Hierarchical Multi-label Classification for tree and DAG Hierarchies

By:

Mallinali Ramírez Corona

Thesis submitted as partial fulfillment
of the requirement for the degree of:

MASTER OF SCIENCE IN COMPUTER SCIENCE

at

Instituto Nacional de Astrofísica, Óptica y Electrónica

October, 2014
Tonantzintla, Puebla

Advisors:

Dr. Luis Enrique Sucar Succar
Dr. Eduardo Morales Manzanares

Agradecimientos

Primero que nada quiero agradecer a mi familia por siempre estar cerca, echarme porras y preguntarme ¿cómo vas?, ¿cuándo terminas?. A mis papás, hermana, abuelos (aunque mis abuelas me miran desde el cielo), tíos y primos.

A Miguel de Jesús Osorio Ramos por tolerarme tantas veces en *modo tesis*, aguantar que le ignorara y tener que escuchar mis interminables soliloquios, ademas de aportarme sus opiniones e ideas y en general su apoyo y amor incondicionales que me dieron fuerza para esta etapa.

A mis compañeros del INAOE muy especialmente a los del Chavira con quienes conviví y comí muchas veces. A a los de las retas de fútbol de los jueves por dejarme jugar y tratar de no pegarme con la pelota, además de los momentos de diversión fuera de la cancha que compartimos. También a mis amigos de la vida Citlalli, Dianita, Isa, Ernesto y más a quienes no es necesario ver para saber que me mandan sus buena vibras.

A mis asesores Dr. Luis Enrique Sucar Succar y Dr. Eduardo Morales Manzanares, por convertir mis ideas a medio cocinar en algo mas tangible digno de escribirse, por impulsarme y exigirme para lograr publicar mi trabajo y por leer y releer mis intentos de artículos y tesis. Al Dr. Morales especialmente por escuchar mis inseguridades sobre los objetivos, contenido y hasta el color de los pósters y, claro, recibirme cada vez que lo fui a buscar. Al Dr. Sucar por su paciencia, apoyo y la presión que de vez en cuando me ponía a prueba (para el lunes terminar la entrega y escribir un articulo). Al Dr. Orihuela por sus duros comentarios y observaciones que finalmente se convertían en explicaciones y apoyo para este trabajo. A los Dres. Montes y Villaseñor por sus comentarios al revisar mi tesis.

Al INAOE por las facilidades y apoyos brindados, al CONACYT por la beca número 401412/280088 que permitió que dedicara todo este tiempo únicamente a la maestría.

El agradecimiento más especial es para Edith Corona Sánchez y Armando Ramírez Medina por convencerme que lo que me proponga hacer es posible. Gracias por su amor, confianza y apoyo que me han traído de la mano desde niña.

Abstract

The core of supervised classification consists in assigning to an object or phenomenon one of a previously specified set of categories or classes. There are more complex problems where, instead of a single label, a set of labels are assigned to each instance, this is called multi-label classification. When the labels in a multi-label classification problem are ordered in a predefined structure, typically a tree or a Direct Acyclic Graph (DAG); the task is called Hierarchical Multi-label Classification (HMC).

There are HMC methods that create a global model which take advantage of the relations (predefined structure) of the labels. However these methods tend to create too complex models unusable for large scale data. Other methods divide the problem in a set of subproblems, which usually does not benefit from the predefined structure.

This thesis addresses the problem of hierarchical classification for tree and DAG structures considering large datasets with a considerable number of labels. A local classifier per parent node is trained for each non-leaf node in the hierarchy. Our method exploits the correlation of the labels with its ancestors in the hierarchy and evaluates each possible path from the root to a leaf node, taking into account the level of the predicted labels to give a score to each path and finally return the one with the best score.

In some cases there are instances whose labels do not reach a leaf node, for this cases we developed an extension of the base method for Non Mandatory Leaf Node Prediction (NMLNP); in which a pruning phase is performed before selecting the best path.

We noticed that many evaluation measures scored the short paths that only predict the most general cases better than longer more specific paths, that is why we also propose a new evaluation measure that avoids the bias toward conservative predictions in the case of NMLNP.

We tested our methods with 18 datasets with tree and DAG structured hierarchies against a number of state-of-the-art methods. The evaluation shows the advantages of these methods, in terms of predictive performance, execution time and scalability compared with other methods for hierarchical

classification. Our methods proved to obtain superior results when dealing with deep hierarchies and competitive with shallower hierarchies.

Resumen

La parte medular de la clasificación supervisada es asignar a un objeto o fenómeno un elemento de un conjunto previamente definido de clases o categorías. Hay problemas más complejos donde se asigna un conjunto de clases en vez de una sola, a este tipo de problemas se les llama multi-etiqueta. Cuando las etiquetas de un problema multi-etiqueta están ordenadas en una estructura predefinida, usualmente un árbol o un Grafo Acíclico Dirigido (DAG), se le denomina Clasificación Jerárquica Multi-Etiqueta (HMC).

Existen métodos de HMC que aprovechan las relaciones entre las etiquetas (estructura predefinida), sin embargo estos métodos tienden a crear modelos demasiado complejos que serían imposibles de utilizar en datos de gran escala. Otro grupo de métodos divide el problema en varios sub-problemas que usualmente no aprovechan la información de la estructura predefinida.

Esta tesis aborda el problema de clasificación jerárquica para estructuras de árbol y DAG considerando datos de gran escala y con muchas etiquetas. Por cada nodo no hoja en la jerarquía se entrena un clasificador local. Nuestro método explota las relaciones de las etiquetas con sus ancestros en la jerarquía y evalúa cada camino posible de la raíz a un nodo hoja, tomando en cuenta el nivel de la etiquetas predichas para asignar una calificación a cada camino y finalmente regresar el que tenga la mejor.

En algunos casos las etiquetas de las instancias no llegan a las etiquetas más específicas, para estos casos desarrollamos una extensión de nuestro método para predicción no obligatoria de nodos hoja, en la que se realiza un fase de poda antes de seleccionar el mejor camino.

Durante la experimentación notamos que muchas medidas de evaluación beneficiaban a las predicciones cortas que solo contienen las clases más generales, por eso proponemos una nueva medida de evaluación que evita devolver predicciones conservadoras (solo contienen las clases más generales de la jerarquía) que no contribuyen a la tarea de clasificación.

Probamos nuestros métodos con 18 conjuntos de datos de estructuras de árbol y DAGs contra varios métodos de estado del arte. La evaluación muestra las ventajas de nuestros métodos en términos de desempeño, tiempo de

ejecución y escalabilidad. Nuestros métodos obtuvieron resultados superiores en los conjuntos de datos con jerarquías profundas y resultados competitivos en jerarquías poco profundas.

Contents

List of Figures v

List of Tables vii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Research Issues . . . 3
  - 1.2.1 Research questions . . . 3
- 1.3 Objectives . . . 4
  - 1.3.1 Specific Objectives . . . 4
- 1.4 Proposed Solution . . . 4
- 1.5 Contributions . . . 5
- 1.6 Document Guide . . . 5

2 Theoretical Framework 7
- 2.1 Supervised Classification . . . 7
  - 2.1.1 Algorithm Selection . . . 8
- 2.2 Multi-label Classification . . . 11
  - 2.2.1 Algorithm Adaptation Methods . . . 11
  - 2.2.2 Problem Transformation Methods . . . 12
- 2.3 Summary . . . 14

3 Related Work 15
- 3.1 Hierarchical Multi-label Classification . . . 15
  - 3.1.1 Flat Classification Approach . . . 17
  - 3.1.2 Local Classifiers Approach . . . 17
  - 3.1.3 Non-mandatory leaf node prediction and the blocking problem . . . 25
  - 3.1.4 Global Classifier (big-bang) approach . . . 26
- 3.2 Evaluation Measures . . . 27
  - 3.2.1 Analysis . . . 30
- 3.3 Summary . . . 30

4 Chained Path Evaluation 31
4.1 Training . . . 31
4.2 Merging Rule . . . 33
4.3 Classification for MLNP . . . 35
4.4 Classification for NMLNP . . . 37
4.5 Gain-Loose Balance . . . 39
4.6 Summary . . . 41

5 Experiments and Results 43
5.1 Experimental Setup . . . 43
5.1.1 Datasets . . . 43
5.1.2 Data Topology . . . 46
5.2 Experiments . . . 48
5.2.1 Base Classifier . . . 49
5.2.2 Weighting Scheme . . . 50
5.2.3 MLNP . . . 53
5.2.4 Hierarchy depth effect over classification performance . 63
5.2.5 NMLNP . . . 65
5.2.6 Time . . . 68
5.3 Summary . . . 70

6 Conclusions 77
6.1 Conclusions . . . 77
6.1.1 Weighting Scheme . . . 78
6.1.2 Depth effect . . . 78
6.1.3 Comparing CPE against other methods . . . 78
6.1.4 NMLNP . . . 79
6.2 Contributions . . . 79
6.3 Future Work . . . 80

References 81

A Base Classifier 91

B Weighting Scheme 97
B.1 RSM vs NW scheme . . . 97
B.2 RSM vs other weighting schemes . . . 100

C Hierarchy depth effect over classification performance 107

D Comparison of MLNP approaches 113
D.1 Tree Structured Datasets . . . 113
D.2 DAG Structured Datasets . . . 117

E Selection of the best NMLNP approach 121

F Comparison of NMLNP approaches 123
F.1 Tree Structured Datasets 123
F.2 DAG Structured Datasets 127

List of Figures

2.1 Examples of methods to learn Bayesian Networks 10

3.1 Example of a tree hierarchy structure. 15
3.2 Hierarchical structures used in HMC 16
3.3 Flat approach example 17
3.4 Local Classifier per Node approach example 20
3.5 Top-Down approach example 20
3.6 Local Classifier per Parent Node approach example 23
3.7 Local Classifier per Level approach example 24
3.8 Global Classifier approach example 27

4.1 Example of the selection of the training set 32
4.2 Example of the computation of the weight of the nodes 34
4.3 Example of the computation of the weight of the nodes with two approaches 34
4.4 Example of the application of the merging rule 36
4.5 Example of hierarchy where pruning was performed 41

5.1 Structure of cellcycle_FUN dataset for MLNP 44
5.2 Plots comparing the performance with hierarchies of different depths 64

C.1 Performance of CPE with hierarchies of different depths 108
C.2 Performance of TD with hierarchies of different depths 109
C.3 Performance of TD-C with hierarchies of different depths . . . 110
C.4 Performance of HIROM with hierarchies of different depths . . 111

D.1 CPE vs other MLNP methods for tree datasets 114
D.1 CPE vs other MLNP methods for tree datasets 115
D.1 CPE vs other MLNP methods for tree datasets 116
D.2 CPE vs other MLNP methods for DAG datasets 117
D.2 CPE vs other MLNP methods for DAG datasets 118

D.2 CPE vs other MLNP methods for DAG datasets 119
D.2 CPE vs other MLNP methods for DAG datasets 120

F.1 CPE vs other NMLNP methods for tree datasets 124

List of Tables

5.1 Description of the datasets for the MLNP experiments 45
5.2 Description of the datasets for the NMLNP experiments 46
5.3 Dimensionality reduction for tree structured datasets 47
5.4 Dimensionality reduction for DAG structured datasets 48
5.5 Performance of the different base classifiers 49
5.6 RSM against a non weighted approach 51
5.7 RSM against other weighting schemes 52
5.8 Accuracy comparing our method against other MLNP methods 53
5.9 Hamming Accuracy comparing our method against other methods 54
5.10 Exact Match comparing our method against other methods . . 54
5.11 F1-macro D comparing our method against other methods . . . 55
5.12 F1-macro L comparing our method against other methods . . . 55
5.13 H-loss comparing our method against other methods 56
5.14 Accuracy comparing our method against other methods 57
5.15 Hamming Accuracy comparing our method against other methods 57
5.16 Exact Match comparing our method against other methods . . 58
5.17 F1-macro D comparing our method against other methods . . . 58
5.18 F1-macro L comparing our method against other methods . . . 58
5.19 H-loss comparing our method against other methods 59
5.20 Accuracy comparing CPE against Flat approach 60
5.21 Hamming Accuracy comparing CPE against Flat approach . . . 60
5.22 Exact Match comparing CPE against Flat approach 61
5.23 Exact Match comparing CPE against Flat approach 61
5.24 F1-macro L comparing CPE against Flat approach 62
5.25 H-loss comparing CPE against Flat approach 62
5.26 CPE-NMLNP against CPE-MLNP using MLNP datasets. 66
5.28 Accuracy comparison of NMLNP methods 67
5.27 CPE-NMLNP against CPE-MLNP using NMLNP datasets . . . 71
5.29 Hamming accuracy comparison of NMLNP methods 72

5.30 Exact Match comparison of NMLNP methods 72
5.31 F1-macro D comparison of NMLNP methods 73
5.32 F1-macro L comparison of NMLNP methods 73
5.33 H-loss comparison of NMLNP methods 74
5.34 Gain-Loose Balance comparison of NMLNP methods 74
5.35 CPE-NMLNP against another NMLNP method for DAG datasets 74
5.36 Time (seconds) required for training MLNP methods. 75
5.37 Time required for training NMLNP datasets. 75
5.38 Time (seconds) required for testing MLNP methods. 76
5.39 Time required for testing NMLNP datasets. 76

A.1 Accuracy using five different base classifiers. 91
A.2 Hamming Accuracy using five different base classifiers. 92
A.3 Exact Match using five different base classifiers. 93
A.4 F1-macro D using five different base classifiers. 94
A.5 F1-macro L using five different base classifiers. 95
A.6 H-loss using five different base classifiers. 96

B.1 Accuracy (%) of RSM vs NW scheme 97
B.2 Hamming Accuracy (%) of RSM vs NW scheme 98
B.3 Exact Approach (%) of RSM vs NW scheme 98
B.4 F1-macro D (%) of RSM vs NW scheme 99
B.5 F1-macro L (%) of RSM vs NW scheme 99
B.6 H-loss of RSM vs NW scheme 100
B.7 Accuracy of RSM vs other weighting schemes 101
B.8 Hamming Accuracy of RSM vs other weighting schemes 102
B.9 Exact Match of RSM vs other weighting schemes 103
B.10 F1-macro D of RSM vs other weighting schemes 104
B.11 F1-macro L of RSM vs other weighting schemes 105
B.12 H-loss of RSM vs other weighting schemes 106

E.1 NGLB (%) for different approaches for pruning phase 122

F.1 Accuracy (%) comparison of MLNP methods. 127
F.2 Hamming Accuracy (%) comparison of MLNP methods. 127
F.3 Exact Match (%) comparison of MLNP methods. 128
F.4 F1-macro D (%) comparison of MLNP methods. 128
F.5 F1-macro L (%) comparison of MLNP methods. 128
F.6 H-loss comparison of MLNP methods. 129
F.7 NGLB (%) comparison of MLNP methods. 129

Abbreviations

BCC Bayesian Chain Classifier

BR Binary Relevance

BU Button-Up

CC Classifier Chain

CPE Chained Path Evaluation

CPE-MLNP Chained Path Evaluation for Mandatory Leaf Node Prediction

CPE-NMLNP Chained Path Evaluation for Non-Mandatory Leaf Node Prediction

DAG Directed Acyclic Graph

GLB Gain-Loose Balance

GO Gene Onthology

H-loss Hierarchical Loss

HMC Hierarchical Multi-Label Classification

IQR Interquartile Range

LCL Local Classifier per Level

LCN Local Classifier per Node

LCPN Local Classifier per Parent Node

LP Label Power Set

MBC Multi-dimensional Bayesian network Classifier

MLNP Mandatory Leaf Node Prediction

MPP Multiple Path Prediction

NB Naïve Bayes

NMLNP Non-Mandatory Leaf Node Prediction

NN Nearest Neighbors

PCA Principal Component Analysis

RSM Ramirez, Sucar, Morales weighting scheme

SPP Single Path Prediction

SVM Support Vector Machine

TD Top-Down

Chapter 1

Introduction

Many important real-world problems are naturally cast as a hierarchical classification task, where the set of classes to be predicted is organized with an underlying, predefined class hierarchy, typically a tree or a Directed Acyclic Graph (DAG). The goal is to make predictions for a problem that has an underlying structure. There is a very large amount of research in conventional, non hierarchical classification problems. However, the problems where the classes are hierarchically organized, make the classification problem much more challenging because the predicted classes have restrictions about the combination of labels that can be predicted to make sense inside the hierarchy.

1.1 Motivation

The task of making a prediction incorporating the underlying structure of the labels into the decision making process is called Hierarchical Multi-Label Classification (HMC). This approach can achieve better classification rates than approaches that do not take into account the structure and return a set of labels that makes sense inside the hierarchy.

Some of the fields and applications where hierarchical classification has been successfully applied are described below.

Text Categorization

The task of text categorization is to assign categories to describe the content of a document in order to have a standardized, universal way for referring or describing the text.

An example of the application of HMC in text categorization can be found in the categorization of articles. Ruiz and Srinivasan (2002) classified biomedical articles from the Medline library into one or several keywords from a specified tree structure named Medical Subject Headings (MeSH); MeSH is a manually built controlled vocabulary for the biomedical domain where terms are arranged into several hierarchical structures called MeSH Trees.

The Reuters-21578 dataset has been widely used for research, even though this is a rather small and tidy collection, Reuters-21578 dataset appeared on the Reuters news wire in 1987 and has been used in the works of Koller and Sahami (1997), Weigend et al. (1999) and Hernandez et al. (2013) among others. Other interesting application is in the categorization of contents in the web, Labrou and Finin (1999) used a collection of names of Yahoo! as the set of labels, McCallum et al. (1998) used a set of web pages classified in a hierarchy of industry sectors and the Yahoo! dataset as well.

Image Classification

Automatic image annotation consist in automatically assigning metadata in the form of captions or keywords to a digital image. A single image may contain different meanings organized semantically in a hierarchy.

An application in the medical field is the classification of x-ray images, in the work of Dimitrovski et al. (2011) the database is accessed via textual information through the standard picture archiving and communication system (PACS). The same author (Dimitrovski et al., 2012) performed automatic *diatom* images classification. Diatoms are a large and ecologically important group of unicellular or colonial organisms (algae) of different kinds. The importance of classifying diatoms lie in the variety of uses, such as water quality monitoring, paleoecology and forensics.

Other interesting task is the photo annotation, where the goal is to predict which visual concepts are present in each image, examples are the works of Dimitrovski et al. (2010), Binder et al. (2010) and Hernandez et al. (2013) among others.

Protein Function Prediction

Assigning functions for unknown proteins is particularly interesting given the large increase in the number of uncharacterized proteins available for analysis.

As proteins often have multiple functions which are described hierarchically, the use of multi-label hierarchical techniques for the induction of classification models in Bioinformatics is a promising research area. Assigning functions for every protein with traditional experimental techniques could take decades, but the currently accumulated data from different biological sources make it possible to generate automated predictions that guide laboratory experiments and speed up the annotation process.

The biological functions that can be performed by proteins are defined in a structured, one standardized dictionary of terms is called Gene Ontology (GO). Costa and Lorena (2008); Alves et al. (2008); Otero et al. (2010) predict protein functions from information associated with the protein's primary sequence.

Secker et al. (2010) addresses the same problem but considering the functional classification of G-Protein-Coupled Receptors (GPCRs).

1.2 Research Issues

HMC incorporates the structure of the labels by using different approaches. Below we describe the two existing approaches.

The first is a global approach where a single classification model is built from the training set; the global approach is effective taking into account the relations of the labels but as the number of attributes and classes increases the model become more and more complex and thus time consuming. Some global approaches are specific to a classification algorithm and can not be adapted to the necessities of the datasets unlike local approaches.

The second is a local approach that divides the problem in several subproblems according to a strategy (can be a local classifier per level, per node or per non leaf node). The main problem of this approach is that it does not incorporate the relations (underlying structure) in the local classification. Another problem is that some of them make local decisions that can not be corrected in further levels, causing a propagation of the error. Some approaches optimize a function to reduce the errors using the local predictions. The limiting factor of these methods is that the resulting predictions do not necessarily optimize the scores of other evaluation measures, focusing on a single aspect in the selection of the best prediction.

Our method belongs to the local approaches to include the advantages over large scale datasets but avoiding its problems. To incorporate the relations it adds an extra attribute with the prediction of the parent node and includes a weighting scheme that assigns more value to the predictions of the more general classes than those of the more particular classes. To avoid error propagation we score all the paths to select the better one when we have a complete overview of the possible predictions.

We noticed that the current evaluation measures benefit conservative predictions that return just the most general classes over the predictions that return more specific labels.

1.2.1 Research questions

This thesis will examine the following research questions:

1. How to incorporate the relations between the labels in local classification to improve the performance of HMC?

2. How to develop a new method that does not necessarily return the most specific classes in the hierarchy and thus prune the predictions but

maintains as much information as possible?

3. How to evaluate HMC that does not return the most specific labels as classification to avoid conservative predictions?

1.3 Objectives

The aim of this thesis is to develop a new approach for hierarchical multi-label classification able to take into account the relations of the labels in an efficient manner to deal with large hierarchies with tree and DAG structures.

1.3.1 Specific Objectives

- Develop a local HMC approach able to deal with both tree and DAG hierarchies. This novel approach should be able to:
  - incorporate the relations of the labels based on chain classifiers;
  - handle large, deep hierarchies (with many nodes and many levels).
- Incorporate the possibility to predict paths that lead to an intermediate node in the hierarchy (Non mandatory Leaf Node Prediction) by pruning the prediction in the node/label where no more information can be extracted.
- Prepare a new evaluation measure that minimizes the bias toward conservative predictions.

1.4 Proposed Solution

We propose a novel HMC local approach, Chained Path Evaluation (CPE-MLNP), to predict single paths in tree and multiple paths in DAG hierarchies (paths from the root down to a leaf node). In the classification stage, the predictions of all the local classifiers are combined, to estimate the probability of all paths in the hierarchy.

We develop an extension of the base method for Non Mandatory Leaf Node Prediction (CPE-NMLNP); in which a pruning phase is performed to select the best path. Additionally, we propose a new evaluation metric for NMLNP hierarchical classifiers, that avoids the bias toward conservative predictions. In general, the closer the predicted class is to the root of the hierarchy, the lower the classification error tends to be. On the other hand, such classification becomes less specific and, as a consequence, less useful.

The proposed approach was experimentally evaluated with 10 tree and 8 DAG hierarchical datasets in the domain of protein function prediction.

We contrasted: (i) the best classifier for the datasets, (ii) the impact of the weighting scheme, (iii) CPE-MLNP against several state of the art approaches, (iv) the hierarchy depth effect over the classification performance, (v) CPE-MLNP vs. CPE-NMLNP, (vi) CPE-NLNP against several state of the art approaches.

1.5 Contributions

This thesis addresses the task of HMC with local classifiers, where the problem is divided in various subproblems. This scheme allows the usage of any classifier allowing the local classifiers to be selected according to the application, the dataset or another project requirements (e.g., users require a comprehensible model).

This thesis contributes with a method for prediction of the most specific labels in the hierarchy (MLNP) [Ramírez et al. 2014a]; and one for intermediate labels in the hierarchy (NMLNP) [Ramírez et al. 2014b]. Both methods improve the predicting performance over related methods, specially for deep hierarchies. The main contributions are:

1. A method for HMC that incorporates several novel aspects:
    - A cost is assigned to each node depending on the level it has in the hierarchy, giving more weight to correct predictions at the top levels; the weight is shared linearly along the structure. This assigned cost tries to incorporate information of the structure in the prediction.
    - The relations between the nodes in the hierarchy are considered by incorporating the parent label as in chained classifiers.
    - The possibility to handle tree or DAG structures interchangeably.
2. An extension of the method for NMLNP.
3. A new hierarchical measure designed for non mandatory leaf node prediction which avoid conservative predictions.

Our claims are supported by experimental analysis and comparisons to related methods with a large collection of datasets under multiple measures for predictive performance.

1.6 Document Guide

This document is structured as follows:

- Chapter 2 provides an overview of the general classification problem.

- Chapter 3 discusses the relevant work in the hierarchical multi-label classification literature.
- Chapter 4 introduces our hierarchical multi-label classification approach.
- Chapter 5 includes the experimental setup and the performed experiments as well as discussion of the results.
- Chapter 6 provides the concluding remarks and the future work.

Chapter 2

Theoretical Framework

In this chapter we present the concepts and existing theories that will allow to comprehend the content of the thesis.

2.1 Supervised Classification

The core of supervised classification is to build a model or general rule that segment the domain into regions that can be associated with the classes of interest to a particular application, it is used to classify new objects from which we do not know the class. In practice those regions may overlap. Different methods vary in the way they identify and describe the regions in the space (Richards and Jia, 1999).

More formally, supervised classification consists of assigning to an object or phenomenon one of a previously specified categories or classes (Araújo, 2006). Suppose that we have a set of predictor variables $X = \{x_1, x_2, ..., x_N\}$ and a y variable that represents the real class. The real class is a member of the set of possible labels L. If |L|=2 , then the learning problem is called a binary classification problem. In a database D there are collected N cases of the problem, in which we know the class to which class they belong, in the form $D = \{(X_1, y_i), (X_2, y_2), ..., (X_N, y_N)\}$, in a problem of |L| different classes. These systems are capable to learn from the existing data to generalize and deal with new cases.

Supervised classification have been used to solve numerous problems like tumor detection (Mithun and Raajan, 2013), concede or reject bank credits (Bee Wah and Ibrahim, 2010), bankruptcy prediction (Aghaie and Saeedi, 2009), hand-written character recognition (Huang et al., 1991), chromosome abnormality detection (Karvelis et al., 2009), etc.

In the classification problem there are two possible cases to express the complexity of the problem. There are problems where the classes are totally separable, when all the objects with similar characteristics belong to the same class. There are more complex problems where the different classes are not

separable. It is produced when objects with similar characteristics belong to different classes.

The process to design a usable pattern recognition system involves the following elements (Kotsiantis, 2007):

1. Collection of the dataset. An expert in the area can suggest the most informative features. If there is not an expert, a commonly used approach is to measure every feature available, this is called brute force. However, brute force often does not retrieve data directly suitable for classification. In most cases it contains noise and missing feature values, and therefore it requires preprocessing.

2. Pre-processing data. This stage handles and cleans the dataset to make the classification stage more robust and generalizable and let the algorithms work faster and more effectively. Some of the task in this stage are: handle missing values, sample instances from large datasets, etc.

3. Variables selection and extraction: Selection of a feature subset to remove irrelevant and redundant features.

4. Decision making. Involves selecting the algorithm and the evaluation of the classification process.

2.1.1 Algorithm Selection

A large number of techniques for supervised classification have been developed. There are three main groups: geometric (template matching), logical (syntactic/symbolic) and statistical (Bayesian Networks, Instance-based techniques)(Jain and Duin, 2000; Flach, 2012; Tsoumakas and Katakis, 2007).

2.1.1.1 Geometric Models

The instance space is the set of all possible instances, whether they are present in our dataset or not. A geometric model is constructed directly in the instance space, using geometric concepts such as lines, planes and distances.

Linear Discriminant Analysis (LDA) and the related Fisher's linear discriminant are simple methods used in statistics and machine learning to find the linear combination of features which best separate two or more classes of objects (Friedman, 1989). LDA works when the measurements made on each observation are continuous quantities. When dealing with categorical variables, the equivalent technique is Discriminant Correspondence Analysis (Mika et al., 1999).

A very useful geometric concept in classification is the notion of distance. Two instances are similar if the distance between them is small, and so nearby instances would be expected to receive the same classification, or belong to

the same cluster. Some examples of the usage of the distance are: Nearest Neighbors (NN) proposed by Cover and Hart (1967) and k-Nearest Neighbors (k-NN) proposed by Fix and Hodges Jr (1951).

Support vector machines (SVM), proposed by (Cortes and Vapnik, 1995), are a powerful kind of linear classifier. The main idea of SVMs is to find the optimal separating hyperplane between the classes by maximizing the margin between the classes closest points, the points lying on the boundaries of the margin are those called *support vectors*.

Template matching applications include image retrieval (Grujic et al., 2008), image recognition (Omachi and Omachi, 2007), image mosaicing and registration (Ding et al., 2001), object detection (Lin and Davis, 2010), and stereo matching (Huan et al., 2011).

2.1.1.2 Logical Models

Logical models adopt a perspective where a pattern is viewed as being composed of simple sub-patters which are composed themselves by simpler sub-patterns. The simplest sub-patterns are called primitives, and the complex pattern is represented in terms of the relations between the primitives. This sub patterns can be ordered in a list or in a hierarchy.

There are two important groups of logical learning methods: decision trees and rule-based classifiers.

Decision trees (Murthy, 1998) classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. An object X is classified by passing it through the tree starting at the root node. The test at each internal node along the path is applied to the attributes of X, to determine the next arc along which X should go down. The label at the leaf node at which X ends up is output as its classification.

Perhaps, the most well-known algorithm in the literature for building decision trees is the C4.5 (Quinlan, 1993). C4.5 is an extension of Quinlan's earlier ID3 algorithm (Quinlan et al., 1979).

Decision trees can be translated into a set of rules by creating a separate rule for each path from the root to a leaf in the tree (Quinlan, 1993). However, rules can also be directly induced from training data using a variety of rule-based algorithms.

The symbolic rule learning algorithms are also called separate-and-conquer or covering algorithms. All members of this family share the same top-level loop: search for a rule that explains a part of its training instances, separate these examples, and recursively conquer the remaining examples by learning more rules until no examples remain (Fürnkranz, 1999). RIPPER is a well-known rule-based algorithm (William et al., 1995).

2.1.1.3 Statistical Models

Statistical approaches are characterized by having an explicit underlying probability model, which yields the probability that an instance belongs to each class, rather than simply a classification.

