, respectively.

   (b) ***Classes imbalance:*** as the specificity of the functional classes increases, less num-



(a) Consistent with the hierarchy constraint (b) Violation of the hierarchy constraint

Figure 2.2: An example of the hierarchy constraint. The nodes are the class labels and the edges represent parent-child relationships. The green nodes are those which the classifier predicts to be positive, while red ones represent the classes where the classifier predicts to be negative. In (a) the predictions are consistent with the class hierarchy, while in (b), the predictions are inconsistent with the class hierarchy because the example is annotated with classes $C_{1.1}$ and $C_{2.1}$ but not with classes C_1 and C_2, though the latter classes are parent classes of the former ones, respectively.

ber of genes are annotated to the more specific functions. The classification models for functional classes having few members are expected to severely be affected. An example of this problem is illustrated in Figure 2.3. For simplicity, we are showing here one path of the functional classes, from the root to the leave. In this example, the most general gene function, *Transcription*, has 474 gene members, while the most specific class, *Transcription initiation*, has only 23 gene members. Going from the general function to the more specific ones, the number of member genes is reduced. This is because the set of genes that are annotated with a non-root function is a subset of the set of genes that belongs to the parent of that function. Therefore, we need an algorithm that takes the classes imbalance issue into consideration.

3. **Multi-source integration:** handling the problem of huge diversity in genomic data effectively is an issue that has limited attempts in the literature. Functionally-similar genes may have huge diversity on their measurements through various datasets. Each data source may provide information about the characteristics of genes to be classified from a different view. Moreover, one source can be useful to learn a specific functional class

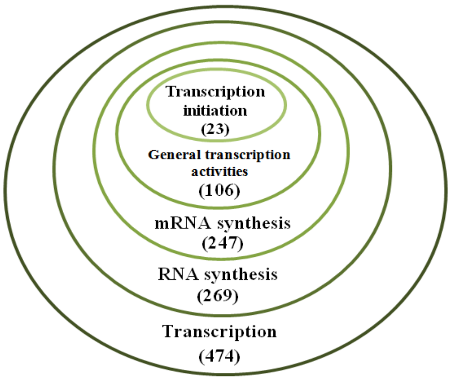

Transcription initiation (23)
General transcription activities (106)
mRNA synthesis (247)
RNA synthesis (269)
Transcription (474)

Figure 2.3: An example of the insufficient number of training examples for building accurate classification models. In this example, five FunCat functional classes that are related through a parent-child relationship with varying sizes are shown.

while being irrelevant to other classes. One promising solution for the problem of huge diversity of genomic data is to use an integrated analysis framework. Integrating data from multiple sources can provide valuable knowledge that can be used to improve the prediction results.

4. **Computational cost:** the inter-relationships among the classes, shown through the hierarchical structure, can be exploited for reducing the computational cost for learning large amount of gene function classifiers. The dependencies between the gene functions can be exploited for improving gene function classifier training and reducing the computational cost.

Considering all the above issues, the problem of accurately predicting the gene functions is a challenging problem. To the best of our knowledge, there is no existing work that handles all of the above challenges in a single framework. The goal of our work is to tackle these problems in a generalized framework.

2.4 Biomolecular Data Sources

We chose to demonstrate the performance of the proposed algorithms, namely, the HML-Boosting, the HiBLADE, the HiBiN and the wHiBiN algorithms for the prediction of gene functions in yeast using four bio-molecular datasets that were used in [118]. In this section, we describe the four common biomolecular datasets that are used by the four algorithms. In Chapter 6, we show two more datasets that are used for showing the effectiveness of the data integration algorithms. Valentini [118] pre-processed the datasets in such a way that for each dataset, only genes that are annotated with FunCat taxonomy are selected. To make this dissertation self-contained, we briefly explain the data collection process and the pre-processing steps performed on the data. Uninformed features that have the same value for all of the examples are removed. Class "99" in FunCat corresponds to an "unclassified protein". Therefore, genes that are annotated only with that class are excluded. Finally, in order to have a good size of positive training examples for each class, selection has been performed to classes with at least 20 positive examples. Dataset characteristics are summarized in Table 2.2. For the integration purposes, we used two additional data sources and we performed additional pre-processing steps, which are described in Chapter 6.

Table 2.2: The characteristics of the four bio-molecular datasets used in our experiments for the HML-Boosting and HiBLADE algorithms.

Dataset	Description	Samples	Features	Classes
Gene-Expr	Gene expression data	4532	250	230
PPI-BG	PPI data from BioGRID	4531	5367	232
Pfam-1	Protein domain binary data	3529	4950	211
PPI-VM	PPI data from Von Mering experiments	2338	2559	177

The gene expression dataset, Gene-Expr, is obtained by merging the results of two studies, gene expression measures relative to 77 conditions [111] and transcriptional responses of yeast to environmental stress measured on 173 conditions [44]. For each gene product in the protein-protein interaction dataset, PPI-BG, a binary vector is generated that implies the presence or absence of protein-protein interaction. Protein-protein interaction data have been downloaded from BioGRID database [112, 119]. Several computational methods were developed to interpret protein interaction networks for many model species to elucidate protein functions [45, 108].

In Pfam-1 dataset, a binary vector is generated for every gene product that reflects the presence or absence of 4950 protein domains obtained from Pfam (Protein families) database [32, 118]. For PPI-VM dataset, Von Mering experiments produced protein-protein data from yeast two-hybrid assay, mass spectrometry of purified complexes, correlated mRNA expression and genetic interactions [124].

CHAPTER 3

LITERATURE REVIEW

Hierarchical multi-label classification deals with the problem of learning multiple class labels that are in turn related through a hierarchical structure. Recently, many approaches have been developed to tackle the hierarchical multi-label classification challenge, often motivated by real-world applications such as text categorization and protein function prediction. For example, web repositories and digital libraries are organized in hierarchical categories. Examples include LookSmart [36], Yahoo! [78] and Reuters [69]. In bioinformatics, a typical multi-label learning example is the gene function prediction problem where a gene can be associated with multiple functional classes [5]. Assigning biological functions to genes is a key challenge in modern biology. This discovery process is usually interacting with laboratory experiments [106]. In other words, after applying machine learning techniques for the prediction of protein functions using a pre-defined taxonomy of the functions, predictions with the highest confidence are taken for further biological analysis in the lab [106].

This chapter gives the necessary background on the work related to hierarchical multi-label classification. First, we discuss the state-of-the-art classification approaches that can used to solve the standard classification problems with two classes, including boosting classifier which is used as the baseline method in our work. Then, we present a formal problem statement of a hierarchical multi-label classification. The general approaches that deal with the problem and the related works are discussed next. Finally, the evaluation metrics that are commonly used by the researchers for evaluating their approaches as well as the evaluation metrics that are tailored to the hierarchical multi-label classification problem will also be explained.

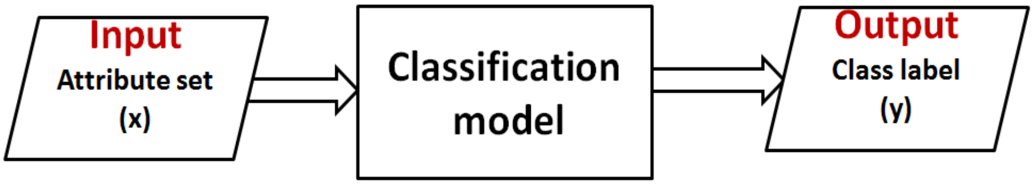
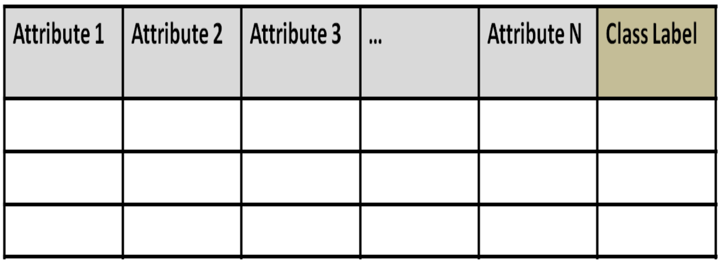


Figure 3.1: Overview of a classification model.

3.1 Traditional Classification Approaches

Traditional classification is the task of assigning objects to one of a predefined labels. It is a supervised learning approach that uses prior knowledge of classes and labeled objects in predicting the class labels for new objects [8]. Each object has a set of attributes that describe the properties of the corresponding object. The classification model is used to predict the class labels of unknown objects. Additionally, the classification model can be considered as a descriptive model that identifies the most discriminative features that can be used to distinguish between the objects of different classes. An overview of a typical classification model is shown in Figure 3.1.



Attribute 1	Attribute 2	Attribute 3	...	Attribute N	Class Label

Figure 3.2: The structure of the classification data.

Mathematically, classification can be defined as the task of learning a target function that maps each attribute set x to one of the predefined class labels y [114]. The classification data is composed of two parts: the attribute sets (x) and the class labels (y) and can be organized in a table as shown in Figure 3.2. Several classification algorithms have been proposed to predict the class labels from the training data [114]. Next, we describe the most well-known classification techniques.

3.1.1 Decision Tree

Decision tree is a data mining algorithm that recursively partitions a set of records until all the data items belong to a particular class [8]. There are three components of a decision tree: root, internal nodes and leaf nodes [94]. In a decision tree, the class labels are represented as leaf nodes, and the remaining nodes in the tree represent test conditions on some attributes [93, 16]. The Decision trees have been used to solve various classification problems such as protein function prediction [121, 27]. Constructing the decision trees can be done using several algorithms such as the Hunts algorithm [54].

3.1.2 Nearest-Neighbor

The nearest-neighbor technique is a simple classification method that assigns a label to an unclassified object [30, 35]. The idea of this method is to find all the training objects that are similar to a given test object [114, 13]. These objects are referred to as the nearest neighbors and are used to assign the class label for the test object. Basically, the similarity between the test object and all the training objects are computed to find the nearest neighbors. The similarity is computed using some distance function such as the Euclidean distance. Using K nearest neighbors (KNN) method, a majority voting scheme is applied in assigning the class labels to the test object [72]. Usually, K is set to an odd number to avoid ties. The main characteristic of the nearest neighbor classification technique is that it does not require model building [114]. The nearest neighbor method has been used in many applications including text categorization and protein function prediction [48, 131].

3.1.3 Support Vector Machine (SVM)

Using the support vector machine, a hyperplane (a decision boundary) is constructed and used for the classification [29, 58]. This hyperplane is constructed to achieve a good separation between two classes [50]. In this case, the classifier is referred to as maximal margin classifier. The decision boundary can be linear or non-linear decision boundary by transforming the training data into a new space. In addition, SVM classifiers can solve binary or multi-class classification problems [114].

3.1.4 Artificial Neural Network (ANN)

The intuition behind using the artificial neural network classifier is to simulate the human brain. The classification model is composed of neurons that are linked together at different layers including input nodes and output nodes [52]. The input nodes (units) are connected to the output nodes using weighted links. This structure of ANN is referred to as perceptron [42]. The classification is performed by calculating the weighted sum of the inputs and subtracting a bias factor. The sign of the result determines the class label $\{0, 1\}$. The weights are computed from the training data during the learning process. The ANN can have multi-layers: input, hidden and output [43]. When the number of the nodes in the hidden layer is large, the training process becomes computationally expensive [114].

3.1.5 Boosting

Boosting is a recent and a powerful learning method that was designed to solve various classification problems. The main idea of boosting is to combine the classification results of many $weak$ classifiers. Weak classifiers are better than random guessing [49]. The most popular boosting algorithm is the AdaBoost [102, 103]. It is an iterative algorithm in which weak classifiers are repeatedly applied to modified versions of the training data. Each object in the training data is assigned a weight. Initially, all of the objects are assigned the same weight. In the next iteration, the weights of the misclassified objects are increased, and the weights of the correctly classified objects are decreased. This weighting strategy allows the model to focus on the objects that are hard to classify. This process is continued for a predefined set of iterations. Finally, the predictions from all of the weak classifiers are combined using a majority voting scheme to produce the final prediction [49]. Boosting algorithms have been used in several applications, due to their theoretical guarantees and efficient performance [116, 122]. We use boosting classifiers as the base classifiers in the proposed work.

3.2 Multi-label Classification

Traditional classification methods assume that each instance is associated with exactly one label among a finite set of candidate labels. On the contrary, in the multi-label classification setting, an instance can be associated with multiple classes simultaneously [14]. In bioinformatics, a typical multi-label learning example is the protein function prediction problem where a protein can be associated with multiple functional classes. Since conventional classification methods, such as binary classification and multi-class classification, were not designed to directly tackle the hierarchical classification problems, such algorithms are referred to as flat classification algorithms [60]. It is important to mention that flat classification and other similar approaches are not considered to be hierarchical classification approaches as they create new (meta) classes instead of taking the advantage of the pre-defined taxonomies.

A popular computational approach for inferring protein functions is by applying clustering techniques to the gene expression data. The goal of clustering is to identify groups of proteins that have similar expression patterns [86, 87]. The assumption here is that co-expressed genes (genes which cluster together) are likely to be functionally similar. However, it was shown that simultaneously expressed genes do not always share common functions [109].

3.3 Single Source Gene Function Prediction

Existing hierarchical multi-label classification approaches are different in many aspects: the learning algorithm they are based upon, whether the hierarchy constraint is always met, and whether they can deal with hierarchies structured as a DAG (such as GO) or with hierarchies structured as a rooted tree (such as MIPS's FunCat).

There are two well-known approaches for gene function classifier training: a) Flat approach, where a classifier for each gene function is learned independently, and b) Hierarchical approach, where the relationships between gene functions are taken into consideration while building the classifiers for gene function prediction. Figure 3.3 illustrates the different approaches. These approaches are described in detail later in this chapter.

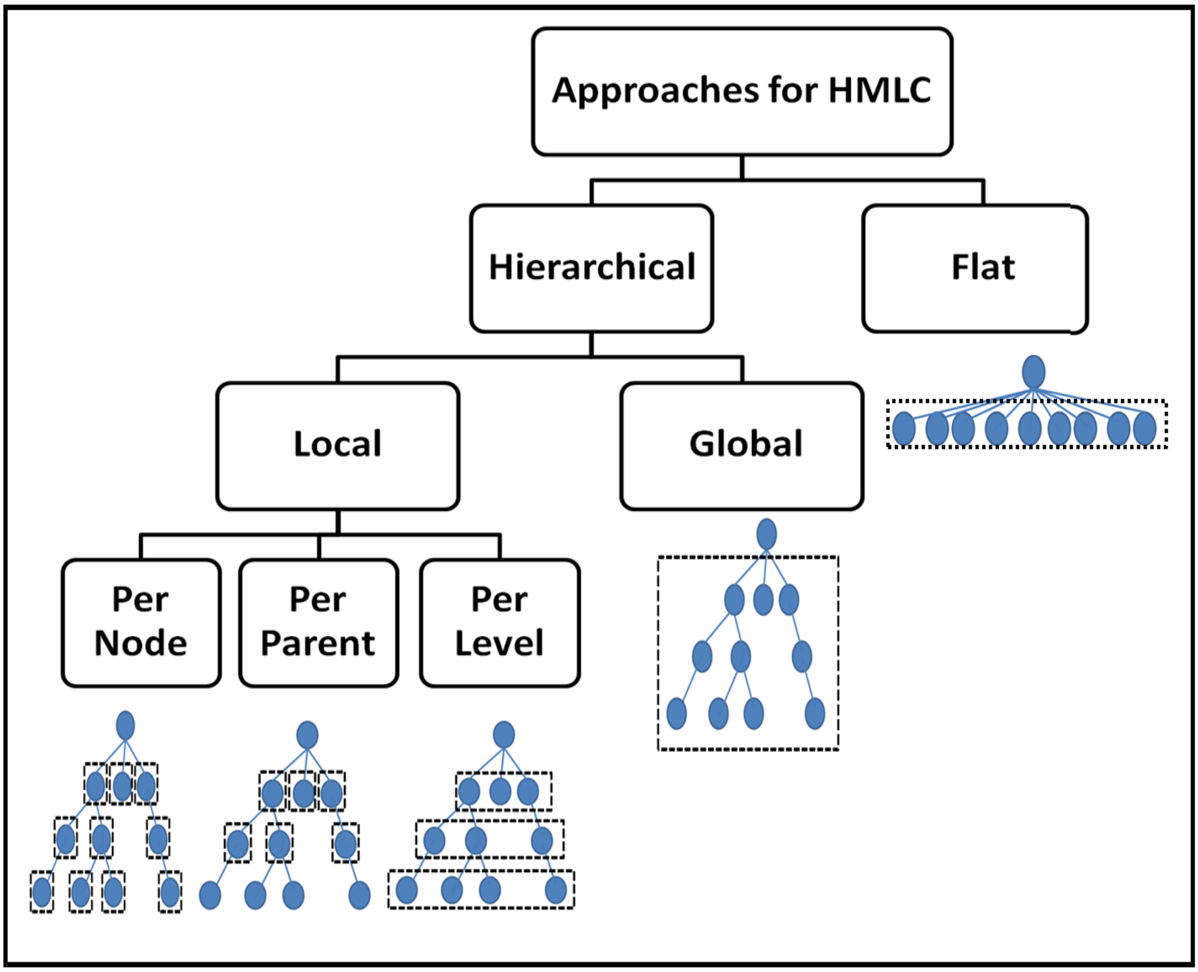


Figure 3.3: An illustration of the existing approaches for hierarchical multi-label classification problem.

3.3.1 Flat Classification Approach

Much of the existing research focuses on a simple and popular approach, namely, “flat” classification approach, that operates on non-hierarchical classification schemes, where a binary classifier is constructed for each label, with instances that are relevant to that label form the positive training set and the rest of the instances form the negative training set [40]. Reducing a hierarchical multi-label classification problem to a standard classification problem allows the possibility of applying the existing methods. However, since the prediction of the class labels has to be performed simultaneously and independently, such transformations are not capable of exploiting the mutual interdependencies and correlations between the labels [21]. Moreover, the flat classification algorithm fails to take advantage of the information inherent in the class hierarchy, when applied to hierarchical classification problems, and hence may be suboptimal in terms of efficiency and/or effectiveness [40].

Furthermore, one of the major problems with the flat approach is that it is too expensive to learn the classifiers for all the functions. When large amounts of gene functions are considered, isolating the gene functions and learning their classifiers independently will produce inferior results. The learning for each classifier is performed on a strongly skewed class distribution [121]. This is because few instances belong to the classes at lower levels in the hierarchy, while more instances are positive examples of the classes at higher levels in the hierarchy. Additionally, flat approach does not take the hierarchy constraint into account and hence, can produce a *hierarchically inconsistent set of predictions*. In other words, a class may predict a test instance to be positive while its parent class (or any of its ancestor classes) predicts it as negative. By leveraging on the hierarchical structure of the classification scheme, the classifiers can be more efficient [40].

As an example of the flat approach, Deng et al. [32] predict gene functions with Markov random fields using protein interaction data. A model is learned for each gene function separately where the hierarchical relationships between the functions is ignored. However, this approach fails to capture one of the key properties of such classification schemes; most schemes not only provide annotations to functional classes, but also capture inter-relations between the functional classes.

3.3.2 Hierarchical Classification Approach

Driven by various domains, such as functional genomics and text categorization, hierarchical multi-label classification is receiving increasing attention now. Hierarchical classifications appears sometimes under the name of structured classification [9, 107] although the term structured classification is broader and includes a classification problem where there is some inherent structure, which could be hierarchical or not, among the classes. Several methods have been proposed to handle the hierarchical multi-label classification task for gene/protein function prediction application [11, 100, 121, 38, 79, 105, 80, 120, 110]. The majority of the hierarchical multi-label classification works have been motivated by the text classification problem. Many proposed methods in that domain use either Bayesian or kernel-based classifiers [100]. For example, Rousu et al. [100] presented a kernel-based algorithm for hierarchical text classification. Their classification model is based on a variant of the maximum margin Markov network for structured output prediction. In that model, the hierarchical structure of the classes is represented as a Markov tree. Their approach does not require a second step to correct for class prediction consistency.

The hierarchical information helps in training and testing the classifiers by identifying the relevant positive and negative examples for each class, according to its location in the hierarchy. The hierarchical approach can provide at least two benefits. First, the large data spaces for learning the classifiers can be partitioned into smaller ones for the relevant more-specific gene functions. Second, the computational complexity for training large amount of gene function classifiers can be reduced significantly by exploiting the hierarchical relationships of the classifiers for the classifier training.

3.3.3 Problem Statement

Now, we introduce the formal notations and terminologies that we use throughout the following discussion. Let $\mathcal{X} = \Re^d$ be the d-dimensional input space and $\mathcal{Y} = \{y_1, y_2, ..., y_{\mathcal{L}}\}$ be the finite set of $\mathcal{L}$ possible labels. The hierarchical relationships among classes in $\mathcal{Y}$ are defined as follows: Given $y_1, y_2 \in \mathcal{Y}, y_1$ is the ancestor of y_2, denoted by $(\uparrow y_2) = y_1$, if and only if y_1 is a superclass of y_2.

Let a hierarchical multi-label training set $\mathcal{D} = \{< x_1, \mathcal{Y}_1 >, ..., < x_N, \mathcal{Y}_N >\}$, where $x_i \in \mathcal{X}$ is a feature vector for instance i and $\mathcal{Y}_i \subseteq \mathcal{Y}$ is the set of labels associated with x_i, such that $y_i \in \mathcal{Y}_i \Rightarrow y'_i \in \mathcal{Y}_i, \forall(\uparrow y_i) = y'_i$. Having $\mathcal{Q}$ as the quality criterion for evaluating the model based on the prediction accuracy, the objective function is defined as follows: a function $\mathfrak{f} : \mathcal{D} \rightarrow 2^{y}$. Here, 2^{y} is the power set of $\mathcal{Y}$, such that $\mathcal{Q}$ is maximized, and $y' \in \mathfrak{f}(x) \Rightarrow y \in \mathfrak{f}, \forall(\uparrow y') = y$. The function $\mathfrak{f}$ is represented here by the ***HML-Boosting*** or ***HiBLADE*** algorithms.

Table 3.1: An example of a synthetically generated dataset. Each example belongs to one or more of the hierarchically structured classes.

Examples	Classes							
	A	B	C	D	E	F	G	H
e1	1	1	1	0	0	1	0	1
e2	1	0	1	0	0	0	0	0
e3	1	1	0	1	1	0	0	0
e4	1	1	1	1	0	0	0	0
e5	1	0	1	0	0	0	0	0
e6	1	1	1	0	1	0	0	0
e7	1	1	0	0	0	0	0	0
e8	1	0	0	1	0	0	0	0
e9	1	1	0	1	1	1	1	0
e10	1	0	1	1	0	0	0	0

Hierarchical Classification Approach Vs. Flat Classification Approach

Table 3.1 shows a set of classes in the columns and a set of examples in the rows. An example is considered to be associated with a certain class if the corresponding value is 1. In this example, we assume that the classes are related in a hierarchical manner as shown in Figure 3.4(a). In this case, class A is considered to be the most general class which all the examples belong to. Going down the tree, more specific classes are reached. Hence, only fewer number of examples are labeled with those specific classes. In other words, for a non-root node in the tree, the set of examples that belongs to this node is a subset of the set of examples that belongs to the parent of that node. Hierarchical classification incorporates the hierarchical information for the purpose of labeling unseen examples. Furthermore, the hierarchical information helps in training and testing the classifiers by identifying the relevant positive and negative examples for each class, according to its location in the hierarchy, which will be discussed later in this thesis. However, the flat classification, which is referred to as “the direct approach” in [17], does not capture such relationship between classes.

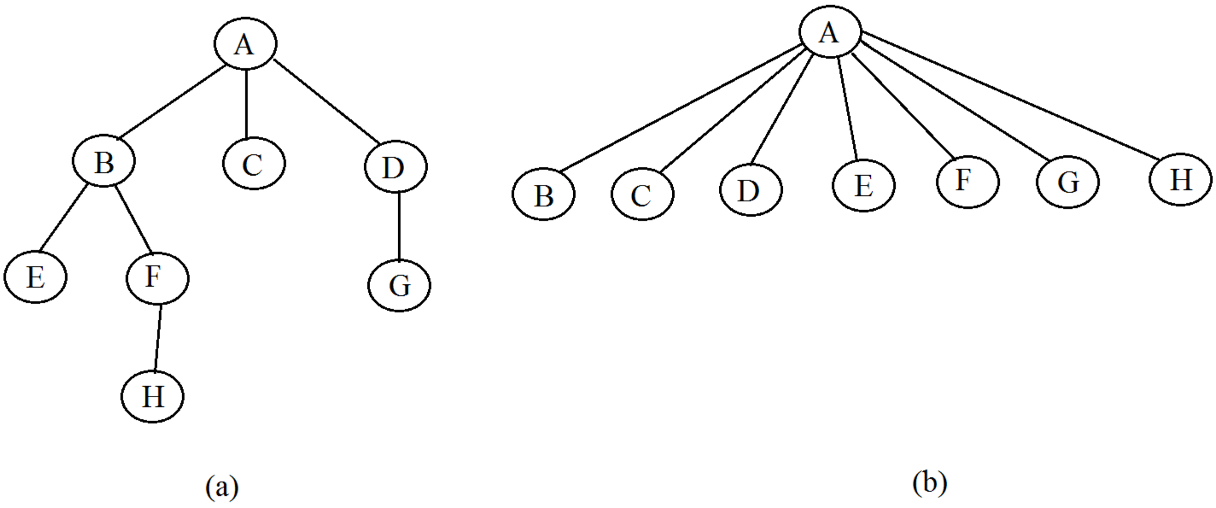


Figure 3.4: (a) A tree structure representation for the classes in Table 3.1, (b) A flat representation for the same set of classes.