Maximum entropy is one general technique for estimating probability distributions from data. The underlying principle in maximum entropy is that when nothing is known, the distribution should be as uniform as possible, that is, have maximal entropy (Csiszár, 1996).

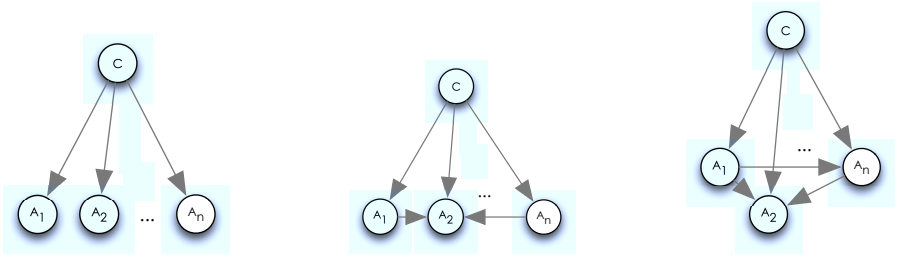
Bayesian networks (BN) are the quintessential instances of statistical learning algorithms (Jensen, 1996). These models capitalize on Bayes theorem (Equation (2.1)) where B would be the label to predict and A the observed attributes.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{2.1}$$

A BN is a graphical model for probability relationships among a set of variables (features). The BN structure S is a directed acyclic graph (DAG) and the nodes in S are in one-to-one correspondence with the features X. The arcs represent casual influences among the features while the lack of possible arcs in S encodes conditional independencies.

Naive Bayesian classifier (NB) are the most simple BN which S has only one parent (the label y) and several children (features X)(Good, 1950).

Two common methods to learn a BN are TAN (Tree Augmented Naïve-Bayes) and BAN (BN Augmented Naïve-Bayes). TAN starts creating a NB net and then incorporates dependencies between attributes by the construction of a tree between them. BAN incorporates a net to model the dependencies between attributes.



(a) Naïve Bayes

(b) Bayes Net TAN induced

(c) Bayes Net BAN induced

Figure 2.1: Examples of methods to learn Bayesian Networks

Statistical approaches have been successfully used in semiautomatic navigation during endoscopies Sucar and Gillies (1994), ozone prediction on Mexico City Sucar et al. (1995) and crime analysis Oatley and Ewart (2003) among others.

2.2 Multi-label Classification

The traditional classification task deals with problems where each example X is associated with a single label $y \in L$, where L is the set of classes, $|L| > 1$. However, some classification problems are more complex and multiple labels are needed ($|L| > 2$). This is called multi-label classification. A multi-label dataset D is composed of N instances (X_1, Y_1) , (X_2, Y_2), ..., (X_N, Y_N) where $Y_i \subseteq L$.

Examples of multi-label tasks can be found in text categorization and medical diagnosis. Text documents usually belong to more than one class. For example, a news story about the marriage between Shakira and Pique can be classified as sports and entertainment (De Comité et al., 2003). Similarly, in medical diagnosis, a patient may be suffering, for example, flu and asthma at the same time (Perotte et al., 2014).

Nowadays, multi-label classification methods are increasingly required by modern applications, such as protein function classification (Zhang et al., 2005), music categorization (Li and Ogihara, 2003), and semantic scene classification (Boutell et al., 2004). In semantic scene classification, a photograph can contain several objects at the same time, e.g. a photograph of a beach can contain sea, palm trees, people. Similarly, in music categorization, a song may belong to more than one genre. For example, a song can be categorized as pop and rock.

Existing methods for multi-label classification are grouped into two main categories (Tsoumakas and Katakis, 2007): a) problem transformation methods and b) algorithm adaptation methods. We define problem transformation methods as those methods that transform the multi-label classification problem either into one or more single-label classification or regression problems. We define algorithm adaptation methods as those methods that extend specific learning algorithms in order to handle multi-label data directly.

2.2.1 Algorithm Adaptation Methods

There is a group of algorithms that has been adapted to handle multi-label data (Tsoumakas et al., 2010). Examples in this group of algorithms are:

The original C4.5 algorithm was adapted for multi-label data (Clare and King, 2001) by modifying the formula for entropy, it is calculated as a weighted sum of the entropy for each subset where weighted sum means if an item appears twice in a subset because it belongs to two classes then we count it twice. A separate rule will be generated for each class at the leafs of the decision tree.

Adaboost.MH and Adaboost.MR (Schapire and Singer, 2000) are two extensions of AdaBoost (Freund and Schapire, 1997) for multi-label classification. They both apply AdaBoost on weak classifiers. In AdaBoost.MH, if the sign

of the output of the weak classifiers is positive for a new example X and a label y_i, then we consider that this example can be labelled with y_I; while if it is negative, then this example is not labelled with y_i. In AdaBoost.MR, the output of the weak classifiers is considered for ranking each of the labels in L.

ML-kNN (Zhang and Zhou, 2005) is an adaptation of the kNN lazy learning algorithm for multi-label data. Actually this method follows the One-Against-All paradigm. In essence, ML-kNN uses the kNN algorithm independently for each label: It finds the k nearest examples to the test instance and considers those that are labelled at least with y_i as positive and the rest as negative.

Elisseeff and Weston (2001) presented a ranking algorithm for multi-label classification. Their algorithm follows the philosophy of support vector machines (SVMs). It is a linear model that minimizes a cost function, while maintains a large separating margin. The cost function they use is ranking loss, which is defined as the average fraction of pairs of labels that are ordered incorrectly.

Godbole and Sarawagi (2004) present two improvements for the SVM classifier in conjunction with the One-Against-All approach for multi-label classification. The main idea is to extend the original data set with |L| extra features containing the predictions of each binary classifier. Then a second round of training |L| new binary classifiers takes place, this time using the extended data sets. For the classification of a new example, the binary classifiers of the first round are initially used, and their output is appended to the features of the example and then classified by the binary classifiers of the second round.

MMAC (Thabtah et al., 2004) is an algorithm that follows the paradigm of associative classification, which deals with the construction of classification rule sets using association rule mining, removes the examples associated with this rule set, and recursively learns a new rule set from the remaining examples until no further frequent items are left. These multiple rule sets might contain rules with similar preconditions but different labels on the right hand side.

2.2.2 Problem Transformation Methods

There exist several simple transformations that can be used to convert a multi-label data set to a single-label dataset with the same set of labels.

The most common approaches of problem transformation are the One-Against-One, the One-Against-All schemes (Lorena et al., 2008) and Label Power set (LP).

One-Against-One approach builds one classifier for each pair of classes. Thus, for a problem with |L| classes, $\frac{|L|(|L|-1)}{2}$ classifiers are trained to distin-

guish the samples of one class from the samples of another class. Usually, classification of an unknown pattern is done according to a voting scheme where each classifier votes for one class.

Binary relevance (BR) is an example of One-Against-All scheme. It transforms any multi-label problem into one binary problem for each label. Hence this method trains $|L|$ binary classifiers $C_1, C_2, ..., C_{|L|}$. Each classifier C_j is responsible for predicting the $0/1$ association for each corresponding label $y_j \in L$. For the classification of a new instance, BR outputs the union of the labels that are positively predicted by the $|L|$ classifiers.

LP considers each unique set of labels that exists in a multi-label training set as one of the classes of a new single-label classification task. It combines the entire label sets into atomic (single) labels to form a single-label problem for which the set of possible single labels represents all distinct label subsets in the original multi-label representation. Each (X, Y) is transformed into (X, y_i) where y_i is the atomic label representing a distinct label subset. Given a new instance, the single-label classifier of LP outputs the most probable class, which is actually a set of labels.

Two current problem transformation methods are explained below.

2.2.2.1 Multi-dimensional Bayesian Network Classifiers

Multi-dimensional Bayesian network Classifiers (MBCs), initially proposed by Gaag and Waal (2006), include one or more class variables and one or more feature variables. It models the relationships between the variables by acyclic directed graphs over the class variables and over the feature variables separately, and further connects the two sets of variables by a bipartite directed graph.

Bielza et al. (2011) extended MBC. Their probabilistic graphical model organizes class and feature variables as three different subgraphs: class subgraph, feature subgraph, and bridge (from class to features) subgraph. Under the standard 0–1 loss function, the most probable explanation (MPE) is computed. Under other loss functions defined in accordance with a decomposable structure, they derived theoretical results on how to minimize the expected loss.

2.2.2.2 Chain Classifiers

One important notion we have used to include the relations of the parent nodes is the idea of chain classifiers proposed by Read et al. (2011) and further extended by Zaragoza et al. (2011b,a); Sucar et al. (2014).

The Classifier Chain model (CC) proposed by Read et al. (2011) involves $|L|$ binary classifiers as in Binary Relevance methods (see Section 2.2.2). The

classifiers are linked along a chain where each classifier deals with the binary classification problem associated with a label $y_j \in L$.

A chain $C_1, ..., C_{|L|}$ of binary classifiers is formed. Each classifier C_j in the chain is responsible for learning and predicting the binary association of label y_j given the feature space, augmented by all prior binary relevance predictions in the chain $y_1, ..., y_{j-1}$ .The classification process begins at C_1 and propagates along the chain: C_1 determines $P(y_1|x)$ and every following classifier $C_2, ..., C_{|L|}$ predicts $P(y_j|x_i, y_1, ..., y_{j-1})$. This chaining method passes label information between classifiers, allowing CC to take into account label correlations.

The order of the labels in the chain is critical. Although there exist several possible heuristics for selecting a chain order for CC, the authors propose to make an ensemble calling it Ensemble Classifier Chains (ECC). It uses a different random chain ordering for each iteration in the building process.

Zaragoza et al. (2011b); Sucar et al. (2014); Zaragoza et al. (2011a) proposed a Bayesian Chain Classifier (BCC), which obtains a class dependency structure out of the data. This structure determines the order of the chain, so that the order of the class variables in the chain is consistent with the structure found in the first stage.

The order of the classes is based in the dependencies between classes given the features. They assume that the dependencies can be represented as a Bayesian Network (DAG) such that they build classifiers starting from the root and propagating the predictions downwards. They consider conditional independencies between classes to create simpler classifiers, constructing $|L|$ classifiers considering only the parent class of each class. For multi-label classification problems where there can be a large number of classes, they considered a small subset of related classes that produce competitive results against a more expensive strategy that uses all the previous classes in the chain.

2.3 Summary

In this thesis we will address the problem of multi-label classification with the particularity that the labels follow a predefined structure, typically a tree or a Direct Acyclic Graph (DAG), the task is called Hierarchical Multi-label Classification (HMC).

This chapter has presented an overview of the general classification task, starting form supervised classification. It describes the pattern recognition system designing process and the main techniques of classification. Multi-label classification is introduced and the main groups in which it is divided are explained.

Chapter 3

Related Work

This chapter presents a review of the work closely related to the HMC problem. It identifies and discuss past approaches and identify similar solutions to the proposed one.

3.1 Hierarchical Multi-label Classification

When the labels in a multi-label classification problems are ordered in a predefined structure, typically a tree or a Direct Acyclic Graph (DAG), the task is called Hierarchical Multi-label Classification (HMC). The class structure represent an "IS-A" relationship, these relations in the structure are asymmetric (e.g., all cats are animals, but not all animals are cats) and transitive (e.g., all Siameses are cats, and all cats are animals; therefore all Siameses are animals).

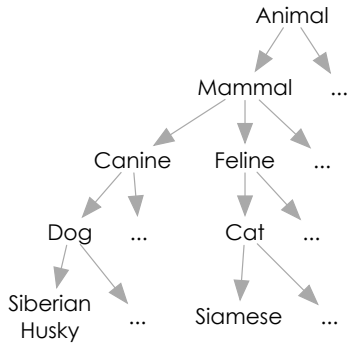


Figure 3.1: Example of a tree hierarchy structure.

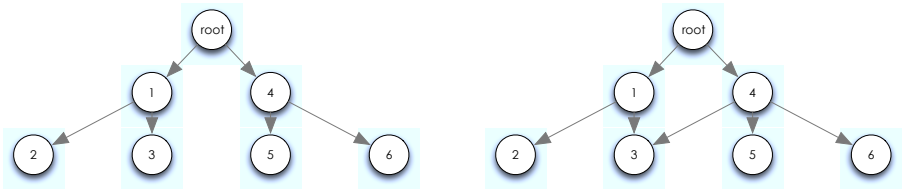
By taking into account the hierarchical organization of the classes, the classification performance may be boosted. In hierarchical classification, an example that belongs to certain class automatically belongs to all its superclasses (hierarchy constraint). When a prediction fulfill the hierarchy constraint it is called a consistent prediction. An example (using the hierarchy in Figure 3.1) of a consistent prediction would be *Animal, Mammal, Feline,*

Cat, Siamese; and and example of an inconsistent prediction would be *Animal, Mammal, Feline, Dog, Siberian Husky.*

Some major applications of HMC can be found in the fields of text categorization (Rousu et al., 2006; Sun and Lim, 2001; Kiritchenko et al., 2004), protein function prediction (Silla Jr. and Freitas, 2009a; Otero et al., 2010; Alves et al., 2008), music genre classification (Silla Jr. and Freitas, 2009b; Li and Ogihara, 2003), phoneme classification (Dekel et al., 2005, 2004), etc.

According to Freitas and de Carvalho (2007); Sun and Lim (2001) hierarchical classification methods differ according to four criteria: hierarchical structure, depth of the classification, number of paths returned, and exploration policy.

The first criterion is the type of hierarchical structure used. This structure is based on the problem structure and it typically is either a tree or a DAG (Figure 3.2).



(a) Tree structure **(b)** DAG structure

Figure 3.2: Hierarchical structures used in HMC

The second criterion is related to how deep the classification in the hierarchy is performed. That is, the hierarchical classification method can be implemented in a way that will always return a classification that includes a leaf node, referred as mandatory leaf-node prediction (MLNP), MLNP can be easily transformed into a flat problem by just taking into account the leaf labels (see Subsection 3.1.1). On the other hand the method can consider stopping the classification at any node in any level of the hierarchy, referred to as Non-Mandatory Leaf Node Prediction (NMLNP).

The third criterion refers to the number of paths returned, if the classifier returns one path from the root to a leaf node in the hierarchy, it is called Single Path Prediction (SPP); if it returns more than one path in the hierarchy from the root to different nodes, it is called Multiple Path Prediction (MPP).

Finally, the last criterion is related to how the hierarchical structure is explored. The current literature often refers to top-down (or local) classifiers, when the system employs a set of local classifiers; big-bang (or global) classifiers, when a single classifier coping with the entire class hierarchy is used; or flat classifiers, which ignore the class relationships, typically predicting only the leaf nodes, that is, a common multi-label classifier.

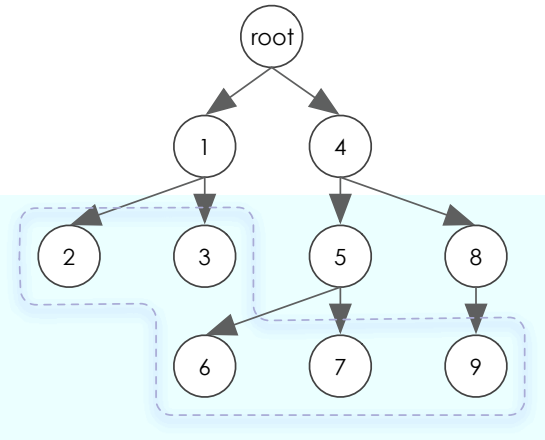


Figure 3.3: Flat Approach Example. A classifier that predicts only leaf nodes

3.1.1 Flat Classification Approach

The flat classification approach (Figure 3.3), is the simplest way to deal with hierarchical classification problems. It involves ignoring the class hierarchy completely, typically predicting only classes at the leaf nodes (MLNP). Traditional multi-label classification algorithms conform this approach. However, it provides an indirect solution to the problem of hierarchical classification; because when a leaf class is assigned to an example, one can consider that all its ancestor classes are also implicitly assigned to that instance. However, this very simple approach has the serious disadvantage of having to build a classifier to discriminate among a possibly large number of classes (all the leaf nodes), without exploiting information about parent-child class relationships present in the class hierarchy.

3.1.2 Local Classifiers Approach

In the local classifier approach the hierarchy is taken into account by using a local information perspective. This approach, can be grouped based on how the local information is grouped and the classifiers are built. More precisely, there are three main ways of using the local information: a local classifier per node (LCN), a local classifier per parent node (LCPN), and a local classifier per level (LCL).

The three types of local hierarchical classification algorithms differ significantly in their training phase but many of them share a very similar top-down approach in their testing phase. In this top-down approach, for each new example in the test set, the system first predicts its root-level (most generic) class, then it uses the predicted class to narrow the choices of classes at the next level (the only valid candidate second-level classes are the children of the class predicted at the first level), and so on, until the most specific prediction

is made.

A disadvantage of the top-down class-prediction approach (which is shared by all the three types of local classifiers) is that an error at a certain class level is going to be propagated downwards the hierarchy, unless some procedure for avoiding this problem is used.

If the problem is non-mandatory leaf node prediction, a blocking approach (where an example is passed down to the next lower level only if the confidence on the prediction at the current level is greater than a threshold) can avoid misclassifications to be propagated downwards, at the expense of providing the user with less specific (less useful) class predictions.

3.1.2.1 Training set selection

There are different ways to define the set of positive and negative examples for training the local classifiers, the most common are described below.

Exclusive Policy (Eisner et al., 2005). The training set of classifier C_j consists of two subsets: one is the positive training set ($Tr^+(C_j)$) where only the examples explicitly labeled as class y_j as the most explicit label are selected as part of it, while everything else belong to the negative training set ($Tr^-(C_j)$). In Figure 3.4, for the node 5 ($j = 5$), $Tr^+(C_5)$ includes all the examples which more specific class is 5; and $Tr^-(C_5)$ includes those examples which most specific class is 1, 2, 3, 4, 6, 7, 8, 9. This approach has several limitations. First, it does not consider the hierarchy when creating the training sets. Second, it is limited to problems that include examples with the most specific class at each node of the hierarchy. Third, using the descendants of y_j ($des(y_j)$ as negative examples is contrary to the notion that they also belong to y_j class. Fourth, it creates unbalanced training sets with few positive examples.

Less Exclusive Policy (Eisner et al., 2005). $Tr^+(C_j)$ consists of the examples which most explicit label is y_j and $Tr^-(C_j)$ includes the rest of the instances except those which most specific label is part of $des(y_j)$. In Figure 3.4, for $j = 5$, $Tr^+(C_5)$ is the set of examples which most specific class is 5, and $Tr^-(C_5)$ includes those examples which most specific class is 1, 2, 3, 4, 8, 9. This policy considers the hierarchy and assumes that the descendants of a label also belong to the positive training set, but the second and fourth problems of the previous approach still remain.

Less Inclusive Policy (Eisner et al., 2005; Fagni and Sebastiani, 2007). $Tr^+(C_j)$ consists of the examples which most explicit label is y_j or one in the set $des(y_j)$ and $Tr^-(C_j)$ includes the rest of the instances except those which most specific label is y_j or $des(y_j)$. In Figure 3.4, for $j = 5$, $Tr^+(C_5)$ is the set of examples which most specific class is 5, 6, 7, and $Tr^-(C_5)$ includes those examples

which most specific class is 1, 2, 3, 4, 8, 9. This approach then is not limited to problems that include examples with the most specific class at each node of the hierarchy.

Inclusive Policy (Eisner et al., 2005). $Tr^+(C_j)$ consists of the examples which most explicit label is y_j or $des(y_j)$ and $Tr^-(C_j)$ includes the instances which most specific label is not y_j, $des(y_j)$ or ancestor of y_j ($anc(y_j)$). In Figure 3.4, for $j = 5$, $Tr^+(C_5)$ is the set of examples which most specific class is 5, 6, 7, and $Tr^-(C_5)$ includes those examples which most specific class is 1, 2, 3, 8, 9.

Siblings (Fagni and Sebastiani, 2007; Ceci and Malerba, 2007) $Tr^+(C_j)$ consists of the examples which most explicit label is y_j or $des(y_j)$, and $Tr^-(C_j)$ includes the instances which most specific label is sibling of y_j ($sib(y_j)$) or $des(sib(y_j))$. In Figure 3.4, for $j = 5$, $Tr^+(C_5)$ is the set of examples which most specific class is 5, 6, 7, and $Tr^-(C_5)$ includes those examples which most specific class is 8, 9.

Exclusive Siblings (Ceci and Malerba, 2007). $Tr^+(C_j)$ consists of the examples which most explicit label is y_j and $Tr^-(C_j)$ includes the instances which most specific label is a $sib(y_j)$. In Figure 3.4, for $j = 5$, $Tr^+(C_5)$ is the set of examples which most specific class is 5 and $Tr^-(C_5)$ includes those examples which most specific class is 8.

3.1.2.2 Local classifier per node approach (LCN)

The local classifier per node approach (Figure 3.4), also called top-down approach, is one of the most used approaches in the literature. The LCN approach trains one binary classifier for each node of the class hierarchy (except the root node).

During the testing phase, regardless of how the positive and negative examples are defined, the output of each binary classifier will be a prediction indicating whether or not a given test example belongs to the classifier's predicted class. One advantage of this approach is that it is naturally multi-label in the sense that it is possible to predict multiple labels per class level. In the case of single-label, a problem that requires only one path of classes in the hierarchy, it is possible to enforce the prediction of a single class label per level by assigning just the class with the highest confidence from the classifier at a level. This approach, however, can lead to inconsistencies in class predictions, as one can predict a class which father has not been predicted. Therefore a correction method should be taken into account. Some of the most common correction approaches are explained below.

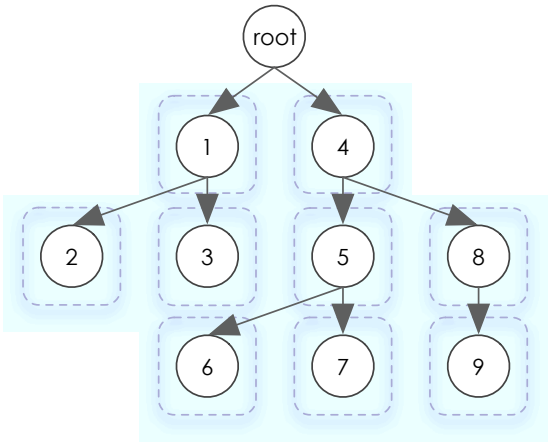


Figure 3.4: Local Classifier per Node approach example. A binary classifier is trained for each non-root node.

Koller and Sahami (1997) proposed the top-down approach. Its essential characteristic is in the testing phase in a top-down fashion. This method is forced to predict a leaf node. For example, in Figure 3.5, at the top level, the output of the local classifier for class 4 is true, and the output of the local classifier for class 1 is false. At the next level, the system will only consider the output of classifiers predicting classes which are children of class 4, labels 5 and 8. The main problem of this approach is that misclassifications at higher levels are propagated toward lower levels.

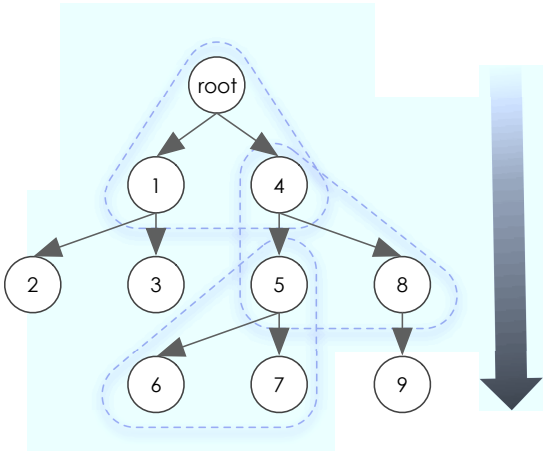


Figure 3.5: Top-Down approach example. The class predicted in a level is based on the class predicted at the parent level.

To deal with inconsistencies generated by the LCN approach, Wu et al. (2005) proposed the "Banalized Structured Label Learning" (BSLL). This method stops the classification once the binary classifier for a given node returns a negative prediction. For example, in Figure 3.5, if the output for the binary classifier of class 4 is true, and the outputs of the binary classifiers

for classes 5 and 8 are false, then this approach ignores the answer of all the lower level classifiers predicting classes that are descendants of classes 5 and 8 and outputs the class 4 to the user. This approach makes it more difficult to propagate the errors but does not completely solves the problem.

Dumais and Chen (2000) propose two class-membership inconsistency correction methods based on thresholds. In order for a class to be assigned to a test example, the probabilities for the predicted class are used. In the first method, they use a binary condition where the posterior probability of the classes at the first and second levels must be higher than a user specified threshold value, in the case of a two-level class hierarchy. The second method uses a multiplicative threshold value that takes into account the product of the posterior probabilities of the classes at the first and second levels. The limitation of this approach is that it is only designed for two-level hierarchies.

The main problem with the approaches that use threshold values is that the selection of a threshold is made by an expert, who is not always available or is unable to provide an adequate numerical value as they use empirical knowledge to perform the task. Otherwise a strategy of automatic threshold selection should be implemented.

DeCoro et al. (2007); Barutcuoglu and DeCoro (2006); Barutcuoglu et al. (2006) proposed a method for inconsistency correction based on a Bayesian aggregation of the output of the base binary classifiers. The method takes the class hierarchy into account by transforming the hierarchical structure of the classes into a Bayesian network. In Barutcuoglu and DeCoro (2006) two baseline methods for conflict resolution are proposed: the first method propagates negative predictions downward (i.e., the negative prediction at any class node is used to overwrite the positive predictions of its descendant nodes) while the second baseline method propagates the positive predictions upward (i.e., the positive prediction at any class node is used to overwrite the negative predictions of all its ancestors).

Valentini and Cesa-Bianchi (2008); Valentini (2009); Valentini and Re (2009) propose another approach for class-membership inconsistency correction based on the output of all classifiers, where the positive decisions for a node influence the decisions of the parents (bottom-up) turning them positive. Negative predictions in a node turn all its descendants negative. This approach is called True Path, latter upgraded to Weighted True Path (the nodes acquire a weight in the hierarchy). This method can propagate the errors, this time in a bottom-up fashion.

Bennett and Nguyen (2009) propose a technique called expert refinements. The refinement consists of using cross-validation in the training phase to obtain a better estimate of the true probabilities of the predicted classes. The refinement technique is then combined with a bottom-up training approach, which consists of training the leaf classifiers using refinement and passing this

information to the parent classifiers. They include the false positive instances which have been misclassified at its ancestor nodes, hoping that these instances will be be rejected by the current classifier and the misclassification errors will not be further propagated to the low level nodes. These instances have the risk of becoming noise at the low level nodes, making the trained classifiers weaker.

Alaydie et al. (2012) developed HiBLADE (Hierarchical multi-label Boosting with LAbel DEpendency); an LCN algorithm that takes advantage of, not only the predefined hierarchical structure of the labels, but also exploits the hidden correlation among the classes that is not shown through the hierarchy. This algorithm attaches the predictions of the parent nodes as well as the related classes. A common problem of this approach is that appending that amount of attributes can create models that over-fit the data.

The previous approaches have issued the problem in the context of a single label (per level) problem with a tree-structured class hierarchy. There are other methods that can handle classification of multiple-label, DAG-structured hierarchies. An example is the work of Bi and Kwok (2012, 2011) where they propose HIROM, a method that uses the local predictions (independently of the way they are trained) to search for the optimal consistent multi-label using a greedy strategy. Using Bayesian decision theory, they derive the optimal prediction rule by minimizing the conditional risk. A limitation of this approach is that it optimizes a function that does not necessarily performs well in other evaluation measures.

3.1.2.3 Local Classifier per Parent Node Approach (LCPN)

The Local Classifier per Parent Approach (Figure 3.6) trains a multi-class classifier, for each parent node in the class hierarchy, to distinguish between its child nodes. In order to train the classifiers the "siblings" policy, as well as the "exclusive siblings" policy, both presented in the Subsection 3.1.2.2, are adequate for this approach. During the testing phase, this approach can also be coupled with a top-down prediction approach.

Usually in the LCPN approach the same classification algorithm is used throughout all the class hierarchy but Secker et al. (2007) proposes a selective top-down approach. This approach produces a tree of classifiers, at each node the training data for that node is split into training and validation sets with randomly selected instances. Different classifiers are trained using this training data and tested using the validation set. The classifier with the highest classification accuracy in the validation set is selected as the classifier for the node. The classifier is retrained using both training and validation sets.

Extending the notion of selecting the training algorithm for each node, Holden and Freitas (2008) used a swarm intelligence algorithm for this pur-

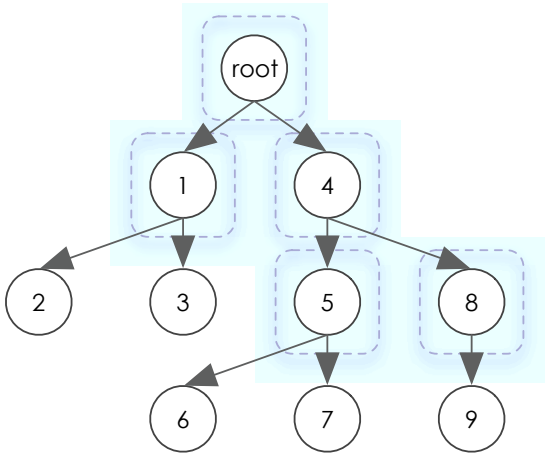


Figure 3.6: Local Classifier per Parent Node approach example, a classifier is trained for level of the hierarchy.

pose. This method optimizes the classification accuracy of the entire hierarchy on the validation set instead of optimizing each classifier separately and thus takes into account interaction among classifiers at different classifier nodes.

Silla Jr. and Freitas (2009b) proposed an LCPN algorithm combined with two selective methods for training. The first method selects the best features to train the classifiers, the second selects both, the best classifier and the best subset of features simultaneously, showing that selecting a classifier and features improves the classification performance. Secker et al. (2010) developed a similar method where, for each node in the hierarchy, the attributes and the classifier are chosen.

The main problem with these methods that train several classifiers for each node, is the huge increment in the training time specially in large hierarchies.

Bi and Kwok (2011, 2012) proposed HIROM, a method that uses the local predictions (independently of the way they are trained) to search for the optimal consistent multi-label using a greedy strategy. Using Bayesian decision theory, they derive the optimal prediction rule by minimizing the conditional risk. The limitation of this approach is that it optimizes a function that does not necessarily maximizes the performance in other evaluation measures.

The approach of Hernandez et al. (2013), used for tree structured taxonomies, learns an LCPN. In the classification phase, it classifies a new instance with the local classifier at each node, and combines the results of all of them to obtain a score for each path from the root to a leaf-node. Two fusion rules were used to achieve this: product rule and sum rule. Finally it returns the path with the highest score. This approach however does not consider that the length of the paths can affect the score; given that the method penalizes/favors longer paths with the product/sum rule. Also, it does not

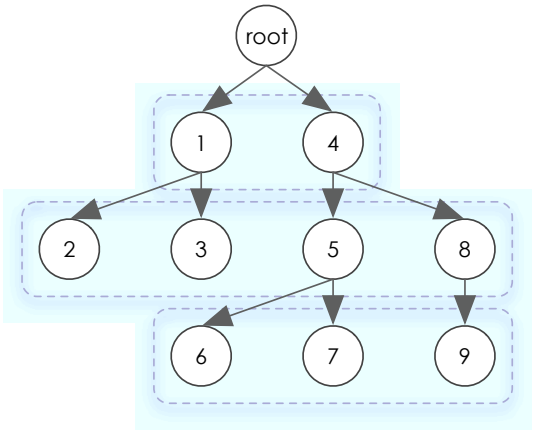


Figure 3.7: Local Classifier per Level approach example. A classifier is trained for level of the hierarchy (dotted squares).

take into account the relations of the nodes when classifying an instance.

The previous approaches fall into the category of single label problems (a single path from the root note to another node), with tree structured class hierarchies.

3.1.2.4 Local Classifier per level Approach (LCL)

The local classifier per level approach (Figure 3.7) consists on training one multi-class classifier for each level of the class hierarchy. The creation of the training sets here is implemented in the same way as in the local classifier per parent node approach.

One possible (although very naïve) way of classifying test examples using classifiers trained by this approach is to get the output of all classifiers (one classifier per level) and use this information as the final classification. The major drawback of this class-prediction approach is that it can cause class-membership inconsistency. Hence, if this approach is used, it should be complemented by a post-processing method to try to correct the prediction inconsistency. To avoid this problem a top-down approach can be used. In this context, the classification is done in a top-down fashion (similar to the standard top-down class-prediction approach), restricting the possible classification output at a given level only to the child nodes of the class node predicted at the previous level (in the same way as it is done in the LCPN approach).

This approach is the least used of the local approaches, it was used as a baseline comparison method in Clare and King (2003) and Costa et al. (2007). Cerri et al. (2013) used this approach combined with neural networks. Predictions made by a neural network at a given level are used as inputs to the network of the next level. The labels are predicted using a threshold

value. Finally, a post processing phase is used to correct inconsistencies. One problem with this approach is the selection of appropriate threshold values since Cerri et al. set a given threshold of 0.5 and Clare and King; Costa et al. do not specify it but the most probable is that they used 0.5 as well.

3.1.3 Non-mandatory leaf node prediction and the blocking problem

The Non-Mandatory Leaf Node (NMLNP) prediction problem, allows the most specific class predicted to any given instance to be a class at any level i.e., internal or leaf node; of the class hierarchy, and was introduced by Sun and Lim (2001). A simple way to deal with the NMLNP problem is to use a threshold value at each class node, if the confidence score or posterior probability of the classifier at a given class node for a given test example is lower than this threshold, the classification stops for that example. A method for automatically computing these thresholds was proposed by Ceci and Malerba (2007).

The use of thresholds can lead to what is called the blocking problem. Blocking occurs when, during the top-down process of classification of a test example, the classifier at a certain level in the class hierarchy predicts that the example in question does not have the class associated with that classifier. In this case the classification of the example will be *blocked*, meaning that the example will not be passed to the descendants of that classifier.

Three strategies to avoid blocking are discussed by Sun et al. (2003):

Threshold reduction method: This method consists of lowering the thresholds of the subtree classifiers. Reducing the thresholds allows more examples to be passed to the classifiers at lower levels. The challenge associated with this approach is how to determine the threshold value of each subtree classifier. This method can be easily used with both tree-structured and DAG-structured class hierarchies.

Restricted Voting: This method consists of creating a set of secondary classifiers that will link a node and its grandparent node. The restricted voting approach gives the low-level classifiers a chance to access these examples before they are rejected. This method was originally designed for tree-structured class hierarchies and extending it to DAG-structured hierarchies would make it considerably more complex and more computationally expensive, as in a DAG-structured class hierarchy each node might have multiple parent nodes.

Extended multiplicative thresholds: This method is an extension of the multiplicative threshold proposed by Dumais and Chen (2000), which

originally only worked for a 2-level hierarchy. The extension consists of establishing thresholds recursively for every two levels.

Recently Hernandez et al. (2013) proposed a new approach to achieve NMLNP and avoid the blocking problem. First the best path of classes in the hierarchy is obtained and afterwards the path is pruned in a bottom-up fashion, if there is not information gain (IG) from a child node to its parent.

A problem with using IG as pruning strategy is that, when applied in button-up (BU) fashion, it does not prune many nodes because there is no gain from children to parent, and often just the leaf nodes are pruned. When applied in top-down fashion it behaves too aggressively, it prunes the majority of the nodes, because there is no gain from parents to children. This is because IG is a difficult condition to comply.

3.1.4 Global Classifier (big-bang) approach

Global classifiers have two main characteristics: they consider the entire class hierarchy at once and they lack the kind of modularity for local training of the classifier that is a core characteristic of the local classifier approach.

The global approach learns a single global model for all classes. This kind of approach is known as the big-bang approach. In the global classifier approach, a single classification model is built from the training set, taking into account the class hierarchy as a whole during a single run of the classification algorithm. When used during the test phase, each test example is classified by the induced model, a process that can assign classes at potentially every level of the hierarchy to the test example.

Silla Jr. and Freitas (2009a) proposed an extension of the Naïve Bayes algorithm for flat classification, where a single global classification model is built by considering all the classes in the hierarchy.

Vens et al. (2008); Blockeel et al. (2006); Blockeel and Bruynooghe (2002) presented a global HMC method. It is a decision tree induction algorithm that is based on Predictive Clustering Trees (PCT) in which a distance measure is employed to calculate how similar are the training examples in the classification tree. PCT were also used by Dimitrovski et al. (2010). They constructed ensembles of PCT to improve the predictive performance in the detection of visual concepts and annotation of images.

Otero et al. (2009) proposed an ant colony optimization (ACO) hierarchical classification algorithm, hAnt-Miner. This method discovers classification rules for DAG hierarchies with single path classification for NMLNP. Later they proposed a similar method that works for multiple paths in the hierarchy, hmAnt-Miner (Otero et al., 2010). The main problem of this approaches is the compromise between the time it spends creating the model and the accuracy obtained.

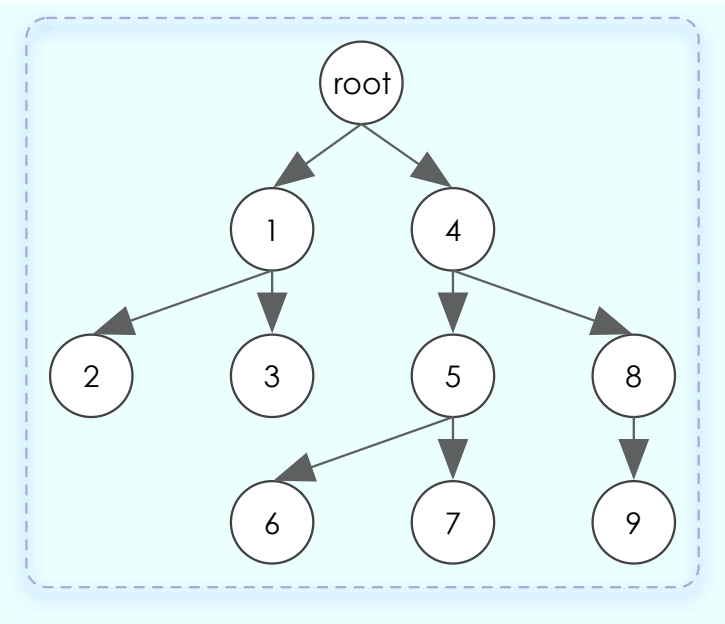


Figure 3.8: Global Classifier approach example. One classifier (dotted square) is trained for the complete hierarchy.

3.2 Evaluation Measures

Generally, the evaluation measures in classification problems are defined from a matrix with the numbers of examples correctly and incorrectly classified for each class, named confusion matrix where the false positive, false negative, true positive, true negative examples are represented. In the case of HMC the definition of this terns is not straightforward since the predictions can be partially correct. If the set of predicted labels is considered as a whole and must match exactly with the real labels the measure can be too demanding, since even a single error makes the prediction incorrect. If each label is evaluated as a separate example, this measure tends to be too soft due to the typical sparsity of labels in multi-label data. For this reason measures for conventional classification are not adequate for hierarchical multi-label problems. Specific measures for HMC have been proposed.

In the multi-label space, predictive performance can be measured in many ways, two of which are:

1. Label-set based evaluation. Each label set is evaluated separately (e.g. exact match, accuracy and F1-macro D).

2. Label-based evaluation. The binary relevance of each individual label is evaluated separately (e.g. hamming-loss and F1 macro L).

Different classifiers perform better under different evaluation measures. Since

it is possible to select evaluation measures to benefit certain methods, in this thesis, we used multiple evaluation measures.

The cardinality of set of labels is represented by $|L|$, N is the number of instances in the training set. The labels of an instance are represented in a vector of size $|L|$ where the predicted/real labels are marked with 1 and the rest with 0. Y_i represents the real set of labels and $\hat{Y}_i$ the predicted set of labels.

Exact Match

Exact match (Equation (3.1)), also known as label-set based accuracy, represents the proportion of the instances that were correctly predicted from the complete testing set.

$$ExactMatch = \frac{1}{N}\sum_{i=1}^{N} 1_{Y_i=\hat{Y}_i} \tag{3.1}$$

Exact Match is a flat performance measure and thus does not considers the problem's hierarchical class structure and ignore that the difficulty of classification usually increases with the depth of the classes to be predicted because they become more similar. Despite that fact, this measure, shows the effectiveness of the method to return complete correct paths.

Accuracy

Accuracy as multi-label measure was introduced by Godbole and Sarawagi (2004) (Equation (3.2)). This is the ratio of the size of the intersection and union of the predicted and actual label sets (represented by the logical AND and OR operations in bit-vector notation, respectively), taken for each example, and averaged over the number of examples.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} \frac{|Y_i \wedge \hat{Y}_i|}{|Y_i \vee \hat{Y}_i|} \tag{3.2}$$

This measure is less harsh than exact match because it consider the correctly classified labels but also ignores the the fact that the classification difficulty tends to increase for deeper levels of the class.

Hamming Loss and Hamming Accuracy

Hamming Loss, depicted in Equation (3.3), where $Y_i \oplus \hat{Y}_i$ is the symmetrical difference between Y_i and $\hat{Y}_i$ (the logical XOR operation), it is a label-based

evaluation measure. Hamming Loss evaluates the frequency that an example-label pair is misclassified.

$$\mathrm{HammingLoss} = \frac{1}{N|L|}\sum_{i=1}^{N}\left|Y_i \oplus \hat{Y}_i\right| \tag{3.3}$$

Hamming accuracy is defined as $\mathrm{HammingAccuracy} = 1 - \mathrm{HammingLoss}$.

F1-measure

F1 measure for multi-label classification, is defined as the scalar F1-measure (Equation (3.4)) but redefining precision and recall:

- $\mathrm{precision}$ as the fraction of predicted labels which are actually part of the true set of labels $\frac{|z_i \wedge \hat{z}_i|}{|\hat{z}_i|}$; and
- recall as the fraction of true labels which are also predicted $\frac{|z_i \wedge \hat{z}_i|}{|z_i|}$.

$$\mathrm{F1} = \frac{2 \times \mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{3.4}$$

We have specified a vector z instead of the y_i vector, because in the multi-label context there are several ways to average this measure, for this thesis in particular two of them are used:

F1-macro D (Equation (3.5)) is averaged by example (macro-averaged); N vectors of $z_i \equiv y_i$. This measure as averaged for instances

$$\mathrm{F1macro\,D} = \frac{1}{N}\sum_{i=0}^{N}\mathrm{F1}(z_i, \hat{z}_i) \tag{3.5}$$

F1-macro L (Equation (3.6)) is averaged by label (macro-averaged); M vectors of $z_i \equiv [y_i^1, ..., y_i^N]$.

$$\mathrm{F1macro\,L} = \frac{1}{|L|}\sum_{i=0}^{|L|}\mathrm{F1}(z_i, \hat{z}_i) \tag{3.6}$$

Precision and recall measures do take into account the hierarchical relationships between classes. The F1-macro L measure is a per class mean performance measure, the F1-macro D measure is a per instance mean performance measure.

Hierarchical Loss (H-loss)

Is a loss function, proposed by Cesa-Bianchi et al. (2006), that takes into account the hierarchical structure . The term y_i represents the real i^{th} label $\hat{y}_i$ the predicted i^{th} labels, and $\mathrm{anc}(i)$ the set of ancestors of node i.

$$\mathrm{Hloss} = \frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{|L|}\left\{\hat{y}_j \neq y_j \wedge \hat{y}_k = y_k, k \in \mathrm{anc}(j)\right\} \tag{3.7}$$

All paths in the hierarchy from a root down to a leaf are examined and, whenever a node i is encountered such that $\hat{y}_i \neq y_i$, then 1 is added to the loss, while all the loss contributions in the subtree rooted at i are discarded.

At the point where the error is discovered it is known that the predicted path is not equal to the real one, but the error propagation problem present in hierarchical classification is not taken into account.

3.2.1 Analysis

These six evaluation measures capture different aspects of the rightness of a predictions, exact match captures the complete correct paths, accuracy captures the rate of the real labels in the predicted set, Hamming loss (and thus Hamming accuracy) captures the error rate by number of labels, F1-macro D captures the compromise between precision and recall along the instances and thus the behavior along the dataset, F1-macro L does the same but for classes, H-loss captures the opposite to exact match, the paths which had an error.

Since an exact prediction is difficult to obtain in HMC, the current existing measures try to describe the rightness of a prediction. They capture some of the characteristics that are needed to determine if the prediction is good or bad, like the number of correctly predicted set labels and the real set of labels; but they ignore some crucial information obtained from the structure itself that allow conservative predictions to obtain good scores. The level on which the error was made and the number of siblings of a node are some examples. This lack of information causes that, in average, current measures to score conservative predictions better than more specific ones that are closer to real set of labels. We propose a new metric to avoid these problems, the new metric is described in Section 4.5.

3.3 Summary

This chapter described the approaches to tackle the HMC problem and collect the main works in the area. It also described common evaluation measures used for multi-label and hierarchical multi-label predictions, we found a gap in current evaluation measures that favor conservative predictions.

Next chapter describes a novel HMC approach and a new evaluation measure that tries to avoid favoring conservative predictions.

Chapter 4

Chained Path Evaluation

We developed a hierarchical classification method, named Chained Path Evaluation (CPE), that exploits the relation of the labels with its ancestors in the hierarchy. We evaluate each possible path from the root to a leaf or intermediate node using a merging rule that takes into account the level of the predicted labels to give a score to each path and finally return the one with the best score.

The method has two variants, one for Mandatory Leaf Node Prediction (MLNP) and one for Non Mandatory Leaf Node Prediction (NMLNP). The difference between them is a pruning step performed before emitting the final prediction included in NMLNP. They both have the same training process.

This chapter explains in detail the training and classification procedure we used for the proposed method, and the extension for NMLNP, as well a a novel metric proposed for NMLNP evaluation.

4.1 Training

Let D be a training set with N examples, $e_e = (X_e, Y_e)$, where X_e is a d-dimensional feature vector and $Y_e \subset L$, $L = \{y_1, y_2, ..., y_{|L|}\}$ is a finite set of $|L|$ possible labels or classes. It bears mentioning that regularly Y_e is a small set compared with $|L|$. Y_e is represented as a $0/1$ vector $Y_e \in \{0,1\}^{|L|}$, where $y_i = 1$ if and only if $y_i \in Y_e$ else $y_i = 0$.

A multi-class classifier C_i is trained for each non leaf node y_i (LCPN), henceforth called *base classifier*. Any classifier can be used as *base classifier* as long as it return a probability or its output can be converted to a probability. The classes in C_i are the labels of the children of y_i ($\mathtt{child}(y_i)$) plus an "*unknown*" label that corresponds to the instances that do not belong to $\mathtt{child}(y_i)$. To select the instances in the training set of C_i we use a modification of the siblings policy (see Subsection 3.1.2.2).

As in multidimensional classification, the class of each node in the hierarchy is not independent from the other nodes. To incorporate these relations,

inspired by chain classifiers, we include the class predicted by the parent node(s) as an additional attribute in the LCPN classifier. That is, the feature space of each node in the hierarchy is extended with the $0/1$ label associated to the parent (tree structure) or parents (DAG structure) of the node, as in a Bayesian Chain Classifier (Section 2.2.2.2).

The set of positive training examples ($Tr^{+}(C_i)$) consists of the instances where $child(y_i) = 1$, that is, all the instances that include a children of y_i in their set of classes ($child(y_i) \in Y_e$). Each instance in this set will be labeled with the corresponding $child(y_i)$ label. In this set, the added features of the parents will have the value true since all the instances are their children.

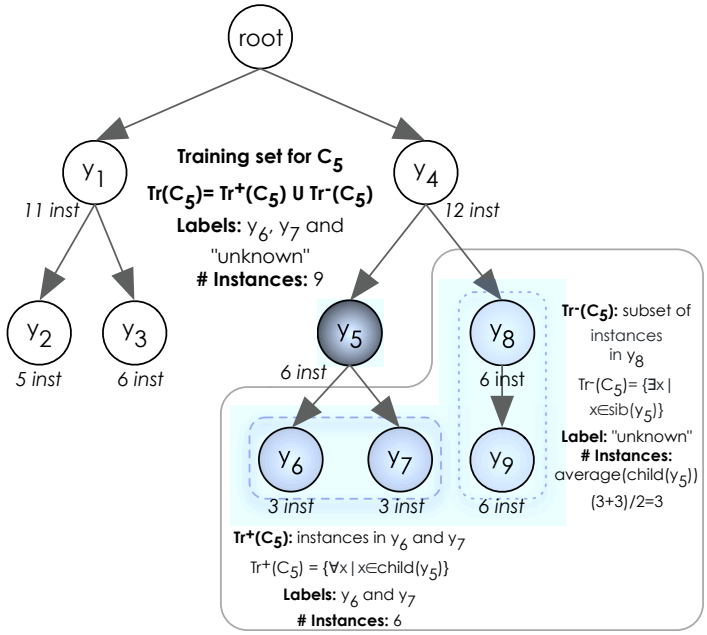


Figure 4.1: Example of the selection of the training set for the classifier (C_5) in node y_5 . The positive set ($Tr^{+}(C_5)$) consists of the instances where $child(y_5) = \{y_6, y_7\} = 1$, labeled with the corresponding $child(y_5)$ label. The negative set ($Tr^{-}(C_5)$) consists of the instances in $sib(y_5) = \{y_8\}$, this set will be labeled as "*unknown*". This set is under-sampled to make the number of instances proportional to the average of the training examples for $child(y_5)$.

The negative training set ($Tr^{-}(C_i)$) consists of the instances in the siblings of y_i ($sib(y_i)$) or, in the case that y_i has no siblings, the uncles ($sib(pa(y_i))$). The siblings include all the children nodes of the parents of y_i ($pa(y_i)$) except y_i. This set will be labeled as "*unknown*". This set is under-sampled to create a balanced training set, to obtain the highest balanced accuracy (Qiong Wei, Roland L. Dunbrack, 2013). The number of under-sampled instances is proportional to the mean of the training examples for each $e \in child(y_i)$. In this set the labels of the associated parent attributes have a zero value, the

idea is to reduce the probability of an instance which parent is predicted as zero. Figure 4.1 depicts an example of the training process.

4.2 Merging Rule

The rule that merges the predictions of each local classifier into one score considers the level in the hierarchy of the node to determine the weight that this node will have in the overall score. Misclassifications at the upper hierarchy levels (which correspond to more generic concepts) are more expensive than those at the lower levels (which correspond to more specific concepts). The intuition behind the selection of the weights is that, in the upper levels there are more examples and the classes are more different, then there is more information to discriminate the classes. Also, an error in the upper levels will cause a wrong path from the beginning, if it is in lower levels just part of the path will be wrong.

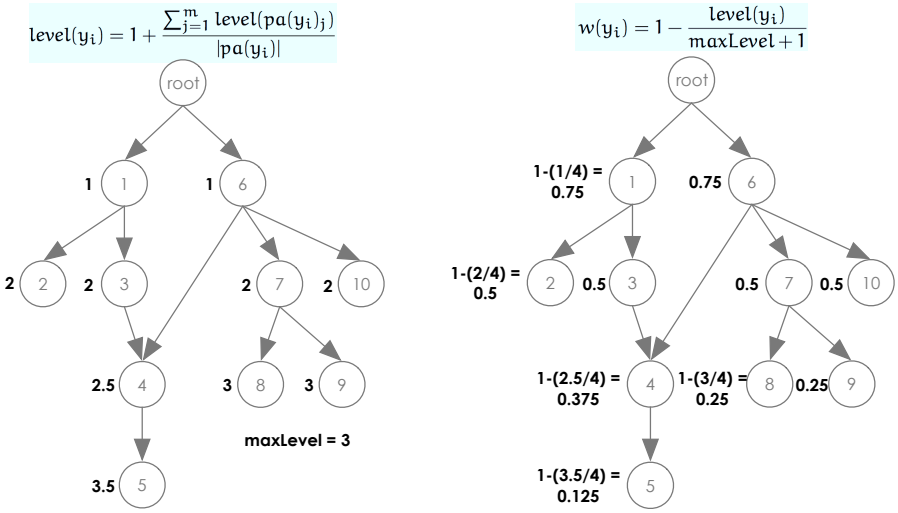
To achieve this task, the weight of a node ($w(y_i)$) is defined by Equation 4.2, where $level(y_i)$ is the level at which the node y_i is placed in the hierarchy (Equation 4.1). For a tree structure it is simply the level of its parent plus one (to avoid the last level to have weight equal to zero), and for DAG structures it is computed as the mean of the levels of the m parents ($pa(y_i)$) of the node (y_i) plus one. Finally, $maxLevel$ is the length of the longest path in the hierarchy (see Figure 4.2). This way of computing the weight of each node assures that the weights are well distributed along the hierarchy; so that the weights of the lower levels do not tend rapidly to zero (Bi and Kwok, 2011), or decrease very slow (Vens et al., 2008).

$$level(y_i) = 1 + \frac{\sum_{j=1}^{m} level(pa(y_i)_j)}{|pa(y_i)|} \tag{4.1}$$

$$w(y_i) = 1 - \frac{level(y_i)}{maxLevel + 1} \tag{4.2}$$

Equation 4.3 describes the merging rule which is the sum of the logarithms of the probabilities on the nodes along the path; where p is the number of nodes in the path, y_i is the i^{th} node in the path and $P(y_i = 1|x_e, pa(y_i))$ is the probability of the node y_i to be predicted as true by the local classifier. Taking the sum of logarithms is used to ensure numerical stability, in other words minimize approximation errors, when computing the probability for long paths. Figure 4.4 depicts the classification procedure.

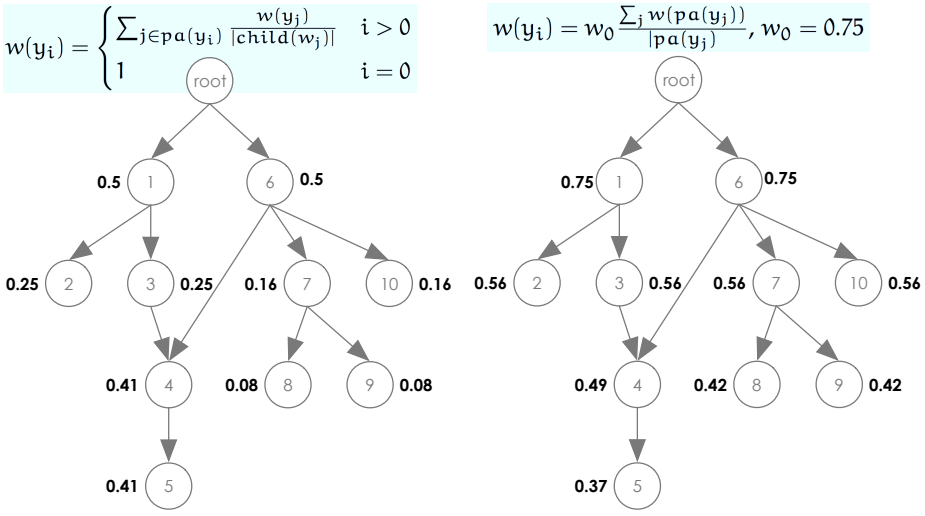
$$score = \sum_{i=1}^{p} w(y_i) * \log(P(y_i|x_e, pa(y_i))) \tag{4.3}$$



(a) Fetch the level of each node and the maximal depth of the hierarchy

(b) Compute the weight for each node using the information obtained in (a)

Figure 4.2: Example of the computation of the weight of the nodes of CPE.



(a) Bi and Kwok (2011) approach

(b) Vens et al. (2008) approach

Figure 4.3: Example of the computation of the weight of the nodes using two different approaches.

To simplify the computation, this scheme assumes independence between the labels, although in an indirect way the dependencies with the parent nodes are considered by incorporating them as additional attributes. As in bayesian chain classifiers, this scheme looks for a balance between classification accuracy and computational complexity.

For DAG structures there might be numerous paths from the root to one

leaf node. In that case, all the paths that end in that leaf node are returned.

A detailed representation of the classification phase of the algorithm is given in Algorithm 4.1. In the algorithm the confidences of each node are obtained by calling the local classification process described in Algorithm 4.2 for MNLP and in Algorithm 4.3 for NMLNP, then all the possible paths are obtained and stored in $allPaths$. Finally, a score is associated with each path and the better is returned as final prediction.

Algorithm 4.1 CPE. Algorithm for classifying a new instance.

Require: x (test example), H (hierarchical structure)
Ensure: $path$, a set of classes that describes a path from the root to a leaf node

```
confidences = LC(H.root)
allPaths ←get all the paths from the root to a leaf node in the hierarchy H
maxScore ← −∞
path ← null
for all p_i ∈ allPaths do
    score ← compute score for p_i using equation Equation (4.3)
    if score > maxScore then
        maxScore = score
        path = p_i
    end if
end for
```

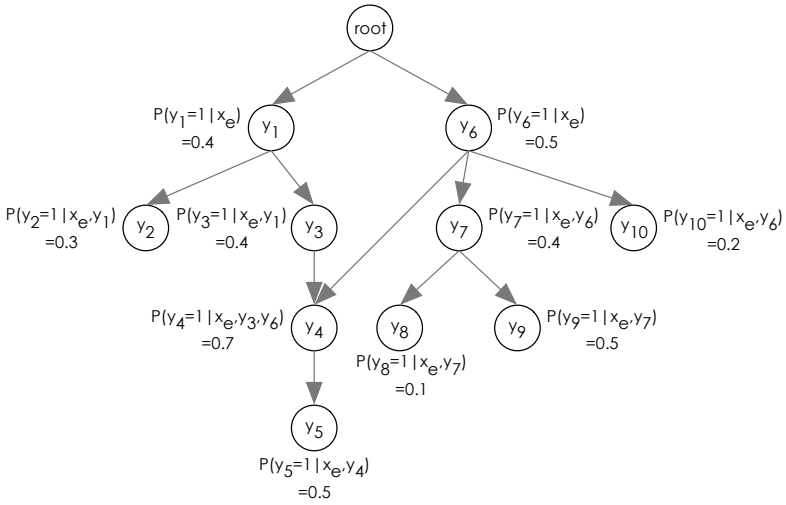
4.3 Classification for MLNP

The classification phase consists in calculating, for each new instance with feature vector X_e, the probability of a label y_i to occur given the feature vector and the prediction of the parents of each label $P(y_i = 1|X_e, pa(y_i))$.

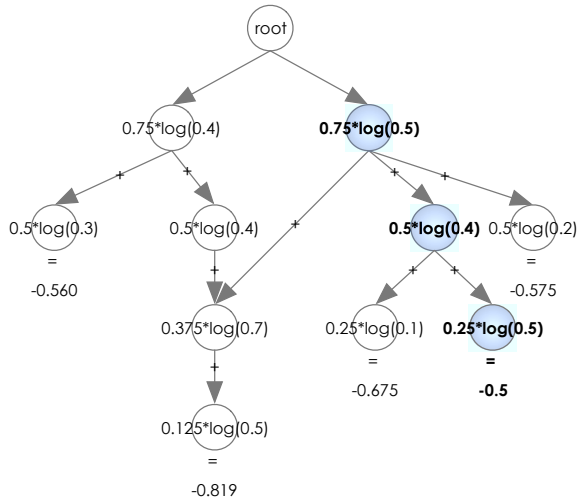
When the structure of the dataset is a DAG it is possible that we obtain more than one prediction for one class, then the associated prediction is the average of all the predictions for that class, in other words the average of the prediction of all the parents for that class, this is because the path to a node includes all its ancestors.