As shown in Figure 3.4(b), flat classifiers work independently of each other. Although class A is shown to be the root class for all of the other classes, it is considered as “*any class*”. For a non-root node in the tree, the set of examples that belongs to this node is a subset of the set of examples that belongs to the parent of that node. Since the flat classifiers do not capture the dependencies among the classes, all the classifiers are considered to be at the same level. Moreover, the learning for each classifier is performed on a strongly skewed class distribution [121]. This is because few instances belong to classes at lower levels of the hierarchy, while more instances are positive examples of classes at higher levels of the hierarchy. The set of positive (and negative) examples used to train each classifier does not follow any constraint. As a result, an example may be classified as a given class while it is not classified as belonging to the parent of that class. Such violation should be avoided when some domain knowledge about the examples is available. One of the primary goals of this work is to compare the performance of hierarchical classification against flat classification. Hierarchical multi-label classification approaches bring equal, if not better, classification accuracy when compared with the flat approach, while allowing exponential time savings at both learning and classification time [41].

3.3.4 Training Strategies for Hierarchical Classification

In general, one may divide the existing hierarchical multi-label classification methods into two main approaches: (i) the local classifier approach and (ii) the global classifier approach.

Local-based Hierarchical Classification

The flat approach treats any label independently of all other labels, and thus it ignores the correlations and the interdependencies between the labels. This approach does not take the hierarchy constraint into consideration and hence, can produce a hierarchically inconsistent set of predictions. In other words, a class may predict a test instance to be positive while its parent class (or any of its ancestor classes) predicts it as negative. To overcome the hierarchy constraint problem of the first approach, the second approach mainly focuses on taking the class hierarchy constraint into account. More specifically, the separate class-wise models are hierarchically combined in the prediction stage, so that a classifier generated for a class c, will predict positive only if the classifier for the parent class of c is also predicted to be positive [11, 121].

The methods that adopted the local classifier strategy use local information to train the classifiers. This approach takes the class hierarchy constraint into account. In most of the existing works, a separate binary classifier is learned for each class in the hierarchy. The separate class-wise models are hierarchically combined in the prediction stage, so that a classifier generated for a class c, will predict positive only if the classifier for the parent class of c is also predicted to be positive [121, 11].

Some pioneering works have been proposed for achieving hierarchical gene function classifier training using the hierarchical approach. In essence, there are three main approaches for using the local information to build the classifiers [60], a local classifier per every node [92, 113, 126], a local classifier per parent node [34, 78, 70] and a local classifier per level [66, 27]. For example, in the work of Barutcuoglu et al. [11], a Bayesian framework is developed for correcting class-membership inconsistency for the separate class-wise models approach. Their method starts by training independent Support Vector Machine (SVM) classifiers for each class with no regard to the hierarchy. Bayesian combination of the output of the base binary classifiers is applied next. Thus, the hierarchical structure of the classes is captured through the use of a Bayesian network.

Several methods have been proposed in the literature to deal with the hierarchial multi-label classification problem [47]. In the work of Jiang et al. [57], a unified Bayesian Markov Random Field framework, PHIPA, is proposed. The algorithm focuses on information fusion through coherent collaboration of methodology and data usage to improve predictive capabilities.

In [118], a true path ensemble method, named TPR, was proposed. In this method, a classifier is built for each functional class in the training phase. A bottom-up approach is followed in the testing phase to correct the class-membership inconsistency. In other words, positive predictions for a certain node are propagated to parental classes and their ancestor nodes in the hierarchical scheme while negative predictions are propagated to descendant nodes so that the hierarchy constraint is always respected. In a modified version of TPR, $TPR - w$, a parent weight is introduced. The weight is used to explicitly modulate the contribution of the local predictions with the positive predictions coming from the descendant nodes. In [19], a hierarchical bottom-up Bayesian cost-sensitive ensemble (HBAYES-CS), is proposed. Basically, a calibrated classifier is trained at each node in the taxonomy.

Global-based Hierarchical Classification

In the global classifier strategy, a single global model is developed from the training set that takes the complete class hierarchy into account during a single run of the algorithm. The single global classifier is able to predict all the classes of an example at once. However, since the entire class hierarchy is considered at once in the global classifier strategy, it lacks the kind of modularity for local training of the classifiers as in the local classifier strategy [60]. CLUS-HMC algorithm, proposed by Ven et al. [121], is an example of the global classifier strategy. Their algorithm is based on the predictive clustering trees, in which a single tree is trained to make prediction for all the classes at once. In order to do that, the classification output is transformed into a vector with boolean components corresponding to the possible classes. Weighted Euclidean distance is used to calculate the degree of similarity between the training examples in the classification tree.

Silla Jr. and Freitas [61] presented a global model approach for hierarchical prediction of protein functions as an extension of the Naive Bayes algorithm. The proposed global classification model is built by considering all the classes in the hierarchy instead of building a number of local classification models. Their algorithm is capable of predicting, as the most specific class for a test example, a class at any level of the hierarchy.

3.3.5 Testing Strategies for Hierarchical Classification

For the testing phase, there are two main strategies for the class predictions, namely, the top-down approach and the bottom-up approach. In the top-down approach that was proposed by Koller and Sahami [65], the test instances are propagated from the root to the leaves. Most of the existing methods use a top-down class prediction strategy in the testing phase [11, 100, 60].

On the other hand, in the bottom-up approach, a bottom-up traversal of the hierarchical structure of the classes is performed. In this approach, the offsprings of a given class, where a positive annotation is predicted, can influence the prediction result at the parent class. Valentini [118] followed a bottom-up approach in correcting the class-membership inconsistency in the testing phase after building a separate classifier for each functional class in the training phase. Basically, after evaluating all the classifier nodes' outputs, a "consensus" probability is computed in a bottom-up fashion to obtain consistent predictions.

3.4 Correlation-based Classification

Studying the correlation between objects has gained much attention recently [3, 2, 4]. In the context of gene function, genes are considered to be linked if they are among the ten closest neighbors within a given distance or similarity cutoff [75]. Several research groups have studied the effective exploitation of correlation information among different labels in the context of multi-label learning. However, these approaches do not assume the existence of any pre-defined taxonomical structure of the classes [130, 96, 129, 62, 21]. Broadly speaking, correlation-based approaches can be classified into two main approaches: (i) instance-instance correlation and (ii) label-label correlation.

3.4.1 Instance-Instance Correlation-based Classification

Instance-based learning methods have been proposed to take the interdependencies between the various labels into consideration for multi-label classification, [21]. Cheng and Hüllermeier [21] proposed to combine both model-based and similarity-based inference for multi-label classification. Basically, a framework based on logistic regression for instance-based learning has been proposed. In other words, this approach considers the label information of neighborhood examples as features of a query instance. Moreover, in the extended version of the proposed algorithm, the authors showed that a combination of instance-based and model-based (attribute-oriented) learning can be achieved by the estimation of the optimal regression coefficients.

Typically, some of the methods that can utilize the neighborhood dependencies [99]:

1. Two stage framework where the labels of the instances are predicted first. The predicted labels are used to generate the "neighborhood" features in the second stage. Different ways can be used to define the set of "neighbor" features. Each neighborhood feature

can be seen as a function of the predicted scores and the distance between the candidate and the neighbors. Such neighborhood feature could be a distance weighted average of the scores using some kernel to translate the Euclidean distance into weights [99]. In the second stage, a number of derived features can be incorporated. The two most important features are the neighbor distances and the predicted scores. In the simplest scenario, the labels of the nearest neighbors are used as additional input features.

2. Penalize the nearest neighbors that have significantly different prediction values. The intuition behind this approach is that the instances that are close to each other based on a distance measure are expected to have similar labels. Such a constraint can be incorporated by using a loss function that is associated with the difference between the predicted labels of the nearest examples.

3. Feature construction from the neighboring candidates. In this approach, the features of the neighbors are included directly into the original feature vectors of the instances instead of adding information about the response of the neighbors' features set. Therefore, the features of the neighbors are added to the original feature vector that contains the mean feature vectors of the neighbors.

3.4.2 Label-Label Correlation-based Classification

Several researchers have studied the effective exploitation of correlation information among different labels in the context of multi-label learning but without assuming the existence of any pre-defined taxonomical structure of the classes [130, 96, 129, 62, 21]. Zhang and Zhang in [130] presented a Bayesian network to represent the joint probability of all the labels conditioned on the feature set. Instead of working on the original labels, the Bayesian network utilizes the classification errors of all the labels. In the work of Yan et al. [129], the authors presented a boosting-type learning algorithm called model-shared subspace boosting (MSS-Boost). This algorithm can find, share and combine a number of random subspace models across multiple labels. This algorithm attempts to reduce the information redundancy in the learning process by jointly optimizing the loss functions over all the labels [129].

Jun and Ghosh [62] presented Adaboost.BHC, a multi-class boosting algorithm. A binary hierarchical classifier (BHC) tree is learnt on the classes based on their closeness to one another. AdaBoost is then applied on each internal node of the BHC tree with separately updated distributions. Basically, the problem is divided into a hierarchical structure where similar classes are grouped together to form meta-classes at each inner level in the tree, resulting in a BHC tree.

3.5 Classification with the Integration of Multiple Data Sources

In the literature, several approaches have been proposed to deal with integrating multiple sources of data. The motivation behind the integration is that each data source might provide information about the characteristics of the genes that are to be classified from a different view. Moreover, one source can be useful to learn a specific functional class while being irrelevant to other classes.

Learning from multiple data sources has been extensively studied in machine learning. Generally, the methods that perform the integration can be grouped into two groups: (i) the feature-level integration and (ii) the semantic integration [74]. Figure 3.5 summarizes these two groups. It is important to mention that in the existing approaches, the hierarchical structure among the classes is not taken into account. However, we are incorporating the hierarchical structure of the classes to be a central component in the hierarchical integration framework.

In the feature integration approach, the features from the different data sources are combined at the feature level, and the learning is performed in the joint feature space. In other words, the learning is performed by considering the various data sources at once. On the other hand, in semantic integration, individual models are built from the separate data sources, then the hypotheses (predictions) are combined using some processes such as mutual information maximization [12].

The obtained joint feature space using the feature integration approach is often more informative than the available feature spaces from each of the individual sources. However, feature integration usually tends to not generalize well [71]. Other problems associated with feature integration include model complexity, computation intensity, and training difficulty [127]. On the other hand, with semantic integration, one can still have control over the individual data sources. For example, with semantic integration, we may exclude some data sources from the classification process and make the classification solely based on a certain data source such as DNA microarray. Moreover, the semantic integration appears to have biological plausibility [71].

In spite of the importance of such integration tasks , only few attempts have been made recently to integrate heterogeneous data sources and the results have demonstrated that the multi-source approach is a powerful tool for understanding the functional role of proteins in various organisms. The value of integrating knowledge obtained from various data sources has been illustrated by several studies where functional predictions were made based on several types of data [117, 95, 81].

Much of the work in protein function prediction focuses on incorporating information from multiple sources for pairs of proteins to predict pair-wise functional relationships [68, 56, 123]. For example, Troynskaya et al. proposed MAGIC (Multisource Association of Genes by Integration of Clusters for Saccharomyces cerevisiae) framework that considers the outputs of several clustering methods applied on each data source separately and incorporate the knowledge of yeast biology experts in the prior probabilities of the Bayesian framework. Marcotte et al. [75] predicted a number of protein functions for Saccharomyces cerevisiae based on a heuristic combination of different data types. However, since the confidence levels of protein-protein links are defined on a case-by-case basis, the data sources are combined in a semi-manual and heuristic fashion.

It is important to mention that the distinction between these approaches and our approach

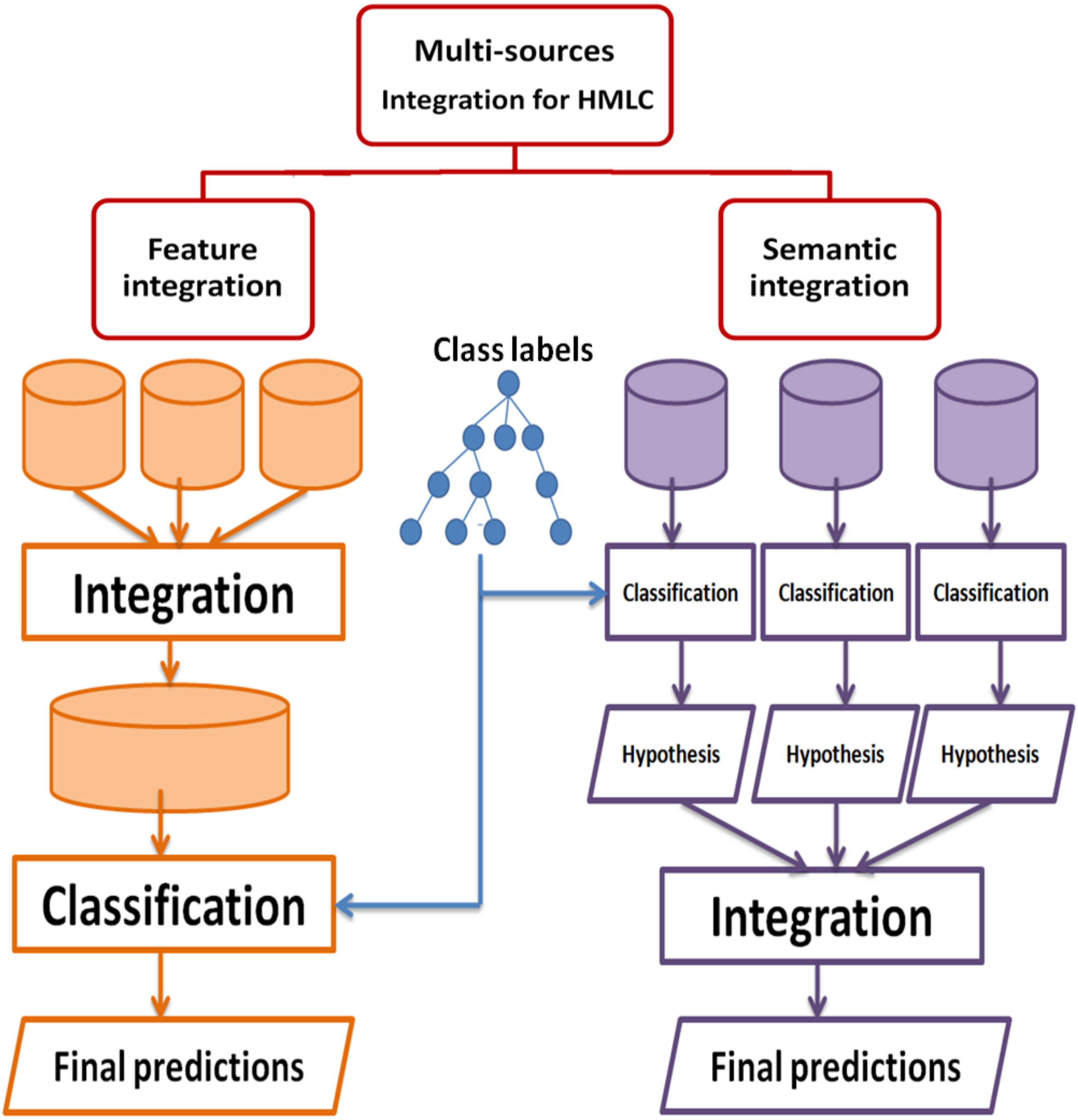


Figure 3.5: An illustration of the integration approaches for protein function prediction.

is that these approaches were targeted specifically towards discovering interacting sets of proteins, while we focus on the problem of discovering the functions of genes/proteins in the context of the hierarchical taxonomies of the functions.

Functional linkage networks, kernel fusion, vector space integration and ensemble systems are the other approaches that have been proposed to deal with the data integration problem [63, 67, 18]. In functional linkage networks, graphs are used to represent the interactions between gene products. In these graphs, genes are represented as nodes and the relationships between them are represented as edges [25]. However, it is hard to apply this approach when the data is not available as relational data. In vector space integration (VSI), vectorial data are concatenated from different sources [91]. However, integrating new data sources make this approach to have very limited modularity. In kernel fusion, the data is represented by means of a kernel function and support vector machines are constructed for each gene function separately. Then, kernel fusion can be used for data integration which uses the closure property of kernels with respect to the sum or other algebraic operators [67]. However, in the proposed framework, only the top-level classes in the hierarchy are predicted.

Guan et al. [46] proposed an ensemble framework as an extension of the method proposed by Barutcuoglu et al. [11]. The ensemble framework is based on three classifiers. In the first step, a classifier that learns a single support vector machine for each gene function is constructed. Next, the Bayesian framework is used to correct the combination of support vector machines for maintaining the hierarchy constraint. Finally, a classifier that constructs a single support vector machine per gene function and per data source and forms a Naive Bayes combination over the data sources is learned. Obozinski et al. [85] presents a two-step approach where support vector machines are learned independently for each gene function, which allows the violation of the hierarchy constraint. Then, reconciliation is performed to obtain a set of probabilistic predictions that are consistent with the topology of the ontology.

The above discussed approaches focus on integrating multiple sources of data without taking into account the hierarchical relationships between the functional classes. As a consequence, naively applying these data integration approaches to the hierarchical gene function prediction problems leads to serious inconsistencies due to the violation of the hierarchy constraint governing the functional annotations of genes in the hierarchical taxonomies. In the proposed integration framework, we respect the hierarchy constraint while integrating the different data sources.

3.6 Evaluation Metrics

Since each example in the hierarchical multi-label classification is associated with multiple labels that are arranged in a hierarchy, evaluating hierarchical multi-label classification algorithms is more complicated than the conventional single-label learning algorithms. One natural solution is to compute the well-known metrics of precision, recall and F-measure for each label independently and then combine the results using micro- or macro-averaging [130]. However, such evaluation measures are used by unstructured classification problems and thus, they are inadequate to address the hierarchical structure of the classes. Another approach that is used for the hierarchical multi-label learning is to utilize the extended versions of the single label metrics (precision, recall and F-measure). In order to evaluate our algorithm, we adopted both, the classical and the hierarchical performance evaluation measures.

F_1 measure considers the joint contribution of both precision (P) and recall (R). F_1 measure is defined as follows:

$$F_1 = \frac{2 \times P \times R}{P + R} = \frac{2TP}{2TP + FP + FN} \tag{3.1}$$

Precision and Recall are defined as follows:

$$P = \frac{TP}{TP + FP} \tag{3.2}$$

$$R = \frac{TP}{TP + FN} \tag{3.3}$$

where TP stands for True Positive, TN for True Negative, FP for False Positive and FN for False Negative. When TP=FP=FN=0, we made F_1 measure to equal to 1 as the classifier has correctly classified all the examples as negative examples [40].

Hierarchical measures are defined as follows [61]:

$$hP = \frac{1}{|l(P(x))|} \sum_{p \in l(P(x))} \frac{|C(x) \cap \uparrow p|}{|\uparrow p|} \tag{3.4}$$

$$hR = \frac{1}{|l(C(x))|} \sum_{c \in l(C(x))} \frac{|\uparrow c \cap P(x)|}{|\uparrow c|} \tag{3.5}$$

$$hF = \frac{2 \times hP \times hR}{hP + hR} \tag{3.6}$$

where hP, hR and hF stands for hierarchical precision, hierarchical recall and hierarchical F-measure, respectively. $P(x)$ is a subgraph formed by the predicted class labels for the instance x while $C(x)$ is a subgraph formed by the true class labels for the instance x. p is one of the predicted class labels and c is one of the true labels for instance x. $l(P(x))$ and $l(C(x)))$ are the set of leaves in graphs $P(x)$ and $C(x)$, respectively. $\uparrow p$ is the set consisting of the most specific predicted class and all of its ancestor classes. Likewise, $\uparrow c$ is the set consisting of the most specific true class and all of its ancestor classes. We also computed both micro-averaged hierarchical F-measure (hF_1^{μ}) and macro-averaged hierarchical F-measure hF_1^{M}. hF_1^{μ} is computed by computing hP and hR for each path in the hierarchical structure of the tree and then applying Equation (3.6). On the other hand, hF_1^{M} is computed by calculating hF_1 for each path in the hierarchical structure of the classes independently and then averaging them.

Having high hierarchical precision means that the predictor is capable of predicting the most general functions of the instance, while having high hierarchical recall indicates that the predictor is able to predict the most specific classes [119]. The hierarchical F-measure takes into account the partially correct paths in the overall taxonomy.

3.7 Summary

This chapter summarizes the existing work related to hierarchical multi-label classification in the gene function prediction domain. Hierarchical multi-label classification problem is an extension of the binary classification problem where an instance can be associated with multiple classes that are related through a hierarchical categorization scheme. In other words, when an instance is labeled with a certain class, it should also be labeled with all of its superclasses. Hierarchical multi-label classification methods are classified into two main approaches in the training phase: (i) the local classifier approach and (ii) the global classifier approach. For the testing phase, there are two main approaches: (i) the top-down approach and (ii) the bottom-up approach. The integrative analysis of multiple heterogenous data sources can be used to generate new knowledge not accessible by the analysis of a single data source alone. In the protein function prediction context, the integrated analysis may be a key to decipher gene function annotation. Apart from the integration, labels may have correlations that are not shown through the hierarchical structure. The target problem of this dissertation has the characteristic of hierarchical structure through a rooted-tree, multi-label, multi-source and multi-correlation.

CHAPTER 4

HIERARCHICAL MULTI-LABEL BOOSTING FOR GENE FUNCTION PREDICTION

Proteins are organized and classified according to a hierarchical classification scheme and each protein will participate in multiple activities. Flat classifiers, that work on non-hierarchical classification problems independently, do not take into account the hierarchical structure of the functional class taxonomies. Therefore, they are not able to utilize the information inherent in the class hierarchy. Moreover, independent classifiers, where each classifier predicts the gene membership to a particular class, may lead to an inconsistent set of predictions for a hierarchically structured classification scheme. In this chapter, HML-Boosting (**H**ierrachical **M**ulti-**L**abel classification using **B**oosting) algorithm for the problem of hierarchical multi-label classification in the context of gene function prediction is proposed. The HML-Boosting algorithm exploits the hierarchical dependencies among the classes. The algorithm takes the hierarchy constraint into account so that no additional step is needed for the correction of the inconsistent predictions.

4.1 Motivation

The functional classification of genes is an important and challenging problem in the field of functional genomics. First, there are many functional classes which may be related to each other in a tree or a graph structure (Funcat or Gene Ontology(GO)). Usually, the goal is to discover the specific unknown functions for a particular gene. Second, a gene may have multiple class labels. Biologically, a gene may be involved in more than one biological activity. Therefore, there is a need for a prediction algorithm that is able to identify all the possible functions of a particular gene.

In this chapter, the hierarchical structure of gene functions in FunCat taxonomy is incorporated into the multi-label classification problem. As a result, the performance of the proposed method that incorporates both the hierarchical and multi-label concepts is significantly better compared to the standard classification approach. In this work, we develop the HML-Boosting algorithm and apply it to bio-molecular data for predicting gene functions. Boosting algorithms are popular machine learning algorithms with rigorous theoretical properties and are competitive to any other classification methods [40]. HML-Boosting algorithm relies on the hierarchical information and utilizes the hierarchy to improve the prediction accuracy. To test the HML-Boosting framework in the context of functional genomics, each classifier is evaluated in a three-fold cross-validation scheme in which FunCat annotations of the test genes are hidden.

Our proposed work falls into the local classifier category. More specifically, for each parent node in the class hierarchy a multi-class multi-label classifier is built using AdaBoost.MH. The rest of this chapter is organized as follows: In section 2, the proposed HML-Boosting algorithm is described. In section 3, extensive experimental evaluation on four bio-molecular datasets is reported. Finally, section 4 concludes the chapter with future directions.

4.2 Our Contributions

The contributions of our algorithm can be summarized as follows:

1. Develop a class-membership consistency correction method in the model building process rather than having it as a separate step.

2. Study the effect of the different testing strategies (the top-down approach and the bottom-up approach) on the prediction results.