Algorithm 4.2 gives an overview of the local classification process for MLNP.

After computing a probability for each node, the predictions are merged using the rule explained in Section 4.2 to obtain a score for each path.



(a) Each node has an associated probability $P(y_i|x_e, pa(_i))$



(b) The probability is used to compute the overall score of the path, the path with the best score is returned as final prediction.

Figure 4.4: Example of the application of the merging rule. Figure 4.2 shows the w_j of the nodes. The best path is marked by the bold, grey nodes in (b).

Algorithm 4.2 LC. Algorithm that describes the local classification process for the MLNP version.

Require: x_i (test example), y_i (root node of the hierarchical structure), C (set of classifiers, one for each non leaf node)

Ensure: $confidences$ (an array with the confidences of each node)

```
1: if y_i is not a leaf and has not been visited before then
2:   if all pa(y_i) have been visited then
3:     mark y_i visited
4:     for all y_j ∈ child(y_i) do
5:       confidences[j] = P(y_j = 1|x_j, y_i)/|pa(y_j)|  {use C_i to compute the
         probabilities for the children}
6:       LC(y_j)
7:     end for
8:   end if
9: end if
```

4.4 Classification for NMLNP

Sometimes the information available is not sufficient to estimate the class of an instance at the lower levels in the hierarchy, so it could be better to truncate the predicted path at some previous level, this is known as non-mandatory leaf node prediction (NMLNP). The set of labels of the instances contained in this kind of datasets does not necessarily contain a leaf node. We introduce a pruning phase to obtain NMLNPs. We consider three decisions that need to be considered for pruning: direction, time and condition.

Pruning time

Determines when to perform the pruning stage:

1. Prune & Choose. Prunes the hierarchy before computing the score to choose the best path.

2. Choose & Prune. Prunes the path that obtained the best score in the classification phase.

Pruning direction

Determines the way the hierarchy is traversed to prune:

1. Top-Down. The hierarchy is traversed starting from the root node. When the pruning condition is met in one node, the traversing is stopped and the descendants of the node are pruned.

2. Bottom-Up. The hierarchy is traversed starting from the leaf nodes. When the pruning condition is met in one node, the traversing is stopped and the node and its descendants are pruned.

Pruning condition

Establishes the condition to prune a node. In this thesis we considered the following options:

1. Sum of children probabilities (SUM). Prunes if the sum of the probabilities of the children is less than the probability of the "*unknown*" label.

2. Most probable child (BEST). Prunes if the probability of the most probable child is less than the probability of the "*unknown*" label.

3. Information Gain (IG). Prunes if there is no information gain when including the child in the prediction.

We compared the different pruning decisions experimentally (Appendix E). The better approach for the task was to *Prune & Choose* in a *Top-Down* fashion, using the most probable child (*BEST*) as the condition to prune a node. We think this is because *Prune & Choose* allow our method to choose from the nodes that really have probabilities to appear in the real label set and avoid that the labels/nodes that does not have probability and thus decrease the score of a correct path that does not reach a leaf node; regarding to *Top-Down* decision it privileges to cut the first label that does not provide information to the prediction instead of deleting the last one as in *Bottom-Up*, finally the pruning condition *BEST* is more conservative than *SUM* because it is needed the only one node has the sufficient probability to surpass the unknown label, in the *SUM* approach even if all the probabilities of the nodes are not very high the sum could surpass the unknown label, *IG* proved to be the worst because it is rarely fulfilled.

Algorithm 4.3 gives an overview of the local classification process for NMLNP adding the pruning step.

The root of some hierarchies has just one child, node "1" in Figure 4.5. During the pruning experimentation we noticed that one pruning combination returned as prediction just node "1", which does not contribute to the discrimination of the classes (since all the instances belong to that node); the interesting thing is that all the evaluation measures explained in Section 3.2 assigned the best scores to that conservative approach and the rest pruning approaches did not had a clear difference in any evaluation measure. That is why we propose a new evaluation measure that avoids that kind of bias by considering the number of children that a node has.

Algorithm 4.3 LC. Algorithm that describes the local classification process for the NMLNP version.

Require: x_i (test example), y_i (root node of the hierarchical structure), C (set of classifiers, one for each non leaf node)

Ensure: confidences (an array with the confidences of each node)

```
1: if y_i is not a leaf and has not been visited before then
2:   if all pa(y_i) have been visited then
3:     mark y_i visited
4:     maxConfidence ← 0
5:     totalConfidence ← 0
6:     bestChild ← null
7:     for all y_j ∈ child(y_i) do
8:       confidences[j] = P(y_j = 1|x_j, y_i)/|pa(y_j)|  {use C_i to compute the
         probability }
9:       totalConfidence ← totalConfidence + confidences[j]
10:      if confidences[j] > maxConfidence then
11:        maxConfidence ← confidences[j]
12:        bestChild ← j
13:      end if
14:    end for
15:    unknown ← 1 − totalConfidences
16:    if maxConfidence > unknown then
17:      for all y_j ∈ child(y_i) do
18:        LC(y_j)
19:      end for
20:    end if
21:  end if
22: end if
```

4.5 Gain-Loose Balance

In this thesis we propose a new evaluation measure for hierarchical classifiers that avoids conservative predictions when using NMLNP. Gain-Loose Balance (GLB) determine the rewards and penalties using the number of siblings of the node and the depth of the node in the hierarchy.

Based on the notion that discriminating few categories is much easier than discriminating many of them, a correctly classified node with few siblings has a minor impact on the rewards than one with many. On the other hand, a misclassified node with few siblings has a mayor impact on the penalty than one with many.

A correctly classified node that belongs to a deep level in the hierarchy has

more impact on the rewards than one in shallow levels, because reaching the most specific node of the real labels is the goal of the prediction. In contrast, a misclassified node in a deep level of the hierarchy has less impact in the penalty than one in shallow levels, because an error in the last levels of the hierarchy will produce a prediction more similar to the real set of labels.

Equation (4.4) describes the GLB measure, where n_p is the number of correct classified labels, n_{fp} is the number of false positive errors, n_{fn} is the number of false negative errors, n_t is the number of true labels, N represents the number of siblings plus one (the node that is being evaluated), and w_i is the weight of the node (see Equation (4.2)). The first term represent the gains and the second the loses, there are two kinds of losses: false positive and false negative which are represented by the summations.

$$GLB = \frac{\sum_{i=0}^{n_p}(1-\frac{1}{N})(1-w_i)}{\sum_{i=0}^{n_t}(1-\frac{1}{N})(1-w_i)} - \left(\sum_{i=0}^{n_{fp}}\frac{1}{N}w_i + \sum_{i=0}^{n_{fn}}\frac{1}{N}w_i\right) \tag{4.4}$$

Gain-Loose Balance ranges from 1 (when the predicted path is equal to the real path) to $-\frac{maxL}{2}$ (see Equation (4.5)), where $maxL$ is the maximum number of levels in the hierarchy (see Equation (4.1)). The worst case scenario, when the loss is bigger (the second term in Equation (4.4)), is where not one label is correctly classified and an incorrect path down to the deepest leaf node has been predicted, if each node of this path has just one sibling the error will reach the maximum value, because an error in a node with less siblings is more expensive, then N was set to $N = 2$.

$$\begin{aligned}
min &= -2\sum_{i=1}^{maxL}\frac{1}{N}w_i \qquad (4.5)\\
&= -2\sum_{i=1}^{maxL}\frac{1}{2}\left(1-\frac{i}{maxL+1}\right)\\
&= -2\left(\sum_{i=1}^{maxL}\frac{1}{2}-\frac{1}{2(maxL+1)}\sum_{i=1}^{maxL}i\right)\\
&= -2\left(\frac{1}{2}\sum_{i=1}^{maxL}1-\frac{1}{2(maxL+1)}\left(\frac{maxL(maxL+1)}{2}\right)\right)\\
&= -2\left(\frac{maxL}{2}-\frac{maxL}{4}\right)\\
&= -maxL+\frac{maxL}{2}\\
&= -\frac{maxL}{2}
\end{aligned}$$

As we know the maximum and minimum values of the GLB measure we transformed it into a (0, 1) range maintaining the ratio. Then Normalized GLB

is described by $NGLB = f(GLB)$, $f(GLB)$ is defined in Equation (4.6), where $max = 1$ and min is defined in Equation (4.5).

$$f(GLB) = \frac{(GLB - min)}{max - min} \tag{4.6}$$

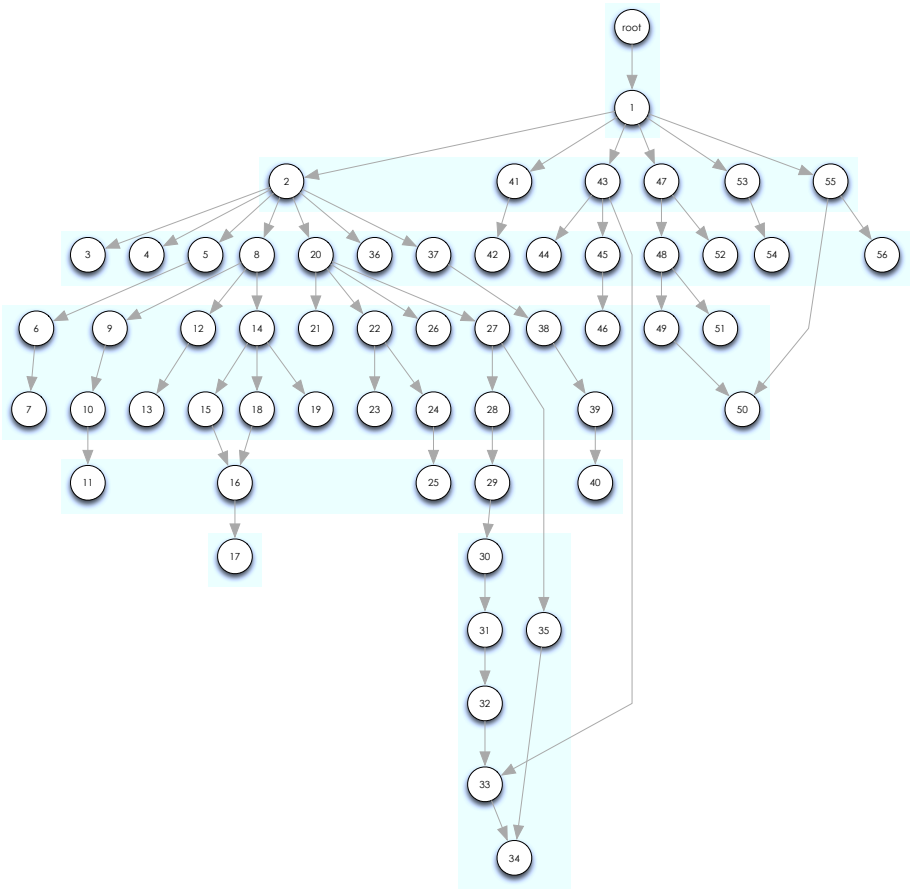


Figure 4.5: Example of the hierarchy cellcycle_GO where the pruning conditions were tested.

4.6 Summary

This chapter has described the proposed HMC method: Chained Path Evaluation (CPE). Our method computes the probability of each label using a local classifier in each non leaf node, unlike previous approaches, we proposed that this predictions were linked with the prediction of the parent node to take into account the relations of the nodes. A score for each path from the root to a leaf node is computed considering the level of the nodes in the path with a novel weighting scheme; finally the best path is selected and returned as prediction. If we need a NMLNP we perform a pruning phase in a top-down

fashion deleting the nodes where the probability of *unknown* is greater than the probability of the most probable node.

Lastly we propose a new evaluation metric that try to avoid the bias towards conservative predictions.

Chapter 5

Experiments and Results

We carried out several experiments on a number of datasets in the field of functional genomics.

First we present the experimental setup, that contains the description and topology of the datasets. Next we present a set of experiments to evaluate several aspects of our method.

5.1 Experimental Setup

This section outlines the framework for the experiments.

5.1.1 Datasets

Eighteen datasets were used in the tests, these datasets are from the field of functional genomics[1]. They were selected because of the relevance of the field and because they perfectly fit the hierarchical approach we are dealing with. Another reason is that they are available online, and accessible for free.

The different data sets describe different aspects of the genes in the yeast genome. They include five types of bioinformatic data: sequence statistics, phenotype and expression. Below, we describe each data set:

pheno contains phenotype data, which represents the growth or lack of growth of knock-out mutants that are missing the gene in question. The gene is removed or disabled and the resulting organism is grown with a variety of media to determine what the modified organism might be sensitive or resistant to. The attributes are discrete, and the data set is sparse, since not all knock-outs have been grown under all conditions.

seq records sequence statistics that depend on the amino acid sequence of the protein for which the gene codes. These include amino acid ratios, sequence length, molecular weight and hydrophobicity. Attributes are

[1] http://dtai.cs.kuleuven.be/clus/hmcdatasets/

mostly real valued, with a few exceptions (like chromosome number or strand) being discrete.

other The use of microarrays to record the expression of genes is popular in biology and bioinformatics. Microarray chips provide the means to test the expression levels of genes across an entire genome in a single experiment. Many expression data sets exist for yeast, and several of these were used. Attributes for these data sets are real valued, representing fold changes in expression levels.

There are two versions of most of the datasets. The input attributes are identical in both versions, but the classes are taken from two different classification schemes. In the first version, they are from FunCat (Ruepp et al., 2004), a scheme for classifying the functions of gene products developed by MIPS (Munich Information Center for Protein Sequences). FunCat is a tree-structured class hierarchy. In the second version of the data sets, the genes are annotated with terms from the Gene Ontology (GO) (Ashburner et al., 2000), which forms a directed acyclic graph instead of a tree: each term can have multiple parents.

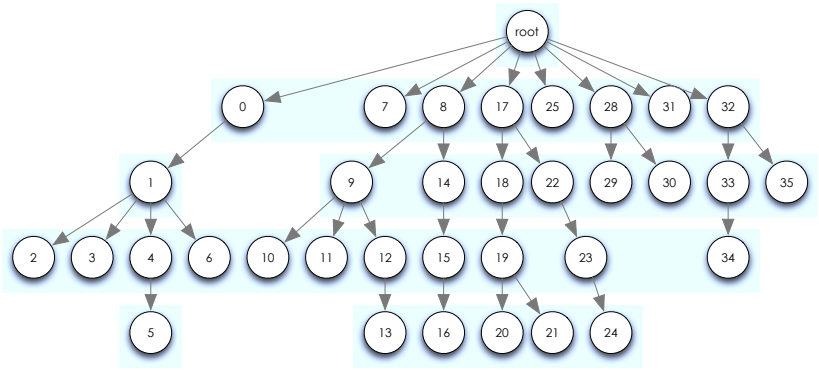


Figure 5.1: Structure of cellcycle_FUN dataset for MLNP

Since the labels of the instances in the datasets included more than one path, and this paths were NMLNP, the labels have been trimmed to get instances with no more than one path in the hierarchy from the root to a leaf node (if one node has two parents both are part of the path). We selected the first path that appeared in each instance.

For the set of experiments regarding MLNP the labels in the datasets were trimmed to get paths that reached one leaf node. Only the leaf nodes with enough instances to train were considered, more than 70 for tree hierarchies and more than 50 for DAG hierarchies to avoid overfitting (little generalization, difficult to apply to new data); thus having a hierarchical tree like structure with up to four levels of increasing specificity and a DAG structure of maximum 11 levels. The datasets are described in Table 5.1.

Table 5.1: Description of the datasets for the MLNP experiments. |L| = Number of Labels, LC = Label Cardinality (average number of labels relevant to each instance), A = Number of Attributes, N = Number of Instances and D = Maximum Depth

(a) Tree Datasets

Dataset	|L|	LC	A	N	D
cellcylcle_FUN	36	2.47	78	2339	4
church_FUN	36	2.45	28	2340	4
derisi_FUN	37	2.48	64	2381	4
eisen_FUN	25	2.26	80	1681	3
expr_FUN	36	2.46	552	2346	4
gasch1_FUN	36	2.47	174	2356	4
gasch2_FUN	36	2.46	53	2356	4
pheno_FUN	17	1.94	70	1162	3
seq_FUN	39	2.43	479	2466	4
spo_FUN	36	2.47	81	2302	4

(b) DAG datasets

Dataset	|L|	LC	A	N	D
cellcycle_GO	53	4.28	78	1708	11
church_GO	53	4.28	28	1711	11
derisi_GO	54	4.46	64	1746	11
expr_GO	53	4.34	552	1720	11
gasch1_GO	53	4.33	174	1716	11
gasch2_GO	53	4.37	53	1720	11
seq_GO	52	4.26	479	1711	11
spo_GO	53	4.31	81	1685	11

For the set of experiments for NMLNP the instances in the datasets were trimmed to get paths that not necessarily reached a leaf node. Only the nodes with more than 50 instances to train were considered. The datasets are described in Table 5.2. Since the pruning is not so strict these datasets have more instances and labels.

Table 5.2: Description of the datasets for the NMLNP experiments. |L| = Number of Labels, LC = Label Cardinality (average number of labels relevant to each instance), A = Number of Attributes, N=Number of Instances and D = Maximum Depth

(a) Tree datasets

Dataset	|L|	LC	A	N	D
cellcylcle_FUN	49	2.45	78	3602	4
church_FUN	49	2.45	28	3603	4
derisi_FUN	49	2.44	64	3675	4
eisen_FUN	35	2.26	80	2335	4
expr_FUN	49	2.44	552	3624	4
gasch1_FUN	49	2.45	174	3611	4
gasch2_FUN	49	2.48	53	3624	4
pheno_FUN	22	1.96	70	1462	3
seq_FUN	51	2.40	479	3765	4
spo_FUN	49	2.45	81	3553	4

(b) DAG datasets

Dataset	|L|	LC	A	N	D
cellcycle_GO	56	3.38	78	3516	11
church_GO	56	3.37	28	3515	11
derisi_GO	57	3.41	64	3485	11
expr_GO	56	3.39	552	3537	11
gasch1_GO	56	3.38	174	3524	11
gasch2_GO	56	3.44	53	3537	11
seq_GO	59	3.45	479	3659	11
spo_GO	56	3.39	81	3466	11

5.1.2 Data Topology

To discover the topology of the datasets we tested the capacity of dimensionality reduction of the datasets to approximate the linearity or non-linearity of the problems. This is because the dependencies in many complex systems have been found to be better approximated by relationships where every time you double one quantity the other one is multiplied by a number which is not two (non-linear relationships) than by linear relationships. A non-linear relation is a more general form of relationship and for this reason it should be a better approximation for a complex system (Bar-Yam, 2011). However, there are cases where complex systems are represented by linear relationships and vice versa.

We used two different methods to find the intrinsic dimension. The intrinsic dimension of a problem is the number of independent variables that explain satisfactorily that problem. A problem which is in appearance high-dimensional, and thus complex, can actually be governed by a few simple variables (Campbell, 1988; Carreira-Perpiñan, 1997).

The first method is a linear approach called Principal Component Analysis (PCA) (Jolliffe, 1986). The method generates a new set of variables, called principal components. Each principal component is a linear combination of the original variables. The full set of principal components is as large as the original set of variables. But it is commonplace for the sum of the variances of the first few principal components to exceed 80% of the total variance of the original data, this is the configuration we used.

The second method is a non linear approach, IsoMap (Tenenbaum et al., 2000). It construct a complete weighted graph with all the attributes, then prunes the graph by using k nearest neighbors, next it uses Floyd's algorithm to compute the shortest paths and reconstruct the graph, finally applies MDS. IsoMap was tested using from 3 to 12 neighbors to find the intrinsic dimensionality that best suited the data. The number of neighbors (k) that best explained the data is reported along with the dimensionality reduction for each method.

For both methods the intrinsic dimensionality is reported as well as the percentage of the data explained by that dimensionality.

Table 5.3: Dimensionality reduction for tree structured datasets. The percentage of data explained by each method is reported after the number of reduced attributes. A = Number of Attributes or Explicit Dimensionality

Dataset	A	Linear Intrinsic Dimensionality [Abs (%)]	Non-Linear Intrinsic Dimensionality [Abs (%)]	Linearity
cellcycle_FUN	78	40 (89.84%)	4 (60.12%) k=11	Non-linear
church_FUN	28	1 (85.12%)	2 (72.06%) k=10	Linear
derisi_FUN	64	2 (87.55%)	4 (76.75%) k=10	Linear
eisen_FUN	80	27 (89.83%)	3 (48.06%) k=7	Non-linear
expr_FUN	552	6 (88.29%)	4 (71.89%) k=4	Unclear
gasch1_FUN	174	45 (89.99%)	4 (55.47%) k=9	Non-linear
gasch2_FUN	53	21 (89.62%)	4 (59.04%) k=4	Non-linear
pheno_FUN	70	2 (87.18%)	4 (76.50%) k=4	Linear
seq_FUN	479	1 (99.99%)	4 (27.74%) k=8	Linear
spo_FUN	81	6 (89.31%)	3 (80.58%) k=12	Unclear

From Table 5.3, it can be inferred that the datasets church_FUN and seq_FUN are two datasets that have a lot of irrelevant or redundant information. derisi_FUN and pheno_FUN are linear problems (most probably non complex problems). expr_FUN and spo_FUN are problems with unclear linearity. While cellcycle_FUN, eisen_FUN, gasch1_FUN and gasch2_FUN are non-linear problems due to the large intrinsic dimensionality and thus most probably the most complex of the group.

Table 5.4: Dimensionality reduction for DAG structured datasets. The percentage of data explained by each method is reported after the number of reduced attributes (%). A = Number of Attributes or Explicit Dimensionality

Dataset	A	Linear Intrinsic Dimensionality [Abs (%)]	Non-Linear Intrinsic Dimensionality [Abs (%)]	Linearity
cellcycle_GO	78	40 (89.56%)	6 (75.05%) k=11	Non-linear
church_GO	28	1 (86.79%)	2 (73.08%) k=9	Linear
derisi_GO	64	2 (88.52%)	4 (69.33%) k=3	Linear
expr_GO	552	6 (89.32%)	3 (60.95%) k=3	Unclear
gasch1_GO	174	39 (89.92%)	3 (47.52%) k=9	Non-linear
gasch2_GO	53	20 (89.35%)	6 (74.80%) k=5	Non-linear
seq_GO	479	1 (99.99%)	4 (28.77%) k=11	Linear
spo_GO	81	6 (89.69%)	3 (80.26%) k=12	Unclear

From the results it for DAG structured datasets, in Table 5.3, can be inferred that the datasets church_GO, seq_GO and derisi_GO are linear problems. While cellcycle_GO, gasch1_GO and gasch2_GO are non-linear problems with the probability in complexity that it entail.

5.2 Experiments

We performed a set of experiments:

- to determine the best base classifier for the datasets,
- to compare the effectiveness of the proposed weighting scheme,
- to compare the MLNP version of our algorithm against other MLNP methods,
- to determine the effect of depth over the classification performance,
- to contrast the NMLNP version of our algorithm against the MLNP version,

- to compare the NMLNP version of our algorithm against other NMLNP methods and
- to estimate training and testing time.

5.2.1 Base Classifier

An experiment was designed to select the base classifier that best suits the datasets used to test our method and select the base classifier for the rest of the experiments. The experiment was performed over the 18 datasets using a ten-cross-fold validation scheme. We compared the performance of our method with six different hierarchical measures (see Section 3.2), using as base classifier five common methods selected from the techniques described in Subsection 2.1.1, the base classifiers were set with the default parameters of Weka.

1. Decision Tables (DT). Accuracy was the measure used to evaluate the attribute combinations which were selected by *Best First* method.
2. C4.5. The confidence factor used for pruning was 0.25. The minimum number of instances per leaf was set to 2.
3. Support Vector Machines (SVMs). The kernel used was a PolyKernel ($k(x,y) = \langle x,y\rangle^p$ or $k(x,y) = (\langle x,y\rangle + 1)^p$)
4. Naïve Bayes (NB). Numeric estimator precision values are chosen based on analysis of the training data.
5. Random Forest (RF). Generated 10 trees, and selected $\log_2(A) + 1$ attributes, where A is the number of attributes.

The results are depicted in Table 5.5. The results represented as the mean over all the datasets. The complete set of results for each dataset is in Appendix A.

Table 5.5: Performance (mean [std]) of the different base classifiers along the evaluation metrics. The best results are marked with bold letter.

Evaluation Measures	DT	C4.5	SVM	NB	RF
Accuracy	19.5 [11.73]	22.03 [6.6]	19.42 [10.7]	26.09 [9.08]	**30.69 [9.41]**
Exact Match	8.34 [2.38]	7.1 [3.32]	11.42 [3.64]	17.5[5.99]	**20.26 [5.67]**
Hamming Accuracy	90.76 [1.91]	90.55 [1.94]	90.31 [1.75]	89.47 [2.32]	**91.07 [1.98]**
F1-macro D	25.68 [17.68]	30.08 [11.94]	24.52 [15.61]	31.48 [12.39]	**36.67 [12.69]**
F1-macro L	3.91 [2.19]	2.86 [0.6]	3.77 [1.75]	13.45 [5.39]	**14.02 [4.63]**
H-loss	1.847 [0.05]	1.871 [0.08]	1.789 [0.06]	1.675 [0.12]	**1.611 [0.11]**

Using Random Forest as base classifier improves the classification performance of our method and even though the standard deviation does intersect when the results are averaged by datasets it does not intersect in most of the cases when the datasets are evaluated separately (see Appendix Chapter A).

Discussion

The reason of RF's better performance could be that RF is efficient for big datasets, can handle many variables and works well in the presence of redundant attributes which is the case of many of the datasets. This suggests that Random Forest is the best base classifier for this set of datasets.

We used RF for as base classifier for further experiments. We tested all the methods using the same base classifier (RF) because we are testing the performance of the approaches by giving them the same base conditions. Also we used it because most of the methods were reported to use RF as base classifier, also the selection of the base classifier was reported to depend on the application for each method.

5.2.2 Weighting Scheme

Two experiments were designed to verify that the proposed weighting scheme (see 4.2), that now on will be addressed by the name of Ramirez, Sucar, Morales (RSM), improves the classification performance.

The first experiment compares RSM against a non weighted scheme. The second compares RSM against two different weighting schemes.

5.2.2.1 RSM vs Non-weighted scheme

The first experiment verifies that RSM improves the performance of our method compared against a Non-Weighted (NW) scheme, where the weight for each node was set to one. The experiment was performed using a ten-cross-fold validation over the 18 datasets. The summary of the results is depicted in Table 5.6, we report the mean along the datasets. The mean of the folds for each dataset is reported in Appendix B.1.

Even though the difference is barely distinguished in the table because the results are averaged along the datasets, the proposed weighting scheme (RSM) outperforms the non-weighted scheme in five out of six measures in many cases with a significant difference (see Appendix B.1). The evaluation measure where the non-weighted scheme obtains better result is hamming accuracy.

Discussion

There is evidence that suggests that using the RSM weighting scheme improves the performance of our method compared with the non weighted scheme.

Table 5.6: RSM against a non weighted approach (mean [std]) along the evaluation metrics. The best results are marked in bold letter.

Evaluation Measures	RSM	NW
Accuracy	**30.36 [9.35]**	29.37 [9.82]
Exact Match	**19.95 [5.52]**	19.36 [5.6]
Hamming Accuracy	91.03 [2.01]	**91.13 [1.98]**
F1-macro D	**36.33 [12.7]**	35.15 [13.38]
F1-macro L	**13.8 [4.19]**	12.51 [4.23]
H-loss	**1.617 [0.11]**	1.629 [0.11]

This is because RSM adds extra information based in the position of the node in the hierarchy, since most general nodes are easier to distinguish because the instances are more different they have more weight in the overall score and the most specific where it is more difficult to distinguish the classes have less weight. NW lead RSM in hamming accuracy, as hamming accuracy deducts the errors, it means there are more false positives and false negatives in RSM predictions than in NW ones, this means that RSM is returning longer paths. In the rest of the evaluation measures RSM obtains best results, this means it has more true positives and true negatives than NW.

5.2.2.2 RSM vs other weighting schemes

The second experiment verifies that our method improves the classification compared against two other weighting schemes:

1. Vens et al. (2008) (V). This weighting scheme defines the weight as $w(y_i) = w_0 \left(\frac{\sum(pa(y_i))}{|pa(y_i)|} \right)$ and $w_0 = 0.75$.
2. Bi and Kwok (2012) (BK). This weighting scheme proposes that the weight $w(y_i)$ for the class c is $w(y_i) = \sum_{j \in pa(y_i)} \frac{w(j)}{|child(j)|}$ if it is a non-root class in the hierarchy, else $w(root) = 1$.

The summary of the results results is depicted on Table 5.7. The results are averaged over the datasets. The complete set of results is in Appendix B.2.

BK obtains usually the best result in the shallowest, tree hierarchies (although with a slight difference) while V usually obtains the best results in the deepest, DAG hierarchies (see Appendix B.2). This can be seen in accuracy, hamming accuracy and F1-macro D metrics. In the case of H-loss V is in most cases the winner, specially in deep hierarchies. In F1-macro L measure RSM is the best method for shallow hierarchies followed by V in deep hierarchies but the difference is not significative. Generally RSM gets the intermediate or better scores along the compared schemes, when it obtains the worst score, the difference with the intermediate approach is very slight.

Table 5.7: RSM against other weighting schemes (mean [std]) along the evaluation metrics. The best results are marked in bold letter.

(a) Tree structured datasets

Evaluation Measures	RSM	V	BK
Accuracy	23.66 [5.61]	23.55 [5.86]	**25.16 [5.31]**
Exact Match	**18.41 [4.7]**	18.4 [5.07]	17.99 [4.54]
Hamming-accuracy	90.11 [2.29]	**90.18 [2.23]**	90.15 [2.19]
F1-macro D	26.31 [6.11]	26.16 [6.32]	**28.78 [5.81]**
F1-macro L	**13.83 [3.83]**	13.25 [4.05]	12.05 [3.16]
H-loss	**1.632 [0.09]**	**1.632 [0.1]**	1.64 [0.09]

(b) DAG structured datasets

Evaluation Measures	RSM	V	BK
Accuracy	**38.73 [5.27]**	28.52 [5.39]	29.07 [4.44]
Exact Match	21.88 [6.16]	**22.15 [6.25]**	17.74 [5.25]
Hamming-accuracy	92.19 [0.55]	**92.20 [0.58]**	91.50 [0.46]
F1-macro D	48.86 [4.57]	**49.46 [4.71]**	48.16 [3.83]
F1-macro L	13.77 [4.88]	**14.38 [5.00]**	13.54 [3.66]
H-loss	1.599 [0.13]	**1.594 [0.13]**	1.68 [0.11]

Discussion

RSM approach does not obtain the best results when competing with V and BK. Since BK and V does not take into account the total number of levels they propagate the weights decreasing in certain rate. BK weights decrease very fast then with less levels it does not get to the lowest values, where the predictions of the lower nodes would not count in the global score, then BK works better in shallow hierarchies than in deep hierarchies. V weights decrease slow, the deep levels get relatively high values then the predictions in lower nodes really affect the overall score then obtaining better results in deeper levels. The advantage of RSM compared with BK is that it RSM weighting scheme is more stable over the two kinds of datasets because RSM takes into account the total number of levels and propagates the weights according to this information, this causes that all the nodes have a weight that assures they affect the final score in a way they does not control the score but affect enough to get good predictions. RSM is at least as good as V with no significant difference in any measure.

5.2.3 MLNP

The proposed method, Chained Path Evaluation (CPE), was evaluated experimentally with a number of tree and DAG structured hierarchies and compared with various hierarchical classification techniques. The results were obtained by a stratified 10-fold cross-validation. The six evaluation measures are reported in different tables for each dataset.

The base classifier used for all the methods was Random Forest because it was the classifier that obtained the best results in the first experiment (see Subsection 5.2.1).

Median and Interquartile Range (IQR) are reported. A Friedman test was carried out to find statistical significance in the results with a confidence level of 95% using Tukey's honestly significant difference criterion.

5.2.3.1 Comparison Against Other MLNP Methods

Tree Structured Datasets For tree structured hierarchies, we used ten hierarchical datasets and compared CPE against four HMC methods:

1. Top-Down LCPN (TD). Proposed by Koller and Sahami (1997), this method trains a LCPN and selects at each level the most probable node. Only the children of this node are explored to preserve the consistency of the prediction. The training set is obtained by a siblings policy.

2. Multidimensional Hierarchical Classifier (MHC). Proposed by Hernandez et al. (2013).

3. HIROM. Proposed by Bi and Kwok (2012).

Table 5.8: Accuracy (median [IQR]) comparing the performance of the MLNP version our method against other MLNP methods. The best results are marked in bold. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	**22.93 [3.21]**	20.00 [1.79]	19.52 [3.31]	2.99 [0.44]↓
church_FUN	11.68 [9.22]	10.58 [9.84]	**19.12 [2.67]**	3.80 [2.99]↓
derisi_FUN	18.19 [2.31]	12.95 [3.11]	**19.75 [3.20]**	2.94 [1.26]↓
eisen_FUN	**27.08 [11.31]**	25.30 [7.74]	22.75 [3.37]	1.78 [0.60]↓
expr_FUN	**30.24 [2.30]**	23.31 [6.22]	20.15 [3.00]↓	1.91 [1.69]↓
gasch1_FUN	**28.12 [5.42]**	20.72 [3.57]	19.54 [3.44]↓	1.70 [0.85]↓
gasch2_FUN	18.32 [10.87]	17.85 [11.60]	**19.78 [2.54]**	0.50 [4.68]↓
pheno_FUN	18.97 [2.08]	**23.56 [2.84]**↓	22.13 [1.91]	6.90 [3.45]↓
seq_FUN	28.70 [9.38]	**31.24 [2.63]**	17.44 [3.65]↓	2.70 [1.23]↓
spo_FUN	**19.92 [3.17]**	15.25 [2.62]↓	19.78 [3.56]	3.43 [1.45]↓

Table 5.9: Hamming accuracy (median [IQR]) comparing the performance of the MLNP version our method against other MLNP methods. The best results are marked in bold. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	**90.98 [0.39]**	88.75 [0.19]↓	90.36 [0.39]	84.82 [0.30]↓
church_FUN	**90.50 [1.13]**	85.43 [3.77]↓	90.19 [1.08]	84.75 [0.89]↓
derisi_FUN	**90.71 [0.33]**	87.16 [0.81]↓	90.54 [0.28]	85.52 [0.28]↓
eisen_FUN	**88.40 [1.81]**	86.80 [1.52]	87.49 [1.64]	79.95 [0.38]↓
expr_FUN	**91.28 [0.43]**	88.97 [0.91]↓	90.38 [0.42]	84.81 [0.25]↓
gasch1_FUN	**91.45 [0.78]**	88.68 [0.50]↓	90.34 [0.47]	84.87 [0.14]↓
gasch2_FUN	**90.81 [1.24]**	88.50 [1.51]	90.18 [0.93]	85.04 [0.63]↓
pheno_FUN	**84.13 [1.08]**	83.67 [0.71]	83.54 [0.87]	78.35 [0.46]↓
seq_FUN	**91.77 [0.72]**	91.18 [0.56]	90.76 [0.32]↓	86.31 [0.16]↓
spo_FUN	**90.64 [0.41]**	87.84 [0.32]↓	90.39 [0.33]	85.07 [0.22]↓

Table 5.10: Exact match (median [IQR]) comparing the performance of the MLNP version our method against other MLNP methods. The best results are marked in bold. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	**17.74 [5.13]**	16.92 [2.14]	9.40 [3.85]	2.99 [0.44]↓
church_FUN	**8.97 [9.40]**	8.76 [9.40]	9.40 [2.56]	3.42 [2.99]↓
derisi_FUN	**13.00 [3.36]**	10.50 [2.94]	9.45 [2.89]	2.94 [1.26]↓
eisen_FUN	18.69 [13.10]	**21.13 [9.52]**	10.42 [4.17]↓	1.78 [0.60]↓
expr_FUN	**24.20 [1.69]**	20.13 [5.51]	10.19 [2.54]↓	1.91 [1.69]↓
gasch1_FUN	**21.74 [4.68]**	17.27 [3.34]	9.36 [4.22]↓	1.70 [0.85]↓
gasch2_FUN	**13.35 [12.34]**	12.29 [11.91]	10.83 [2.93]	0.42 [4.68]↓
pheno_FUN	**15.09 [2.59]**	12.07 [2.67]	9.44 [1.72]	6.90 [3.45]↓
seq_FUN	23.28 [10.24]	**26.72 [2.33]**	8.70 [2.40]	2.64 [1.21]↓
spo_FUN	**14.35 [2.17]**	11.96 [3.48]	9.54 [3.86]↓	3.26 [1.74]↓

Table 5.11: F1-macro D (median [IQR]) comparing the performance of the MLNP version our method against other MLNP methods. The best results are marked in bold. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	**25.55 [1.95]**	21.81 [1.38]	24.98 [2.65]	2.99 [0.44]↓
church_FUN	14.03 [9.79]	11.69 [9.72]	**24.51 [4.62]**	3.87 [2.99]↓
derisi_FUN	21.18 [2.33]	14.33 [2.81]	**25.08 [3.34]**	2.94 [1.26]↓
eisen_FUN	**31.31 [11.71]**	27.35 [7.74]	28.57 [5.36]	1.78 [0.60]↓
expr_FUN	**32.97 [2.61]**	25.02 [6.65]	25.40 [3.52]↓	1.91 [1.69]↓
gasch1_FUN	**31.36 [5.47]**	22.26 [3.85]↓	24.82 [2.85]	1.70 [0.85]↓
gasch2_FUN	20.79 [10.11]	20.69 [11.45]	**24.95 [3.26]**	0.53 [4.68]↓
pheno_FUN	20.73 [1.89]	**29.27 [2.91]**	28.88 [2.59]↑	6.90 [3.45]
seq_FUN	31.65 [8.41]	**33.57 [2.52]**	21.98 [3.69]	2.74 [1.23]↓
spo_FUN	22.86 [3.63]	16.83 [2.26]	**25.30 [3.45]**	3.48 [1.39]↓

Table 5.12: F1-macro L (median [IQR]) comparing the performance of the MLNP version our method against other MLNP methods. The best results are marked in bold. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	13.74 [3.32]	**17.78 [3.18]**	2.09 [0.25]	0.49 [0.12]↓
church_FUN	6.47 [4.55]	**6.75 [4.45]**	2.06 [0.24]	0.75 [0.73]↓
derisi_FUN	**12.43 [2.97]**	10.47 [1.06]	2.04 [0.23]↓	0.47 [0.20]↓
eisen_FUN	**20.27 [3.37]**	17.51 [4.30]	3.33 [0.36]↓	0.45 [0.19]↓
expr_FUN	18.53 [2.54]	**20.12 [6.67]**	2.12 [0.24]	0.32 [0.27]↓
gasch1_FUN	**17.82 [2.67]**	16.11 [4.00]	2.08 [0.26]↓	0.28 [0.07]↓
gasch2_FUN	6.43 [7.97]	**10.05 [5.02]**	2.11 [0.23]	0.10 [0.76]↓
pheno_FUN	**10.98 [2.90]**	9.95 [2.06]	4.81 [0.31]↓	3.91 [1.51]↓
seq_FUN	15.85 [2.56]	**20.60 [5.05]**	1.79 [0.27]	0.49 [0.30]↓
spo_FUN	11.56 [1.58]	**12.61 [1.68]**	2.11 [0.27]↓	0.64 [0.41]↓

Table 5.13: H-loss (median [IQR]) comparing the performance of the MLNP version our method against other MLNP methods. The best results are marked in bold. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	**1.645 [0.1]**	1.662 [0.04]	1.812 [0.08]	1.94 [0.01]↓
church_FUN	1.821 [0.19]	1.825 [0.19]	**1.812 [0.05]**	1.923 [0.06]↓
derisi_FUN	**1.74 [0.07]**	1.79 [0.06]	1.811 [0.06]	1.941 [0.03]↓
eisen_FUN	1.626 [0.26]	**1.577 [0.19]**	1.792 [0.08]↓	1.964 [0.01]↓
expr_FUN	**1.516 [0.03]**	1.598 [0.11]	1.796 [0.05]↓	1.962 [0.03]↓
gasch1_FUN	**1.565 [0.09]**	1.655 [0.07]	1.813 [0.08]↓	1.966 [0.02]↓
gasch2_FUN	**1.733 [0.25]**	1.754 [0.24]	1.783 [0.06]	1.989 [0.09]↓
pheno_FUN	**1.698 [0.05]**	1.759 [0.05]	1.811 [0.03]	1.862 [0.07]↓
seq_FUN	1.534 [0.2]	**1.466 [0.05]**	1.826 [0.05]	1.945 [0.02]↓
spo_FUN	**1.713 [0.04]**	1.761 [0.07]	1.809 [0.08]↓	1.93 [0.03]↓

Discussion

In tree structured datasets CPE outperforms the other methods in most datasets along all the measures and in some cases with statistical significance (see Section D). CPE obtains in most of the datasets better results in accuracy then it has a higher rate of real labels from the predicted set. In hamming accuracy CPE obtains the best results of the bunch then having the lower error rate from the total number of labels. In exact match also obtains most of the best results which is good because it return more correct complete paths than the other methods. In the case of F1-macro D shares the best results with MHC which means they both get a good compromise between precision and recall along the instances, but MHC obtains really low rate regarding F1-macro L which means there are labels that are very bad predicted in MHC method. In F1-macro L CPE shares the best results with TD but TD has a lower rate in F1-macro D than CPE then even though CPE is not the best in F1-macro D and F1-macro L it obtains competitive result in both measures unlike other methods that obtains high results in one measure and low in the other.

DAG Structured Datasets For DAG structured datasets, we used eight hierarchical datasets and compared CPE against tree HMC methods:

1. Top-Down LCPN (TD). Proposed by Koller and Sahami (1997), this method trains a LCPN and selecting at each level the most probable node. Only the children of this node are explored to preserve the

consistency of the prediction. The training set is obtained by siblings policy.

2. Top-Down LCPN Corrected (TD-C). The only difference between this method and TD is that when a leaf node is reached, all the paths to that node are appended to the final prediction. TD returns a single path.

3. HIROM. Proposed by Bi and Kwok (2012), the variant for DAG structures.

Table 5.14: Accuracy (median [IQR]) in percentage comparing our method against other methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	TC-C	HIROM
cellcycle_GO	**36.95 [4.79]**	34.07 [4.88]	33.83 [4.52]	14.76 [0.90]↓
church_GO	**32.13 [3.59]**	30.82 [1.94]	30.62 [2.10]	16.49 [1.12]↓
derisi_GO	**33.15 [3.10]**	31.69 [2.27]	31.56 [2.11]	15.02 [0.96]↓
expr_GO	**42.36 [2.31]**	37.76 [4.79]	37.65 [4.67]↓	14.20 [0.32]↓
gasch1_GO	**42.43 [4.97]**	39.19 [2.62]	39.01 [2.65]	14.02 [1.51]↓
gasch2_GO	**39.83 [3.55]**	37.03 [2.75]	36.76 [3.29]↓	15.39 [1.11]↓
seq_GO	**48.87 [2.26]**	46.23 [4.57]	46.76 [4.57]	13.85 [1.17]↓
spo_GO	**34.06 [0.96]**	32.52 [2.51]	32.50 [2.73]↓	15.19 [1.50]↓

Table 5.15: Hamming Accuracy (median [IQR]) in percentage comparing our method against other methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	TD-C	HIROM
cellcycle_GO	**92.11 [0.98]**	90.02 [0.79]	89.65 [0.76]↓	85.83 [0.76]↓
church_GO	**91.37 [0.43]**	89.65 [0.38]	89.36 [0.43]↓	87.01 [1.51]↓
derisi_GO	**91.58 [0.34]**	89.65 [0.30]	89.35 [0.24]↓	85.99 [0.84]↓
expr_GO	**92.46 [0.56]**	90.56 [0.75]	90.30 [0.81]↓	85.90 [0.29]↓
gasch1_GO	**92.43 [0.76]**	90.82 [0.64]	90.52 [0.60]↓	85.98 [0.60]↓
gasch2_GO	**92.39 [0.59]**	90.52 [0.64]	90.00 [0.65]↓	86.26 [0.28]↓
seq_GO	**93.38 [1.14]**	91.76 [0.86]	91.68 [0.77]↓	84.67 [0.48]↓
spo_GO	**91.64 [0.32]**	89.94 [0.23]	89.57 [0.42]↓	86.10 [0.66]↓

Table 5.16: Exact Match (median [IQR]) in percentage comparing our method against other methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	TD-C	HIROM
cellcycle_GO	**19.30 [2.92]**	15.25 [6.94]	15.84 [6.43]	0.00 [0.00]↓
church_GO	**13.45 [4.68]**	12.28 [1.75]	12.28 [2.34]	0.58 [1.17]↓
derisi_GO	**15.43 [2.95]**	13.51 [2.21]	13.79 [2.29]	0.00 [0.00]↓
expr_GO	**27.33 [2.91]**	21.51 [5.81]↓	21.51 [6.40]	0.00 [0.00]↓
gasch1_GO	**27.11 [5.68]**	21.93 [2.91]	22.16 [3.95]	0.00 [0.00]↓
gasch2_GO	**22.67 [5.23]**	19.77 [4.07]↓	20.06 [4.65]	0.00 [0.00]↓
seq_GO	**33.63 [3.51]**	28.86 [5.85]↓	31.19 [5.85]	0.00 [0.00]↓
spo_GO	**16.57 [1.79]**	13.35 [1.19]↓	13.35 [1.70]	0.00 [0.00]↓

Table 5.17: F1-macro D (median [IQR]) in percentage comparing our method against other methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	TD-C	HIROM
cellcycle_GO	**47.42 [4.57]**	44.98 [4.02]	44.65 [3.74]	24.55 [1.38]↓
church_GO	**43.50 [3.01]**	41.88 [1.48]	41.60 [1.78]	27.30 [1.31]↓
derisi_GO	**43.88 [3.24]**	42.67 [2.28]	42.43 [2.33]	24.91 [1.39]↓
expr_GO	**51.67 [2.77]**	47.73 [3.98]	47.56 [3.90]↓	23.97 [0.44]↓
gasch1_GO	**51.71 [4.59]**	49.19 [1.67]	48.88 [1.71]	23.87 [1.78]↓
gasch2_GO	**49.74 [3.25]**	47.37 [2.45]	46.99 [2.82]↓	25.33 [1.30]↓
seq_GO	**57.90 [2.51]**	55.35 [3.92]	55.56 [3.68]	23.33 [1.54]↓
spo_GO	**44.74 [1.22]**	43.41 [3.06]	43.31 [3.38]	25.25 [1.93]↓

Table 5.18: F1-macro L (median [IQR]) in percentage comparing our method against other methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	TD-C	HIROM
cellcycle_GO	10.73 [1.68]	13.43 [3.00]	**14.68 [2.54]**↑	2.52 [0.19]
church_GO	8.84 [2.57]	10.51 [1.64]	**10.53 [1.64]**	2.71 [0.57]↓
derisi_GO	9.40 [2.42]	12.76 [2.17]	**13.26 [2.48]**↑	2.66 [0.27]
expr_GO	17.00 [3.56]	16.41 [5.75]	**17.01 [5.09]**	2.38 [0.18]↓
gasch1_GO	16.82 [5.07]	18.01 [3.19]	**18.10 [3.22]**	2.35 [0.36]↓
gasch2_GO	12.87 [3.40]	14.95 [2.79]	**15.16 [3.42]**	2.58 [0.42]
seq_GO	22.46 [3.70]	27.31 [2.36]	**31.61 [3.49]**↑	2.48 [0.28]
spo_GO	9.79 [1.79]	**11.89 [3.28]**	11.85 [3.18]	2.73 [0.49]

Table 5.19: H-loss (median [IQR]) comparing our method against other methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	TD	TD-C	HIROM
cellcycle_GO	**1.67 [0.08]**	1.693 [0.19]	1.736 [0.09]	3.044 [0.06]↓
church_GO	**1.787 [0.1]**	1.798 [0.06]	1.807 [0.05]	2.737 [0.39]↓
derisi_GO	**1.728 [0.03]**	1.751 [0.05]	1.765 [0.04]↓	3.014 [0.12]↓
expr_GO	**1.486 [0.08]**	1.596 [0.12]	1.619 [0.12]↓	3.111 [0.08]↓
gasch1_GO	**1.504 [0.13]**	1.597 [0.08]	1.62 [0.07]↓	3.056 [0.06]↓
gasch2_GO	**1.584 [0.1]**	1.642 [0.12]	1.651 [0.12]↓	2.927 [0.08]↓
seq_GO	**1.351 [0.06]**	1.411 [0.11]	1.414 [0.08]	3.234 [0.09]↓
spo_GO	**1.712 [0.05]**	1.766 [0.02]	1.786 [0.03]↓	2.973 [0.08]↓

Discussion

In DAG-structured hierarchies our method is superior to the other approaches, and the difference is statistically significant in practically all the measures and datasets (see Section D.2). The exception is F1-macro L measure, where the results are averaged by labels, that means that there are probably a few labels, which does not have many instances classified, where our method tend to obtain bad results.

There are different factors that could explain this difference for DAG structures. One could be that our method was developed considering this type of hierarchies, while TD was not. That is why we tested with TD-C version which proved no to be as good as TD, it even causes more errors. However, HIROM also considers DAG structures, and our method is superior.

Another possible explanation is that our approach is better for deeper hierarchies, as the tree datasets have a maximum depth of 3 or 4 levels, while the DAG datasets have a maximum depth of 11 levels. The weighting scheme per level in our method could be one of the reasons for this difference.

5.2.3.2 Comparison Against the Flat Approach

We compared our method against the flat approach (explained at 3.1.1). A non-paired, one-tail t-test was carried out to find statistical significance in the results with a confidence level of 95%.

Discussion

CPE is not a competitor for the Flat approach in shallow hierarchies because in almost every shallow dataset along most of the metrics, Flat approach obtains significantly better results. Exceptions are redundant datasets (eisen_FUN, expr_FUN,pheno_FUN and seq_FUN), where CPE obtain better scores (not

Table 5.20: Accuracy percentage (mean [std]) comparing our method against flat approach. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	CPE	Flat
cellcycle_FUN	22.76 [2.41]	**25.15 [2.25]**↑
church_FUN	14.24 [8.01]	**18.09 [6.39]**
derisi_FUN	18.58 [2.15]	**21.86 [1.54]**↑
eisen_FUN	**31.07 [9.28]**	28.80 [9.14]
expr_FUN	29.46 [2.66]	**29.64 [1.72]**
gasch1_FUN	28.39 [2.87]	**30.61 [2.82]**↑
gasch2_FUN	22.32 [8.68]	**26.15 [7.75]**
pheno_FUN	18.39 [3.02]	**21.44 [3.64]**↑
seq_FUN	**30.52 [6.68]**	29.32 [5.94]
spo_FUN	20.57 [1.94]	**23.24 [2.29]**

(b) DAG datasets

Dataset	CPE	Flat
cellcycle_GO	36.60 [2.68]	**36.92 [3.10]**
church_GO	**32.09 [1.94]**	31.16 [2.70]
derisi_GO	33.42 [1.76]	**34.22 [1.61]**
expr_GO	**42.80 [1.98]**	40.00 [2.46]
gasch1_GO	**42.03 [3.17]**	41.18 [2.39]
gasch2_GO	**39.53 [2.10]**	38.76 [1.84]
seq_GO	**48.99 [2.87]**	47.54 [2.79]
spo_GO	**34.45 [1.54]**	34.13 [1.64]

Table 5.21: Hamming Accuracy percentage (mean [std]) comparing our method against flat approach. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	CPE	FLAT
cellcycle_FUN	**91.03 [0.32]**	90.38 [0.29]↓
church_FUN	**90.51 [0.74]**	89.14 [1.48]↓
derisi_FUN	**90.64 [0.25]**	90.17 [0.33]↓
eisen_FUN	**88.79 [1.44]**	88.06 [1.49]
expr_FUN	**91.27 [0.41]**	90.94 [0.27]↓
gasch1_FUN	**91.27 [0.44]**	91.02 [0.38]
gasch2_FUN	**91.00 [1.26]**	90.40 [1.28]
pheno_FUN	**84.26 [0.63]**	83.31 [0.61]↓
seq_FUN	**91.98 [0.75]**	91.34 [0.75]↓
spo_FUN	**90.69 [0.29]**	90.14 [0.34]↓

(b) DAG datasets

Dataset	CPE	FLAT
cellcycle_GO	**91.98 [0.62]**	90.88 [0.57]↓
church_GO	**91.52 [0.35]**	89.47 [1.36]↓
derisi_GO	**91.57 [0.26]**	90.51 [0.42]↓
expr_GO	**92.56 [0.34]**	91.33 [0.38]↓
gasch1_GO	**92.51 [0.49]**	91.36 [0.59]↓
gasch2_GO	**92.31 [0.31]**	91.00 [0.40]↓
seq_GO	**93.33 [0.57]**	92.52 [0.51]↓
spo_GO	**91.71 [0.30]**	90.52 [0.27]↓

Table 5.22: Exact Match percentage (mean [std]) comparing our method against flat approach. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	CPE	FLAT
cellcycle_FUN	17.53 [2.61]	**18.30 [2.72]**
church_FUN	11.20 [7.00]	**12.48 [6.35]**
derisi_FUN	13.27 [2.19]	**14.87 [1.47]**↑
eisen_FUN	**23.32 [9.91]**	20.71 [9.74]
expr_FUN	**23.47 [2.31]**	20.75 [7.51]
gasch1_FUN	22.12 [2.72]	**24.34 [2.42]**↑
gasch2_FUN	17.49 [8.94]	**19.48 [8.16]**
pheno_FUN	**14.90 [3.47]**	13.26 [3.89]
seq_FUN	**25.26 [7.14]**	24.13 [6.44]
spo_FUN	14.73 [1.68]	**16.38 [2.19]**↑

(b) DAG datasets

Dataset	CPE	FLAT
cellcycle_GO	19.26 [2.19]	**20.90 [3.73]**
church_GO	13.79 [2.85]	**14.78 [3.07]**
derisi_GO	15.41 [1.83]	**17.18 [2.02]**↑
expr_GO	**27.33 [2.51]**	24.36 [2.87]↓
gasch1_GO	**26.28 [3.69]**	25.53 [2.81]
gasch2_GO	22.62 [2.56]	**22.79 [1.85]**
seq_GO	**33.31 [3.32]**	32.55 [3.13]
spo_GO	16.97 [2.20]	**17.27 [2.05]**

Table 5.23: F1-macro D percentage (mean [std]) comparing our method against flat approach. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	CPE	FLAT
cellcycle_FUN	25.40 [2.44]	**28.63 [2.11]**↑
church_FUN	15.77 [8.58]	**20.93 [6.51]**
derisi_FUN	21.27 [2.22]	**25.37 [1.66]**↑
eisen_FUN	**34.98 [9.04]**	32.89 [8.91]
expr_FUN	32.47 [2.86]	**33.12 [1.64]**
gasch1_FUN	31.53 [2.95]	**33.78 [3.01]**
gasch2_FUN	24.76 [8.58]	**29.54 [7.57]**
pheno_FUN	20.16 [2.83]	**25.56 [3.72]**↑
seq_FUN	**33.17 [6.44]**	31.96 [5.74]
spo_FUN	23.49 [2.22]	**26.73 [2.35]**↑

(b) DAG datasets

Dataset	CPE	FLAT
cellcycle_GO	**47.03 [2.70]**	46.77 [2.77]
church_GO	**43.24 [1.61]**	41.47 [2.51]↓
derisi_GO	44.25 [1.67]	**44.55 [1.52]**
expr_GO	**52.14 [1.72]**	49.51 [2.20]↓
gasch1_GO	**51.52 [2.80]**	50.60 [2.17]
gasch2_GO	**49.60 [1.83]**	48.43 [1.75]
seq_GO	**57.94 [2.60]**	56.27 [2.54]
spo_GO	**45.12 [1.26]**	44.45 [1.38]

Table 5.24: F1-macro L percentage (mean [std]) comparing our method against flat approach. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	CPE	FLAT
cellcycle_FUN	14.34 [2.50]	**14.52 [2.10]**
church_FUN	7.72 [6.22]	**9.53 [5.39]**
derisi_FUN	11.54 [1.76]	**12.95 [1.30]**↑
eisen_FUN	**19.59 [3.15]**	17.25 [3.73]
expr_FUN	**18.45 [2.19]**	18.31 [2.31]
gasch1_FUN	17.24 [1.85]	**19.76 [2.99]**↑
gasch2_FUN	8.74 [4.17]	**9.99 [3.16]**
pheno_FUN	11.14 [3.11]	**14.16 [3.06]**↑
seq_FUN	**17.21 [3.36]**	17.00 [2.66]
spo_FUN	11.97 [1.36]	**13.43 [2.60]**

(b) DAG datasets

Dataset	CPE	FLAT
cellcycle_GO	10.45 [1.21]	**16.12 [3.75]**↑
church_GO	8.72 [1.29]	**10.54 [2.13]**↑
derisi_GO	9.50 [1.26]	**14.33 [2.15]**↑
expr_GO	17.45 [1.90]	**18.24 [3.08]**
gasch1_GO	16.70 [2.34]	**18.48 [3.03]**
gasch2_GO	12.52 [1.90]	**15.82 [2.08]**↑
seq_GO	24.28 [4.25]	**32.88 [4.74]**↑
spo_GO	9.79 [1.24]	**12.71 [1.90]**↑

Table 5.25: H-loss (mean [std]) comparing our method against flat approach. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	CPE	FLAT
cellcycle_FUN	1.649 [0.05]	**1.634 [0.05]**
church_FUN	1.776 [0.14]	**1.750 [0.13]**
derisi_FUN	1.735 [0.04]	**1.703 [0.03]**↑
eisen_FUN	**1.534 [0.20]**	1.586 [0.19]
expr_FUN	**1.531 [0.05]**	1.485 [0.17]
gasch1_FUN	1.558 [0.05]	**1.513 [0.05]**↑
gasch2_FUN	1.650 [0.18]	**1.610 [0.16]**
pheno_FUN	**1.702 [0.07]**	1.735 [0.08]
seq_FUN	**1.495 [0.14]**	1.517 [0.13]
spo_FUN	1.705 [0.03]	**1.672 [0.04]**↑

(b) DAG dataset

Dataset	CPE	FLAT
cellcycle_GO	1.654 [0.05]	**1.638 [0.07]**
church_GO	**1.766 [0.06]**	1.783 [0.09]
derisi_GO	1.729 [0.03]	**1.706 [0.04]**
expr_GO	**1.483 [0.05]**	1.563 [0.06]↓
gasch1_GO	**1.509 [0.08]**	1.541 [0.07]
gasch2_GO	**1.584 [0.06]**	1.597 [0.04]
seq_GO	**1.361 [0.07]**	1.393 [0.07]
spo_GO	**1.704 [0.04]**	1.710 [0.03]

significant) in most metrics. An special case is eisen_FUN dataset which in addition to be redundant it is also a complex dataset, this is the one shallow dataset where CPE obtains significantly better results.