3. Compare the performance of the flat classification approach with the proposed local approach of the hierarchical multi-label classification.

4.3 The Proposed HML-Boosting Algorithm

HML-Boosting [5] is a hierarchical multi-label gene functional classification algorithm that is inspired by TREEBOOST.MH [40], a multi-label hierarchical text classification algorithm. HML-Boosting uses ADABOOST.MH as its base step and recurs over the class tree structure. HML-Boosting exploits the hierarchical taxonomy of the classes to improve prediction performance.

4.3.1 ADABOOST.MH

ADABOOST.MH [104] is a popular multi-class variant of the AdaBoost algorithm and works well in practice [62]. It handles multi-class and multi-label problems. ADABOOST.MH (see Algorithm 1) works by transferring a multi-class problem into a binary classification problem by replicating positive instances for the given class labels [62]. More details of the ADABOOST.MH algorithm are given in [104]. The input to the algorithm is a training set $\mathcal{T} = < g_1, \mathcal{S}_1 >, ..., < g_n, \mathcal{S}_n >$, where $\mathcal{S}_i \subseteq \mathcal{C}$ is the set of classes to which gene/gene-product g_i belongs to.

ADABOOST.MH maintains a distribution D, that is updated at each iteration. The initial distribution D_1 is uniform. At each iteration, s, a weak learner is built to form a weak hypothesis $\hat{\phi}_s$. The weak hypothesis is generally a decision stump. Next, all the weights $D_s(g_i, c_j)$ are updated to $D_{s+1}(g_i, c_j)$ using the following rule:

$$D_{s+1}(g_i, c_j) = \frac{D_s(g_i, c_j) \exp(-\phi(g_i, c_j)\hat{\phi}_s(g_i, c_j))}{Z_s} \qquad (4.1)$$

Here, the target function $\phi(g_i, c_j)$ is defined as follows:

$$\phi(g_i, c_j) = \begin{cases} 1, & g_i \in c_j \\ -1, & \text{Otherwise} \end{cases}$$

and

$$Z_s = \sum_{i=1}^{n}\sum_{j=1}^{m} D_s(g_i, c_j)\exp(-\phi(g_i, c_j)\hat{\phi}_s(g_i, c_j)) \tag{4.2}$$

is a normalization factor chosen to make $\sum_{i=1}^{n}\sum_{j=1}^{m} D_{s+1}(g_i, c_j) = 1$.

The sign of the weak hypothesis is used to decide upon the prediction made by the weak learner, while the absolute value can be interpreted as the strength of the belief. In other words, if $\phi_s(g_i, c_j) > 0$, then g_i is predicted to belong to c_j, while if $\phi_s(g_i, c_j) < 0$, then g_i is predicted as a negative example of c_j. After finishing all the iterations, the final hypothesis is generated by summing up all of the weak hypothesis, $\hat{\phi}(g, c) = \sum_{s=1}^{S}\hat{\phi}_s(g, c)$ for $c \in C = c_1, ..., c_m$.

Algorithm 1 $ADABOOST.MH$

```
Input:
A training set $\mathcal{T} = < g_1, \mathcal{S}_1 >, ..., < g_n, \mathcal{S}_n >$ where $\mathcal{S}_i \subseteq \mathcal{C} = \{c_1, ..., c_m\}$ for all $i = 1, ..., n$.
Output: A final hypothesis $\hat{\phi}(g, c) = \sum_{s=1}^{S}\hat{\phi}_s(g, c)$ for $c \in C = c_1, ..., c_m$.
Algorithm:
Set $D_1(g_i, c_j) = \frac{1}{mn}$ for all $i = 1, ..., n$ and for all $j = 1, ..., m$
for $s = 1, ..., S$ do
    Pass distribution $D_s(g_i, c_j)$ to the weak learner
    Get the weak hypothesis $\hat{\phi}_s$ from the weak learner
    Set $D_{s+1}(g_i, c_j) = \frac{D_s(g_i, c_j)\exp(-\phi(g_i, c_j)\hat{\phi}_s(g_i, c_j))}{Z_s}$
    Where $Z_s = \sum_{i=1}^{n}\sum_{j=1}^{m} D_s(g_i, c_j)\exp(-\phi(g_i, c_j)\hat{\phi}_s(g_i, c_j))$
end for
```

4.3.2 HML-Boosting

According to the True Path Rule (TPR), that governs both FunCat and GO taxonomies annotations, the TPR indicates a two-way asymmetric flow of information. A gene g that is annotated with a particular functional class, c_j, is also annotated with the parent class and with all of the ancestor classes of c_j in a recursive way [118]. In this scenario, gene g is called as "bubbled-up" positive example in the sense that it has been bubbled up to c_j from somewhere down below [40]. On the other hand, if a gene is not annotated with a class c_j, that gene cannot be annotated with any of the descendant classes of c_j. In that case, gene g is called an "own" positive example of c_j [40].

At each classifier, we need to carefully choose the set of positive and negative examples. We followed the same approach discussed in [40], in which the positive training examples of a non-leaf category, c_j, is a superset of the union of sets of positive training examples of all of its descendent (leaf) classes.

HML-Boosting algorithm is explained in Algorithm 2. Each non-root class, c_j, has a binary classifier, $\hat{\phi}_j$, that is associated with it. The classifier should act as a "filter" to prevent unsuitable examples from spreading out to the lower levels in the hierarchy. Hence, only the test genes that a classifier $\hat{\phi}_j$ decides as belonging to c_j are passed to all the binary classifiers corresponding to the children classes of c_j. While the genes that classifier $\hat{\phi}_j$ marks as not belonging to c_j are "blocked" and no further analysis is carried out on them.

```
Algorithm 2 $HML - Boosting$
  Input: A pair $< C, L >$ where $C$ is a tree-structured set of classes and $L$ is the total number
  of classes of $C$.
  Output: For each non-leaf class $c_t \in C$, a final hypothesis $\hat{\phi}(g, c) = \sum_{s=1}^{S} \hat{\phi}_s(g, c)$ for
  $c \in children(c_t)$.
  Algorithm:
  for $i = 1, ..., L$ do
    if class $i$ is a leaf class then
      Do nothing
    else
      Let $children(i) = c_{1i}, ..., c_{ki}$ be the $k$ children classes of $i$
      Run ADABOOST.MH on $children(i)$
    end if
  end for
```

Relying on the hierarchical topology of the classification scheme, the training of each of the binary classifiers, $\hat{\phi}_j$, is performed locally using *the siblings policy* where the training dataset for each classifier is composed of the positive instances at the current class and the siblings of the current class. During classification, the classifier $\hat{\phi}_j$ at class c_j will only be presented with examples that are positive at the parent class of c_j. Hence, the reached examples at $\hat{\phi}_j$ are positive examples to c_j and/or to the siblings of c_j. In other words, the training for classifier $\hat{\phi}_j$ is performed by feeding as negative training examples, the positive examples at the parent of c_j that are not positive examples at c_j.

It should be noted that the selected negative training examples at $\hat{\phi}_j$ are the most informative negative examples for training [40]. Moreover, since as we go down in the hierarchy, fewer training examples are survived, this will be reflected in the efficiency of the algorithm, for both training and classification time.

HML-Boosting converts the hierarchical multi-label classification problem to multiple flat multi-label classification problems, in which one classifier is built for every internal node in the tree. HML-Boosting iteratively calls ADABOOST.MH to generate a multi-label flat classifier for the children of every internal node. In other words, a binary classifier, $\hat{\phi}$, for each non-root class $c_j \in C$ is generated so that the hierarchical classification can be performed. The algorithm first checks whether the reached class is a leaf node or not. The classifiers are built for internal nodes only and hence, no work needs to be done if a leaf function class has been reached.

If an internal class, c_j, has been reached, we identify the set of children classes of c_j. Next, ADABOOST.MH is called for a multi-label flat classification for the children classes of c_j. Note that the negative examples of a class c_j are the set of positive examples at the parent class but are not positive examples at class c_j. Next, for each class $c_q \in children(c_j)$, where $children(c_j)$ refers to the set of children of class c_j, HML-Boosting is called iteratively on the children of a particular class. HML-Boosting results in a tree of binary classifiers, one for each non-root node, where each one consisting of the combination of decision stumps.

We evaluated the performance of the HML-Boosting algorithm using two different policies for the testing phase, the top-down class prediction approach and the bottom-up class prediction approach. In the top-down class prediction approach, the negative predictions are propagated from top to bottom along the hierarchy. In other words, for an unseen example, the positive predictions of the higher-levels (most generic) classes are propagated from top to bottom nodes along the hierarchy.

On the other hand, in the bottom-up class prediction strategy, the propagation of positive decisions are carried out from bottom to top nodes along the hierarchy. In other words, positive decisions of descendant nodes contribute to the decision of their ancestors. More specifically, the prediction of a node/class is dependent on the local prediction of that node/class and on the prediction of the descendant nodes of that node according to the following [118]:

$$P(x) = \frac{P_{local}(x) + \sum P_{child}(x)}{1 + |children(x)|} \tag{4.3}$$

where $P_{local}(x)$ is the local prediction at the node x and $\sum P_{child}(x)$ is the summation of the prediction of the descendant nodes of x. $children(x)$ represents the set of children classes of the class x.

4.4 Experimental Results

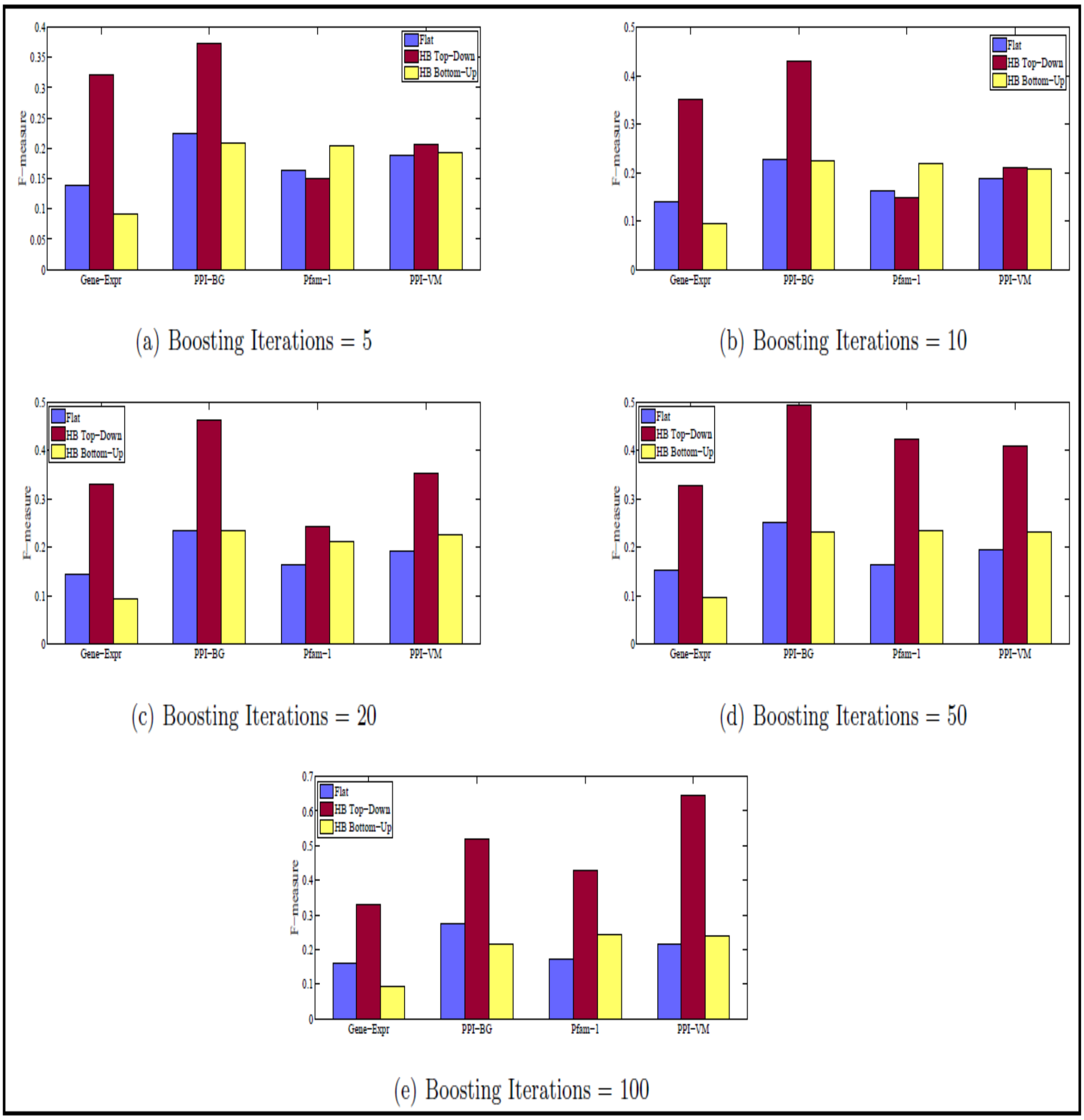
4.4.1 Experimental Setup

For our experiments, we used the functional classification of yeast genes at genome-wide level, where the classes are structured according to FunCat taxonomy. Each dataset provides a different description of a specific gene aspect. For each dataset, we report the evaluation of the performance of the flat ADABOOST.MH multi-label classifiers and the HML-Boosting algorithm using precision, recall and F_1 evaluation metrics that are described earlier.

The flat multi-label classifier does not incorporate the hierarchical structure of the classes as discussed earlier. Each base classifier in the flat multi-label classification framework, is separately trained to identify the set of genes belonging to a specific functional class without considering the hierarchical relationship between the classes.

The comparison between HML-Boosting algorithm and the Flat method is based on the "per-class" F_1 measure that is obtained by averaging the F-measure for all the classes in the

FunCat hierarchy for each dataset. In other words, an overall F_1 measure is obtained by computing F_1 measure for each class separately and then averaging F_1 measure across all the classes.



(a) Boosting Iterations = 5

(b) Boosting Iterations = 10

(c) Boosting Iterations = 20

(d) Boosting Iterations = 50

(e) Boosting Iterations = 100

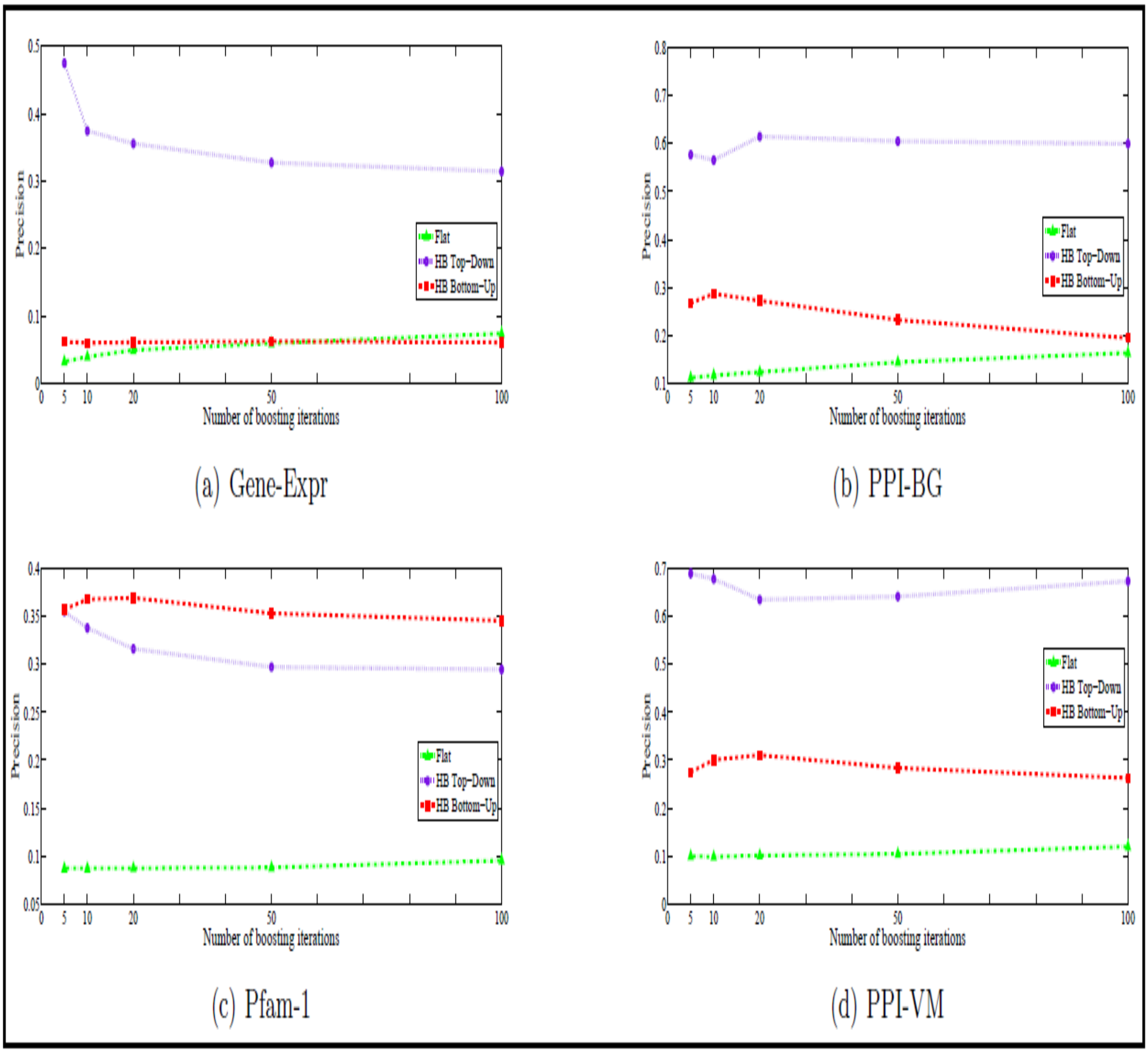
Figure 4.1: The Overall per-class F_1 measure comparison between flat method, HML-Boosting top-down and HML-Boosting bottom-up.

4.4.2 Results and Discussion

Consistent with most of the other works in the literature, in the hierarchical multi-label classification setting, precision, recall and F_1 measure are used as appropriate evaluation metrics for comparing the performance. The performance of the flat method, the HML-Boosting algorithm using the top-down and the HML-Boosting algorithm using the bottom-up class prediction strategies were compared using per-class F_1 measure; then, we analyzed the performance of the HML-Boosting at each level of FunCat hierarchy. We also studied the influence of changing the number of boosting iterations on the performance.

Figure 4.1 shows the F_1 measure for each dataset using the HML-Boosting with the top-down class prediction strategy during the testing phase, the HML-Boosting with the bottom-up class prediction strategy during the testing phase and the flat classification methods with different boosting iterations. In our experimental setting, five values of boosting iterations: $5, 10, 20, 50$ and 100 have been examined. Each barplot group refers to the results of applying the flat method and the HML-Boosting algorithm (with the top-down and the bottom-up strategies) on a particular dataset. The difference between the HML-boosting and the flat classification is significant in most of the cases. We noticed that as the number of boosting iterations (s) is increased, HML-Boosting significantly outperforms the flat classification method on all the datasets. In other words, the improvements become much higher as we increase the number of boosting iterations. Additionally, in most of the cases, the HML-Boosting with the top-down class prediction strategy during the testing phase outperforms its counterpart, the HML-Boosting with the bottom-up class prediction strategy.

Figure 4.2 and Figure 4.3 show the overall "per-class" precision and the overall "per-class" recall, as a function of boosting iterations, for each dataset using the HML-Boosting with the top-down class prediction strategy, the HML-Boosting with the bottom-up class prediction strategy and the flat method. We observed that the HML-Boosting with the top-down class prediction strategy tends to have the best results with respect to precision and F_1 measure



(a) Gene-Expr

(b) PPI-BG

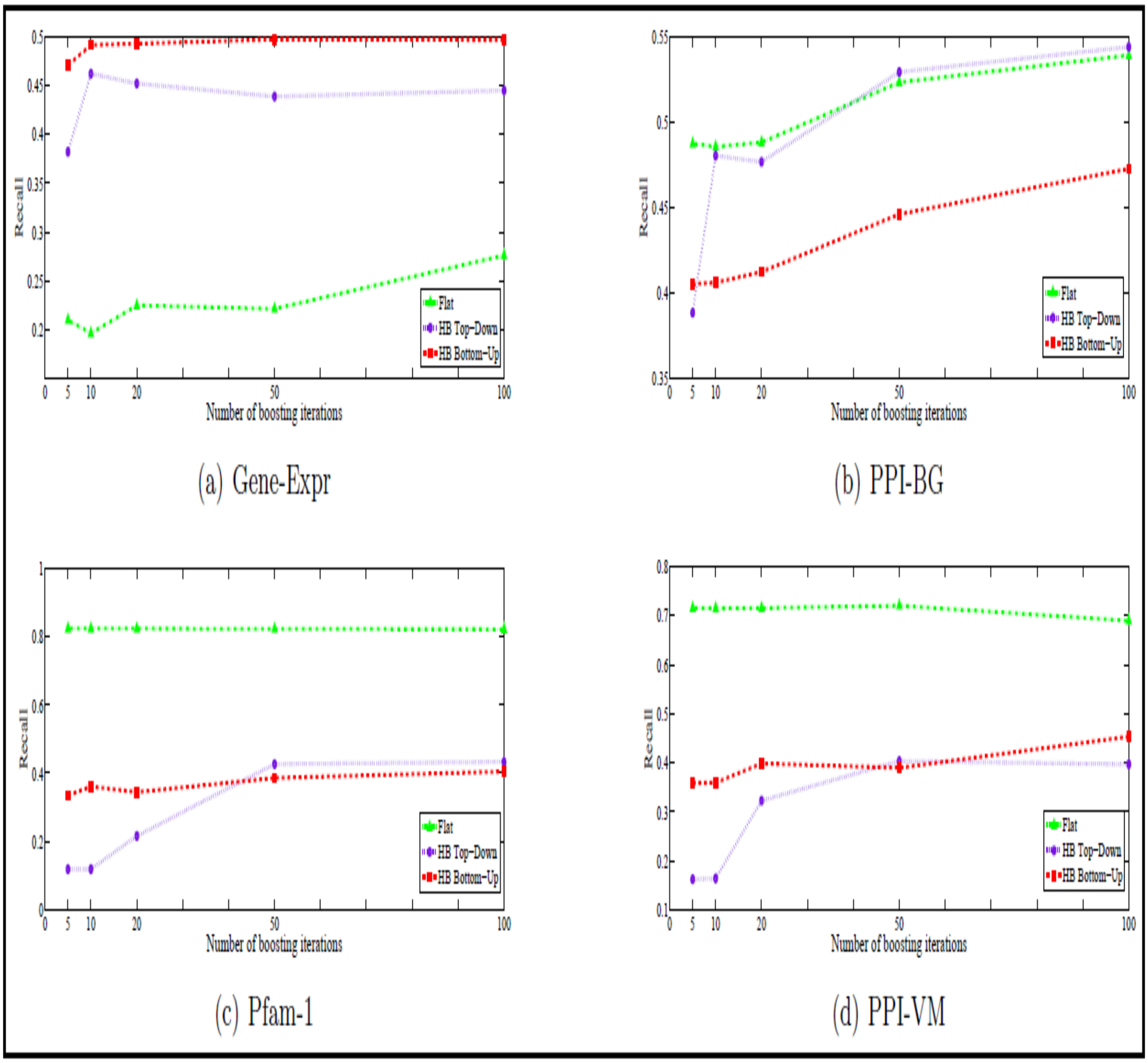
(c) Pfam-1

(d) PPI-VM

Figure 4.2: The overall Precision for each dataset using the flat method, HML-Boosting top-down and HML-Boosting bottom-up algorithms.

while the flat method tends to achieve better results with respect to recall. In fact, both of the class-membership inconsistency correction have similar behavior in terms of the recall.

To get more insights into the performance of the HML-Boosting algorithm with the different testing strategies, we performed a level-wise analysis of the precision, recall, F_1 measure and accuracy on the four datasets. In measuring the level-wise performance, level 1 reflects



(a) Gene-Expr

(b) PPI-BG

(c) Pfam-1

(d) PPI-VM

Figure 4.3: The overall recall for each dataset using the flat method, HML-Boosting top-down and HML-Boosting bottom-up algorithms.

the root nodes while all other classes are at depth i, where $2 \leq i \leq 5$. We show the results for the top four levels in the hierarchy. Moreover, we show the performance of the HML-Boosting algorithm when the number of boosting iterations is varied. In general, the results improve as we increase the number of iterations. Tables 4.1, 4.2, 4.3 and 4.4 show the results of per-level evaluation for Gene-Expr, PPI-BG, Pfam-1 and PPI-VM datasets, respectively. The best results for each level are highlighted.