In the case of deep hierarchies in many cases CPE obtains better results than the Flat approach. As in shallow datasets it obtains significantly better results in redundant datasets.

Hamming Accuracy metric is one metric where in every dataset CPE obtains better significantly better results against the Flat approach, this metric is like an error ratio, this means that if CPE does not get the complete path it at least obtain a more similar path than the Flat approach.

These results are due to the fact that in presence of few leaf nodes the Flat approach will return good results, when there are more labels it becomes difficult to difference between them with a single classifier. In shallow hierarchies there are approximately 15 labels but in shallow hierarchies the number of labels increases to 24 therefore our method return better results.

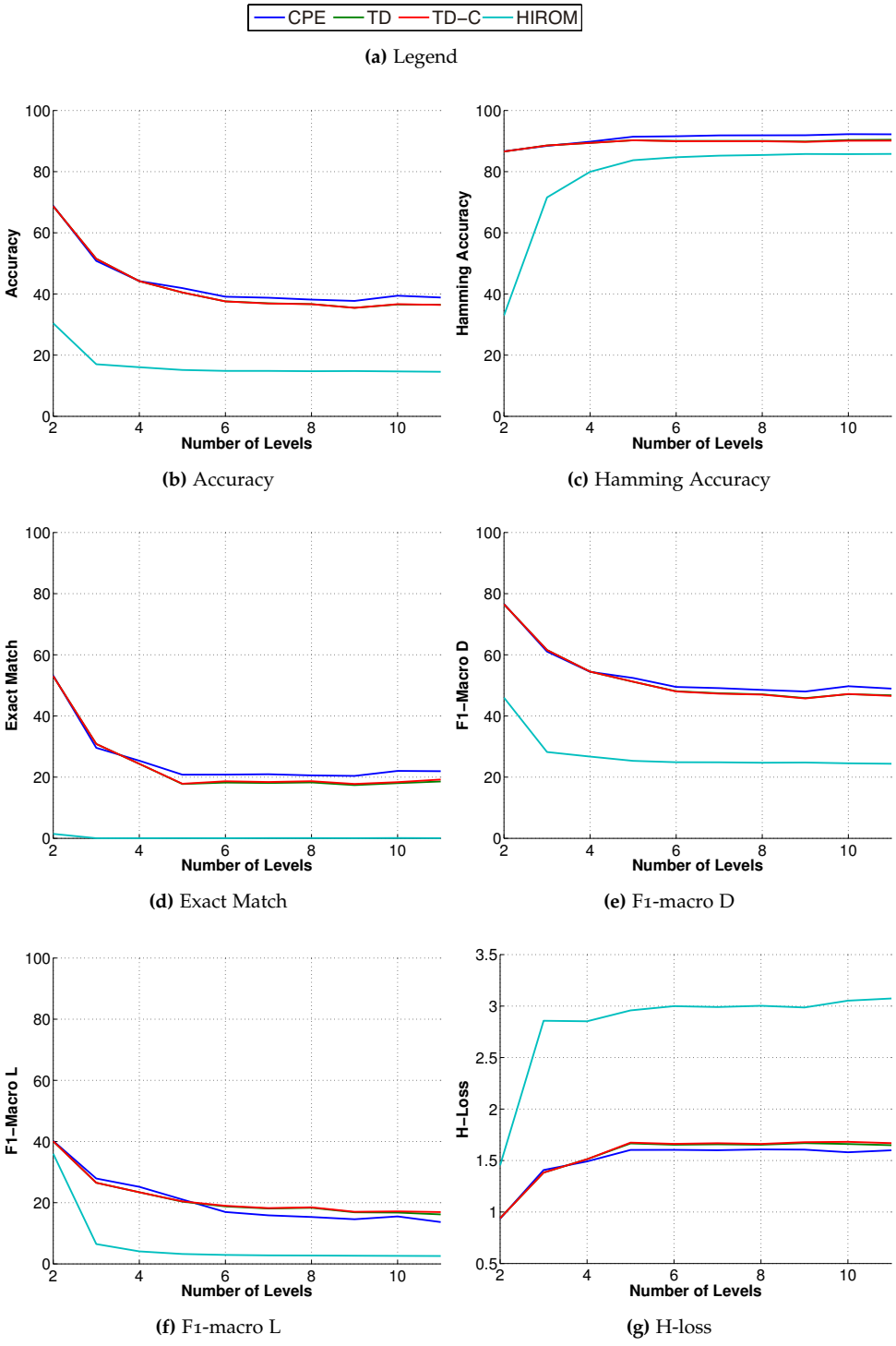
5.2.4 Hierarchy depth effect over classification performance

An experiment was set to determine the effect of the depth of the hierarchy in the classification performance of our method. To achieve this task the hierarchy of the DAG datasets was pruned from 11 to 2 levels maintaining the same instances, this means that the starting structures are DAGs but in shallower levels became trees due to the elimination of nodes. This is the reason why TD-C obtains exactly the same results as TD at shallower hierarchies. We performed a ten-cross fold-validation at each level. The results are depicted in Figure 5.2. Each plot represents a different evaluation measure performance (y-axis) and the number of levels (x-axis), the lines represent the performance along the different classification methods averaged along all the datasets.

When there are less levels, there is also a decrease in the number of labels. In general it is easier to classify when there are less labels. The performance decreases in a lower rate in the deeper hierarchies because less nodes/labels are pruned in the deeper levels, in other words the difference between the number of labels between levels 8 and 10 is lower than the difference between levels 2 and 4.

Discussion

The classification performance of all the methods in accuracy, exact match, F1-macro D and F1-macro L metrics drop off on the first two or three levels, then it steadily falls approximately on the 4th or 5th levels and then it stabilizes. In



(a) Legend

(b) Accuracy

(c) Hamming Accuracy

(d) Exact Match

(e) F1-macro D

(f) F1-macro L

(g) H-loss

Figure 5.2: Plots comparing the performance of CPE, TD, TD-C and HIROM with hierarchies of different depths, using different metrics.

the case of H-loss the behavior is similar but inverted since it is a loss function, the smaller the better.

We expected hamming accuracy to behave as the first metrics but it increased with the number of levels. Hamming accuracy (Equation (3.1)) comes from Hamming loss function (Equation (3.3)), this function evaluates how many times a set of labels for an example is misclassified normalized by the number of labels. The growth in the measure when there are more levels can be due to this normalization, while there are more labels an error cost less than an error when there are fewer labels.

Other methods perform equal or better than CPE with shallow hierarchies, but CPE stabilizes sooner when the number of levels starts to grow which allows our method to outperform the rest of them whose performance drop rapidly after four levels.

From this analysis we consider that our method works well with deep hierarchies since its performance is less affected when there are many levels.

5.2.5 NMLNP

Since NMLNP is the general case of MLNP we developed an extension for CPE called CPE-NMLNP. CPE-NMLNP and CPE-MLNP obtain almost the same performance when classifying MLNP datasets (Table 5.1). The results of NMLNP are slightly better for tree datasets and slightly worst for DAG than MLNP in any case significant. This suggest that our method is pruning less nodes when the datasets are MLNP which means it can be used for both MLNP and NMLNP because it has about the same performance. The pruning is even avoiding some errors specially in tree structures where is obtaining an slight (not significant) advantage. In larger paths, like the ones in DAG structures it is pruning more nodes and thus obtaining an slight disadvantage.

5.2.5.1 CPE-NMLNP vs CPE-MLNP

We compared the variant for NMLNP of CPE method against MLNP using the NMLNP datasets described in Table 5.2, to determine if it is worth to prune the predicted paths. Table 5.27 depicts the results. A non-paired, one-tail t-test was carried out to find statistical significance in the results with a confidence level of 95%.

Discussion

In the case of tree datasets the difference between both methods is not very clear, in fact, in neither of the databases the results are significant different. In every DAG dataset the results obtained by NMLNP version are significantly superior compared to the MLNP version for NMLNP datasets. This means

Table 5.26: Comparing CPE-NMLNP against CPE-MLNP using MLNP datasets (mean [std]).

(a) Tree datasets

Evaluation Measure	NMLNP	MLNP
Accuracy	**23.86 [5.76]**	23.65 [5.97]
Hamming Accuracy	**90.15 [2.23]**	90.12 [2.22]
Exact Match	**18.57 [4.83]**	18.48 [5.09]
F1-macro D	**26.53 [6.31]**	26.26 [6.46]
F1-macro L	**14.1 [4.2]**	14.05 [4.4]
H-loss	**1.629 [0.1]**	1.63 [0.1]
NGLB	**61.89 [2.62]**	61.76 [2.77]

(b) DAG datasets

Evaluation Measure	NMLNP	MLNP
Accuracy	38.83 [5.51]	**39.12 [5.59]**
Hamming Accuracy	92.18 [0.6]	**92.21 [0.6]**
Exact Match	22.07 [6.35]	**22.47 [6.38]**
F1-macro D	48.88 [4.82]	**49.11 [4.88]**
F1-macro L	13.69 [4.8]	**14.16 [5.24]**
H-loss	**1.596 [0.13]**	1.587 [0.13]
NGLB	78.18 [1.92]	**78.28 [1.93]**

that the pruning works better in the presence of deeper hierarchies like the DAG ones.

5.2.5.2 Comparison Against Other NMLNP Methods

The proposed method, CPE-NMLNP, was evaluated experimentally with a number of tree and DAG structured hierarchies and compared with various hierarchical classification techniques. The results were obtained by a stratified 10-fold cross-validation.

The four evaluation measures are reported in different tables for each dataset. A Friedman test was carried out to find statistical significance in the results with a confidence level of 95% using Tukey's honestly significant difference criterion.

Tree Structured Datasets For tree structured hierarchies, we used ten hierarchical datasets and compared CPE against three HMC-NMLNP methods:

1. HIROM. Proposed by Bi and Kwok (2012).

2. True Path Rule (TPR) by Valentini (2009). Using a threshold of 0.5.

3. Weighted True Path Rule (TPRw) by Valentini (2009). Using a threshold of 0.5 and a parent weight (w_p) of 0.5 as well. We tried to obtain the best trade off between precision and recall, the authors report that it is achieved with $0.5 \leqslant w_p \leqslant 0.8$. We performed several tests obtaining the best results with 0.5.

Table 5.28: Accuracy (median [IQR]) percentage comparison of MLNP methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	HIROM	TPR	TPRw
cellcycle_FUN	**16.55 [0.84]**	2.98 [0.74]↓	12.16 [2.67]	11.89 [2.68]↓
church_FUN	**7.89 [6.23]**	3.63 [2.47]	6.28 [6.41]	6.45 [9.74]↓
derisi_FUN	**13.92 [1.68]**	3.10 [0.38]↓	9.67 [2.01]	9.89 [1.27]↓
eisen_FUN	**18.86 [8.46]**	1.86 [1.85]↓	13.78 [6.39]↓	14.19 [6.83]↓
expr_FUN	**21.38 [2.41]**	2.48 [0.27]↓	14.75 [1.77]↓	14.93 [1.17]
gasch1_FUN	**21.62 [3.57]**	2.03 [0.37]↓	14.81 [2.95]↓	15.68 [3.14]
gasch2_FUN	**14.26 [7.10]**	0.69 [4.97]↓	10.34 [3.33]	10.07 [5.59]
pheno_FUN	**12.97 [4.11]**	5.57 [1.83]↓	10.06 [2.67]	10.32 [1.93]
seq_FUN	**23.29 [5.28]**	3.08 [0.84]↓	16.21 [3.61]↓	16.78 [4.73]
spo_FUN	**13.82 [2.47]**	3.36 [1.04]↓	11.28 [1.57]	10.97 [1.93]↓

DAG Structured Datasets For DAG structured hierarchies, we used ten hierarchical datasets and compared CPE against three HMC-NMLNP methods:

1. HIROM. Proposed by Bi and Kwok (2012).

Discussion

The results on tree structured datasets (see Section F.1) are statistically significant against all the compared methods on accuracy, exact match, F1macro D and NGLB. Regarding to Hamming-accuracy CPE is statistically superior to HIROM and TPR but TPRw is statistically superior in datasets with more redundancy as cellcycle_FUN, expr_FUN, gasch1_FUN and seq_FUN, and in the cases of problems with medium complexity CPE obtains the best results. F1-macro L metric does not have a clear winner: CPE, TPR and TPRw obtain almost the same low results compared to F1-macro D which means that there are some labels that are very bad classified; HIROM obtain the lower values. TPRw method obtains the best results in the H-loss metric.

Regarding to DAG structured datasets (see Section F.2) CPE obtains the best results with statistical significancy, it is only compared against HIROM.

5.2.6 Time

We tested the training and testing time required by our method against the time required by other methods. This test was performed using an OS X version 10.9.3 personal computer with a 1.4 GHz Intel Core 2 Duo processor and 5GB 1067 MHz DDR3 memory. This is an empirical evaluation which is not generalizable but we performed it to provide a framework of the training and testing times in comparison with other methods.

For MLNP datasets we tested with approximately 1956 training instances and 217 testing instances depending on the dataset for each fold in tree datasets; and approximately 1543 training instances and 171 testing instances for each fold in DAG datasets. The results were averaged along the 10 folds. The results are depicted in Tables 5.36 and 5.38.

For NMLNP datasets we tested with approximately 2948 training instances and 328 testing instances depending on the dataset for each tree dataset; and approximately 3177 training instances and 353 testing instances for each DAG dataset. We performed a ten-fold cross-validation.

Discussion

Regarding the training time, the Flat approach spends around 3 times less time, TD and TD-C around 1.5 times less and MHC almost spends the same time than CPE. As for testing time the difference is much more significant Flat approach spends approximately 10 times less time than CPE, TD around 6

times less, TD-C close to 5 times less and MHC spends similar time. In MLNP CPE is outperformed by most of the methods but, as the time is measured in seconds, the difference is not very relevant.

Those results were the expected ones because the Flat approach trains just one classifier with as many labels as the leaf nodes of the hierarchy, and same for testing. TD is the simplest hierarchical approach where not even all the labels/nodes are visited during testing phase; its training phase does not take into account the relations which makes the selection of the training set, an easier task. As TD-C performs an extra step where the paths are completed is spends more time than normal TD but not as much as CPE. A more complex approach MHC spends almost the same time as CPE.

With respect to HIROM approach it is a complex method where a function is optimized using a greedy approach. This causes the training and testing time to be at least twice as long as CPE in both MLNP and NMLNP.

In general local classifiers spend less time than global classifiers, since all the compared methods are local ones the difference is small. For example one global method (hAntMiner) spent approximately 6 hours to train one of the datasets used in this thesis.

CPE spends more time than the most simple approaches but performs better, it does not spend so much time as the complex approach and also outperforms them.

5.3 Summary

In this chapter we performed a series of experiments from which we concluded that our method, using a novel weighting scheme (RSM), performs better with deep, DAG hierarchies and redundant, complex datasets. Our method spends a reasonable amount of time in training a classification tasks. Our methods is competitive against methods of the state of the art.

Table 5.27: Normalized Gain-Loose Balance (mean [std]) percentage comparing NMLNP against MLNP. Statistically inferior results against NMLNP are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	NMLNP	MLNP
cellcycle_FUN	**58.96 [0.55]**	58.88 [0.97]
church_FUN	**56.40 [1.89]**	56.29 [2.48]
derisi_FUN	**57.66 [1.00]**	57.51 [0.83]
eisen_FUN	60.96 [3.02]	**61.23 [2.98]**
expr_FUN	**60.48 [1.09]**	60.44 [1.44]
gasch1_FUN	60.69 [1.13]	**60.97 [1.09]**
gasch2_FUN	**59.26 [3.19]**	59.21 [3.10]
pheno_FUN	59.60 [1.55]	**59.61 [2.09]**
seq_FUN	**62.16 [1.44]**	61.88 [1.90]
spo_FUN	**57.90 [0.94]**	57.77 [0.99]

(b) DAG datasets

Dataset	NMLNP	MLNP
cellcycle_GO	**83.83 [0.43]**	79.91 [0.44]↓
church_GO	**83.57 [0.67]**	79.16 [1.43]↓
derisi_GO	**83.73 [0.53]**	78.98 [0.43]↓
expr_GO	**83.55 [0.65]**	81.17 [0.40]↓
gasch1_GO	**83.96 [0.59]**	81.03 [0.65]↓
gasch2_GO	**84.12 [1.56]**	80.00 [1.62]↓
seq_GO	**84.01 [0.53]**	82.26 [0.61]↓
spo_GO	**83.84 [0.53]**	79.59 [0.42]↓

Table 5.29: Hamming accuracy (median [IQR]) percentage comparison of MLNP methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	HIROM	TPR	TPRw
cellcycle_FUN	92.69 [0.11]	89.25 [0.24]↓	89.78 [0.42]	**93.13 [0.39]**
church_FUN	**92.67 [0.50]**	89.33 [0.33]↓	88.79 [3.59]↓	91.92 [1.18]
derisi_FUN	**92.49 [0.29]**	89.23 [0.13]↓	87.49 [1.04]↓	91.58 [0.48]
eisen_FUN	90.59 [0.90]	85.29 [0.23]↓	86.65 [1.46]↓	**90.71 [1.14]**
expr_FUN	92.56 [0.34]	89.24 [0.14]↓	90.18 [0.52]	**93.45 [0.46]**
gasch1_FUN	92.79 [0.33]	89.26 [0.12]↓	89.77 [0.57]	**93.39 [0.38]**
gasch2_FUN	**92.60 [0.69]**	89.19 [0.99]↓	88.98 [0.66]↓	92.59 [1.35]
pheno_FUN	**86.02 [1.03]**	79.19 [0.39]↓	84.13 [2.24]↓	84.88 [1.65]
seq_FUN	93.07 [0.50]	89.78 [0.17]↓	90.94 [0.83]	**93.80 [0.26]**
spo_FUN	**92.56 [0.34]**	89.36 [0.10]↓	89.42 [0.72]↓	92.48 [0.43]

Table 5.30: Exact Match (median [IQR]) percentage comparison of NMLNP methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	HIROM	TPR	TPRw
cellcycle_FUN	**10.68 [1.39]**	1.94 [0.83]↓	4.58 [1.39]↓	5.28 [0.82]
church_FUN	**4.71 [5.81]**	2.50 [1.93]	0.97 [4.72]↓	1.25 [5.00]↓
derisi_FUN	**8.66 [0.86]**	2.23 [0.57]↓	2.66 [1.40]↓	3.63 [1.39]
eisen_FUN	**12.42 [9.44]**	0.86 [1.28]↓	5.79 [8.17]↓	6.65 [8.61]
expr_FUN	**13.66 [1.66]**	1.38 [1.10]↓	6.77 [1.64]↓	7.59 [1.38]
gasch1_FUN	**13.85 [1.97]**	1.25 [0.55]↓	5.54 [2.49]↓	7.06 [1.94]
gasch2_FUN	**7.86 [8.30]**	0.41 [3.04]↓	2.49 [5.81]↓	3.31 [8.29]
pheno_FUN	**7.85 [2.74]**	4.44 [1.37]↓	3.42 [1.36]↓	3.75 [2.07]↓
seq_FUN	**14.74 [4.52]**	1.99 [0.54]↓	8.36 [3.99]↓	9.42 [4.22]
spo_FUN	**8.86 [2.79]**	2.39 [0.85]↓	3.80 [1.97]↓	4.22 [1.41]

Table 5.31: F1-macro D (median [IQR]) percentage comparison of NMLNP methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	HIROM	TPR	TPRw
cellcycle_FUN	**19.14 [1.01]**	3.42 [0.86]↓	15.18 [3.01]	14.35 [2.93]↓
church_FUN	**10.12 [7.59]**	4.09 [2.76]↓	9.21 [6.91]	9.12 [10.90]
derisi_FUN	**16.27 [2.32]**	3.53 [0.43]↓	12.73 [2.08]	12.35 [1.23]↓
eisen_FUN	**21.89 [7.08]**	2.21 [2.17]↓	17.06 [5.91]	16.85 [6.19]↓
expr_FUN	**24.68 [2.57]**	2.96 [0.30]↓	18.20 [1.59]	17.78 [1.07]↓
gasch1_FUN	**24.98 [4.39]**	2.49 [0.44]↓	18.91 [3.32]	18.86 [3.72]↓
gasch2_FUN	**17.46 [6.20]**	0.80 [5.77]↓	14.21 [2.86]	12.83 [4.50]
pheno_FUN	**15.39 [4.54]**	6.08 [1.92]↓	12.62 [3.22]	12.87 [2.46]
seq_FUN	**27.20 [5.12]**	3.49 [1.02]↓	19.74 [3.53]	19.56 [4.61]↓
spo_FUN	**16.68 [2.93]**	3.80 [0.87]↓	14.61 [2.02]	13.75 [2.29]↓

Table 5.32: F1-macro L (median [IQR]) percentage comparison of NMLNP methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	HIROM	TPR	TPRw
cellcycle_FUN	**12.12 [2.60]**	0.36 [0.08]↓	11.48 [1.72]	11.54 [2.92]
church_FUN	4.96 [3.78]	0.58 [0.61]↓	**6.35 [2.96]**	4.97 [3.98]
derisi_FUN	**8.66 [0.57]**	0.38 [0.05]↓	8.62 [1.48]	8.44 [1.22]
eisen_FUN	12.81 [4.79]	0.43 [0.45]↓	14.74 [5.36]	**15.48 [6.90]**
expr_FUN	**16.34 [1.17]**	0.30 [0.03]↓	15.17 [1.58]	15.21 [2.16]
gasch1_FUN	15.65 [0.83]	0.24 [0.06]↓	15.61 [2.40]	**16.22 [2.04]**
gasch2_FUN	8.45 [4.14]	0.14 [0.98]↓	10.36 [2.45]	**10.50 [2.83]**
pheno_FUN	9.87 [3.77]	1.64 [0.61]↓	**12.24 [5.09]**	11.96 [3.88]
seq_FUN	**16.52 [2.10]**	0.37 [0.17]↓	14.36 [2.30]	14.78 [2.19]
spo_FUN	9.70 [1.55]	0.43 [0.13]↓	**10.46 [1.44]**	9.97 [2.44]

Table 5.33: H-loss (median [IQR]) comparison of MLNP methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	HIROM	TPR	TPRw
cellcycle_FUN	1.753 [0.03]	1.933 [0.02]	1.513 [0.05]	**1.285 [0.06]**↑
church_FUN	1.897 [0.11]	1.922 [0.05]	1.696 [0.14]	**1.520 [0.14]**↑
derisi_FUN	1.804 [0.03]	1.93 [0.01]	1.759 [0.06]	**1.508 [0.1]**↑
eisen_FUN	1.72 [0.18]	1.957 [0.04]	1.451 [0.12]	**1.286 [0.15]**↑
expr_FUN	1.668 [0.03]	1.944 [0]	1.444 [0.05]	**1.236 [0.05]**↑
gasch1_FUN	1.672 [0.06]	1.95 [0.01]	1.472 [0.09]	**1.256 [0.06]**↑
gasch2_FUN	1.808 [0.18]	1.985 [0.11]	1.581 [0.13]	**1.351 [0.17]**↑
pheno_FUN	1.823 [0.07]	1.881 [0.03]	1.647 [0.2]	**1.582 [0.16]**↑
seq_FUN	1.647 [0.07]	1.935 [0.02]	1.404 [0.12]	**1.202 [0.06]**↑
spo_FUN	1.803 [0.04]	1.927 [0.02]	1.592 [0.05]	**1.383 [0.06]**↑

Table 5.34: Gain-Loose Balance (median [IQR]) percentage comparison of NMLNP methods. Statistically different results compared with CPE are marked with ↑ if they are superior and ↓ if they are inferior.

Dataset	CPE	HIROM	TPR	TPRw
cellcycle_FUN	**59.00 [0.62]**	40.36 [0.93]↓	50.92 [0.79]↓	57.32 [1.66]
church_FUN	**55.82 [1.98]**	40.86 [1.74]↓	46.73 [8.63]↓	51.65 [4.35]
derisi_FUN	**57.61 [1.18]**	40.31 [0.31]↓	45.32 [1.99]↓	53.15 [1.11]
eisen_FUN	**59.94 [3.43]**	38.95 [0.69]↓	52.54 [4.27]↓	57.57 [3.60]
expr_FUN	**60.30 [2.03]**	40.07 [0.94]↓	53.06 [1.52]↓	59.28 [0.91]
gasch1_FUN	**61.09 [1.81]**	39.91 [0.53]↓	51.79 [2.87]↓	59.17 [1.57]
gasch2_FUN	**58.42 [3.50]**	38.95 [4.95]↓	48.99 [2.83]↓	55.98 [4.83]
pheno_FUN	**59.30 [3.08]**	47.78 [0.63]↓	56.90 [3.38]↓	57.41 [1.82]
seq_FUN	**62.02 [2.75]**	40.82 [0.84]↓	54.15 [2.63]↓	60.00 [2.07]
spo_FUN	**57.88 [0.58]**	40.76 [0.29]↓	49.85 [2.96]↓	55.84 [1.12]

Evaluation Measure	CPE	HIROM
Accuracy	**32.29 [3.14]**	14.92 [0.45]↓
Hamming Accuracy	**93.15 [0.38]**	84.99 [0.24]↓
Exact Match	**9.31 [2.76]**	0.02 [0.03]↓
F1-macro D	**44.68 [2.91]**	24.81 [0.59]↓
F1-macro L	**8.56 [2.76]**	3.33 [0.21]↓
H-loss	**1.604 [0.07]**	3.021 [0.06]↓
NGLB	**80.18 [1.20]**	64.03 [0.41]↓

Table 5.35: Comparison of CPE-NMLNP against another NMLNP method for DAG datasets.

Table 5.36: Time (seconds) required for training MLNP methods.

(a) Tree datasets

Dataset	CPE	Flat	TD	MHC	HIROM
cellcycle_FUN	3.12 [0.74]	**1.30 [0.26]**	1.74 [0.26]	2.50 [0.44]	6.21 [0.75]
church_FUN	1.54 [0.16]	**0.49 [0.03]**	0.85 [0.12]	1.28 [0.09]	3.83 [0.26]
derisi_FUN	2.91 [0.34]	**1.56 [0.10]**	1.79 [0.02]	2.58 [0.09]	7.12 [0.76]
eisen_FUN	1.48 [0.05]	**0.72 [0.04]**	0.90 [0.05]	1.39 [0.19]	3.23 [0.32]
expr_FUN	7.49 [0.36]	**2.92 [0.14]**	4.28 [0.14]	7.77 [1.44]	16.30 [0.60]
gasch1_FUN	4.08 [0.61]	**1.58 [0.05]**	2.20 [0.16]	3.21 [0.14]	8.26 [0.54]
gasch2_FUN	3.04 [0.42]	**1.38 [0.03]**	2.08 [0.65]	3.17 [0.80]	5.76 [0.12]
pheno_FUN	0.73 [0.04]	**0.30 [0.01]**	0.47 [0.05]	0.90 [0.19]	1.76 [0.09]
seq_FUN	7.19 [0.64]	**2.57 [0.10]**	5.07 [0.82]	6.23 [0.72]	17.03 [2.89]
spo_FUN	3.37 [0.42]	**1.40 [0.03]**	2.24 [0.04]	2.50 [0.14]	8.72 [3.00]

(b) DAG datasets

Dataset	CPE	Flat	TD	TD-C	HIROM
cellcycle_GO	2.45 [0.53]	**1.01 [0.25]**	1.79 [0.40]	3.19 [0.85]	5.93 [1.63]
church_GO	1.21 [0.16]	**0.38 [0.07]**	0.92 [0.16]	1.03 [0.19]	3.57 [0.26]
derisi_GO	2.60 [0.24]	**1.06 [0.03]**	1.73 [0.04]	2.08 [0.35]	5.83 [0.43]
expr_GO	11.01 [1.74]	**2.16 [0.08]**	5.72 [0.20]	8.64 [2.03]	16.46 [2.44]
gasch1_GO	4.75 [0.36]	**1.14 [0.04]**	2.57 [0.07]	3.65 [0.78]	7.61 [0.80]
gasch2_GO	2.74 [0.56]	**0.97 [0.06]**	1.53 [0.02]	1.77 [0.17]	4.94 [0.46]
seq_GO	7.00 [0.46]	**1.79 [0.26]**	4.78 [0.13]	6.39 [1.07]	12.14 [0.97]
spo_GO	2.41 [0.11]	**1.07 [0.07]**	1.77 [0.02]	2.29 [0.70]	5.37 [0.43]

Table 5.37: Time required for training NMLNP datasets.

(a) Tree datasets

Dataset	CPE	HIROM
cellcycle_FUN	**3.81 [0.68]**	13.66 [1.59]
church_FUN	**1.78 [0.11]**	8.90 [1.11]
derisi_FUN	**3.91 [0.14]**	15.27 [1.40]
eisen_FUN	**1.96 [0.08]**	6.29 [0.34]
expr_FUN	**10.70 [0.17]**	35.62 [1.78]
gasch1_FUN	**5.10 [0.11]**	18.69 [1.37]
gasch2_FUN	**3.35 [0.06]**	13.20 [1.35]
pheno_FUN	**0.75 [0.05]**	2.77 [0.30]
seq_FUN	**9.37 [0.35]**	35.35 [1.61]
spo_FUN	**3.74 [0.09]**	14.78 [0.86]

(b) DAG datasets

Dataset	CPE	HIROM
cellcycle_GO	**3.83 [0.80]**	13.36 [1.34]
church_GO	**1.85 [0.21]**	10.76 [1.25]
derisi_GO	**3.87 [0.20]**	15.14 [0.68]
expr_GO	**13.36 [0.45]**	38.08 [2.74]
gasch1_GO	**5.57 [0.52]**	18.09 [1.74]
gasch2_GO	**3.24 [0.09]**	11.55 [1.01]
seq_GO	**12.46 [1.68]**	37.77 [7.37]
spo_GO	**5.03 [0.48]**	13.70 [0.43]

Table 5.38: Time (seconds) required for testing MLNP methods.

Dataset	CPE	Flat	TD	MHC	HIROM
cellcycle_FUN	0.25 [0.18]	0.04 [0.03]	**0.03 [0.01]**	0.20 [0.14]	0.37 [0.21]
church_FUN	0.11 [0.02]	**0.01 [0.01]**	0.02 [0.01]	0.09 [0.01]	0.17 [0.02]
derisi_FUN	0.16 [0.02]	**0.02 [0.01]**	**0.02 [0.00]**	0.14 [0.01]	0.33 [0.13]
eisen_FUN	0.06 [0.01]	**0.01 [0.00]**	**0.01 [0.00]**	0.05 [0.01]	0.10 [0.02]
expr_FUN	0.87 [0.09]	**0.10 [0.01]**	0.14 [0.02]	0.96 [0.21]	1.58 [0.34]
gasch1_FUN	0.44 [0.07]	**0.04 [0.00]**	0.06 [0.03]	0.31 [0.04]	0.60 [0.08]
gasch2_FUN	0.21 [0.05]	**0.02 [0.00]**	0.05 [0.05]	0.20 [0.07]	0.29 [0.11]
pheno_FUN	0.02 [0.01]	**0.01 [0.00]**	**0.01 [0.01]**	0.02 [0.01]	0.05 [0.01]
seq_FUN	1.13 [0.17]	**0.11 [0.01]**	0.19 [0.04]	0.98 [0.13]	1.79 [0.56]
spo_FUN	0.25 [0.05]	**0.02 [0.01]**	0.05 [0.01]	0.19 [0.05]	0.47 [0.23]

(a) Tree datasets

Dataset	CPE	Flat	TD	TD-C	HIROM
cellcycle_GO	0.25 [0.09]	**0.04 [0.03]**	**0.04 [0.01]**	0.07 [0.02]	0.47 [0.22]
church_GO	0.15 [0.02]	**0.02 [0.01]**	0.03 [0.00]	0.03 [0.00]	0.29 [0.03]
derisi_GO	0.25 [0.05]	**0.02 [0.00]**	0.04 [0.00]	0.06 [0.03]	0.42 [0.05]
expr_GO	1.90 [0.47]	**0.10 [0.01]**	0.22 [0.02]	0.40 [0.24]	2.23 [0.19]
gasch1_GO	0.78 [0.12]	**0.04 [0.01]**	0.09 [0.01]	0.15 [0.04]	0.87 [0.07]
gasch2_GO	0.32 [0.15]	**0.02 [0.01]**	0.03 [0.01]	0.06 [0.02]	0.46 [0.07]
seq_GO	1.17 [0.14]	**0.11 [0.03]**	0.21 [0.02]	0.28 [0.05]	1.96 [0.34]
spo_GO	0.28 [0.03]	**0.03 [0.01]**	0.05 [0.01]	0.07 [0.01]	0.48 [0.05]

(b) DAG datasets

Table 5.39: Time required for testing NMLNP datasets.

(a) Tree datasets

Dataset	CPE	HIROM
cellcycle_FUN	**0.39 [0.12]**	0.98 [0.32]
church_FUN	**0.25 [0.07]**	0.77 [0.30]
derisi_FUN	**0.34 [0.03]**	0.92 [0.16]
eisen_FUN	**0.15 [0.02]**	0.32 [0.07]
expr_FUN	**2.00 [0.13]**	4.22 [0.31]
gasch1_FUN	**0.75 [0.02]**	1.67 [0.16]
gasch2_FUN	**0.35 [0.04]**	0.90 [0.12]
pheno_FUN	**0.04 [0.01]**	0.11 [0.02]
seq_FUN	**1.97 [0.15]**	4.66 [0.59]
spo_FUN	**0.43 [0.02]**	1.12 [0.19]

(b) DAG datasets

Dataset	CPE	HIROM
cellcycle_GO	**0.62 [0.27]**	1.23 [0.24]
church_GO	**0.31 [0.03]**	0.96 [0.23]
derisi_GO	**0.57 [0.06]**	1.23 [0.16]
expr_GO	**3.72 [0.37]**	5.85 [0.56]
gasch1_GO	**1.20 [0.08]**	2.12 [0.26]
gasch2_GO	**0.55 [0.05]**	1.11 [0.26]
seq_GO	**3.72 [0.69]**	6.60 [1.31]
spo_GO	**0.98 [0.21]**	1.28 [0.12]

Chapter 6

Conclusions

This thesis addressed the Hierarchical Multi-label Classification problem, where examples can be associated to multiple labels and these labels belong to a predefined structure.

We have presented a novel approach for hierarchical multi-label classification for tree and DAG structures. The method estimates the probability of each path by combing LCPNs, incorporating two additional features: (i) a weighting scheme that gives more importance to correct predictions at the top levels; (ii) an extension of the chain idea for hierarchical classification, incorporating the label of the parent nodes as additional attributes. Experiments with 18 tree and DAG hierarchies were performed. Our method obtained superior results when dealing with deep hierarchies and competitive performance with shallower hierarchies when compared to state-of-the-art algorithms.

A version for NMLNP of our method was developed; we compared several pruning approaches varying the pruning direction, pruning time and pruning condition; we discovered that the best prune was obtained in a top-dow fashion, pruning the hierarchy and then selecting the best path and pruning when the unknown probability surpass the most probable node al a local classifier. We tested with 18 tree and DAG structured datasets Our method resulted superior against several state of the art methods in DAG and competitive for tree structures.

A new metric was introduced to avoid conservative classifications, this metric takes into account the level of the labels and the number of siblings in the hierarchy in order to include extra information in the evaluation of the predictions and obtain predictions that contribute with relevant information and not just the most general.

6.1 Conclusions

The proposed method was evaluated empirically and compared with several state of the art methods.

The experimental evaluation of this thesis provides an empirical analysis with respect to the number and variety of evaluation measures (including the introduction of Gain-Loose Balance metric), the number of datasets used in experimentation and the evaluation of the characteristics of the method. This framework allowed a more conscious evaluation and comparison of the performance of existing methods in the literature. Next we summarize our findings.

6.1.1 Weighting Scheme

We proposed a weighting scheme (RSM) which takes into account the level of the nodes and divides the weight linearly along the levels, it improves the results in most metrics compared with a non-weighted scheme. Our weighting scheme showed to be more consistent than other current approach along both shallow and deep hierarchies, when combined with the proposed method and at least as good as other current approach.

6.1.2 Depth effect

When there are more levels in the hierarchy, hence more classes, the classification performance decreases, this is because as the number of labels increases the complexity of separating their spaces grow. The performance suffers a drop and then stabilizes when the number of levels increases, this can be due to the fact that in the last levels less nodes are eliminated from the hierarchy. The proposed method reaches the stabilization point slightly sooner than the other methods with which it was compared. Our method is a good alternative when dealing with deep hierarchies.

It is worth mentioning that HIROM was the method that stabilized sooner (approximately at 4 levels) in all the evaluation measures though it obtained the lower performance in the bunch.

6.1.3 Comparing CPE against other methods

By including information of the hierarchy, in the form of taking into account the prediction of the parent node and the position of the label in the hierarchy the performance of the HMC using local classifiers can be boosted. This resulted in CPE outperforming in many cases other classifiers and even returned better results when dealing with redundant and complex (many relevant attributes) datasets with deep hierarchies and many labels.

Interestingly, HIROM was the most stable of the methods obtaining approximately the same performance along all the datasets in all the evaluation measures.

6.1.4 NMLNP

We proposed a pruning condition for NMLNP that tries to minimize the reduction of the specificity of the predictions. Our method tries to prune the prediction in the moment where no more information can be extracted, thus reducing specificity errors and returning too general predictions that does not provide enough information.

6.2 Contributions

This thesis identified the scarcity of HMC methods based on local classifiers that exploit the relationships between the nodes and the relevance of the labels according to the level.

The major contributions include a novel hierarchical classification method applicable for both MLNP and NMLNP, problems with tree or DAG hierarchy structures. This method fills a gap in local HMC methods in the literature.

Another contribution is the proposal of a new evaluation measure designed for NMLNP which deals with the bias towards conservative predictions.

The versatility of our method for a variety of conditions and the scalability ensures that it can remain relevant in the hierarchical multi-label classification task.

The proposed CPE method, where the relations of the labels are taken into account even when the approach divide the classification problem, can be powerful when dealing with problems with deep hierarchies structures, but is not so effective in shallow hierarchies (surpassed performance by simpler approaches when dealing with less than 3 levels of specificity). Furthermore, this method performs well in the presence of redundant and complex datasets.

CPE incorporates the relations of the labels in the hierarchy by adding the parent prediction as an appended attribute and by taking into account the level at which the label is placed assigning a weight to each level. Our method combines the modularity of local classifiers with the overall view of global classifiers.

As compared with other methods in the literature, CPE:

- keeps a training and classifying time in the range of simpler classifiers;
- achieves competitive performance with shallow structured dataset; and
- superior performance with deep structured datasets.

The CPE-NMLNP deals with the problem of not necessarily selecting the most specific label by making a pruning step before returning the prediction. Pruning is performed before choosing the best path (prune the whole set

of possible paths in the hierarchy), in a top-down fashion, using the most probable child as the condition to prune a node.

The NMLNP variant of our method is competitive with other methods in the literature. The two variants of our method are able to use different base classifiers; to fit the domain and complexity of the datasets.

During this research we developed an extension to the multi-label framework Meka[1], to implement and evaluate our methods and other methods from the literature.

6.3 Future Work

Hierarchical multi-label classification is becoming a trend for big datasets with a large number of labels which have a predefined structure.

This thesis has discussed an alternative for taking advantage of the predefined structure of the labels to boost the performance of the classification. As future work we would like to deal with multiple paths from the root to different nodes in the hierarchy by using automatically selected thresholds obtained from the data.

[1] http://sourceforge.net/projects/meka/

References

Aghaie, A. and Saeedi, A. (2009). Using bayesian networks for bankruptcy prediction: Empirical evidence from iranian companies. In *International Conference on Information Management and Engineering, 2009. ICIME '09.*, pages 450–455.

Alaydie, N., Reddy, C. K., and Fotouhi, F. (2012). Exploiting Label Dependency for Hierarchical Multi-label Classification. *Advances in Knowledge Discovery and Data Mining*, 7301:294–305.

Alves, R., Delgado, M., and Freitas, A. a. (2008). Multi-label hierarchical classification of protein functions with artificial immune systems. *Advances in Bioinformatics and Computational Biology*, pages 1–12.

Araújo, B. S. (2006). *Aprendizaje automático: conceptos básicos y avanzados*. Pearson Prentice Hall.

Ashburner, M., Ball, C. A., and Blake, J. A. (2000). Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29.

Bar-Yam, Y. (2011). Concepts: Linear and nonlinear. http://necsi.edu/guide/concepts/linearnonlinear.html.

Barutcuoglu, Z. and DeCoro, C. (2006). Hierarchical shape classification using Bayesian aggregation. *IEEE International Conference on Shape Modeling and Applications, 2006. SMI 2006*.

Barutcuoglu, Z., Schapire, R., and Troyanskaya, O. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–6.

Bee Wah, Y. and Ibrahim, I. (2010). Using data mining predictive models to classify credit card applicants. In *6th International Conference on Advanced Information Management and Service (IMS)*, pages 394–398.

Bennett, P. N. and Nguyen, N. (2009). Refined experts: Improving classification in large taxonomies. In *Proceedings of the 32nd Annual ACM SIGIR Conference*, pages 11–18. Association for Computing Machinery, Inc.

Bi, W. and Kwok, J. T. (2011). Multi-label classification on tree-and dag-structured hierarchies. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 17–24.

Bi, W. and Kwok, J. T. (2012). Hierarchical Multilabel Classification with Minimum Bayes Risk. *2012 IEEE 12th International Conference on Data Mining*.

Bielza, C., Li, G., and Larranaga, P. (2011). Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727.

Binder, A., Kawanabe, M., and Brefeld, U. (2010). Efficient classification of images with taxonomies. *Computer Vision–ACCV 2009*.

Blockeel, H. and Bruynooghe, M. (2002). Hierarchical multi-classification. pages 21–35.

Blockeel, H., Schietgat, L., Struyf, J., Džeroski, S., and Clare, A. (2006). Decision trees for hierarchical multilabel classification: A case study in functional genomics. In *Knowledge Discovery in Databases: PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 18–29.

Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. (2004). Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771.

Campbell, D. J. (1988). Task Complexity: A Review and Analysis. *Academy of Management Review*, 13(1):40–52.

Carreira-Perpiñan, M. A. (1997). A Review of Dimension Reduction Techniques. Technical report, Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09.

Ceci, M. and Malerba, D. (2007). Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):37–78.

Cerri, R., Barros, R. C., and Carvalho, A. C. (2013). Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, 1:1–18.

Cesa-Bianchi, N., Gentile, C., and Zaniboni, L. (2006). Incremental algorithms for hierarchical classification. *The Journal of Machine Learning Research*, 7:31–54.

Clare, A. and King, R. (2003). Predicting gene function in Saccharomyces cerevisiae. *Bioinformatics*, 19:ii42–ii49.

Clare, A. and King, R. D. (2001). Knowledge discovery in multi-label phenotype data. In *Principles of data mining and knowledge discovery*, pages 42–53. Springer.

Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Intelligent Systems and their Applications*, 297:273–297.

Costa, E. and Lorena, A. (2008). Top-down hierarchical ensembles of classifiers for predicting g-protein-coupled-receptor functions. In *Advances in Bioinformatics and Computational Biology, Third Brazilian Symposium on Bioinformatics, BSB*, pages 35–46.

Costa, E. P., Lorena, A. C., Carvalho, A. C., Freitas, A. A., and Holden, N. (2007). Comparing several approaches for hierarchical classification of proteins with decision trees. In *Advances in Bioinformatics and Computational Biology*, pages 126–137. Springer.

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

Csiszár, I. (1996). Maxent, mathematics, and information theory. In *Maximum entropy and Bayesian methods*, pages 35–50. Springer.

De Comité, F., Gilleron, R., and Tommasi, M. (2003). Learning multi-label alternating decision trees from texts and data. In Perner, P. and Rosenfeld, A., editors, *Machine Learning and Data Mining in Pattern Recognition*, volume 2734 of *Lecture Notes in Computer Science*, pages 35–49. Springer Berlin Heidelberg.

DeCoro, C., Barutcuoglu, Z., and Fiebrink, R. (2007). Bayesian Aggregation for Hierarchical Genre Classification. In *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007*, pages 77–80.

Dekel, O., Keshet, J., and Singer, Y. (2004). Large margin hierarchical classification. *Twenty-first international conference on Machine learning - ICML '04*, page 27.

Dekel, O., Keshet, J., and Singer, Y. (2005). An online algorithm for hierarchical phoneme classification. *Machine Learning for Multimodal Interaction*, pages 146–158.

Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2010). Detection of visual concepts and annotation of images using ensembles of trees for hierarchical multi-label classification. *Recognizing patterns in signals, speech, images and videos*, pages 152–161.

Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2011). Hierarchical annotation of medical images. *Pattern Recognition*, 44(10-11):2436–2449.

Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2012). Hierarchical classification of diatom images using ensembles of predictive clustering trees. *Ecological Informatics*, 7(1):19–29.

Ding, L., Goshtasby, A., and Satter, M. (2001). Volume image registration by template matching. *Image and Vision Computing*, 19(12):821–832.

Dumais, S. and Chen, H. (2000). Hierarchical classification of Web content. In *SIGIR*, pages 256–263.

Eisner, R., Poulin, B., Szafron, D., Lu, P., and Greiner, R. (2005). Improving protein function prediction using the hierarchical structure of the gene ontology. In *Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, 2005. CIBCB'05.*, pages 1–10. IEEE.

Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In *NIPS*, volume 14, pages 681–687.

Fagni, T. and Sebastiani, F. (2007). On the selection of negative examples for hierarchical text categorization. *Proceedings of LTC-07*, pages 24–28.

Fix, E. and Hodges Jr, J. L. (1951). Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document.

Flach, P. (2012). *The Art and Science of Algorithms that Make Sense of Data*, volume 2010. Cambridge University Press.

Freitas, A. A. and de Carvalho, A. (2007). *A Tutorial on Hierarchical Classification with Applications in Bioinformatics.* IGI Publishing.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54.

Gaag, L. V. D. and Waal, P. D. (2006). Multi-dimensional Bayesian Network Classifiers. *Probabilistic graphical models*, pages 107–114.

Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In Dai, H., Srikant, R., and Zhang, C., editors, *Advances in Knowledge Discovery and Data Mining*, volume 3056 of *Lecture Notes in Computer Science*, pages 22–30. Springer Berlin Heidelberg.

Good, I. J. (1950). *Probability and the Weighing of Evidence*. Charles Griffin.

Grujic, N., Ilic, S., Lepetit, V., and Fua, P. (2008). 3d facial pose estimation by image retrieval. In *8th IEEE Internationall Conference on Automatic Face and Gesture Recognition*, pages 1–6.

Hernandez, J., Sucar, L. E., and Morales, E. F. (2013). A Hybrid Global-Local Approach for Hierarchical Classification. In *Twenty-Sixth International Florida Artificial Intelligence Research Society Conference*, pages 432–437.

Holden, N. and Freitas, A. A. (2008). Improving the Performance of Hierarchical Classification with Swarm Intelligence. *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 48–60.

Huan, M., Wang, K., Zuo, W., and Li, Z. (2011). Template based stereo matching using graph-cut. In *Instrumentation, Measurement, Computer, Communication and Control, 2011 First International Conference on*, pages 303–306. IEEE.

Huang, S.-C., Huang, Y. F., and Jou, I.-C. (1991). Analysis of perceptron training algorithms and applications to hand-written character recognition. In *International Conference on Acoustics, Speech, and Signal Processing, 1991. ICASSP-91.*, pages 2153–2156 vol.3.

Jain, A. K. and Duin, P. (2000). Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37.

Jensen, F. V. (1996). *An introduction to Bayesian networks*, volume 210. UCL press London.

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer Series in Statistics.

Karvelis, P. S., Fotiadis, D., Tsalikakis, D., and Georgiou, I. (2009). Enhancement of multichannel chromosome classification using a region-based classifier and vector median filtering. *IEEE Transactions on Information Technology in Biomedicine*, 13(4):561–570.

Kiritchenko, S., Matwin, S., and Famili, F. (2004). Hierarchical text categorization as a tool of associating genes with gene ontology codes. In *Proceedings of the Second European Workshop on Data Mining and Text Mining for Bioinformatics*, pages 26–30.

Koller, D. and Sahami, M. (1997). *Hierarchically classifying documents using very few words*. Stanford InfoLab.

Kotsiantis, S. (2007). Supervised machine learning: a review of classification techniques. *Informatica (03505596)*.

Labrou, Y. and Finin, T. (1999). Yahoo! as an ontology: using Yahoo! categories to describe documents. In *Proceedings of the eighth international conference on Information and knowledge management.*, pages 180–187.

Li, T. and Ogihara, M. (2003). Detecting emotion in music. In *ISMIR*, volume 3, pages 239–240.

Lin, Z. and Davis, L. S. (2010). Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):604–618.

Lorena, A. C., De Carvalho, A. C., and Gama, J. M. (2008). A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*, 30(1-4):19–37.

McCallum, A., Rosenfeld, R., Mitchell, T., and Ng, A. (1998). Improving Text Classification by Shrinkage in a Hierarchy of Classes. *ICML*.

Mika, S., Ratsch, G., Weston, J., Scholkopf, B., and Muller, K. (1999). Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.*, pages 41–48.

Mithun, P. and Raajan, N. R. (2013). Neural Network Based Augmented Reality for Detection of Brain Tumor. *International Journal of Engineering & Technology (0975-4024)*, 5(2):1688–1692.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389.

Oatley, G. C. and Ewart, B. W. (2003). Crimes analysis software: 'pins in maps', clustering and bayes net prediction. *Expert Systems with Applications*, 25(4):569 – 588.

Omachi, S. and Omachi, M. (2007). Fast template matching with polynomials. *IEEE Transactions on Image Processing*, 16(8):2139–2149.

Otero, F. E. B., Freitas, A. A., and Johnson, C. (2009). A hierarchical classification ant colony algorithm for predicting gene ontology terms. *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 68–79.

Otero, F. E. B., Freitas, A. a., and Johnson, C. G. (2010). A hierarchical multi-label classification ant colony algorithm for protein function prediction. *Memetic Computing 2.3*, 2(3):165–181.

Perotte, A., Pivovarov, R., Natarajan, K., Weiskopf, N., Wood, F., and Elhadad, N. (2014). Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association*, 21(2):231–237.

Qiong Wei, Roland L. Dunbrack, J. (2013). The Role of Balanced Training and Testing Data Sets for Binary Classifiers in Bioinformatics. *PLoS ONE*, 8(7):e67863.

Quinlan, J. R. (1993). *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann.

Quinlan, J. R. et al. (1979). *Discovering rules by induction from large collections of examples*. Expert systems in the micro electronic age. Edinburgh University Press.

Ramírez, M., Sucar, L. E., and Morales, E. F. (2014a). Chained Path Evaluation for Hierarchical Multi-label Classification. In *Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference, FLAIRS 2014, Pensacola Beach, Florida, May 21-23, 2014.*, pages 502–507.

Ramírez, M., Sucar, L. E., and Morales, E. F. (2014b). Multi-label classification for tree and directed acyclic graphs hierachies. In van der Gaag, L. and Feelders, A., editors, *Probabilistic Graphical Models*, volume 8754 of *Lecture Notes in Computer Science*, pages 409–425. Springer International Publishing.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, pages 254–269.

Richards, J. A. and Jia, X. (1999). *Remote sensing digital image analysis*, volume 3. Springer.

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

Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Güldener, U., Mannhaupt, G., Münsterkötter, M., and Mewes, H. W. (2004). The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic acids research*, 32(18):5539–45.

Ruiz, M. E. and Srinivasan, P. (2002). Hierarchical text categorization using neural networks. *Information Retrieval*, pages 87–118.

Schapire, R. E. and Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168.

Secker, A., Davies, M., and Freitas, A. (2007). An experimental comparison of classification algorithms for hierarchical prediction of protein function. *Expert Update (Magazine of the British Computer Society's Specialist Group on AI) 9.3*, pages 17–22.

Secker, A., Davies, M., and Freitas, A. (2010). Hierarchical classification of G-Protein-Coupled Receptors with data-driven selection of attributes and classifiers. *International journal of data mining and bioinformatics 4.2*.

Silla Jr., C. N. and Freitas, A. A. (2009a). A Global-Model Naive Bayes Approach to the Hierarchical Prediction of Protein Functions. *IEEE International Conference on Data Mining*, pages 992–997.

Silla Jr., C. N. and Freitas, A. A. (2009b). Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. *IEEE International Conference on Systems, Man, and Cybernetics*, (October):3499–3504.

Sucar, L. E., Bielza, C., Morales, E. F., Hernandez-Leal, P., Zaragoza, J. H., and Larrañaga, P. (2014). Multi-label classification with bayesian network-based chain classifiers. *Pattern Recognition Letters*, 41(0):14 – 22. Supervised and Unsupervised Classification Techniques and their Applications.

Sucar, L. E. and Gillies, D. F. (1994). Probabilistic reasoning in high-level vision. *Image and Vision Computing*, 12(1):42 – 60.

Sucar, L. E., Perez-Brito, J., and Ruiz-Suárez, J. (1995). Induction of dependence structures from data and its application to ozone prediction. In *Proceedings Eight International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, pages 57–63. DTIC Document.

Sun, A. and Lim, E. (2001). Hierarchical text classification and evaluation. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 521–528.

Sun, A., Lim, E., and Ng, W. (2003). Performance measurement framework for hierarchical text classification. *Journal of the American Society for Information Science and Technology*, 54:1014–1028.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science (New York, N.Y.)*, 290(5500):2319–23.

Thabtah, F. A., Cowling, P., and Peng, Y. (2004). Mmac: A new multi-class, multi-label associative classification approach. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 217–224. IEEE.

Tsoumakas, G. and Katakis, I. (2007). Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining*, 3:1–13.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2010). Mining multi-label data. *Data mining and knowledge discovery handbook*.

Valentini, G. (2009). True path rule hierarchical ensembles. In *Multiple Classifier Systems*, volume 5519 of *Lecture Notes in Computer Science*, pages 232–241. Springer Berlin Heidelberg.

Valentini, G. and Cesa-Bianchi, N. (2008). HCGene: a software tool to support the hierarchical classification of genes. *Bioinformatics*, 24(5):729–31.

Valentini, G. and Re, M. (2009). Weighted True Path Rule: a multilabel hierarchical algorithm for gene function prediction. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases: 1st International Workshop on learning from Multi-Label Data*, pages 132–145.

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

Weigend, A., Wiener, E., and Pedersen, J. (1999). Exploiting hierarchy in text categorization. *Information Retrieval*.

William, C. et al. (1995). Fast effective rule induction. In *Twelfth International Conference on Machine Learning*, pages 115–123.

Wu, F., Zhang, J., and Honavar, V. (2005). Learning classifiers using hierarchically structured class taxonomies. In *Abstraction, Reformulation and Approximation*, pages 313–320. Springer.

Zaragoza, J. H., Sucar, L., and Morales, E. (2011a). A Two-Step Method to Learn Multidimensional Bayesian Network Classifiers Based on Mutual Information Measures. *FLAIRS Conference*, pages 644–649.

Zaragoza, J. H., Sucar, L. E., Morales, E. F., Bielza, C., and Larra, P. (2011b). Bayesian Chain Classifiers for Multidimensional Classification. *IJCAI*, 11:2192–2197.

Zhang, M.-L. and Zhou, Z.-H. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *Conference on Granular Computing, 2005 IEEE International*, volume 2, pages 718–721. IEEE.

Zhang, Y., Zincir-Heywood, N., and Milios, E. (2005). Narrative text classification for automatic key phrase extraction in web document corpora. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 51–58. ACM.

Appendix A

Base Classifier

The individual behavior for each dataset using different base classifiers along the six accuracy measures used in this thesis. The results were obtained using a ten-fold cross-validation, the mean and standard deviation along the folds is reported.

Table A.1: Accuracy percentage (mean [std]) using five different base classifiers.

Dataset	DT	C4.5	SVM	NB	RF
cellcycle_FUN	5.84 [1.21]	16.69 [2.65]	9.33 [0.99]	20.14 [1.29]	**22.24 [2.94]**
church_FUN	8.75 [3.79]	10.55 [4.35]	6.71 [2.62]	12.73 [6.34]	**14.65 [7.54]**
derisi_FUN	4.44 [1.41]	17.02 [1.62]	10.06 [2.95]	13.42 [2.14]	**19.20 [2.13]**
eisen_FUN	11.70 [2.99]	20.66 [2.80]	22.05 [2.63]	27.31 [8.88]	**29.87 [9.33]**
expr_FUN	10.47 [2.57]	17.74 [1.56]	7.49 [1.61]	23.18 [3.54]	**30.37 [1.63]**
gasch1_FUN	9.54 [2.19]	18.12 [1.58]	9.65 [2.30]	20.03 [2.89]	**29.57 [3.31]**
gasch2_FUN	8.74 [1.73]	17.33 [3.61]	9.83 [2.85]	19.27 [7.95]	**23.41 [8.33]**
pheno_FUN	15.43 [3.65]	13.03 [2.94]	13.54 [3.24]	18.20 [3.29]	**18.87 [2.62]**
seq_FUN	13.74 [5.45]	18.45 [4.72]	10.20 [7.10]	**32.74 [4.97]**	30.93 [5.95]
spo_FUN	7.00 [1.62]	16.22 [1.49]	6.90 [1.10]	15.20 [1.48]	**21.26 [2.29]**
cellcycle_GO	30.98 [1.79]	28.87 [1.63]	30.10 [2.36]	35.18 [3.04]	**38.05 [2.74]**
church_GO	31.40 [1.70]	28.86 [1.08]	30.77 [1.89]	26.57 [2.34]	**31.89 [1.57]**
derisi_GO	30.16 [2.32]	28.76 [1.10]	29.92 [2.51]	27.48 [2.57]	**35.00 [2.20]**
expr_GO	31.97 [1.84]	28.83 [1.29]	29.74 [2.12]	35.93 [2.68]	**42.25 [3.17]**
gasch1_GO	32.74 [2.36]	28.85 [1.40]	30.53 [2.35]	33.67 [3.18]	**41.93 [2.34]**
gasch2_GO	33.04 [2.28]	28.83 [1.74]	30.09 [1.90]	34.99 [1.71]	**38.88 [2.24]**
seq_GO	33.90 [1.69]	28.74 [1.59]	32.06 [2.81]	45.80 [2.30]	**49.15 [2.65]**
spo_GO	31.20 [1.93]	28.92 [1.40]	30.63 [3.23]	27.76 [2.29]	**34.97 [1.86]**

Table A.2: Hamming Accuracy percentage (mean [std]) using five different base classifiers.