It should be noted that the accuracy of the HML-Boosting using the top-down class prediction strategy in the testing phase reduces as we go down in the tree, from the higher levels to the lower levels. The main reason for this performance reduction is related to the testing mechanism followed by the algorithm. A classifier will be able to test any example only if that example was predicted to be a positive example by the parent of the corresponding class. Therefore, if a classifier at a particular class misclassifies a positive example, this example will be considered as a negative example by all the descendant classes of that class. Hence, the accuracy of the lower levels will be affected by the behavior of the classifiers at the higher levels. For the same reason, fewer examples will be propagated to the classifiers in the lower levels.

Table 4.1: Per-level precision, recall, F_1 measure and accuracy for Gene-Expr dataset using HML-Boosting with top-down class prediction and HML-Boosting with bottom-up class prediction algorithms for the different choice of boosting iterations.

Level	**Measurement**	**HML-Boosting top-down**					**HML-Boosting bottom-up**				
		5 iter	**10 iter**	**20 iter**	**50 iter**	**100 iter**	**5 iter**	**10 iter**	**20 iter**	**50 iter**	**100 iter**
Level 1	Precision	**0.5405**	0.4776	0.4895	0.4719	0.4425	0.1573	0.1620	0.1662	0.1673	**0.1679**
	Recall	0.0527	0.1020	0.1245	0.1606	**0.1809**	0.8337	0.8889	0.9058	0.9290	**0.9351**
	F1	0.0960	0.1681	0.1986	0.2396	**0.2568**	0.2646	0.2740	0.2808	0.2836	**0.2847**
	Accuracy	**0.8627**	0.8602	0.8608	0.8589	0.8550	**0.3479**	0.3371	0.3471	0.3393	0.3388
Level 2	Precision	**0.4385**	0.3448	0.3524	0.3308	0.3050	0.0814	0.0809	**0.0815**	0.0805	0.0793
	Recall	0.5238	0.4904	0.5345	**0.5529**	0.5485	0.6322	0.6723	0.6963	0.7260	**0.7333**
	F1	**0.4774**	0.4049	0.4248	0.4139	0.3921	0.1442	0.1444	0.1460	**0.1449**	0.1431
	Accuracy	**0.8370**	0.8151	0.8150	0.8095	0.8025	**0.6512**	0.6298	0.6214	0.6019	0.5918
Level 3	Precision	**0.3780**	0.2908	0.2739	0.2743	0.2557	**0.0446**	0.0436	0.0440	0.0425	0.0424
	Recall	**0.7619**	0.6547	0.6165	0.6291	0.6402	0.5629	0.5972	0.6141	0.6216	**0.6283**
	F1	**0.5053**	0.4028	0.3793	0.3820	0.3654	**0.0827**	0.0812	0.0821	0.0795	0.0794
	Accuracy	0.7981	**0.8299**	0.8058	0.8152	0.8215	**0.7336**	0.7118	0.7071	0.6931	0.6894
Level 4	Precision	0.18990	**0.2083**	0.2000	0.1905	0.1929	0.0309	0.0308	**0.0311**	0.0304	0.0309
	Recall	0.7143	0.7576	**0.7750**	0.6496	0.7154	0.6192	0.6277	0.6433	0.6449	**0.6651**
	F1	0.3000	**0.3268**	0.3179	0.2946	0.3039	0.0588	0.0588	**0.0594**	0.0580	0.0591
	Accuracy	**0.6889**	0.6460	0.6518	0.6353	0.6365	**0.6487**	0.6434	0.6387	0.6283	0.6246

Table 4.2: Per-level precision, recall, F_1 measure and accuracy for PPI-BG dataset using HML-Boosting with top-down class prediction and HML-Boosting with bottom-up class prediction algorithms for the different choice of boosting iterations.

Level	**Measurement**	**HML-Boosting top-down**					**HML-Boosting bottom-up**				
		5 iter	**10 iter**	**20 iter**	**50 iter**	**100 iter**	**5 iter**	**10 iter**	**20 iter**	**50 iter**	**100 iter**
Level 1	Precision	0.7127	0.7177	0.7319	0.7447	**0.7526**	0.1572	0.1606	0.1637	**0.1681**	0.1675
	Recall	0.0708	0.0747	0.0848	0.1086	**0.1405**	0.7285	0.7102	0.7362	0.7780	**0.8182**
	F1	0.1288	0.1354	0.1520	0.1895	**0.2368**	0.2586	0.2619	0.2678	0.2764	**0.2780**
	Accuracy	0.8667	0.8671	0.8683	0.8707	**0.8739**	0.4121	**0.4366**	0.4335	0.4268	0.4019
Level 2	Precision	0.5911	0.5914	0.6337	**0.6487**	0.6287	0.0955	0.0966	**0.0983**	0.0959	0.0926
	Recall	0.8304	0.8506	**0.8600**	0.8252	0.8348	0.5758	0.5797	0.6060	0.6488	**0.6804**
	F1	0.6906	0.6977	0.7297	0.7264	**0.7447**	0.1638	0.1657	**0.1692**	0.1671	0.1630
	Accuracy	0.8223	0.8210	0.8404	0.8481	**0.8574**	0.7268	**0.7286**	0.7235	0.6995	0.6754
Level 3	Precision	0.5625	0.5413	0.6077	0.6143	**0.6722**	0.0531	0.0541	**0.0556**	0.0554	0.0538
	Recall	0.7246	0.7170	0.7645	0.7798	**0.7871**	0.5410	0.5501	0.5771	0.6254	**0.6559**
	F1	0.6333	0.6169	0.6771	0.6872	**0.6990**	0.0967	0.0985	0.1014	**0.1018**	0.0994
	Accuracy	0.8667	0.8520	0.8661	**0.8712**	0.8711	0.7879	**0.7886**	0.7852	0.7682	0.7504
Level 4	Precision	0.6078	0.5139	0.6133	**0.6226**	0.6163	0.0353	0.0326	0.0349	**0.0358**	**0.0358**
	Recall	0.6940	0.7115	0.7500	0.8075	**0.8328**	0.6352	0.6584	0.6793	0.7002	**0.7235**
	F1	0.6481	0.5968	0.6748	0.7031	**0.7084**	0.0669	0.0621	0.0664	0.0681	**0.0683**
	Accuracy	**0.7918**	0.7331	0.7837	0.7770	0.7789	**0.6846**	0.6456	0.6598	0.6587	0.6482

Table 4.3: Per-level precision, recall, F_1 measure and accuracy for Pfam-1 dataset using HML-Boosting with top-down class prediction and HML-Boosting with bottom-up class prediction algorithms for the different choice of boosting iterations.

Level	**Measurement**	**HML-Boosting top-down**					**HML-Boosting bottom-up**				
		5 iter	**10 iter**	**20 iter**	**50 iter**	**100 iter**	**5 iter**	**10 iter**	**20 iter**	**50 iter**	**100 iter**
Level 1	Precision	0.9065	0.9065	**0.9128**	0.8817	0.8842	0.1584	**0.1585**	0.1584	0.1529	0.1541
	Recall	0.0433	0.0433	0.0468	0.0564	**0.0578**	0.7018	0.7001	0.6994	0.7312	**0.7417**
	F1	0.0827	0.0827	0.0890	0.1060	**0.1085**	0.2584	**0.2585**	0.2583	0.2529	0.2552
	Accuracy	**0.9240**	0.8515	0.8520	0.8530	0.8532	0.3678	**0.3695**	**0.3695**	0.3217	0.3205
Level 2	Precision	0.8521	0.8898	**0.8994**	0.8398	0.8178	0.0977	0.0979	0.0984	0.0981	**0.0993**
	Recall	0.8345	0.7793	0.8512	0.8782	**0.9154**	0.5764	0.5767	0.5804	0.6098	**0.6221**
	F1	0.8432	0.8309	**0.8746**	0.8586	0.8638	0.1670	0.1674	0.1683	0.1690	**0.1713**
	Accuracy	0.8515	0.9223	**0.9341**	0.9231	0.9230	0.7000	**0.7007**	**0.7007**	0.6870	0.6860
Level 3	Precision	0.7500	0.7857	**0.8000**	0.7500	0.7143	0.0544	0.0546	0.0551	**0.0570**	0.0561
	Recall	0.8438	**0.8750**	0.8421	0.8298	0.8224	0.5209	0.5235	0.5286	0.5497	**0.5726**
	F1	0.7941	**0.8280**	0.8205	0.7879	0.7645	0.0985	0.0990	0.0999	**0.1032**	0.1022
	Accuracy	0.9288	0.9340	**0.9351**	0.9186	0.9136	0.7662	0.7664	**0.7665**	0.7659	0.7534
Level 4	Precision	0.7714	**0.8636**	**0.8636**	0.7736	0.7368	0.0363	0.0349	0.0373	**0.0382**	0.0366
	Recall	**0.7297**	0.5135	0.5135	0.6949	0.7000	0.6079	0.6221	0.6239	0.6390	**0.6513**
	F1	**0.7500**	0.6441	0.6441	0.7321	0.7179	0.0685	0.0660	0.0704	**0.0720**	0.0692
	Accuracy	**0.8571**	0.8108	0.8108	0.8113	0.8146	0.6448	0.6219	**0.6463**	0.6462	0.6238

Table 4.4: Per-level precision, recall, F_1 measure and accuracy for PPI-VM dataset using HML-Boosting with top-down class prediction and HML-Boosting with bottom-up class prediction algorithms for the different choice of boosting iterations.

Level	**Measurement**	**HML-Boosting top-down**					**HML-Boosting bottom-up**				
		5 iter	**10 iter**	**20 iter**	**50 iter**	**100 iter**	**5 iter**	**10 iter**	**20 iter**	**50 iter**	**100 iter**
Level 1	Precision	0.7355	0.7266	0.7654	0.7641	**0.7661**	0.1716	0.1650	0.1672	0.1700	**0.1801**
	Recall	0.0445	0.0465	0.0619	0.0744	**0.0834**	0.7393	0.6603	0.6723	0.6943	**0.8357**
	F1	0.0838	0.0873	0.1146	0.1356	**0.1505**	0.2786	0.2640	0.2678	0.2731	**0.2963**
	Accuracy	0.8440	0.8440	0.8463	0.8476	**0.8487**	0.3850	0.4087	**0.4094**	0.4064	0.3626
Level 2	Precision	0.5859	0.7094	**0.7170**	0.7026	0.7000	0.1025	**0.1069**	0.1050	0.1066	0.1014
	Recall	0.6818	0.7217	**0.7651**	0.7446	0.7624	0.5588	0.5270	0.5527	**0.5750**	0.6517
	F1	0.6303	0.7155	**0.7403**	0.7230	0.7299	0.1732	0.1777	0.1764	**0.1799**	0.1755
	Accuracy	0.8151	0.8759	**0.8818**	0.8754	0.8781	0.6979	**0.7238**	0.7078	0.7031	0.6533
Level 3	Precision	0.6180	0.6598	**0.6797**	0.5989	0.6244	0.0596	0.0629	0.0621	**0.0638**	0.0611
	Recall	0.8088	0.8421	0.8365	0.8195	**0.8477**	0.5250	0.5063	0.5288	0.5569	**0.6045**
	F1	0.7006	0.7399	**0.7500**	0.6921	0.7191	0.1070	0.1119	0.1111	**0.1146**	0.1110
	Accuracy	**0.8618**	0.8427	0.8528	0.8170	0.8339	0.7277	**0.7503**	0.7369	0.7324	0.6990
Level 4	Precision	0.5238	0.7059	**0.8158**	0.6935	0.7356	0.0385	**0.0421**	0.0409	0.0414	0.0415
	Recall	0.6875	0.7059	0.8611	0.8431	**0.8767**	0.6839	0.6667	0.6868	0.7011	**0.7385**
	F1	0.5946	0.7059	**0.8378**	0.7611	0.8000	0.0730	**0.0793**	0.0772	0.0782	0.0786
	Accuracy	0.7619	0.8182	**0.8537**	0.8029	0.8025	0.5147	**0.5676**	0.5413	0.5385	0.5165

4.5 Summary

The functional classification of genes is an important and challenging problem in the area of functional genomics. In this chapter, we presented the HML-Boosting algorithm and applied it to the problem of gene function prediction. The HML-Boosting algorithm leverages the hierarchical structure of the classes. Hence, the classifiers that are built tend to be more efficient and effective compared to the flat classification approach. The incorporation of the hierarchical structure of the classes improves the performance of the predictions. In addition, the HML-Boosting algorithm is compared with the flat classification approach and the results of the experiments on four bio-molecular datasets showed that the HML-Boosting algorithm significantly outperforms the flat classification approach. In addition, the performance of the HML-Boosting algorithm using the top-down and the bottom-up class prediction approaches is evaluated.

CHAPTER 5

EXPLOITING LABEL DEPENDENCY FOR HIERARCHICAL MULTI-LABEL CLASSIFICATION

In this chapter, we propose HiBLADE (Hierarchical multi-label Boosting with LAbel DEpendency), a novel algorithm that takes advantage of not only the pre-established hierarchical taxonomy of the classes, but also effectively exploits the hidden correlation among the classes that is not shown through the class hierarchy, thereby improving the quality of the predictions made by hierarchical multi-label classification algorithm. According to our approach, first, the pre-defined hierarchical taxonomy of the labels is used to decide upon the training set for each classifier. Second, the dependencies of the children for each label in the hierarchy are captured and analyzed using Bayes method and instance-based similarity. The primary objective of the proposed algorithm is to find and share a number of base models across the correlated labels.

5.1 Motivation

Most of the existing research focuses on a “flat” classification approach, that operates on non-hierarchical classification schemes, where a binary classifier is constructed for each label separately. Reducing a hierarchical multi-label classification problem to a conventional classification problem allows the possibility to apply several existing methods. However, since the prediction of the class labels has to be performed independently, such transformations are not capable to exploit the interdependencies and correlations between the labels [21]. Moreover, the flat classification algorithm fails to take advantage of the information inherent in the class hierarchy, and hence may be suboptimal in terms of efficiency and effectiveness [40]. Therefore, some methods perform an additional step to correct inconsistent predictions.

In this chapter, we propose a hierarchical multi-label classification framework that takes the hierarchy constraint into account along with the labels correlation that goes beyond the scope of the hierarchical classification scheme and exploits such correlations in building a shared-model boosting classifiers. This is a novel approach to solve the problem since most of the existing approaches focus on learning models independently for each label while our approach learns models for correlated siblings together by using shared boosting models. By leveraging on the hierarchical structure of the classification scheme, the classifiers can be more efficient [40]. Notably, most of the state-of-the-art hierarchical multi-label classification methods are designed to focus mainly on taking into account the hierarchical structure of the classes by obeying the hierarchy constraint for the final predictions [11, 80, 100]. In our work, we attempt to make a fundamental advance in this area by explicitly exploiting the pre-defined hierarchical taxonomy of the classes and discovering new relationships among the classes, that are not shown through the hierarchical taxonomy, and making use of such knowledge in the learning process. Moreover, the majority of the existing approaches for the gene function prediction problem take a separate step to correct inconsistent predictions. While in our approach, the hierarchy constraint is met naturally during the classification process and no separate step is needed for the correction of the inconsistent predictions.

Figure 5.1 illustrates a simple example of exploiting label correlation. In this figure, there are three labels. These three labels are three siblings from the set of labels in the hierarchy. Initially, a base model is learned for each label independently of the other labels; then, we incorporate the predefined label correlation to build shared models pool. For instance, the decision function of the first label is based on the aggregations of the base models 1 and 2. Similarly, the decision function of the second label is based on the aggregations of the base models 2 and 3. While the decision function of the third label is based on the base model of the third label only.

The rest of the chapter is organized as follows: in the next section, we summarize our contributions for incorporating the correlation in the hierarchical multi-label learning process. Our novel method, HiBLADE, is presented in Section 5.3. In Section 5.4, we report experimental

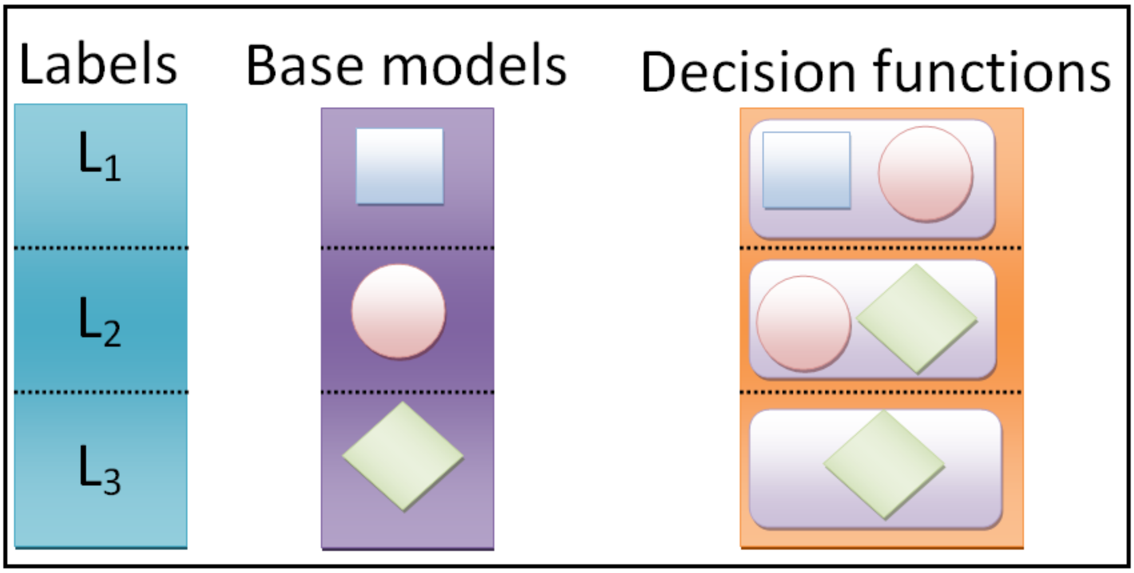


Figure 5.1: An illustration of the correlations between three sibling labels. In this example, a base model is learned for each label and from these models the decision functions are formed from one or more base models. The selection of the base models is controlled by the correlation between the corresponding labels.

results with several biomolecular datasets. Then, we conclude the chapter in Section 5.5.

5.2 Our Contributions

In this chapter, we propose, HiBLADE, a hierarchical multi-label classification framework for modeling the pre-defined hierarchical taxonomy of the labels as well as for exploiting the existing correlations between different labels, that are not explicitly given by the taxonomical classification of the labels, to facilitate the learning process [7].

To the best of our knowledge, *there is no work related to the hierarchical multi-label classification setting that addresses the exploitation of correlations between different labels other than the domain-based pre-established taxonomical classification of the classes*. Our intuition is that the domain-based pre-established taxonomical classification of the classes should be used as additional features while label dependencies are inferred from the data. Specifically, a novel approach to learn the label dependencies using a Bayesian framework and instance-based similarity is proposed. Bayesian framework is used to characterize the dependency relations among the labels represented by the directed acyclic graph (DAG) structure of the Bayesian network. By studying and identifying the undiscovered correlations among the classes, additional insights about the relationships among the classes are revealed. After that, classifiers for the children classes for each class in the hierarchy are built together along with the revealed correlations in the learning process. Additionally, in order to gain the inherent information in the class hierarchy, ancestor labels known from the pre-established taxonomy are used as additional features for learning the classifier for each label. As opposed to other hierarchical multi-label classification algorithms, our algorithm has the following advantages:

1. The underlying pre-defined taxonomy of the labels is explicitly expressed by adding the ancestor labels to be part of the feature vector, which allows us to gain further insights into the learning problem.

2. It is capable of exploiting correlations and interdependencies among the children of a particular label using the Bayesian framework and instance-based similarity.

3. The use of a shared boosting model for the children labels for each label in the hierarchy using the found correlations leads to effective results in an efficient manner.

5.3 HiBLADE Algorithm

Hierarchical multi-label learning aims to model and predict $p(child\ class \mid parent\ class)$. Our goal is to make use of hierarchical dependencies as well as the extracted dependencies among the labels y_k where $1 \leq k \leq \mathcal{L}$ and $(\uparrow y_k) = y_m$ such that for each example we can better predict its labels [7]. The problem then becomes how to identify and make use of such dependencies in an efficient way. For this work, we have adopted Naive Bayesian framework and instance-based similarity to encode the label dependencies.

Bayesian methods range from simple Naive Bayes, in which no dependency among the labels is assumed, to Bayesian networks, where the joint probability distribution for a large set of variables is encoded efficiently [51]. We focus on Naive Bayes algorithm due to its simplicity and computational efficiency.

5.3.1 Training Scheme

The training of each classifier is performed locally. During the classification, the classifier for each class will be presented only with the examples that are positive at the parent of the current class. Hence, the reached examples are positive examples to the current class and/or to the siblings of that class. In other words, the training for each classifier is performed by feeding as negative training examples, the positive examples at the parent of the current that are not positive examples at the current class. It should be noted that the selected negative training examples are the most informative negative examples for training. Moreover, since as we go down the hierarchy, fewer training examples are survived, this will be reflected in the efficiency of the algorithm, for both training and classification time.

5.3.2 Extending the Features

There are two types of dependencies among the labels. One is due to the parent-child relationship that is known from the taxonomy of classes. Based on this type of correlation, the labels of the classes at the levels higher than the level of the current class are added as additional input features for each instance that belong to that class. In other words, the feature vector for each example is extended to include the class labels of the levels higher in the hierarchy than the current level as explained in line 9 of Algorithm 3. More formally, the feature vector for each instance j that belongs to a class i will have the following form:

$$f_{i,j} =< x_{j,1}, ..., x_{j,d}, \bar{x}_{j,d+1}, ..., \bar{x}_{j,d+k} >$$

where $< x_{j,1}, ..., x_{j,d} >$ is the original feature vector and $< \bar{x}_{j,d+1}, ..., \bar{x}_{j,d+k} >$ are the additional features that are added based on the hierarchical structure. In other words, the additional features are the labels of the classes at levels higher in the tree than the level of the current class i.

Augmenting the ancestor labels to the original feature will provide important information that reflects the hierarchical structure of the classes above the current one. Interestingly, similar ideas have been exploited in the multi-label classification. In a multi-label classification setting, the labels of the correlated classes are used as additional features. For this purpose, in the multi-label setting, researchers have developed a variety of methods to find the correlated labels [21, 129, 130]. However, in our algorithm, the correlation between the classes is pre-defined through the hierarchical taxonomy of the classes. Such domain-driven information about the relationship among the classes can be utilized during the learning process. We exploited this knowledge through the employment of the labels of the classes that are at higher levels in the hierarchy as additional features.

Algorithm 3 $HiBLADE$

```
1:  Input: A pair $< \mathcal{Y}, \mathcal{L} >$ where $\mathcal{Y}$ is a tree-structured set of classes and $\mathcal{L}$ is the total
    number of classes of $\mathcal{Y}$.
2:  Output: For each class $y_l \in \mathcal{Y}$, the final composite classifier $y_l = sign[F_l(x)]$.
3:  Algorithm:
4:  for $i = 1, ..., \mathcal{L}$ do
5:    if class $i$ is a leaf class then
6:      Do nothing
7:    else
8:      Let $children(i) = y_{1i}, ..., y_{ki}$ be the $k$ children classes of $i$
9:      Form the new feature vectors by adding the labels of the classes at the higher levels
        to the current feature vectors.
10:     Learn classifiers for $children(i)$ using the shared models Algorithm
11:   end if
12: end for
```

5.3.3 Label Correlation

The other type of dependencies will be modeled through the use of the Bayesian structure and instance-based similarity as shown in line 10 of Algorithm 3. The local perspective of the correlation principle postulates that correlated sibling labels are more likely to have similar functions. In the local correlation principle, only information from the immediate label siblings are used for the seek of the predictions.

For each class i, we get the children classes of class i and we call $sharedModels$ algorithm shown in Algorithm 4. In each boosting iteration t, we search the entire pool for the best fitted model other than the model that was built directly for that label and its corresponding combination weights, the best fitted model is called h_l^t. We refer to the selected best fitted model as the candidate model. The chosen model h_c^t is then updated based on the following formula:

$$\gamma_{ij} = \frac{\epsilon_{ii}}{\epsilon_{ii} + \epsilon_{ji}} * \beta_{ij} \tag{5.1}$$

where ϵ_{ji} is the error results from applying model h_j^t on the examples in class i and ϵ_{ii} is the error results from applying the model h_i^t on the examples in class i. β_{ij} controls the proportional contribution of Bayesian-based and instance-based similarities. β_{ij} is computed as follows:

$$\beta_{ij} = \phi * b_{ij} + (1 - \phi) * s_{ij} \tag{5.2}$$

where b_{ij} is the Bayesian correlation between class i and class j, and it is estimated as $b_{ij} = |i \cap j|/|j|$, where $|i \cap j|$ is the number of positive examples in class i and class j and $|j|$ is the number of positive examples in class j. s_{ij} is the instance-based similarity between class i and class j. In the proposed method, s_{ij} is computed using the Euclidean distance between the positive examples in both classes as follows:

$$s_{ij} = \sqrt{\sum_l (i_l - j_l)^2} \tag{5.3}$$

where l is the corresponding feature in the two vectors. s_{ij} is normalized to be in the range of $[0, 1]$. ϕ is a threshold parameter that has a value in the range $[0, 1]$. Setting ϕ to zero means that only instance-based similarity is taken into consideration in the learning process. While setting it to one means that only Bayesian-based correlation is taken into consideration. On the other hand, any value of ϕ between 0 and 1 combines both types of correlation. In the next section, we show the results of the experiments we performed with the different values of ϕ to highlight the effect of each type of the correlation and the effect of the combination of them. It is important to emphasize that these computations are performed only for the class that is found to be the most useful class with respect to the current class.

In the general case, both classes, the current class and the candidate class, contribute to final prediction. In other words, any value of ϵ_{ji} other than 0, indicates the level of contribution from the candidate class. More specifically, if the error of the candidate class, ϵ_{ji}, is greater than the error of the current class, ϵ_{ii}, the value of γ_{ij} will be small, indicating that only a limited contribution of the candidate class is considered. In contrast, if the error of the current class, ϵ_{ii}, is greater than the error of the candidate class, ϵ_{ji}, then γ_{ij} will be high, and hence, the prediction decision will be dependable more on the candidate class. Finally, the models for the current class and the used candidate class are replaced by the new learned models. At the end, the composite classifiers F_c provide the prediction results.

For a given class k, we are interested in finding the class, l, that minimizes the error in Equation (5.1) ($\operatorname{argmin}_l \gamma_{kl}$). Our intuition is that correlated labels have some shared information. By reducing such redundant shared information during the learning process, the overall classification process efficiency will be improved. Algorithm 4 shows the details of the shared models algorithm. The shared models algorithm takes as input the children classes of a particular class together with the feature vectors for the instances that are positive at the parent class. These instances will form the positive and negative examples for each one of the children classes.

The algorithm begins by initializing a pool of M models, where M is the number of children classes, one for each class that is learned using ADABOOST. In our experiments, we used ADABOOST.MH as the base classifier. The description of this algorithm can be found in Chapter 4. The number of base models to be generated is determined by T. In each iteration t and for each label in the set of the children labels of a particular class, we search for the best fitted model, $h_l^t(x)$ and the corresponding combination weights, α_l^t.

Algorithm 4 $SharedModels$

```
Input: $D = \{(x_i, Y_i) : i = 1, ..., N\}$, where $x_i \in X$ is a feature vector for instance $i$ and $Y_i \subset Y$ is the set of labels associated with $x_i$ and $M$ is the number of labels under study. $\phi$: a threshold parameter.
Output: $y_c = sign[F_c(x)]$
Algorithm:
Set $F_c(x) = 0$ for each label $c = 1, .., M$
Initialize a pool of candidate shared models: $SM_p = h_1(.), ..., h_M(.)$ where $h_i(.)$ is a model learned on the label $i$ using the boosting-type algorithm.
for $t = 1, ..., T$ do
    for $c = 1, ..., M$ do
        Find $\alpha_l^t$ and $h_l^t \in SM_p$ where $c \neq l$ that minimize the loss function on label $c$.
        $F_c(x) = F_c(x) + h_c^t(x) * \alpha_c^t * (1 - \gamma_{cl}) + h_l^t(x) * \alpha_l^t * \gamma_{cl}$
        Replace $h_c^t(x)$ and $h_l^t(x)$ in $SM_p$ with the new learned models using the boosting-type algorithm.
    end for
end for
```

The contribution of the selected base model, $h_l^t(x)$, to the overall classifier, $F_c(x)$, depends on the current label. In other words, if the error, $\epsilon_{ji} = 0$ of the candidate classifier is 0, this will be a perfect model for the current label. Hence, Equation (5.1) will be reduced to $\gamma_{ij} = \beta_{ij}$. In this case, the contribution of that model depends on the level of correlation between the candidate class and the current one. If the current model is a perfect model, i.e., the error $\epsilon_{ii} = 0$, then Equation (5.1) will be reduced to $\gamma_{ij} = 0$, which means that for the current iteration, there is no need to look at any other classifier.

5.4 Experimental Results

In this section, we show the experimental results on four biomolecular datasets.

5.4.1 Experimental Setup

We analyzed the performance of the proposed framework at each level of the FunCat taxonomy, and we also compared the proposed method with four other methods that follow the local classifier approach, namely, HBAYES-CS, HTD, TPR and TPR-w. HBAYES-CS, TPR and TPR-w are described in the Literature Review Chapter. HTD (Hierarchical Top-Down) method is the baseline method that belongs to the local classifier strategy and performs hierarchical classification in a top-down fashion. Since HiBLADE also belongs to the local classifier strategy, it is fair to have a comparison against a local classifier approach that does not consider any type of correlations between the labels. We also analyzed the effect of the proper choice of the threshold ϕ on the performance of the algorithm. The setup for the experiments is summarized as follows:

- Flat: This is the baseline method that does not take the hierarchical taxonomy of the classes into account and does not consider label dependencies. A classifier is built for each class independently of the others. We used AdaBoost as the base learner to form a baseline algorithm for the comparison with the other methods.

- $HiBLADE_I$: The proposed Hierarchical Multi-Label Classification algorithm that considers Instance-based similarities only. Here ϕ is set to zero.

- $HiBLADE_B$: The proposed Hierarchical Multi-Label Classification algorithm that considers classes correlation based on Bayesian probabilities only. Here ϕ is set to one.

- $HiBLADE_C$: The proposed Hierarchical Multi-Label Classification algorithm that considers a combination of both instance-based similarity and classes correlation. Here ϕ is set to 0.5.

5.4.2 Results and Discussion

First, we performed a level-wise analysis of the F-measure of the FunCat classification tree on the four datasets. In measuring the level-wise performance, level 1 reflects the root nodes while all other classes are at depth i, where $2 \leq i \leq 5$. We show the results for the top four levels in the hierarchy for the proposed method and the flat method. Moreover, we show the performance of the proposed framework with different ϕ values while setting the number of boosting iterations to 50 iterations.

Tables 5.1, 5.2, 5.3 and 5.4 show the results of per-level evaluation for Gene-Expr, PPI-BG, Pfam-1 and PPI-VM datasets, respectively. The best results for each level are highlighted.

The proposed algorithm outperforms the flat classification method in most of the cases with significant improvements in the performance measurements.

Table 5.1: Per-level F_1 measure for Gene-Expr dataset using Flat, $HiBLADE_I$, $HiBLADE_C$ with $\phi = 0.5$ and $HiBLADE_B$ for boosting iterations=50.

Level	**Flat**	$HiBLADE_I$	$HiBLADE_C$	$HiBLADE_B$
		$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$
1	**0.3537**	0.2328	0.2301	0.2336
2	0.1980	0.4052	**0.4427**	0.4094
3	0.1000	0.3575	**0.4019**	0.3742
4	0.2000	0.2714	**0.3598**	0.2874

Table 5.2: Per-level F_1 measure for PPI-BG dataset using Flat, $HiBLADE_I$, $HiBLADE_C$ with $\phi = 0.5$ and $HiBLADE_B$ for boosting iterations=50.

Level	**Flat**	$HiBLADE_I$	$HiBLADE_C$	$HiBLADE_B$
		$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$
1	0.0808	0.2014	0.1833	**0.2052**
2	0.0267	0.6904	0.6984	**0.6998**
3	0.0001	0.6446	0.6304	**0.6520**
4	0.0001	0.6743	0.6454	**0.6747**

These results also indicate that the deeper the level the better the performance of the proposed algorithm compared to the flat classification method. For example, in all of the datasets, the proposed algorithm outperformed the flat classification method in all the levels that are higher than level 1. This result is consistent with our understanding of both of the classification schemes. In other words, the proposed method and the flat classification method have a similar learning procedure for the classes at the first level. However, the proposed method achieve better results for the deeper levels in the hierarchy.

To get more insights into the best choice of threshold ϕ, we compare hierarchical precision, hierarchical recall, hierarchical F_1^{μ} measure and hierarchical F_1^{M} measure for Gene-Expr, PPI-BG, Pfam-1 and PPI-VM datasets for $\phi = 0.0, 0.5$ and 1.0, respectively for 50 boosting iterations. The choice of ϕ threshold controls the contribution of instance-based, class depen-

Table 5.3: Per-level F_1 measure for $Pfam-1$ dataset using Flat, $HiBLADE_I$, $HiBLADE_C$ with $\phi = 0.5$ and $HiBLADE_B$ for boosting iterations=50.

Level	**Flat**	$HiBLADE_I$	$HiBLADE_C$	$HiBLADE_B$
		$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$
1	**0.1133**	0.0924	0.0827	0.1085
2	0.0267	0.8524	**0.8702**	0.7273
3	0.1000	0.7473	**0.7946**	0.6824
4	0.2222	0.5122	**0.5135**	0.5085

Table 5.4: Per-level F_1 measure for PPI-VM dataset using Flat, $HiBLADE_I$, $HiBLADE_C$ with $\phi = 0.5$ and $HiBLADE_B$ for boosting iterations=50.

Level	**Flat**	$HiBLADE_I$	$HiBLADE_C$	$HiBLADE_B$
		$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$
1	**0.1631**	0.1266	0.1029	0.1193
2	0.1786	0.6033	**0.6758**	0.6601
3	0.0001	0.5802	0.6822	**0.6957**
4	0.0001	**0.6931**	0.5246	0.5417

dencies or a combination of both of them to the classification task. Table 5.5 shows the results of the comparisons. The best results are highlighted. As shown in Table 5.5, the combination of Bayesian-based correlation and instance-based similarity achieved the best performance results in most of the cases. For example, six of the highest performance values, in general, in this table are achieved when $\phi = 0.5$.

Furthermore, we conducted comparisons of hierarchical F-measure with HBAYES-CS, HTD, TPR and TPR-w methods. HBAYES-CS is using Guassian SVMs as the base learners, while HTD, TPR and TPR-w are using Linear SVMs as the base learners. Figure 5.2 shows the F-measure of the different methods. By exploiting the label dependencies, the classifiers performance are effected positively. Our results show that the proposed algorithm significantly outperforms the local learning algorithms. Although there is no clear winner among the different versions of HiBLADE algorithm, HiBLADE always achieved significantly better results compared to the other methods.

Table 5.5: Comparison of hierarchical precision, hierarchical recall, hierarchical F_1^M and hierarchical F_1^μ measures of HiBLADE for Gene-Expr, PPI-BG, Pfam-1 and PPI-VM datasets using $\phi = 0.0, 0.5 and 1.0$, respectively for boosting iterations=50.

Measure	**Gene-Expr**			**PPI-BG**		
	$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$	$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$
hP	0.820	0.808	**0.826**	0.878	**0.924**	0.875
hR	**0.644**	0.630	0.627	0.662	0.686	**0.701**
hF_1^M	**0.702**	0.689	0.692	0.735	**0.769**	0.756
hF_1^μ	**0.722**	0.708	0.712	0.755	**0.787**	0.778

Measure	**Pfam-1**			**PPI-VM**		
	$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$	$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$
hP	0.763	0.836	**0.875**	0.716	**0.748**	0.719
hR	0.625	**0.663**	0.637	0.542	0.551	**0.557**
hF_1^M	0.669	**0.720**	0.714	0.590	**0.605**	0.601
hF_1^μ	0.687	**0.740**	0.737	0.617	**0.635**	0.628

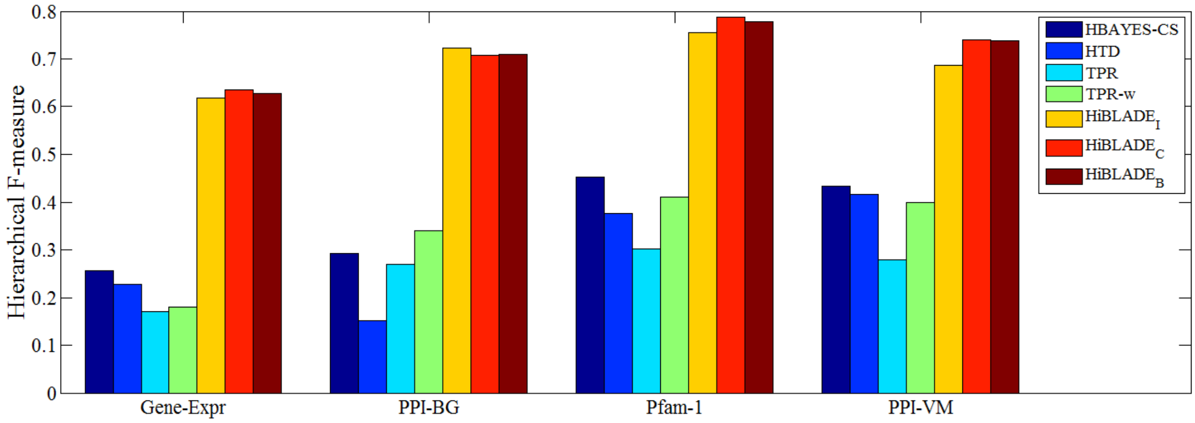


Figure 5.2: Hierarchical F-measure comparison between HBAYES-CS, HTD, TPR, TPR-w, $HiBLADE_I$, $HiBLADE_C$ and $HiBLADE_B$. For the $HiBLADE$ algorithm, the number of boosting iterations is 50 and $\phi = 0.5$ for $HiBLADE_C$.

5.5 Summary

Hierarchical multi-label classification is a challenging research problem that arises in many real-world applications. In this chapter, we presented a hierarchical multi-label classification framework for incorporating information about the hierarchical relationships among the labels as well as the label-label and the instance-instance correlations. The experimental results showed that the proposed HiBLADE algorithm outperforms the flat classification method and the local classifiers method that builds independent classifier for each class without considering correlation among the classes.

CHAPTER 6

A BAYESIAN INTEGRATION MODEL OF HETEROGENEOUS DATA SOURCES

High-throughput technologies make available increasing quantities of data of different types. This opens up new opportunities to accurately predict the functional annotation of gene/protein on a large scale, which involves evidences from multiple types of data. However, no single biological data source analysis can fully unravel the complexities of hierarchical gene function classification. Therefore, the integration of multiple data sources is required to acquire a precise picture of the role of genes in the living organisms through uncovering novel biology in the form of previously unknown functional annotations. Unlike most of the previous works, which mostly look at a single data source for gene function prediction, we explore the integration of heterogeneous data sources for genome-wide gene function prediction. We address this problem with a novel **Hi**erarchical **B**ayesian **iN**tegration algorithm *HiBiN*.

6.1 Motivation

Recently, diverse high-throughput functional genomic data such as protein-protein interaction, gene expression microarrays and genome sequences have become broadly available. Since the functions of significant number of genes are still unknown, such high throughput data can play a key role in assigning accurate functional annotations on a large scale through computational prediction [76]. Providing accurate predictions can advance experimental studies by providing specific hypothesis for targeted experimental testing [117].

Learning reliable classification models for several hierarchically structured functional classes from a single dataset is often hard due to several complex issues including noise in the data, low relevance of the dataset for some functional classes and an insufficient number of training examples for building accurate classification models [89]. Although a wide variety of functional genomic data is available, there is no work in exploiting the full potential of these data in the context of hierarchical prediction of gene functions. This is because the incorporation of heterogeneous functional data in an integrated framework is a key challenge in modern systems biology. Several reasons contribute to the difficulty of the integration of heterogenous data. The variation in the reliability of experimental results among datasets and technologies and the heterogeneity of the datasets are two of the most important reasons [84].

Taking these issues into account, it is hard to believe that functional genomics can be accurately predicted by depending on expression level of genes only or any single source of data. Although a wide variety of functional genomic data is available, there is no research work in exploiting the full potential of these data in the context of hierarchical prediction of gene functions. Considering all the above issues, the problem of accurately predicting the gene functions from multiple sources is a challenging problem. To the best of our knowledge, there is no existing that handles all of the above challenges in a single framework. In this chapter, we tackle these problems in a unified framework.

6.2 Our Contributions

To address the above challenges, to accurately predicting hierarchical gene functions from diverse functional data, we propose **Hi**erarchical **B**ayesian **iN**tegration algorithm *HiBiN*, a flexible probabilistic framework for integrated analysis of multiple heterogeneous biological data [6]. The system is based on a Bayesian approach for the integration of the heterogenous data sources where posterior belief is used to assign class memberships to samples. In other words, evidences from diverse data sources are combined in a Bayesian reasoning framework to predict whether a gene is annotated with a specific function or not.

To our knowledge, there is no work that addresses all of the above challenges in a single framework. More specifically, HiBiN is a new scheme that aims at achieving multi-source integration of diverse genome data for hierarchical gene function classification. The major contributions of our new scheme can be summarized as follows:

1. To predict multi-labels for genes, the HiBiN allows the prediction of more than one functional class simultaneously.

2. To maintain the hierarchy constraint, the parent-child inter-relationships are exploited during the training and the testing phases.

3. To handle the imbalance class issue, the positive and negative examples for each classifier are chosen based on the hierarchical taxonomy of the classes.

4. To handle the problem of source diversity, the proposed framework integrates multiple data sources to characterize the genes effectively.

5. To minimize the computational cost for training the classifiers, HiBiN filters out unsuitable genes from percolating to lower levels in the hierarchy.

The rest of the Chapter is organized as follows. Section 6.3 presents the preliminaries we use through the discussion. The proposed algorithm is presented in Section 6.4. The results and evaluation are presented in Section 6.5. Finally, Section 6.6 concludes the chapter.

6.3 Preliminaries

First, we present the formal notations and terminologies that are being used throughout the following discussion. Let $\mathcal{G} = \Re^d$ be the d-dimensional input space of genes and $\mathcal{Y} = \{y_1, y_2, ..., y_{\mathcal{L}}\}$ be the finite set of $\mathcal{L}$ possible labels. The hierarchical relationships among classes in $\mathcal{Y}$ are defined as follows: Given $y_1, y_2 \in \mathcal{Y}, y_1$ is the ancestor of y_2, denoted by $(\uparrow y_2) = y_1$, if and only if y_1 is a superclass of y_2. We denote by $(\downarrow y_1) = y_2$, the set of children classes of class y_1. The set of labels $\mathcal{L}$ are structured according to a hierarchical structure T (in our case according to a FunCat tree). A gene g is represented with a vector of d different features. The features could represent the presence or absence of interactions with other genes products or gene expression levels in d different conditions.

Let a hierarchical multi-label training set $\mathcal{M} = \{< g_1, \mathcal{Y}_1 >, ..., < g_n, \mathcal{Y}_n >\}$, where

$g_i \in \mathcal{G}$ is a feature vector for gene i and $\mathcal{Y}_i \subseteq \mathcal{Y}$ is the set of labels associated with g_i, such that $y_i \in \mathcal{Y}_i \Rightarrow y'_i \in \mathcal{Y}_i, \forall(\uparrow y'_i) = y_i$. A gene g is assigned to one or more functional classes. The assignments are represented through a vector $\mathcal{F}_g = (y_1, y_2, ..., y_{\mathcal{L}}) \in \{0,1\}^{\mathcal{L}}$, where $y_i = 1$ if the gene g is annotated with class y_i while $y_i = 0$ otherwise.

Problem Statement: Given a set of datasets $\{\mathcal{M}_1, ..., \mathcal{M}_Q\}$ that describe the same set of genes but from different aspects, where $\mathcal{M}_i = \{< g_1, \mathcal{Y}_1 >, ..., < g_n, \mathcal{Y}_n >\}$, $g_i \in \mathcal{G}$ and $\mathcal{Y}_i \in \mathcal{Y}$. The goal is to integrate the different datasets so that predictions are improved. The final output will be a vector $\mathcal{F}_g = (y_1, y_2, ..., y_{\mathcal{L}}) \in \{0,1\}^{\mathcal{L}}$, where $y_i = 1$ if and only if $(\uparrow y_i) = 1$.

6.3.1 Training Scheme

Relying on the hierarchical topology of the classification scheme, the training of each of the binary classifiers, is performed locally. At each classifier, we need to carefully choose the set of positive and negative examples. The positive training examples of a non-leaf class, c_j, is a superset of the union of sets of positive training examples of all of its offsprings (leaf) classes [40]. In other words, during classification, the classifier at each class will only be presented with examples that are positive at the parent class of the current class. Hence, the reached examples are positive examples to the current class and/or to the siblings of that class. The training for each classifier is performed by feeding as negative training examples, the positive examples at the parent of the current class that are not positive examples at the current class. It should be noted that the selected negative training examples are the most informative negative examples for training.

6.4 The proposed HiBiN Algorithm

In this section, we present the details of the HiBiN algorithm, a general framework that uses Bayesian reasoning to integrate heterogeneous data sources for accurate gene function prediction. The system formally uses posterior probabilities to assign class memberships to samples using multiple data sources while maintaining the hierarchical constraint that governs the annotation of genes [6].

6.4.1 AdaBoost for Estimating Conditional Probabilities

Bayesian integration is based on defining probability models for the independent data sources. We used boosting-based classifiers for generating the probability models. Given $\hat{\phi}(g,y) = \sum_{s=1}^{S} \hat{\phi}_s(g,y)$ for $y \in \mathcal{Y} = \{y_1, y_2, ..., y_m\}$, Adaboost minimizes $E(e^{-y\hat{\phi}(g,y)})$, where $y = \{-1, 1\}$, since $E(e^{-y\hat{\phi}(g,y)})$ is minimized at $\hat{\phi}(g,y) = \frac{1}{2}log\frac{P(y=1|g)}{P(y=-1|g)}$. Hence,

$$P(g|y=1) = \frac{e^{\hat{\phi}(g,y)}}{e^{-\hat{\phi}(g,y)} + e^{\hat{\phi}(g,y)}} \tag{6.1}$$

$$P(g|y=-1) = \frac{e^{-\hat{\phi}(g,y)}}{e^{-\hat{\phi}(g,y)} + e^{\hat{\phi}(g,y)}} \tag{6.2}$$

These equations give the normalized likelihood value that the gene (g_i) from dataset j is annotated with label k, $p(g_i, \mathcal{M}_j|y_k)$, where $i \in \{1, ..., N\}$, $j \in \{1, ..., Q\}$ and $k = 1, ..., \mathcal{L}$ classes. In our experiments, we used ADABOOST.MH as the base classifier. The description of this algorithm can be found in Chapter 4.

6.4.2 Bayesian Inference

Bayesian statistical inference is an approach that allows to make probabilistic inferences from the data. Since the data are treated as random variables, integration or summation will allow the removal of nuisance parameters [125]. Through the use of Bayesian statistics, prior information are utilized and new models for integrating diverse data sources can be developed. Let Θ denote the set of unknown parameters, where Θ is of n dimensions, $\Theta = (\theta_1, \theta_2, ..., \theta_n)$ and let Z denote the observed data resulting from an experimental study. The likelihood function is defined as a statistical model $P(Z|\Theta)$ that reflects our belief about the observed data Z given the unknown parameter Θ [97].

The Bayesian inference is made by obtaining the posterior distributions of the unknown quantities of interest. Using Bayes theorem, the posterior distribution is obtained as follows:

$$P(\Theta|Z) = \frac{P(Z|\Theta).P(\Theta)}{p(Z)} \tag{6.3}$$

$$p(\Theta|Z) = \frac{p(Z|\Theta).p(\Theta)}{\sum_j p(Z|\Theta).p(\Theta)} \tag{6.4}$$

where $P(\Theta)$ is the prior probability of Θ that expresses our belief about a parameter Θ before we see any data.

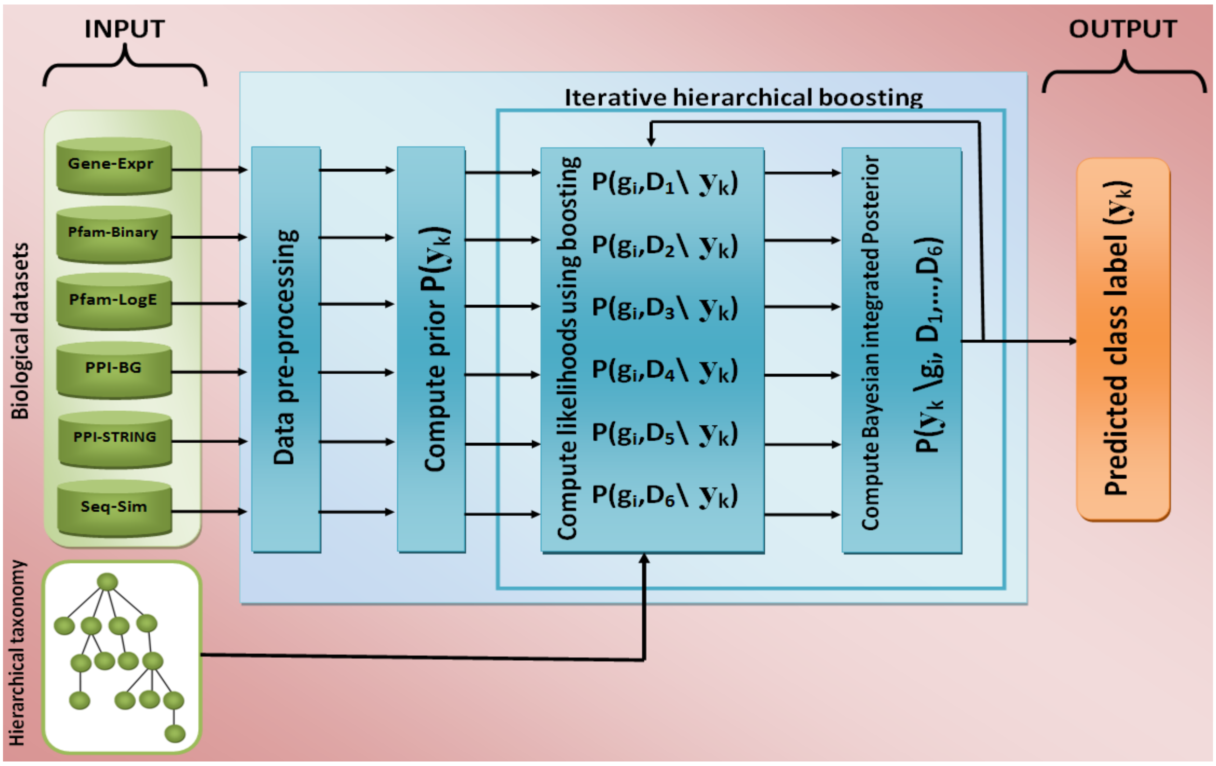

INPUT
OUTPUT
Biological datasets
Gene-Expr
Pfam-Binary
Pfam-LogE
PPI-BG
PPI-STRING
Seq-Sim
Hierarchical taxonomy
Iterative hierarchical boosting
Data pre-processing
Compute prior $P(y_k)$
Compute likelihoods using boosting
$P(g_i,D_1\backslash\ y_k)$
$P(g_i,D_2\backslash\ y_k)$
$P(g_i,D_3\backslash\ y_k)$
$P(g_i,D_4\backslash\ y_k)$
$P(g_i,D_5\backslash\ y_k)$
$P(g_i,D_6\backslash\ y_k)$
Compute Bayesian integrated Posterior
$P(y_k \backslash g_i, D_1,\ldots,D_6)$
Predicted class label (y_k)

Figure 6.1: The proposed analysis pipeline to obtain integrated posterior probabilities across the heterogeneous data sources in the context of FunCat hierarchy.

6.4.3 Bayesian Integration and Classification

For our experiments, the probabilities obtained from boosting classifiers are the likelihood of observing gene (g_i) associated with dataset $\mathcal{M}_j$) for a specific functional class. However, the posterior probability is the important one:

$$P(y_k|g_i, \mathcal{M}_j) = \frac{P(g_i, \mathcal{M}_j|y_k).P(y_k)}{\sum_{k=1}^{m} P(g_i, \mathcal{M}_j|y_k).P(y_k)} \tag{6.5}$$

Thus, for each sample, a set of probabilities are obtained which sum to one.

Bayes formula can be used to integrate and compute posterior probabilities for multiple datasets. With the assumption that each dataset is independent but describe the same genes, the integrated likelihood is computed as the product of the individual likelihoods:

$$P(g_i, \mathcal{M}_1, ..., M_Q|y_k) = P(g_i, \mathcal{M}_1|y_k) \times ... \times P(g_i, \mathcal{M}_Q|y_k) \tag{6.6}$$

Where Q is the number of the datasets. Given that the datasets share common samples, the datasets likely have some correlation. The probability of a particular gene is given by

$$P(y_k|g_i, \mathcal{M}_1, ..., \mathcal{M}_Q) = \frac{\prod_{j=1}^{Q}[P(g_i, \mathcal{M}_j|y_k)]P(y_k)}{\sum_{k=1}^{m}[\prod_{j=1}^{Q} P(g_i, \mathcal{M}_j|y_k)]P(y_k)} \tag{6.7}$$

The HiBiN algorithm is explained in Algorithm 5. Each non-root class, y_j, has a binary classifier $\hat{\phi}_j$ that is associated with it. The classifier should act as a “filter” to prevent unsuitable examples from spreading out to the lower levels in the hierarchy. Hence, only the test genes that a classifier $\hat{\phi}_j$ decides to belong to y_j are passed to all the binary classifiers corresponding to the children classes of y_j. While the genes that classifier $\hat{\phi}_j$ sees not to belong to y_j are ”blocked” and no further analysis is carried out.

The HiBiN algorithm works by first computing the prior probabilities for all the classes. Next, the children classes of the current class are extracted; then ADABOOST.MH will be called on these classes. Next Bayesian posteriors will be calculated for the children classes of the current class using Algorithm 6. The analysis pipeline of the HiBiN algorithm is described in Figure 6.1 with respect to the yeast datasets and is easily generalizable. The number of datasets used in our experiments was six.

Algorithm 5 $HiBiN(T, r, \mathcal{M}_1, ..., \mathcal{M}_Q)$

```
Input: $T$: is a tree-structured set of classes
       $r$: is the root of T
       $\mathcal{M}_1, ..., \mathcal{M}_Q$: are the input datasets
Output: Predicted class labels: $f_g \in \{0, 1\}^{\mathcal{L}}$
Algorithm:
Compute the prior probabilities, $P(y_k), k = 1, ..., \mathcal{L}$.
Let $\downarrow (r) = \{c_{1r}, ..., c_{kr}\}$ be the $k$ children classes of $r$
for $j = 1, ..., Q$ do
    $\mathcal{A} = \{g_i, C_i\}_{i=1}^{n}, n \leq N, C_i \in (\downarrow r)$ and $g_i \in \mathcal{A}_{(\uparrow r)}$
    $\hat{\phi}_{M_i}(g, y)$=ADABOOST.MH($\mathcal{A}$)
end for
$/*$ Run Algorithm 6 for computing posterior probabilities $*/$
$f_{g_i} = BIN(P(y_k), \{\hat{\phi}_{M_j}(g, y)\}_{j=1}^{Q})$ for $k \in \downarrow r$
for $c = 1, ..., v$ do
    Let $T_c$ be the subtree of T rooted at $y_{cr}$
    $/*$ Run HiBiN recursively on the children classes $*/$
    $f_{g_i} = HiBiN(T_c, y_{cr}, \{\mathcal{M}_j\}_{j=1}^{Q})$
end for
```

In algorithm 6, to make a decision whether a gene is annotated with a particular class or not, we compute the posterior probability for each gene. We consider the following Bayesian decision rule:

$$Decide\ y_k\ if\ P(y_k|g_i)\ >\ P(y_k^{'}|g_i);\ otherwise\ decide\ y_k^{'}$$

where $y_k^{'}$ is the complement of the class y_k, i.e., $y_k^{'} = 0$. The computation of $P(y_k^{'}|g_i)$ is similar to the computation of $P(y_k|g_i)$, except that here we compute the probability of the negative class ($y_k^{'}$) and the likelihood given by $P(g|y_k^{'} = 0)$.

We followed the same schemes for the training and the testing of the classifiers as the schemes used for the HML-BOOSTING approach.

Algorithm 6 $BIN(P(y), \hat{\phi}_{M_1}(g,y), ..., \hat{\phi}_{M_Q}(g,y))$

```
Input: $P(y)$: Prior probabilities
        $\{\hat{\phi}_{M_j}(g,y)\}_{j=1}^{Q}$: the final boosting hypothesis
Output: $f_{g_i}$ The prediction of classes for $g_i$.
Algorithm:
for $j = 1, ..., Q$ do
    $P(g|y=1) = \frac{e^{\hat{\phi}(g,y)}}{e^{-\hat{\phi}(g,y)}+e^{\hat{\phi}(g,y)}}$
    $P(y_k|g_i, \mathcal{M}_1, ..., \mathcal{M}_Q) = \frac{\prod_{j=1}^{Q}[P(g_i,\mathcal{M}_j|y_k)]P(y_k)}{\sum_{k=1}^{m}[\prod_{j=1}^{Q}P(g_i,\mathcal{M}_j|y_k)]P(y_k)}$
    Compute the final decisions using the Bayesian decision rule
end for
```

6.5 Experimental Results

6.5.1 Baseline Comparison Methods

To understand the potential effect of data integration, we have constructed two baseline algorithms: flat integration and hierarchical single source. In flat integration, the hierarchy constraint is not taken into consideration when building the classifiers and the positive and negative examples used to build the classifier for each functional class does not follow any constraint. The integration is performed using the same framework we used for HiBiN method. In hierarchical single source, the hierarchical scheme of the classes is taken into account during building the classifiers. However, no integration is performed here. Therefore, hierarchical single source method is applied on each dataset separately.

In order to assess the generalization capabilities of HiBiN, 3-fold cross validation with 100 boosting iterations have been adopted. Moreover, we have tested two strategies to compute the prior probabilities. In one strategy, we computed the prior probabilities without considering the hierarchical structure of the classes. In other words, the prior probabilities are computed by considering the set of annotated genes at each functional label with respect to the total number of genes. This version of the algorithm is called HiBiN$_a$, where a stands for all. In the second variation, prior probabilities are computed while the hierarchical scheme of the classes is taken into account. In other words, the prior probabilities are computed by considering the set of annotated genes at each functional label with respect to the number of genes that are annotated with the parent functional class of the label of interest. This version of the algorithm is called HiBiN$_p$, where p stands for parent.

6.5.2 Real-world Datasets

Six different data sources of yeast biomolecular data were integrated. The datasets include two types of protein domain data (Pfam-Binary and Pfam-LogE), gene expression data (Gene-Expr), predicted and experimentally supported protein-protein interaction data (PPI-STRING and PPI-BioGRID) and pairwise sequence similarity data (Seq-Sim). Datasets characteristics (before preprocessing them for the integration) are summarized in Table 6.1.

Table 6.1: The characteristics of the six bio-molecular datasets used in our experiments.

Dataset	**Description**	**Samples**	**Features**
Gene-Expr	Gene expression data	4532	250
PPI-BG	PPI data from BioGRID	4531	5367
Pfam-Binary	Protein domain binary data	3529	4950
Pfam-LogE	Pfam protein domains with log E values computed by the HMMER software toolkit	3529	5724
PPI-STRING	PPI data from Von Mering experiments	2338	2559
Seq-Sim	Smith and Waterman log-E values between all pairs of yeast sequences	3527	6349

For the integration purposes, we considered only common yeast genes to all the datasets. Moreover, for each dataset, we selected FunCat annotated genes for the classes with at least 20 positive examples so that the set of positive examples used for training is not too small. This pre-processing yielded 1901 yeast genes annotated to 168 FunCat classes distributed across 16 trees and 5 hierarchical levels. We added a “dummy” root node to obtain a single rooted-tree from all of the FunCat forests.

The comparison between HiBiN algorithm and the flat method is based on the “per-class” F_1-measure that is obtained by averaging the F_1-measure for all the classes in the FunCat hierarchy for each dataset. In other words, an overall F_1-measure is obtained by computing the F_1-measure for each class separately and then averaging them across all the classes. Furthermore, to get more insights into the performance of the HiBiN algorithm, we performed a level-wise analysis of the precision, recall and F_1-measure on the baselines and the proposed algorithm. In measuring the level-wise performance, level 1 represents the top level in the hierarchy while level 5 is the deepest level in the hierarchy.

Table 6.2 shows the minimum, maximum and average number of class membership at each level in the hierarchy. Surprisingly, there are few classes at the higher levels in the hierarchy having low number of genes. This explains the low performance of the flat approach, compared to the hierarchical approaches as the classes imbalance membership issue may occur at any level in the hierarchy.

Table 6.2: Minimum, maximum and average number of class membership at each level in the FunCat heirarchy for yeast genes.

	Level 1	**Level 2**	**Level 3**	**Level 4**	**Level 5**
Min	24	20	20	20	20
Max	749	348	260	211	36
Average	312	115	63	56	26

6.5.3 Results and Discussion

In our experiments, after pre-processing the datasets as described above, the prior probabilities are computed for each label in the FunCat scheme. Prior probabilities are computed based on the number of positive examples annotated with each label obtained from all the datasets. Next, boosting classifiers are used to obtain the likelihoods from the different datasets. Finally, the Bayesian posteriors are computed to obtain the probabilities that are used to make the final decision about the classification.

We mentioned earlier that one of the complex issues in the hierarchical classification problems is the imbalance between positive and negative examples at each class. The insufficient number of training examples leads to the difficulty in building accurate classification. Figure 6.2 shows the number of positive examples at each level in the FunCat hierarchy obtained for the yeast genes after preprocessing the datasets as mentioned earlier. The blue bars represent the number of positive examples at each functional class at a particular level. Note that the number of positive instances at level 2 is more than the number of positive instances at level 1. This is acceptable as a gene may be annotated with multiple functional classes at the same level. Hence, a gene may be counted more than one time, one for each class its associated with. On the other hand, the red bars represent the number of unique positive examples from all the functional classes at a particular level; therefore, a gene will be counted only one time if it is annotated with any of the functional classes in that level. The figure shows that going down in the hierarchy, the number of positive examples (genes that are annotated with the more specific functions) decreases dramatically, which affects the learning capability of the classification models.

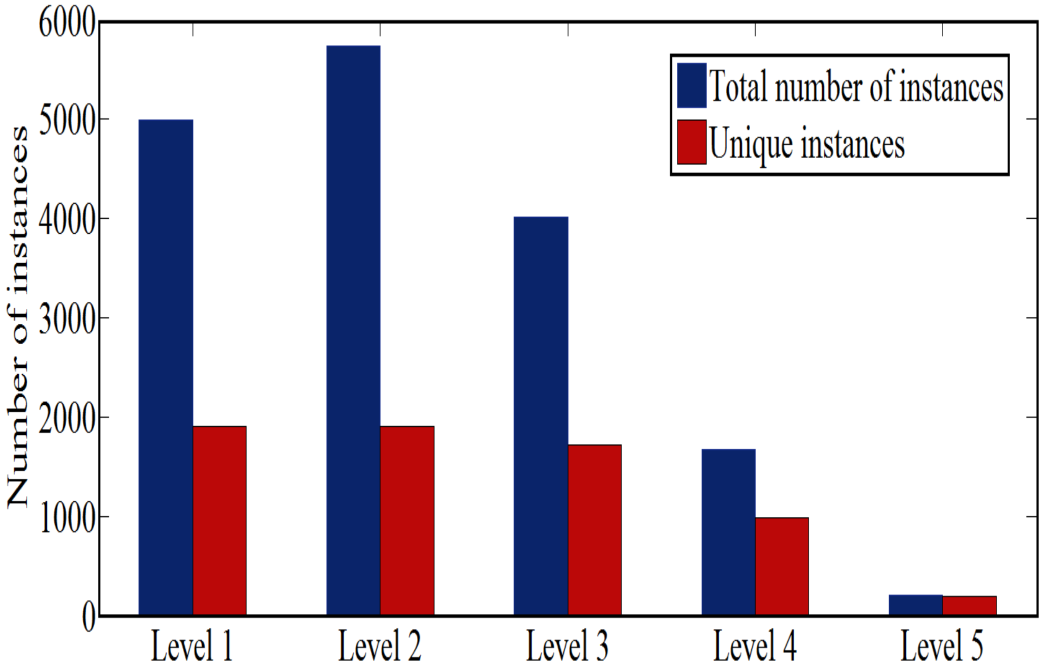


Figure 6.2: Level-wise class membership statistics obtained from the six datasets. The blue bars represent the number of positive examples at each functional class at a particular level, while the red bars represent the number of unique positive examples from all the functional classes at a particular level.

Table 6.3 summarizes the results of the comparisons of the average per-class precision, recall and F-measure for the hierarchical single source, flat integration and HiBiN method. We can observe that HiBiN algorithm brings considerable improvement over the baseline methods. In particular, HiBiN, with its two variations, outperforms the baseline methods in terms of the per-class precision, recall and F-measure. Comparing the flat integration with the hierarchical single source method, there is no clear trend of the winner. For example, the datasets PPI-BG and PPI-STRING performed better than the flat integration in terms of the per-class precision and recall, while flat integration performed slightly better on these datasets, and the other datasets, in terms of per-class F-measure.

Table 6.3: Average per-class precision, recall and F-measure obtained from hierarchical single source (on each dataset separately), flat integration and HiBiN methods.

Single Source			
Dataset	**Precision**	**Recall**	**F-measure**
PPI-BG	0.5211	0.3176	0.4081
PPI-STRING	0.4882	0.3030	0.3823
Pfam-Binary	0.2977	0.1035	0.3901
Pfam-LogE	0.2244	0.2076	0.3837
Gene-Expr	0.2301	0.2107	0.3805
Seq-Sim	0.2136	0.2074	0.3545
Integration methods			
Flat	0.3592	0.307	0.4521
HiBiN$_a$	**0.7436**	0.4535	0.6175
HiBiN$_p$	0.7083	**0.4779**	**0.6222**

In Table 6.4 we compare the hierarchical precision, hierarchical recall, hierarchical F-micro (hF_1^M) and hierarchical F-macro (hF_1^μ) measures for both hierarchical single source and HiBiN methods. Note that hierarchical precision and hierarchical recall are not applicable to flat integration method as flat integration does not take the hierarchy constraint into account. Hence, the obtained predictions may be inconsistent with the hierarchy constraint. As we can observe, HiBiN achieved the best results in terms of all hierarchical performance metrics. The two variations of HiBiN have similar performances, which indicates that incorporating the parent-child relationship in the computation of the prior probabilities does not add a noticeable improvement to the overall integration scheme.

Table 6.4: Hierarchical precision, hierarchical recall, hierarchical F-micro and hierarchical F-macro measures obtained from hierarchical single source (on each dataset separately) and HiBiN methods.

Single Source				
Dataset	**Precision**	**Recall**	**F-Micro**	**F-Macro**
PPI-BG	0.9259	0.6071	0.6953	0.7147
PPI-STRING	0.8882	0.5933	0.6846	0.6926
Pfam-Binary	0.8974	0.5912	0.6882	0.7006
Pfam-LogE	0.7340	0.5593	0.6115	0.6348
Gene-Expr	0.7426	0.5801	0.6414	0.6644
Seq-Sim	0.7274	0.5574	0.6067	0.6312
Integration methods				
HiBiN$_a$	**0.9492**	0.6298	**0.7361**	0.7572
HiBiN$_p$	0.9442	**0.6329**	0.7355	**0.7579**

A closer analysis in terms of per-level precision, recall and F-measure highlights differences between the proposed method and the baselines. Figures 6.3, 6.4 and 6.5 show the level-by-level performance comparisons between hierarchical single source applied on PPI-BioGrid dataset, flat integration, HiBiN$_a$ and HiBiN$_p$ methods in terms of precision, recall and F-measure. Since PPI-BioGrid dataset performed the best for hierarchical single source method, compared with other datasets, We chose it as a representative dataset for the hierarchical single source method. The per-level analysis reveals a degradation in the performance in all of the methods with respect to the depth of the functional classes. However, this degradation is significantly lower when the data integration and hierarchial relationships among the classes are taken into account (which are the major components of HiBiN algorithm).

It is worth pointing out that HiBiN algorithm has a considerably shorter learning process than the baseline classifiers. Table 6.5 shows the comparison of the training time needed for building the classifiers for the baseline methods, which include the flat integration and the hierarchical single-source methods, and HiBiN$_a$ and HiBiN$_p$. We measured the training time for the baseline methods and HiBiN algorithm on all the datasets, running under Windows XP on

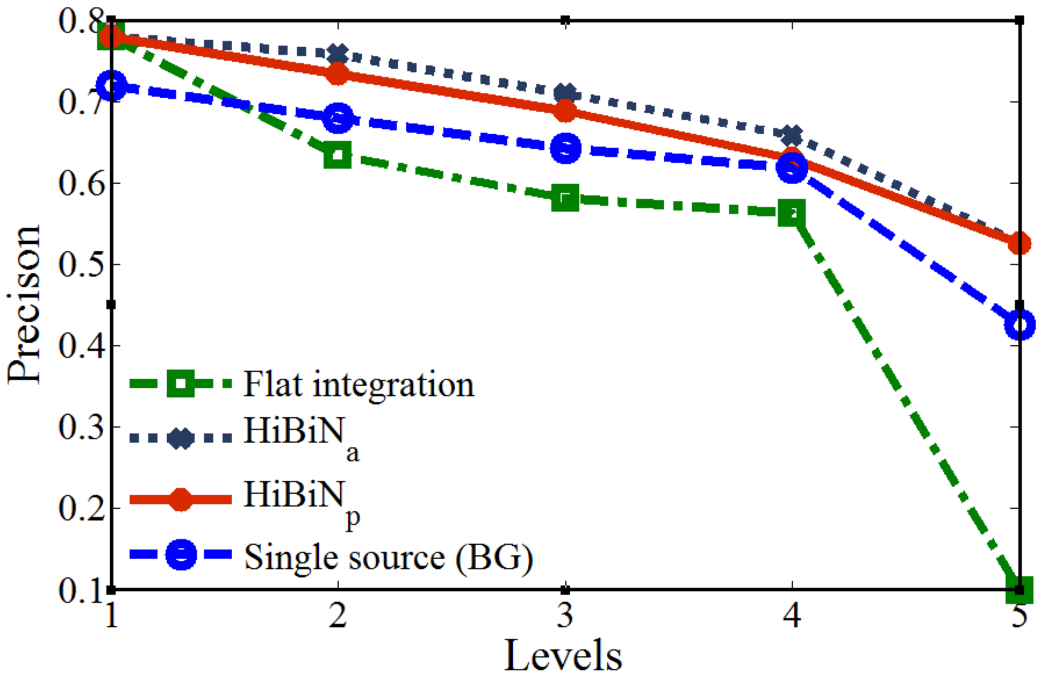


Figure 6.3: Comparison of the per-level average precision across the five levels of the FunCat taxonomy using flat integration, hierarchical single source (applied on PPI-BG dataset), HiBiN$_a$ and HiBiN$_p$ methods. BG stands for PPI-BioGrid dataset.

a 2.4 GHz with 12.0 GB of RAM. The training time for the hierarchical single-source method is obtained by summing up the time needed to train the classifiers on each dataset separately. We can observe that HiBiN, with its two variations, shows considerable improvements over the baseline methods, especially over the flat integration method.

Table 6.5: Comparison of the training time needed to build the classifiers for the flat integration, hierarchical single source and HiBiN methods. The training time is reported in hours.

Single source	**Flat integration**	**HiBiN$_a$**	**HiBiN$_p$**
44.3921	147.20	43.0389	35.8944

6.6 Summary

High-throughput technologies increase the need for integrated computational analysis methods of data generated by such studies. One promising solution for the problem of huge diversity of genomic data is to use an integrated analysis framework. We have developed HiBiN, a gen-

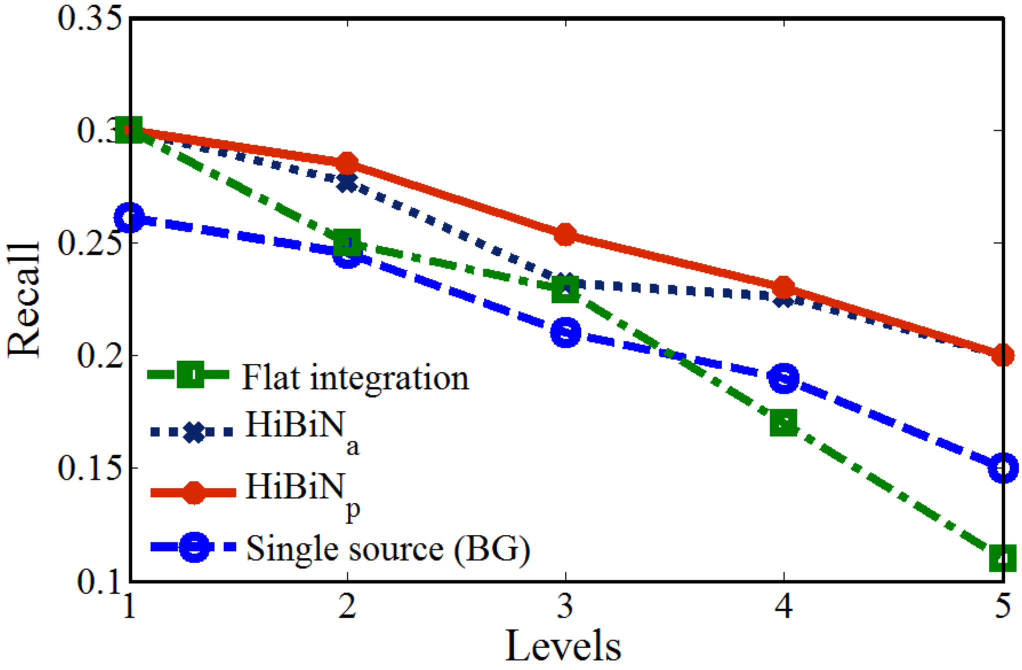


Figure 6.4: Comparison of the per-level average recall across the five levels of the FunCat taxonomy using flat integration, hierarchical single source (applied on PPI-BG dataset), HiBiN$_a$ and HiBiN$_p$ methods. BG stands for PPI-BioGrid dataset.

eral probabilistic framework for genome-wide gene function prediction through the integration of diverse heterogeneous data sources while maintaining the hierarchy constraint among the functions. Our results showed that the integration can help improve the performance of the standard classification-based gene function prediction algorithms.

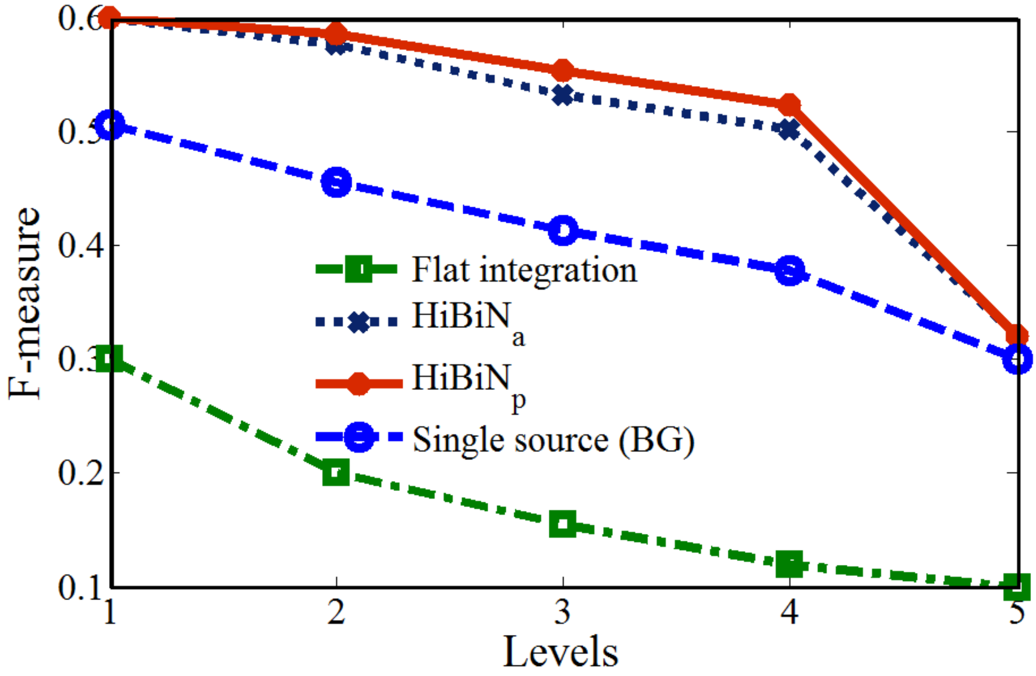


Figure 6.5: Comparison of the per-level average F-measure across the five levels of the FunCat taxonomy using flat integration, hierarchical single source (applied on PPI-BG dataset), HiBiN$_a$ and HiBiN$_p$ methods. BG stands for PPI-BioGrid dataset.

CHAPTER 7

A WEIGHTED FRAMEWORK FOR INTEGRATING HETEROGENEOUS MULTI-SOURCE INFORMATION

7.1 Motivation

Automated protein function annotation methods using high-throughput data is a major challenge in the post-genomic era [59]. The multi-source integration model proposed in the previous chapter can integrate and exploit multiple sources for hierarchical multi-label classification. The integration solution often demonstrates superior classification performance than the single source models in many scenarios. In the current model, the data sources are treated with equal importance. The basic integration model aimed at equate the contributions of the individual data sources. However, data sources may vary in their measurements and the inherent noise. Moreover, one data source may be more informative than the others in learning certain set of labels. We therefore extend our model to permit a weighted combination of the individual data sources over each label in the hierarchy. This extension allows the separate data sources to be exploited while training the classifiers.

A natural way to extend the HiBiN algorithm is by considering a weighting scheme to control the contribution of the individual data sources to the final predictions. In particular, the goal of the weighting scheme is to determine the relative effectiveness or usefulness of different types of data in terms of protein function prediction. Incorporating various biological sources into the classification process can enhance the quality of the final predictions to maximize the exploitation of the increasing wealth of biological knowledge available. This approach can minimize the contributions of undesirable effects from irrelevant and noisy sources for building classification model for each label in the hierarchy.

In this chapter, we present the proposed weighting scheme for integrating heterogeneous data sources for hierarchical multi-label classification problem. In the next section, we show the proposed method to weigh the different data sources followed by the experimental results.

7.2 Our Contribution

Several methods have been proposed to tackle the problem of protein function prediction. Most of these methods attempt to predict protein functions solely based on a certain type of data. However, few methods have come into widespread use. The problem is that most methods deal solely with independent data sources and ignore the fact that data sources may provide complimentary information to each other.

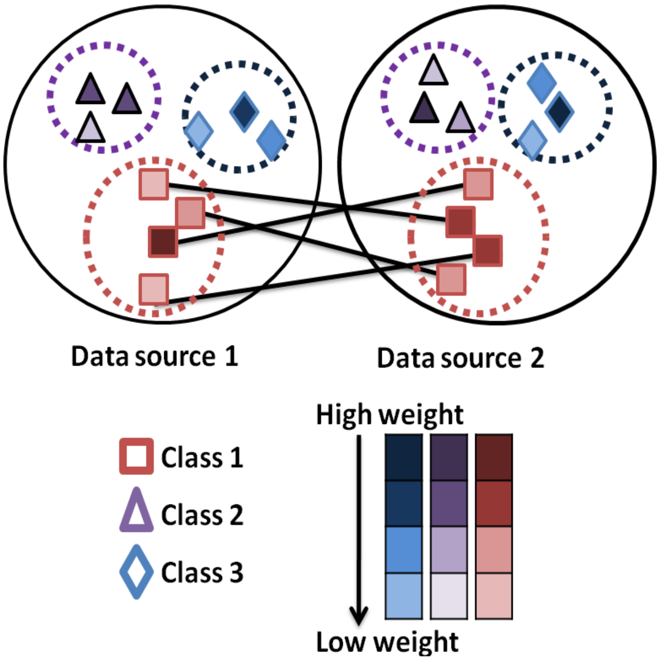


Figure 7.1: An illustration of the proposed weighting scheme for integrating heterogeneous data sources for hierarchical multi-label classification problem

The proposed weighting procedure is illustrated in Figure 7.1. This figure shows an example of two data sources, where each data source has three classes (labels). Each label is represented by a different shape. In this example, three instances belong to two classes while four instances belong to the third class. The weight of each instance is represented as a shade.

Having a darker shade means that more weight is assigned to that instance, while having a lighter shade mean less weight is assigned to that instance. The weights are determined based on the centrality of the instance with respect to all the data instances that are associated to that class. Let us assume that in each class there is a virtual data instance (we call it a representative instance). This instance is computed based on the data instances belonging to each class. Specifically, it is computed as the average of the feature vectors of the instances in that class. Then, the weights are computed to be the similarity between each data instance and the representative feature vector of each class. The weight that is assigned to a particular instance is different from one class to another, and hence, an instance may have a different weight from one class to another and from one data source to another. The proposed weighting scheme for heterogeneous multi-source integration model has the following steps:

1. **Generate a reference vector** ρ. From each data source and for each label, we compute a reference feature vector from the set of instances that belong to the current label.

2. **Compute the weight** ω. For each test instance and for each label, we compute the weight of the test instance with respect to each data source. The weight is computed based on the distance between the test instance and the reference feature vector.

3. **Incorporate the weights** $\omega_1, ..., \omega_Q$ **into the prediction**. The Bayesian posteriors that are used for performing the integration of the different data sources are modified so that the calculated weight obtained from each data source and each label for each test instance is taken into consideration.

7.3 The proposed wHiBiN Algorithm

The goal of using the proposed scheme is to effectively integrate the prediction results from different data sources. For a given test protein, we are interested in determining the annotation of that protein to a particular label in a particular data source.

The weight assigned to an individual instance from a certain data source represents the usefulness of that instance/source in predicting a given gene function. The wHiBiN algorithm aims at minimizing the disagreement between individual models built from each separate data source.

We develop a new solution to this problem by creating a representative feature vector for each label in each data source. We compute the corresponding weight to each instance for each class in each dataset. For a given label in each data source, the vector is used to find the relativeness of a given protein in that data source. Following this strategy, different data sources contribute differently to the prediction of labels for the same test instances.

The representative feature vector is calculated for each label based on the data instances that belong to that label in the same data source. To improve the performance of the integration, a weighting formula, Equation (7.1), is proposed to exploit the heterogenous data sources in an efficient manner for the individual protein function prediction as follows:

$$\omega_M^y = 1 - \sqrt{\sum_i (\rho_i - g_i)^2} \tag{7.1}$$

where ρ is the representative feature vector of a class y from data source M, and g is an instance where its description is found in data source M.

Equation (7.1) calculates the feature weight of each instance (protein) from each data source. A new feature vector is formed by averaging the feature vectors for the instances annotated with a particular label (function) from one data source. This new feature vector is called the reference feature vector. The reference feature vector is serving as a representative vector of the features for the instances that belong to the current label. When a test instance is presented for classification, we compute the distance between that test instance and the reference feature vector for the current label from one data source. This process is performed for the test instance from all the data sources separately. As the distance increases between the test instance and the reference feature vector, less weight is assigned to the prediction that is resulted from the current data source. This means that it is more likely that the test instance is not belonging to the current label. On the other hand, when the computed distance is small, more weight is assigned to the test instance as it has a better chance to be annotated with the current label.

For computing the distance, we used Euclidean distance; however, other distance functions (such as Pearson correlation coefficient) are also applicable. The final prediction from multiple data sources is computed as follows:

$$P(y_k|g_i, M_1, ..., M_Q) = \frac{\prod_{j=1}^{Q} \omega_{(g_i,y_k)}^{j} P(g_i, M_j|y_k)] \cdot P(y_k)}{\sum_{k=1}^{m} [\prod_{j=1}^{Q} P(g_i, M_j|y_k)] \cdot P(y_k)} \tag{7.2}$$

Where $\omega_{(g_i,y_k)}^{j}$ is the weight computed for gene g_i for class label y_k from dataset M_j. The training process for the extended model is the same as before, however, the inference procedure has to be adjusted to fit the weighting scheme. The weighting scheme is described in Algorithm 7.

```
Algorithm 7 $CalculateWeight$
Input: $T$: a tree-structured set of classes
           $r$: the root of T
           $v$: the number of children classes of a particular class in T
           $\mathcal{M}_1, ..., \mathcal{M}_Q$: the input datasets
Output: The weights, $\omega_{i,j}^{k}$, for each test gene $g_i$, with respect to class $y_j \in \mathcal{L}$ in dataset $\mathcal{M}_k$.
Algorithm:
for $c = 1, ..., v$ do
    Find the subset $\delta_c^i$ = $\{< g_1, \mathcal{Y}_1 >, ..., < g_n, \mathcal{Y}_n >\}$ where $i = 1, ...Q$ that contains the instances that are annotated with label $c$
    Compute the reference feature $\rho$ = $\delta_c$ / $n$
    $\omega_D = 1 - (\sqrt{\sum_i (\rho_i - g_i)^2})$
end for
```

7.4 Experimental Results

In our experiments, we compare the performance of wHiBiN algorithm with HiBiN and the baseline flat integration algorithms. We performed the prediction on the testing sets, using the weighting scheme calculated for each of the test instances. To make a fair comparison between the integration methods, we also have tested two ways to compute the prior probabilities. In one variation, the prior probabilities are computed by considering the set of annotated genes for each functional label with respect to the total number of genes. This version of the algorithm is called wHiBiN$_a$, where a stands for all. In the second variation, the prior probabilities are computed by considering the set of annotated genes at each functional label with respect to the number of genes that are annotated with the parent functional class of the label of interest. This version of the algorithm is called wHiBiN$_p$, where p stands for parent.

The results were evaluated using the per class precision, recall and F-measure and compared with flat integration and the two versions of the HiBiN algorithm where all the data sources were equally important. Table 7.1 shows the average per-class evaluation results. The key trend one can glean from this study is that by introducing the weights to the diverse data sources, better classification results are achieved as the focus will be on the more useful and informative data sources.

Table 7.1: Average per-class precision, recall and F-measure obtained from hierarchical multi-source integration method, HiBiN, and the weighted algorithm, wHiBiN.

Integration methods			
Dataset	**Precision**	**Recall**	**F-measure**
Flat	0.3592	0.307	0.4521
HiBiN$_a$	**0.7436**	0.4535	0.6175
HiBiN$_p$	0.7083	0.4779	0.6222
wHiBiN$_a$	0.7231	**0.4910**	0.6031
wHiBiN$_p$	0.7011	0.4690	**0.6322**

Additionally, hierarchical precision, hierarchical recall, hierarchical F-micro and hierarchical F-macro measures are also computed for the wHiBiN algorithm and compared against its counterpart, the HiBiN algorithm. Table 7.2 shows the hierarchical evaluation results. On two of the hierarchical measures, namely hierarchical precision and hierarchical F-micro, we find that wHiBiN$_a$ achieves marginally better results. On the other hand, wHiBiN$_p$ achieves marginally better results in terms of hierarchical recall and hierarchical F-macro measures. The weighted combination of data from different data sources led to increased prediction performance relative to a classifier trained on a single type of data. When some data sources provide more information than the others, weighted combination of the different data sources increase the prediction performance. The existence of noise in a data source can disrupt the learning process. Therefore, low weights are associated with inferior data sources.

Table 7.2: Hierarchical precision, hierarchical recall, hierarchical F-micro and hierarchical F-macro measures obtained from hierarchical multi-source integration HiBiN and weighted integration wHiBiN methods.

Integration methods				
Dataset	**hP**	**hR**	**hF-Micro**	**hF-Macro**
HiBiN$_a$	0.9492	0.6298	0.7361	0.7572
HiBiN$_p$	0.9442	0.6329	0.7355	0.7579
wHiBiN$_a$	**0.9502**	0.6318	**0.7416**	0.7580
wHiBiN$_p$	0.9424	**0.6432**	0.7361	**0.7641**

7.5 Summary

Predicting gene functions using high-throughput data is a major challenge in the post-genomic era. In this chapter, we have presented wHiBiN algorithm for weighted integration of heterogeneous data source for the hierarchical multi-label classification problem. The weighted integration of the heterogeneous high-throughput data helps in reducing the contribution of the noise level for each type of data. Such method provides an efficient hypothesis-generation tool where the results are based on the highly confident data sources. The predictions found by the algorithm provide biologists with hypotheses to design specific experiments for validating the predicted functions. Combining computational methods and experiments may reveal the biological functions of hypothetical proteins much more efficiently compared to traditional methods.

CHAPTER 8

CONCLUSION AND FUTURE WORK

The Hierarchical multi-label classification is a challenging research problem that arises in many domains such as computational biology, text categorization and image annotation. In particular, the functional classification of genes, the application problem dealt with in this thesis, is an important and challenging problem in the area of functional genomics.

The task is to annotate unknown gene products to a set of predefined hierarchically structured functions. Each gene may belong to multiple classes at the same time (multi-label classification), where the classes are hierarchically structured (hierarchical classification). The number of functional classes is large and the functional classes are heavily imbalanced. Moreover, functionally-similar genes may have huge diversity on their measurements through various datasets. Each data source may provide information about the characteristics of genes to be classified from a different point of view. One source can be useful to learn a specific functional class while being irrelevant to other classes.

In addition to the above mentioned challenges, functions may have correlations and dependencies among each other which are not captured by the pre-defined hierarchically structured functions. In a consistent direction, gene products that are annotated by functions which are in turn children of the same parent label may have hidden information that when revealed will lead to improved classification results.

Seeking to address all these challenges in a single unified framework, this dissertation offers practical solutions for “hierarchical multi-label classification” focusing on the protein function prediction application where the mutual relationships among the classes are specified by a rooted-tree structure.

8.1 Summary and Contributions

The first part of this thesis examines the problem of hierarchical multi-label classification problem based on a single data source. We followed a local approach of inducing a separate binary classifier for each child class of a specific parent class and then building them in parallel using a multi-label classifier. The performance of a hierarchical approach is compared to a flat approach using the same experimental settings. It is important to note that in this part, single-source algorithms were developed, where the algorithms were applied on each data source separately.

Our original attempt to deal with these problems from each data source independently was implemented in two-folds: in the first fold, the HML-Boosting algorithm was applied for the gene function prediction problem. This algorithm leverages the hierarchical structure of the classes. Hence, the classifiers that are built tend to be more efficient and effective compared to the flat classification methods. The incorporation of the hierarchical structure of the classes improves the performance of the prediction. In addition, HML-Boosting is compared with flat classification and the results of the experiments on four bio-molecular datasets showed that the HML-Boosting significantly outperforms flat classification. The performance of the HML-Boosting algorithm using the top-down and the bottom-up class prediction strategies is evaluated.

In the second fold, HiBLADE, a hierarchical multi-label classification framework for incorporating the information about the hierarchical relationships among the labels as well as the label correlations was proposed. In particular, the possible correlations among the labels that are not captured by the hierarchical taxonomy of the labels were explored. A function may contribute to its siblings while building their classification models. This knowledge was incorporated during the training process of the classifiers. In other words, such inherent knowledge about the correlations is used to reduce the information redundancy in the learning process. The experimental results showed that the proposed algorithm, HiBLADE, outperforms the flat classification method and the local classifier methods that build independent classifier for each class without considering the correlation among the classes.

The second part of the dissertation seeks to integrate multiple heterogeneous data sources for improving hierarchical protein function classification. In an effort to integrate multiple heterogeneous data sources, HiBiN algorithm, a general probabilistic framework for genome-wide gene function prediction through the integration of diverse heterogeneous data sources while preserving the hierarchy constraints among the gene functions has been proposed. Our results showed that the integration can help in improving the performance of the standard classification-based gene function prediction algorithms. This inspired us to extend HiBiN algorithm to include a weighting scheme to control the contributions from the individual data sources based on their relevance to the label under-study.

8.2 Future Directions

Despite the extensive research effort that has been performed in this thesis, there are still many scientific challenges that can be explored in the future. The goal is to develop algorithms that lead to computational discoveries which confirm the existing biological knowledge. In the first direction, we are planning to apply the proposed solutions to other domains such as text categorization and medical image annotation. The frameworks presented in this thesis are generic and can be adapted and applied in other domains for predicting the classes of unknown examples in the presence of hierarchies.

Developing a cost sensitive strategy for training the labels is an interesting direction for future work. One of the major challenges in the hierarchical multi-label learning process is the issue of dealing with datasets with imbalanced class distributions. This issue is quite common in may real applications with the degree of imbalance varies from one application to the other. A cost-sensitive classification uses a cost matrix during the model building process. The cost matrix encodes the penalty of classifying records from one class as another. One of the future directions is to use the idea of transfer learning to improve the classification performance in the presence of few training samples [1].

Moreover, in the current algorithm, HiBLADE algorithm, we consider only the information revealed from the most similar label/class to the current one. Studying the effectiveness of exploiting the correlation among multiple labels/classes to the current label/class is one of the possible future research directions. We may consider a weighting scheme that combines all of the siblings to the current class in the class hierarchy. The weights control the amount of contribution from each of those labels. The weight associated with each label is measured by the similarity between this label and the current label. Eventually, the final prediction score will be composed of two major components. The first component is the prediction score obtained by applying the current model, that is trained on the data points belonging to the current label. While the second component is the prediction score obtained by applying the sibling models on the data points belonging to the current label [7].

Another possible direction is to investigate the integration of heterogeneous data sources with heterogeneous instances. In other words, the integration of multiple data sources with uncommon instances (genes). We have shown a sematic integration approach across heterogeneous data sources. Instance-level correspondences is adopted by our current approach. Instance-level correspondences, which is known as linkage or instance matching, states that an instance in different data sources has to exist in all of the data sources. In the current integration solutions, we considered only the common genes between the different data sources. This results in reducing the number of genes that are used to build the classifiers for the different labels. Such reduction results in loosing part of the information that may reveal useful information that could lead to improved prediction results [10].

Finally, developing more scalable solutions using some of the recently proposed Boosting strategies is another possible direction for future work [88, 98]. We believe that the proposed solutions are amenable to effective parallelization and worth investigating further. In the current era, huge amounts of data in various domains are available that are hierarchically structured

[90]. Parallel solutions can help in alleviating the performance degradation issues especially when dealing with data of huge size and reduce the computation times immensely.

BIBLIOGRAPHY

[1] Al-Stouhi, S., and Reddy, C. K. Adaptive boosting for transfer learning using dynamic updates. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases - European Conference, ECML/PKDD 2011* (Athens, Greece, September 5-9 2011), pp. 60–75.

[2] Alaydie, N., and Fotouhi, F. Unraveling complex relationships between heterogeneous omics datasets using local principal components. In *IRI* (2011), IEEE Systems, Man, and Cybernetics Society, pp. 136–141.

[3] Alaydie, N., and Fotouhi, F. Exploratory Analysis of the Relationships Between Heterogeneous Omics Datasets Using Local Principal Components. In *In Information Reuse and Integration in Academia and Industry*, T. Özyer, K. Kian-Mehr, M. Tan, , and J. Zeng, Eds. Springer-Verlag, 2012.

[4] Alaydie, N., Fotouhi, F., Reddy, C. K., and Soltanian-Zadeh, H. Noise and outlier filtering in heterogeneous medical data sources. In *DEXA Workshops* (2010), A. M. Tjoa and R. Wagner, Eds., IEEE Computer Society, pp. 115–119.

[5] Alaydie, N., Reddy, C. K., and Fotouhi, F. Hierarchical multi-label boosting for gene function prediction. In *Proceedings of the 9th International Conference on Computational Systems Bioinformatics (CSB)* (Stanford, CA, USA, August 2010), pp. 14–25.

[6] Alaydie, N., Reddy, C. K., and Fotouhi, F. A Bayesian Integration Model of Heterogeneous Data Sources for Improved Gene Functional Inference. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine (ACM-BCB)* (Chicago, IL, USA, August 2011), pp. 376–380.

[7] Alaydie, N., Reddy, C. K., and Fotouhi, F. Exploiting label dependency for hierarchical multi-label classification. In *Proceedings of the 16th Pacific-Asia Conference in Knowledge Discovery and Data Mining (PAKDD)* (2012), pp. 294–305.

[8] Anyanwu, M. N. Comparative analysis of serial decision tree classification algorithms. *Journal of Computer Science 3*, 3 (2009), 230–240.

[9] Astikainen, K., Holm, L., Pitkänen, E., Szedmak, S., and J, R. Towards structured output prediction of enzyme function. *BMC Proc. 2*, Suppl 4:S2 (Dec 2008).

[10] Aziz, M. S., and Reddy, C. K. Robust prediction from multiple heterogeneous data sources with partial information. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010* (Toronto, Ontario, Canada, October 26-30 2010), pp. 1857–1860.

[11] Barutcuoglu, Z., Schapire, R. E., and Troyanskaya, O. G. Hierarchical multi-label prediction of gene function. *Bioinformatics 22*, 7 (Jan 2006), 830–836.

[12] Becker, S. Mutual Information Maximization: Models of cortical self-organization. *Network: Computation in Neural Systems 7*, 1 (February 1996), 7–31.

[13] Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. When is nearest neighbor meaningful? In *Database Theory ICDT99*, vol. 1540 of *Lecture Notes in Computer Science*. 1999, pp. 217–235.

[14] Bielza, C., Li, G., and Larrañaga, P. Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning 52* (2011), 705–727.

[15] Boutell, M., Luo, J., Shen, X., and Brown, C. Learning multi-label scene classification1. *Pattern Recognition 37*, 9 (Sept. 2004), 1757–1771.

[16] Breslow, L. A., and Aha, D. W. Simplifying decision trees: A survey. *Knowledge Engineering Review 12*, 1 (Jan. 1997), 1–40.

[17] Burred, J. J., and Lerch, A. A Hierarchical Approach To Automatic Musical Genre Classification. In *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)* (London, UK, September 2003), pp. 8–11.

[18] Cesa-Bianchi, N., Re, M., and Valentini, G. Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference. *Machine Learning 88*, 1-2 (2012), 209–241.

[19] Cesa-Bianchi, N., and Valentini, G. Hierarchical cost-sensitive algorithms for genome-wide gene function prediction. In *Machine Learning in Systems Biology, Proceedings of the Third international workshop* (Ljubljana, Slovenia, 2009), pp. 25–34.

[20] Chen, Y., and Xu, D. Global protein function annotation through mining genome-scale data in yeast Saccharomyces cerevisiae. *Nucleic Acids Research 32*, 21 (2004), 6414–6424.

[21] Cheng, W., and Hüllermeier, E. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning 76*, 2–3 (2009), 211–225.

[22] Chitale, M., and Kihara, D. Computational Protein Function Prediction: Framework and Challenges. In *Protein Function Prediction for Omics Era*, D. Kihara, Ed. Springer Netherlands, 2011, pp. 1–17.

[23] Chua, H. N., Liu, G., and Wong, L. Protein Function Prediction Using Protein-Protein Interaction Networks. In *Protein Function Prediction for Omics Era*, D. Kihara, Ed. Springer Netherlands, 2011, pp. 243–270.

[24] Chua, H. N., Sung, W.-K., and Wong, L. Exploiting indirect neighbours and topological weight to predict protein function from protein-protein interactions. *Bioinformatics 22*, 13 (July 2006), 1623–1630.

[25] Chua, H. N., Sung, W.-K., and Wong, L. An efficient strategy for extensive integration of diverse biological data for protein function prediction. *Bioinformatics 23* (December 2007), 3364–3373.

[26] Clare, A., and King, R. D. Knowledge discovery in multi-label phenotype data. In *PKDD* (2001), L. D. Raedt and A. Siebes, Eds., vol. 2168 of *Lecture Notes in Computer Science*, Springer, pp. 42–53.

[27] Clare, A., and King, R. D. Predicting gene function in Saccharomyces cerevisiae. *Bioinformatics 19*, 2 (2003), 42–49.

[28] Consortium, T. G. O. Gene ontology: tool for the unification of biology. *Nature Genetics 25*, 1 (May 2000), 25–29.

[29] Cortes, C., and Vapnik, V. Support-vector networks. *Mach. Learn. 20*, 3 (Sept. 1995), 273–297.

[30] Cover, T., and Hart, P. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on 13*, 1 (Jan. 1967), 21–27.

[31] Darko Aleksovski, D. K., and Deroski, S. Evaluation of distance measures for hierarchical multilabel classification in functional genomics. In *ECML/PKDD 2009 Workshop on Learning from Multi-Label Data* (Bled, Slovenia, September 2009).

[32] Deng, M., Chen, T., and Sun, F. An integrated probabilistic model for functional prediction of proteins. In *in Proc 7th Int Conf Comp Mol Biol* (2003), pp. 95–103.

[33] Diplaris, S., Tsoumakas, G., Mitkas, P. A., and Vlahavas, I. P. Protein classification with multiple algorithms. In *Panhellenic Conference on Informatics* (2005), P. Bozanis and E. N. Houstis, Eds., vol. 3746 of *Lecture Notes in Computer Science*, Springer, pp. 448–456.

[34] Do Prado, H. A., and Ferneda, E. *Emerging Technologies of Text Mining: Techniques and Applications*. Information Science Reference, Hershey, PA, 2007.

[35] Dudani, S. The distance-weighted k-nearest-neighbour rule. *IEEE Transactions on Systems, Man and Cybernetics SMC-6(4)* (1975), 325–327.

[36] Dumais, S. T., and Chen, H. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR Conference on Research and development in information retreival* (2000), pp. 256–263.

[37] Eisenberg, D., Marcotte, E. M., Xenarios, I., and Yeates, T. O. Protein function in the post-genomic era. *Nature 405*, 6788 (jun 2000), 823–826.

[38] Eisner, R., Poulin, B., Szafron, D., Lu, P., and Greiner, R. Improving protein function prediction using the hierarchical structure of the gene ontology. In *CIBCB* (2005), IEEE, pp. 354–363.

[39] Engelhardt, B. E., Jordan, M. I., Muratore, K. E., and Brenner, S. E. Protein molecular function prediction by bayesian phylogenomics. *PLoS Comput Biol 1*, 5 (10 2005), e45.

[40] Esuli, A., Fagni, T., and Sebastiani, F. Boosting multi-label hierarchical text categorization. *Information Retrieval 11* (2008), 287–313.

[41] Fagni, T., and Sebastiani, F. Selecting negative examples for hierarchical text classification: An experimental comparison. *Journal of the American Society for Information Science and Technology (JASIST) 61*, 11 (2010), 2256–2265.

[42] Freund, Y., and Schapire, R. E. Large margin classification using the perceptron algorithm. *Machine Learning 37* (1999), 277–296.

[43] Gardner, M., and Dorling, S. Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences. *Atmospheric Environment 32*, 1415 (1998), 2627 – 2636.

[44] Gasch, A. P., Spellman, P. T., Kao, C. M., Carmel-Harel, O., Eisen, M. B., Storz, G., Botstein, D., and Brown, P. O. Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell 11* (2000), 4241–4257.

[45] Gillis, J., and Pavlidis, P. The role of indirect connections in gene networks in predicting function. *Bioinformatics 27*, 13 (July 2011), 1860–1866.

[46] Guan, Y., Myers, C. L., Hess, D., Barutcuoglu, Z., Caudy, A., and Troyanskaya, O. G. Predicting gene function in a hierarchical context with an ensemble of classifiers. *Genome Biology 9*, Suppl 1:S3 (2008).

[47] Guo, Y., and Gu, S. Multi-label classification using conditional dependency networks. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)* (Barcelona, Catalonia, July 16-22 2011), pp. 1300–1305.

[48] Han, E.-H., Karypis, G., and Kumar, V. Text categorization using weight adjusted-nearest neighbor classification. In *Advances in Knowledge Discovery and Data Mining*, D. Cheung, G. Williams, and Q. Li, Eds., vol. 2035 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2001, pp. 53–65.

[49] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[50] Hearst, M. A. Support vector machines. *IEEE Intelligent Systems 13*, 4 (July 1998), 18–28.

[51] Heckerman, D. A tutorial on learning with Bayesian networks. Tech. rep., Microsoft, Technical Report MSR-TR-95-06, 1995.

[52] Hepner, G. F., Logan, T., Ritter, N., and Bryant, N. Artificial neural network classification using a minimal training set - comparison to conventional supervised classification. *Photogramm Eng Remote Sensing 56*, 4 (1990), 469–473.

[53] Horan, K., Jang, C., Bailey-Serres, J., Mittler, R., Shelton, C., Harper, J. F., Zhu, J.-K., Cushman, J. J., Gollery, M., and Girke, T. Annotating genes of known and unknown function by large-scale co-expression analysis. *Plant Physiology 147*, 1 (May 2008), 41–57.

[54] Hunt, E. B., Marin, J., and Stone, P. J. *Experiments in induction*. Academic Press, New York, NY, 1966.

[55] Hvidsten, T. R., Komorowski, J., Sandvik, A. K., and Laegreid, A. Predicting gene function from gene expressions and ontologies. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing 6* (2001), 299–310.

[56] Jaimovich, A., Elidan, G., Margalit, H., and Friedman, N. Towards an integrated protein-protein interaction network. In *Research in Computational Molecular Biology: 9th Annual International Conference, RECOMB* (Cambridge, MA, May 14-15 2005), pp. 14–30.

[57] Jiang, X., N, N. N., Steffen, M., Kasif, S., Gold, D., and Kolaczyk, E. Combining hierarchical inference in ontologies with heterogeneous data sources improves gene function prediction. In *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2008), pp. 411–416.

[58] Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, C. Ndellec and C. Rouveirol, Eds.,

vol. 1398 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1998, pp. 137–142.

[59] Joshi, T., Chen, Y., Becker, J. M., Alexandrov, N., and Xu, D. Genome-scale gene function prediction using multiple sources of high-throughput data in yeast Saccharomyces cerevisiae. *OMICS 8*, 4 (2004), 322–333.

[60] Jr., C. N. S., and Freitas, A. A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*.

[61] Jr, C. N. S., and Freitas, A. A. A global-model naive bayes approach to the hierarchical prediction of protein functions. In *IEEE International Conference on Data Mining (ICDM)* (December 2009), pp. 992–997.

[62] Jun, G., and Ghosh, J. Multi-class boosting with class hierarchies. In *Multiple Classifier Systems* (2009), pp. 32–41.

[63] Karaoz, U., Murali, T. M., Letovsky, S., Zheng, Y., Ding, C., Cantor, C. R., and Kasif, S. Whole-genome annotation by using evidence integration in functional-linkage networks. *Proc Natl Acad Sci U S A 101*, 9 (March 2004), 2888–2893.

[64] Kim, W., Krumpelman, C., and Marcotte, E. Inferring mouse gene functions from genomic-scale data using a combined functional network/classification strategy. *Genome Biology 9*, Suppl 1 (2008), S5+.

[65] Koller, D., and Sahami, M. Hierarchically classifying documents using very few words. In *Proceedings of the 14th international conference on machine learning* (Nashville, Tennessee, July 1997), pp. 170–178.

[66] KRIEGEL, H.-P., KRÖGER, P., PRYAKHIN, A., AND SCHUBERT, M. Using Support Vector Machines for Classifying Large Sets of Multi-Represented Objects. In *in Proc. of 4th SIAM International Conference on Data Mining* (2004), pp. 102–114.

[67] LANCKRIET, G. R. G., BIE, T. D., CRISTIANINI, N., JORDAN, M. I., AND NOBLE, W. S. A statistical framework for genomic data fusion. *Bioinformatics 20*, 16 (2004), 2626–2635.

[68] LEE, I., DATE, S. V., ADAI, A. T., AND MARCOTTE, E. M. A probabilistic functional network of yeast genes. *Science (New York, N.Y.) 306*, 5701 (Nov. 2004), 1555–1558.

[69] LEWIS, D. D., YANG, Y., ROSE, T. G., AND LI, F. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research 5* (2004), 361–397.

[70] LI, T., AND OGIHARA, M. Music genre classification with taxonomy. In *Proceedings of the International Conference on Acoustic, Speech and Signal Processing (ICASSP'05)* (2005), vol. 5, pp. 197–200.

[71] LI, T., ZHU, S., LI, Q., AND OGIHARA, M. Gene functional classification by semisupervised learning from heterogeneous data. In *SAC* (2003), ACM, pp. 78–82.

[72] LIAO, Y., AND VEMURI, V. R. Use of k-nearest neighbor classifier for intrusion detection. *Computers & Security 21*, 5 (2002), 439–448.

[73] LIPPERT, C., GHAHRAMANI, Z., AND BORGWARDT, K. M. Gene function prediction from synthetic lethality networks via ranking on demand. *Bioinformatics 26*, 7 (2010), 912–918.

[74] LIU, Y. Yeast Gene Function Prediction from Different Data Sources: An Empirical Comparison. *The Open Bioinformatics Journal 5* (2011), 69–76.

[75] Marcotte, E. M., Pellegrini, M., Ng, H.-L., Rice, D. W., Yeates, T. O., and Eisenberg, D. Detecting protein function and protein-protein interactions from genome sequences. *Science 285*, 5428 (1999), 751–753.

[76] Marcotte, E. M., Pellegrini, M., Thompson, M. J., Yeates, T. O., and Eisenberg, D. A combined algorithm for genome-wide prediction of protein function. *Nature 402* (2000), 83–86.

[77] Mateos, A., Dopazo, J., Jansen, R., Tu, Y., Gerstein, M., and Stolovitzky, G. Systematic Learning of Gene Functional Classes From DNA Array Expression Data by Using Multilayer Perceptrons. *Genome Research 12*, 11 (nov 2002), 1703–1715.

[78] McCallum, A., Rosenfeld, R., Mitchell, T. M., and Ng, A. Y. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)* (Madison, Wisconsin, USA, July 24-27 1998), pp. 359–367.

[79] Mewes, H., Heumann, K., Kaps, A., Mayer, K., Pfeiffer, F., Stocker, S., and Frishman, D. MIPS: a database for genomes and protein sequences. *Nucleic Acids Research 27*, 1 (1999), 44–48.

[80] Mostafavi, S., and Morris, Q. Using the gene ontology hierarchy when predicting gene function. In *In Conference on Uncertainty in Artificial Intelligence (UAI)* (Montreal, Canada, September 2009), pp. 22–26.

[81] Mostafavi, S., and Morris, Q. Fast integration of heterogeneous data sources for predicting gene function with limited annotation. *Bioinformatics 26*, 14 (2010), 1759–1765.

[82] Mostafavi, S., Ray, D., Warde-Farley, D., Grouios, C., and Morris, Q. GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome biology 9 Suppl 1*, Suppl 1 (2008), S4+.

[83] Murali, T. M., Wu, C.-J., and Kasif, S. The art of gene function prediction. *Nature Biotechnology 24*, 12 (2006), 1474 – 1475.

[84] Myers, C. L., Robson, D., Wible, A., Hibbs, M. A., Chiriac, C., Theesfeld, C. L., Dolinski, K., and Troyanskaya, O. G. Discovery of biological networks from diverse functional genomic data. *Genome biology 6*, 13 (2005), R114+.

[85] Obozinski, G., Lanckriet, G., Grant, C., Jordan, M. I., and Noble, W. S. Consistent probabilistic outputs for protein function prediction. *Genome Biology 9*, Suppl 1:S6 (2008).

[86] Odibat, O., and Reddy, C. K. Ranking differential hubs in gene co-expression networks. *Journal of Bioinformatics and Computational Biology (JBCB) 10*, 1, 1240002.

[87] Odibat, O., and Reddy, C. K. A generalized framework for mining arbitrarily positioned overlapping co-clusters. In *Proceedings of the SIAM International Conference on Data Mining (SDM)* (2011), pp. 343–354.

[88] Palit, I., and Reddy, C. K. Scalable and Parallel Boosting with MapReduce. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2012. In press).

[89] Pandey, G., Myers, C., and Kumar, V. Incorporating functional inter-relationships into protein function prediction algorithms. *BMC Bioinformatics 10*, 1 (2009), 142+.

[90] Park, J. H., and Reddy, C. K. Scale-space based weak regressors for boosting. In *Proceedings of the 18th European Conference on Machine Learning, ECML 2007* (Warsaw, Poland, September 17-21 2007), pp. 666–673.

[91] Pavlidis, P., Weston, J., Cai, J., and Noble, W. S. Learning gene functional classifications from multiple data types. *Journal of Computational Biology 9*, 2 (2002), 401–411.

[92] Punera, K., and Ghosh, J. Enhanced hierarchical classification via isotonic smoothing. In *Proceedings of the 17th International Conference on World Wide Web (WWW)* (Beijing, China, 2008), pp. 151–160.

[93] Quinlan, J. R. Induction of decision trees. *Machine Learning 1*, 1 (Mar. 1986), 81–106.

[94] Quinlan, J. R. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[95] Ray, S. S., Bandyopadhyay, S., and Pal, S. K. Combining Multisource Information Through Functional-Annotation-Based Weighting: Gene Function Prediction in Yeast. *IEEE Transactions on Biomedical Engineering 56*, 2 (2009), 229–236.

[96] Read, J., Pfahringer, B., Holmes, G., and Frank, E. Classifier chains for multi-label classification. *Machine Learning 85*, 3 (2011), 333–359.

[97] Reddy, C. K., Chiang, H.-D., and Rajaratnam, B. Trust-tech-based expectation maximization for learning finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 7 (2008), 1146–1157.

[98] Reddy, C. K., and Park, J.-H. Multi-resolution Boosting for Classification and Regression Problems. *Knowledge and Information Systems (KAIS) 29*, 2 (November 2011), 435–456.

[99] Rosset, S., Perlich, C., Swirszcz, G., Melville, P., and Liu, Y. Medical data mining: Insights from winning two competitions. *Data Mining and Knowledge Discovery 20*, 3 (2010), 439–468.

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

[101] Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Güldener, U., Mannhaupt, G., Münsterktter, M., and Mewes, H. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research 32*, 18 (Oct 2004), 5539–5545.

[102] Schapire, R. E., and Freund, Y. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics 26* (1998), 322–330.

[103] Schapire, R. E., and Singer, Y. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn. 37*, 3 (Dec. 1999), 297–336.

[104] Schapire, R. E., and Singer, Y. BOOSTEXTER: Aboosting-based system for text categorization. *Machine Learning 39*, 2/3 (2000), 135–168.

[105] Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., and Džeroski, S. s. Predicting gene function in S. cerevisiae and A. thaliana using hierarchical multi-label decision trees. In *European Conference on Computational Biology* (Cagliari, Italy, September 2008), pp. 22–26.

[106] Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., and Džeroski, S. s. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics 11*, 2 (2010).

[107] Seeger, M. W. Cross-validation optimization for large scale structured classification kernel methods. *Journal of Machine Learning Research*, 9 (2008), 1147–1178.

[108] Sharan, R., Ulitsky, I., and Shamir, R. Network-based prediction of protein function. *Molecular Systems Biology 3*, 1 (2007), 88.

[109] Shatkay, H., Edwards, S., Wilbur, W. J., and Boguski, M. Genes, Themes, and Microarrays: Using Information Retrieval for Large-Scale Gene Analysis. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)* (La Jolla / San Diego, CA, USA, August 19-23 2000), vol. 8, pp. 317–328.

[110] Sokolov, A., and Ben-Hur, A. Hierarchical classification of gene ontology terms using the gostruct method. *Journal Bioinformatics and Computational Biology 8*, 2 (2010), 357–376.

[111] Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher, B. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomices cerevisiae by microarray hybridization. *Mol. Biol. Cell 9* (1998), 3273–3297.

[112] Stark, C., Breitkreutz, B., Reguly, T., Boucher, L., Breitkreutz, A., and Tyers, M. BioGRID: a general repository for interaction datasets. *Nucleic Acids Research 34* (2006), D535–D539.

[113] Sun, A., and Lim, E.-P. Hierarchical text classification and evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining* (Washington, DC, USA, 2001), ICDM '01, IEEE Computer Society, pp. 521–528.

[114] Tan, P.-N., Steinbach, M., and Kumar, V. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[115] Tian, W., Zhang, L. V., Taşan, M., Gibbons, F. D., King, O. D., Park, J., Wunderlich, Z., Cherry, J. M., and Roth, F. P. Combining guilt-by-association and guilt-by-profiling to predict Saccharomyces cerevisiae gene function. *Genome biology 9 Suppl 1*, Suppl 1 (2008), S7+.

[116] Tieu, K., and Viola, P. Boosting Image Retrieval. *International Journal of Computer Vision 56*, 1 (Jan. 2004), 17–36.

[117] Troyanskaya, O. G., Dolinski, K., Owen, A. B., Altman, R. B., and Botstein, D. A Bayesian framework for combining heterogeneous data sources for gene function prediction (in Saccharomyces cerevisiae). *Proceedings of the National Academy of Sciences of the United States of America 100*, 14 (July 2003), 8348–8353.

[118] Valentini, G. True path rule hierarchical ensembles for genome-wide gene function prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 8* (2011), 832–847.

[119] Valentini, G. True path rule hierarchical ensembles for genome-wide gene function prediction. *IEEE ACM Transactions on Computational Biology and Bioinformatics 8*, 3 (2011), 832–847.

[120] Valentini, G., and Re, M. Weighted true path rule: a multilabel hierarchical algorithm for gene function prediction. In *1st International Workshop on learning from Multi-Label Data (MLD-ECML 2009)* (Bled, Slovenia, 2009), pp. 133–146.

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

[122] Viola, P., and Jones, M. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing System 14* (2001), MIT Press, pp. 1311–1318.

[123] Von Mering, C., Huynen, M., Jaeggi, D., Schmidt, S., Bork, P., and Snel, B. STRING: a database of predicted functional associations between proteins. *Nucleic Acids Research 31*, 1 (jan 2003), 258–261.

[124] Von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S., Fields, S., and Bork, P. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature 417* (2002), 399–403.

[125] Wasserman, L. *All of Statistics: A Concise Course in Statistical Inference (Springer Texts in Statistics)*. Springer, Dec. 2003.

[126] Wu, F., 0002, J. Z., and Honavar, V. Learning classifiers using hierarchically structured class taxonomies. In *SARA* (2005), pp. 313–320.

[127] Wu, L., Oviatt, S. L., and Cohen, P. R. Multimodal integration - a statistical view. *IEEE Transactions on Multimedia 1*, 4 (1999), 334–341.

[128] Wu, L. F., Hughes, T. R., Davierwala, A. P., Robinson, M. D., Stoughton, R., and Altschuler, S. J. Large-scale prediction of Saccharomyces cerevisiae gene function using overlapping transcriptional clusters. *Nature Genetics 31* (2002), 255 – 265.

[129] Yan, R., Tesic, J., and Smith, J. R. Model-shared subspace boosting for multi-label classification. In *13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)* (New York, NY, USA, 2007), pp. 834–843.

[130] Zhang, M.-L., and Zhang, K. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'10)* (Washington D. C., USA, 2010), pp. 999–1007.

[131] Zhang, M.-L., and Zhou, Z.-H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition 40*, 7 (July 2007), 2038–2048.

[132] Zhou, Y., Young, J. A., Santrosyan, A., Chen, K., Yan, S. F., and Winzeler, E. A. In silico gene function prediction using ontology-based pattern identification. *Bioinformatics 21*, 7 (2005), 1237–1245.

ABSTRACT

HIERARCHICAL MULTI-LABEL CLASSIFICATION FOR PROTEIN FUNCTION PREDICTION GOING BEYOND TRADITIONAL APPROACHES

by

NOOR ALAYDIE

August 2012

Advisor: Farshad Fotouhi and Chandan Reddy

Major: Computer Science

Degree: Doctor of Philosophy

Functional classification of genes is a challenging problem in functional genomics due to several reasons. First, each gene participates in multiple biological activities. Second, the genes are classified according to a hierarchical classification scheme that represents the relationships between genes functions. In addition, various biomolecular data sources, such as gene expression data and protein-protein interaction data, can be used to assign biological functions to genes.

In order to address these issues, this thesis proposes new algorithms for the hierarchical multi-label classification. Hierarchical multi-label classification is a variant of conventional classification in which the instances can belong to several labels, that are in turn organized in a hierarchy. The purpose of this thesis is threefold: first, **H**ierarchical **M**ulti-**L**abel classification algorithm using **B**oosting classifiers, *HML-Boosting*, for the hierarchical multi-label classification problem in the context of gene function prediction is proposed. Moreover, we propose the HiBLADE algorithm (**Hi**erarchical multi-label **B**oosting with **LA**bel **DE**pendency), a novel algorithm that takes advantage of not only the pre-established hierarchical taxonomy of the classes, but also effectively exploits the hidden correlation among the classes, thereby improving the quality of the predictions. The primary objective of the proposed algorithm is to find

and share a number of base models across the correlated labels. The HiBLADE algorithm is different from the conventional algorithms in two ways. First, it allows the prediction of multiple labels at the same time while maintaining the hierarchy constraint. Second, the classifiers are built based on the label under-study and its most similar sibling. Experimental results on several real-world biomolecular datasets show that the proposed method can improve the performance of hierarchical multi-label classification.

More important, however, is the third part that focuses on the integration of multiple heterogeneous data sources for improving hierarchical multi-label classification. We explore the integration of heterogeneous data sources for genome-wide gene function prediction with a novel **Hi**erarchical **B**ayesian **iN**tegration algorithm, *HiBiN*, a general framework that uses Bayesian reasoning to integrate heterogeneous data sources for accurate gene function prediction. The system formally uses posterior probabilities to assign class memberships to samples using multiple data sources while maintaining the hierarchical constraint. We demonstrate that the integration of the diverse datasets significantly improves the classification quality for gene function prediction in terms of several measures, compared to single-source prediction and fused-flat models, which are the baseline methods compared against. Moreover, the system has been extended to include a weighting scheme to control the contributions from each data source according to its relevance to the label under-study. The results show that the new weighting scheme compares favorably with the other approach along various performance criteria.

AUTOBIOGRAPHICAL STATEMENT

NOOR ALAYDIE

Dr. Alaydie was born in Amman, Jordan, to Dr. Naser Alaydie and Farah Naami. She spent her early childhood in several remote cities, where her father participated medicine. Dr. Alaydie finished her high school with being top in her school. In 2001, Dr. Alaydie was awarded a Bachelor of Science degree in Computer Science from the University of Jordan in Jordan. She received a Master of Science degree in Computer Science in 2005. In 2006, she married Omar Odibat, who has been her best friend and companion. They have two daughters, Sarah, who is 4 years old, and Hala, who is 1 year old.

Following three semesters of teaching at the University of Jordan in Jordan, Dr. Alaydie then moved to Detroit, Michigan to pursue graduate studies in Computer Science - data mining. She earned another Master of Science degree in Computer Science from Wayne State University in 2012. Her Ph.D. thesis demonstrates that using hierarchical knowledge about class labels and integrating multiple heterogeneous data sources improves the prediction results of the hierarchical multi-label classification problems.