Dataset	DT	C4.5	SCM	NB	RF
cellcycle_FUN	90.55 [0.19]	90.34 [0.37]	90.34 [0.22]	89.10 [0.28]	**90.90 [0.33]**
church_FUN	90.51 [0.40]	**90.54 [0.42]**	90.12 [0.19]	86.03 [2.16]	90.49 [0.77]
derisi_FUN	**90.75 [0.19]**	90.57 [0.22]	90.56 [0.24]	87.71 [0.55]	90.66 [0.31]
eisen_FUN	87.34 [0.58]	87.40 [0.98]	87.73 [0.90]	87.07 [1.48]	**88.58 [1.51]**
expr_FUN	90.71 [0.33]	90.35 [0.34]	90.49 [0.22]	88.80 [0.54]	**91.35 [0.32]**
gasch1_FUN	90.68 [0.18]	90.37 [0.22]	90.21 [0.26]	88.85 [0.43]	**91.43 [0.46]**
gasch2_FUN	90.59 [0.44]	90.32 [0.57]	90.06 [0.60]	89.01 [1.30]	**91.12 [1.27]**
pheno_FUN	**84.59 [0.68]**	84.16 [0.56]	84.27 [0.60]	83.72 [0.51]	84.32 [0.66]
seq_FUN	91.59 [0.56]	91.09 [0.55]	91.55 [0.64]	91.51 [0.73]	**92.03 [0.68]**
spo_FUN	90.64 [0.19]	90.32 [0.14]	90.27 [0.19]	88.04 [0.40]	**90.74 [0.25]**
cellcycle_GO	91.89 [0.41]	91.83 [0.36]	91.22 [0.58]	91.50 [0.59]	**92.13 [0.53]**
church_GO	**91.92 [0.28]**	91.84 [0.25]	91.32 [0.38]	90.60 [0.39]	91.51 [0.28]
derisi_GO	91.71 [0.34]	**91.82 [0.26]**	91.17 [0.33]	90.70 [0.26]	91.76 [0.31]
expr_GO	91.95 [0.30]	91.83 [0.28]	91.26 [0.35]	91.50 [0.25]	**92.48 [0.43]**
gasch1_GO	92.07 [0.17]	91.84 [0.22]	91.31 [0.34]	91.38 [0.40]	**92.53 [0.37]**
gasch2_GO	92.07 [0.37]	91.83 [0.36]	91.25 [0.31]	91.57 [0.30]	**92.22 [0.32]**
seq_GO	92.09 [0.28]	91.61 [0.31]	91.24 [0.31]	92.62 [0.32]	**93.34 [0.48]**
spo_GO	**91.95 [0.35]**	91.82 [0.34]	91.28 [0.53]	90.69 [0.37]	91.73 [0.31]

Table A.3: Exact Match percentage (mean [std]) using five different base classifiers.

Dataset	DT	C4.5	SCM	NB	RF
cellcycle_FUN	5.47 [1.07]	9.06 [3.08]	7.82 [0.90]	**17.23 [1.54]**	16.38 [2.84]
church_FUN	6.62 [2.16]	7.09 [2.36]	6.20 [2.27]	10.85 [6.23]	**11.28 [6.91]**
derisi_FUN	4.41 [1.42]	8.99 [2.16]	7.64 [1.94]	10.71 [2.09]	**13.78 [1.40]**
eisen_FUN	6.72 [2.60]	10.47 [3.00]	15.17 [2.96]	**23.44 [8.95]**	21.10 [9.82]
expr_FUN	7.72 [2.12]	9.59 [1.49]	7.13 [1.59]	19.90 [3.59]	**24.36 [1.10]**
gasch1_FUN	7.08 [1.41]	10.02 [2.31]	8.74 [2.28]	17.09 [2.89]	**23.83 [3.44]**
gasch2_FUN	6.79 [1.98]	9.51 [3.43]	8.15 [2.57]	15.75 [8.59]	**18.64 [8.92]**
pheno_FUN	14.29 [3.21]	12.74 [3.07]	13.08 [3.02]	10.68 [2.59]	**15.41 [2.80]**
seq_FUN	11.19 [5.51]	11.88 [5.09]	10.13 [7.12]	**28.18 [4.99]**	25.71 [6.02]
spo_FUN	6.34 [1.75]	8.56 [2.01]	5.65 [1.20]	12.42 [1.48]	**16.16 [2.33]**
cellcycle_GO	7.90 [2.08]	3.75 [2.24]	14.23 [2.46]	20.37 [3.63]	**21.08 [3.42]**
church_GO	8.77 [3.36]	3.74 [1.24]	**14.61 [2.27]**	11.92 [2.83]	13.50 [2.19]
derisi_GO	8.53 [4.21]	3.67 [0.95]	14.26 [2.87]	12.49 [2.94]	**17.18 [2.39]**
expr_GO	9.48 [2.51]	3.72 [0.96]	13.08 [2.38]	21.51 [3.21]	**26.22 [3.94]**
gasch1_GO	9.97 [3.51]	3.73 [1.46]	14.34 [2.88]	19.06 [3.59]	**25.82 [3.17]**
gasch2_GO	10.93 [3.51]	3.72 [1.76]	13.95 [2.80]	20.06 [2.77]	**22.21 [3.49]**
seq_GO	10.17 [2.39]	3.74 [1.49]	16.83 [3.49]	30.74 [3.41]	**33.43 [3.79]**
spo_GO	7.66 [3.23]	3.74 [1.94]	14.60 [3.36]	12.58 [2.62]	**18.16 [2.38]**

Table A.4: F1-macro D percentage (mean [std]) using five different base classifiers.

Dataset	DT	C4.5	SCM	NB	RF
cellcycle_FUN	6.03 [1.30]	20.56 [2.55]	10.08 [1.18]	21.61 [1.22]	**25.21 [3.08]**
church_FUN	9.83 [4.81]	12.31 [5.53]	6.97 [2.82]	13.81 [6.32]	**16.37 [7.93]**
derisi_FUN	4.45 [1.41]	21.10 [1.44]	11.29 [3.54]	14.91 [2.19]	**21.95 [2.60]**
eisen_FUN	14.19 [3.25]	25.76 [3.19]	25.49 [2.62]	29.34 [8.85]	**34.05 [9.14]**
expr_FUN	11.85 [2.82]	21.88 [1.66]	7.68 [1.63]	24.82 [3.57]	**33.37 [2.01]**
gasch1_FUN	10.78 [2.64]	22.23 [1.38]	10.11 [2.33]	21.52 [2.95]	**32.47 [3.28]**
gasch2_FUN	9.74 [1.70]	21.31 [3.81]	10.68 [3.04]	21.09 [7.64]	**25.81 [8.07]**
pheno_FUN	16.01 [3.93]	13.17 [2.89]	13.77 [3.37]	**21.97 [4.22]**	20.64 [2.57]
seq_FUN	15.05 [5.50]	21.86 [4.58]	10.24 [7.09]	**35.06 [4.99]**	33.57 [5.86]
spo_FUN	7.34 [1.57]	20.12 [1.51]	7.54 [1.15]	16.57 [1.52]	**23.86 [2.37]**
cellcycle_GO	43.85 [1.59]	42.67 [1.37]	40.54 [2.24]	44.86 [2.65]	**48.26 [2.44]**
church_GO	**44.10 [1.35]**	42.66 [0.98]	41.24 [1.65]	36.83 [2.01]	43.07 [1.26]
derisi_GO	42.64 [1.92]	42.57 [1.08]	40.24 [2.23]	37.76 [2.24]	**45.69 [2.04]**
expr_GO	44.55 [1.68]	42.63 [1.33]	40.51 [1.89]	45.40 [2.33]	**51.80 [2.69]**
gasch1_GO	45.37 [1.81]	42.66 [1.28]	41.05 [2.01]	43.39 [2.81]	**51.56 [2.03]**
gasch2_GO	45.44 [1.98]	42.63 [1.63]	40.63 [1.49]	44.75 [1.31]	**48.92 [1.73]**
seq_GO	46.82 [1.39]	42.53 [1.49]	42.18 [2.36]	54.76 [1.77]	**58.09 [2.19]**
spo_GO	44.26 [1.51]	42.73 [1.17]	41.08 [2.98]	38.13 [2.03]	**45.37 [1.58]**

Table A.5: F1-macro L percentage (mean [std]) using five different base classifiers.

Dataset	DT	C4.5	SCM	NB	RF
cellcycle_FUN	0.96 [0.36]	2.46 [0.45]	2.96 [0.41]	**18.12 [2.22]**	13.49 [1.82]
church_FUN	1.47 [0.90]	1.85 [0.90]	1.47 [0.79]	**7.21 [3.69]**	7.14 [5.35]
derisi_FUN	0.48 [0.15]	2.31 [0.24]	2.61 [0.73]	10.72 [1.37]	**11.98 [2.14]**
eisen_FUN	2.96 [0.51]	3.64 [0.64]	6.84 [0.95]	17.67 [3.85]	**18.24 [3.29]**
expr_FUN	2.84 [0.85]	2.58 [0.21]	2.59 [1.02]	**20.09 [4.03]**	19.48 [1.60]
gasch1_FUN	2.32 [0.59]	2.70 [0.32]	3.20 [0.97]	16.08 [2.75]	**19.22 [3.13]**
gasch2_FUN	1.99 [0.38]	2.49 [0.55]	3.15 [1.02]	**10.75 [3.41]**	8.84 [3.58]
pheno_FUN	3.71 [1.81]	1.61 [0.36]	2.28 [0.81]	7.60 [1.58]	**11.68 [2.99]**
seq_FUN	3.16 [0.95]	2.86 [0.96]	1.67 [1.42]	**21.28 [3.36]**	17.54 [3.08]
spo_FUN	1.55 [0.42]	2.27 [0.24]	2.00 [0.40]	12.34 [1.30]	**13.09 [1.54]**
cellcycle_GO	5.72 [0.59]	3.33 [0.11]	4.73 [0.40]	**12.98 [2.18]**	11.30 [1.26]
church_GO	5.42 [0.78]	3.33 [0.07]	3.34 [0.26]	6.14 [0.67]	**8.77 [1.25]**
derisi_GO	5.91 [0.71]	3.30 [0.05]	3.68 [0.58]	8.05 [1.72]	**10.51 [1.29]**
expr_GO	6.37 [0.67]	3.33 [0.06]	5.50 [0.86]	15.93 [3.73]	**16.40 [3.00]**
gasch1_GO	6.61 [0.88]	3.33 [0.11]	5.18 [0.70]	12.78 [2.59]	**16.51 [1.78]**
gasch2_GO	6.71 [0.72]	3.33 [0.10]	4.51 [0.61]	12.17 [1.25]	**13.44 [1.73]**
seq_GO	6.38 [1.00]	3.40 [0.10]	7.85 [1.60]	24.73 [3.99]	**24.76 [4.41]**
spo_GO	5.87 [0.75]	3.34 [0.06]	4.24 [0.69]	7.52 [2.00]	**9.89 [2.14]**

Table A.6: H-loss (mean [std]) using five different base classifiers.

Dataset	DT	C4.5	SCM	NB	RF
cellcycle_FUN	1.89 [0.02]	1.82 [0.06]	1.84 [0.02]	1.66 [0.03]	1.67 [0.06]
church_FUN	1.87 [0.04]	1.86 [0.05]	1.88 [0.05]	1.78 [0.12]	**1.77 [0.14]**
derisi_FUN	1.91 [0.03]	1.82 [0.04]	1.85 [0.04]	1.79 [0.04]	**1.72 [0.03]**
eisen_FUN	1.87 [0.05]	1.79 [0.06]	1.70 [0.06]	**1.53 [0.18]**	1.52 [0.27]
expr_FUN	1.85 [0.04]	1.81 [0.03]	1.86 [0.03]	1.60 [0.07]	**1.51 [0.02]**
gasch1_FUN	1.86 [0.03]	1.80 [0.05]	1.83 [0.05]	1.66 [0.06]	**1.52 [0.07]**
gasch2_FUN	1.86 [0.04]	1.81 [0.07]	1.84 [0.05]	1.69 [0.17]	**1.63 [0.18]**
pheno_FUN	1.71 [0.06]	1.75 [0.06]	1.74 [0.06]	1.79 [0.05]	**1.69 [0.06]**
seq_FUN	1.78 [0.11]	1.76 [0.10]	1.80 [0.14]	**1.44 [0.10]**	1.49 [0.12]
spo_FUN	1.87 [0.04]	1.83 [0.04]	1.89 [0.02]	1.75 [0.03]	**1.68 [0.05]**
cellcycle_GO	1.87 [0.05]	1.95 [0.05]	1.75 [0.06]	1.65 [0.07]	**1.62 [0.07]**
church_GO	1.86 [0.08]	1.96 [0.03]	**1.74 [0.05]**	1.82 [0.07]	1.77 [0.05]
derisi_GO	1.86 [0.09]	1.96 [0.02]	1.75 [0.05]	1.81 [0.06]	**1.69 [0.05]**
expr_GO	1.84 [0.05]	1.96 [0.03]	1.78 [0.05]	1.63 [0.06]	**1.51 [0.08]**
gasch1_GO	1.83 [0.07]	1.96 [0.03]	1.76 [0.06]	1.68 [0.08]	**1.52 [0.07]**
gasch2_GO	1.81 [0.08]	1.96 [0.04]	1.76 [0.06]	1.65 [0.05]	**1.59 [0.07]**
seq_GO	1.82 [0.04]	1.95 [0.03]	1.71 [0.06]	1.43 [0.07]	**1.36 [0.08]**
spo_GO	1.88 [0.07]	1.96 [0.05]	1.74 [0.06]	1.80 [0.06]	**1.68 [0.05]**

Appendix B

Weighting Scheme

B.1 RSM vs NW scheme

The individual behavior for each dataset using a RSM and non weighted scheme along the six accuracy measures used in this thesis. We performed a paired t-test with a confidence level of 95%.

Table B.1: Accuracy percentage (mean[std]) using the proposed weighting scheme against a non weighted approach, statistically superior results compared with RSM scheme are marked with ↑ and statistically inferior with ↓.

(a) Tree datasets

Dataset	RSM	NW
cellcycle_FUN	**22.39 [2.77]**	20.20 [1.57]↓
church_FUN	**14.72 [8.50]**	13.36 [7.29]↓
derisi_FUN	**19.37 [2.01]**	18.38 [2.43]
eisen_FUN	**29.92 [9.74]**	27.92 [10.32]↓
expr_FUN	**29.73 [2.28]**	27.23 [2.81]↓
gasch1_FUN	**28.16 [2.31]**	26.50 [2.73]↓
gasch2_FUN	**22.62 [7.81]**	20.90 [8.08]↓
pheno_FUN	**18.02 [2.63]**	17.45 [2.19]
seq_FUN	**30.54 [5.76]**	29.66 [4.92]
spo_FUN	**21.11 [1.51]**	19.39 [2.11]↓

(b) DAG datasets

Dataset	RSM	NW
cellcycle_GO	**37.19 [2.97]**	36.84 [3.63]
church_GO	31.62 [1.18]	**32.57 [2.25]**
derisi_GO	34.35 [1.62]	**34.55 [1.84]**
expr_GO	**42.35 [1.70]**	41.76 [1.90]
gasch1_GO	**42.13 [3.67]**	41.57 [3.64]
gasch2_GO	**39.34 [2.71]**	38.57 [1.49]
seq_GO	47.86 [2.73]	**48.13 [2.78]**
spo_GO	**34.97 [1.68]**	33.61 [1.70]↓

Table B.2: Hamming Accuracy percentage (mean[std]) using the proposed weighting scheme against a non weighted approach, statistically superior results compared with RSM scheme are marked with ↑ and statistically inferior with ↓.

(a) Tree datasets

Dataset	RSM	NW
cellcycle_FUN	90.90 [0.39]	**91.06 [0.25]**↑
church_FUN	90.56 [0.79]	**90.66 [0.70]**
derisi_FUN	90.71 [0.25]	**90.91 [0.32]**↑
eisen_FUN	88.57 [1.63]	**88.67 [1.63]**
expr_FUN	91.30 [0.31]	**91.39 [0.43]**
gasch1_FUN	91.18 [0.43]	**91.44 [0.35]**↑
gasch2_FUN	91.03 [1.28]	**91.15 [1.19]**
pheno_FUN	84.09 [0.57]	**84.23 [0.55]**
seq_FUN	91.99 [0.66]	**92.27 [0.52]**↑
spo_FUN	90.76 [0.16]	**90.98 [0.30]**↑

(b) DAG datasets

Dataset	RSM	NW
cellcycle_GO	92.06 [0.53]	**92.09 [0.70]**
church_GO	91.48 [0.14]	**91.56 [0.41]**
derisi_GO	91.65 [0.25]	**91.74 [0.22]**
expr_GO	92.54 [0.27]	**92.55 [0.27]**
gasch1_GO	92.52 [0.56]	**92.55 [0.54]**
gasch2_GO	**92.27 [0.43]**	92.24 [0.36]
seq_GO	93.17 [0.53]	**93.25 [0.57]**
spo_GO	**91.81 [0.37]**	91.67 [0.35]↓

Table B.3: Exact Match percentage (mean[std]) using the proposed weighting scheme against a non weighted approach, statistically superior results compared with RSM scheme are marked with ↑ and statistically inferior with ↓.

(a) Tree datasets

Dataset	RSM	NW
cellcycle_FUN	**16.63 [2.59]**	15.65 [1.58]↓
church_FUN	**11.50 [7.73]**	10.51 [6.40]
derisi_FUN	**14.24 [2.33]**	13.61 [2.24]
eisen_FUN	**22.43 [9.86]**	21.36 [9.71]
expr_FUN	**23.51 [2.39]**	21.90 [2.70]↓
gasch1_FUN	**22.64 [2.61]**	21.19 [2.93]↓
gasch2_FUN	**17.57 [8.17]**	16.72 [8.41]
pheno_FUN	**14.81 [2.75]**	14.46 [2.49]
seq_FUN	25.34 [6.19]	**25.50 [5.25]**
spo_FUN	**15.42 [1.55]**	14.68 [1.66]

(b) DAG datasets

Dataset	RSM	NW
cellcycle_GO	**20.03 [3.26]**	19.50 [3.88]
church_GO	13.21 [1.94]	**14.61 [2.41]**↑
derisi_GO	**17.01 [2.08]**	16.78 [2.11]
expr_GO	**26.34 [2.66]**	25.76 [2.64]
gasch1_GO	**26.11 [4.41]**	25.82 [4.13]
gasch2_GO	**22.85 [3.69]**	21.57 [1.95]
seq_GO	32.08 [2.74]	**32.67 [3.77]**
spo_GO	**17.39 [1.89]**	16.26 [2.47]

Table B.4: F1-macro D percentage (mean[std]) using the proposed weighting scheme against a non weighted approach, statistically superior results compared with RSM scheme are marked with ↑ and statistically inferior with ↓.

(a) Tree datasets

Dataset	RSM	NW
cellcycle_FUN	**25.28 [2.97]**	22.50 [1.63]↓
church_FUN	**16.35 [8.96]**	14.81 [7.91]↓
derisi_FUN	**21.98 [2.06]**	20.80 [2.61]
eisen_FUN	**33.68 [9.80]**	31.20 [10.72]↓
expr_FUN	**32.84 [2.34]**	29.91 [2.94]↓
gasch1_FUN	**30.98 [2.17]**	29.21 [2.66]↓
gasch2_FUN	**25.18 [7.64]**	23.02 [7.97]↓
pheno_FUN	**19.66 [2.66]**	18.95 [2.21]
seq_FUN	**33.19 [5.54]**	31.82 [4.77]↓
spo_FUN	**23.96 [1.65]**	21.75 [2.36]↓

(b) DAG datasets

Dataset	RSM	NW
cellcycle_GO	**47.56 [2.75]**	47.29 [3.35]
church_GO	42.85 [0.97]	**43.55 [2.23]**
derisi_GO	44.99 [1.40]	**45.26 [1.66]**
expr_GO	**51.92 [1.32]**	51.38 [1.60]
gasch1_GO	**51.68 [3.22]**	51.12 [3.24]
gasch2_GO	**49.28 [2.33]**	48.74 [1.45]
seq_GO	56.97 [2.59]	**57.06 [2.39]**
spo_GO	**45.65 [1.56]**	44.29 [1.39]↓

Table B.5: F1-macro L percentage (mean[std]) using the proposed weighting scheme against a non weighted approach, statistically superior results compared with RSM scheme are marked with ↑ and statistically inferior with ↓.

(a) Tree datasets

Dataset	RSM	NW
cellcycle_FUN	**14.06 [1.97]**	12.08 [1.34]↓
church_FUN	**7.99 [7.49]**	5.84 [3.81]
derisi_FUN	**11.76 [1.84]**	10.54 [1.43]↓
eisen_FUN	**18.33 [3.69]**	16.30 [4.13]↓
expr_FUN	**18.18 [1.51]**	15.20 [1.50]↓
gasch1_FUN	**17.25 [2.79]**	15.37 [2.61]
gasch2_FUN	**8.83 [4.46]**	7.59 [4.37]↓
pheno_FUN	**12.21 [3.52]**	10.32 [3.36]↓
seq_FUN	**17.37 [3.12]**	15.26 [2.58]↓
spo_FUN	**12.27 [1.12]**	11.22 [1.91]

(b) DAG datasets

Dataset	RSM	NW
cellcycle_GO	**11.82 [1.58]**	10.03 [1.26]↓
church_GO	**8.71 [1.24]**	8.68 [0.96]
derisi_GO	9.91 [0.99]	**10.13 [1.28]**
expr_GO	**16.59 [2.15]**	15.87 [2.17]
gasch1_GO	**16.07 [2.54]**	16.01 [3.15]
gasch2_GO	**13.19 [1.53]**	12.08 [1.65]
seq_GO	23.59 [4.46]	**23.69 [4.17]**
spo_GO	**10.29 [1.29]**	8.94 [1.01]↓

Table B.6: H-loss (mean[std]) using the proposed weighting scheme against a non weighted approach, statistically superior results compared with RSM scheme are marked with ↑ and statistically inferior with ↓.

(a) Tree datasets

Dataset	RSM	NW
cellcycle_FUN	**1.667 [0.05]**	1.687 [0.03]↓
church_FUN	**1.770 [0.15]**	1.790 [0.13]
derisi_FUN	**1.715 [0.05]**	1.728 [0.04]
eisen_FUN	**1.551 [0.20]**	1.573 [0.19]
expr_FUN	**1.530 [0.05]**	1.562 [0.05]↓
gasch1_FUN	**1.547 [0.05]**	1.576 [0.06]↓
gasch2_FUN	**1.649 [0.16]**	1.666 [0.17]
pheno_FUN	**1.704 [0.05]**	1.711 [0.05]
seq_FUN	1.493 [0.12]	**1.490 [0.11]**
spo_FUN	**1.692 [0.03]**	1.706 [0.03]

(b) DAG datasets

Dataset	RSM	NW
cellcycle_GO	**1.634 [0.07]**	1.645 [0.08]
church_GO	1.781 [0.04]	**1.748 [0.06]**↑
derisi_GO	1.702 [0.04]	**1.701 [0.04]**
expr_GO	**1.505 [0.05]**	1.517 [0.05]
gasch1_GO	**1.514 [0.09]**	1.522 [0.08]
gasch2_GO	**1.583 [0.08]**	1.610 [0.05]
seq_GO	1.381 [0.06]	**1.377 [0.07]**
spo_GO	**1.694 [0.04]**	1.717 [0.05]

B.2 RSM vs other weighting schemes

The individual behavior for each dataset using a RSM and other weighting schemes along the six accuracy measures used in this thesis. Median and IQR are reported. The best results are marked with green, the worst with red and the rest in yellow. We performed a Friedman test with a confidence level of 95% using Tukey's honestly significant difference criterion.

Table B.7: Accuracy (median [IQR]) percentage using the proposed weighting scheme against other weighting schemes. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	RSM	V	BK
cellcycle_FUN	22.52 [3.24]	23.20 [4.05]	24.55 [0.72]
church_FUN	12.26 [10.19]	12.02 [10.43]	12.82 [7.80]
derisi_FUN	19.18 [2.46]	19.41 [4.59]	22.07 [2.17]↑
eisen_FUN	28.19 [10.12]	26.07 [9.25]	29.19 [8.06]
expr_FUN	29.58 [2.84]	29.45 [3.79]	29.62 [4.29]
gasch1_FUN	28.26 [3.05]	28.24 [3.34]	30.40 [4.11]
gasch2_FUN	21.18 [9.26]	19.96 [9.08]	24.45 [5.09]
pheno_FUN	18.46 [2.01]	19.11 [3.25]	19.96 [2.73]
seq_FUN	29.15 [7.11]	28.96 [6.67]	29.96 [7.70]
spo_FUN	20.96 [1.64]	19.88 [2.75]	24.46 [3.91]↑

(b) DAG datasets

Dataset	RSM	V	BK
cellcycle_GO	36.93 [3.84]	37.48 [5.42]	35.60 [3.06]↓
church_GO	31.49 [1.50]	31.93 [1.78]	31.76 [1.87]
derisi_GO	34.50 [2.50]	34.39 [4.68]	33.44 [2.26]
expr_GO	42.96 [3.22]	42.70 [2.53]	39.09 [2.98]↓
gasch1_GO	42.69 [5.06]	42.82 [3.65]	39.66 [3.56]
gasch2_GO	38.77 [3.61]	40.75 [3.45]	37.73 [2.91]
seq_GO	47.61 [4.33]	48.80 [2.04]	45.53 [5.24]
spo_GO	34.96 [3.57]	34.60 [1.68]	34.34 [1.59]

Table B.8: Hamming Accuracy (median [IQR]) percentage using the proposed weighting scheme against other weighting schemes. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	RSM	V	BK
cellcycle_FUN	90.89 [0.44]	91.16 [0.33]	90.91 [0.40]
church_FUN	90.47 [1.06]	90.38 [1.13]	90.22 [0.61]
derisi_FUN	90.68 [0.30]	90.73 [0.37]	90.76 [0.20]
eisen_FUN	88.32 [1.50]	88.35 [1.52]	88.52 [1.21]
expr_FUN	91.30 [0.26]	91.17 [0.65]	91.29 [0.49]
gasch1_FUN	91.20 [0.54]	91.34 [0.63]	91.41 [0.81]
gasch2_FUN	90.89 [0.86]	90.95 [1.31]	90.92 [1.18]
pheno_FUN	84.18 [0.61]	84.53 [0.72]	84.28 [0.41]
seq_FUN	91.91 [0.58]	91.86 [0.44]	91.78 [0.56]
spo_FUN	90.71 [0.24]	90.69 [0.10]	90.90 [0.52]

(b) DAG datasets

Dataset	RSM	V	BK
cellcycle_GO	92.05 [0.66]	91.89 [0.97]	91.46 [0.66]↓
church_GO	91.45 [0.27]	91.49 [0.67]	91.26 [0.57]↓
derisi_GO	91.65 [0.31]	91.71 [0.36]	90.83 [0.41]↓
expr_GO	92.53 [0.40]	92.40 [0.49]	91.50 [0.74]↓
gasch1_GO	92.50 [0.68]	92.49 [0.69]	91.74 [0.58]↓
gasch2_GO	92.25 [0.49]	92.53 [0.44]	91.74 [0.57]↓
seq_GO	93.14 [0.87]	93.27 [0.79]	92.32 [0.79]↓
spo_GO	91.90 [0.46]	91.67 [0.25]	91.15 [0.30]↓

Table B.9: Exact Match (median [IQR]) percentage using the proposed weighting scheme against other weighting schemes. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	RSM	V	BK
cellcycle_FUN	16.45 [5.13]	16.70 [4.70]	16.03 [1.71]
church_FUN	7.91 [9.83]	8.33 [9.83]	7.48 [5.98]
derisi_FUN	14.08 [2.94]	14.47 [4.20]	14.29 [2.10]
eisen_FUN	19.05 [10.21]	18.45 [10.12]	18.45 [7.74]
expr_FUN	23.35 [3.81]	23.36 [4.24]	21.82 [3.91]
gasch1_FUN	23.19 [3.83]	22.77 [2.56]	21.96 [5.02]
gasch2_FUN	15.47 [11.49]	14.41 [10.69]	16.77 [4.32]
pheno_FUN	15.09 [2.59]	15.52 [2.59]	14.66 [3.45]
seq_FUN	23.94 [8.50]	23.17 [6.48]	24.09 [9.43]
spo_FUN	15.00 [1.30]	14.53 [2.61]	16.09 [3.48]

(b) DAG datasets

Dataset	RSM	V	BK
cellcycle_GO	19.35 [4.68]	19.94 [5.39]	15.25 [2.34]↓
church_GO	13.45 [0.58]	13.45 [3.51]	12.87 [2.92]
derisi_GO	17.71 [3.45]	16.86 [5.06]	12.64 [2.86]↓
expr_GO	27.33 [4.65]	26.74 [2.91]	20.06 [4.65]↓
gasch1_GO	26.74 [6.84]	26.82 [5.85]	21.51 [4.22]↓
gasch2_GO	21.80 [3.49]	23.84 [4.07]	18.90 [4.07]↓
seq_GO	31.87 [4.68]	33.24 [4.09]	27.49 [6.43]↓
spo_GO	17.80 [2.96]	17.21 [3.66]	13.35 [2.89]↓

Table B.10: F1-macro D (median [IQR]) percentage using the proposed weighting scheme against other weighting schemes. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	RSM	V	BK
cellcycle_FUN	25.71 [3.36]	26.06 [3.92]	28.79 [0.56]↑
church_FUN	14.81 [10.94]	14.10 [10.78]	16.09 [9.62]
derisi_FUN	21.67 [3.06]	21.94 [4.70]	26.12 [1.82]↑
eisen_FUN	32.04 [10.71]	30.51 [10.17]	34.57 [8.93]
expr_FUN	32.70 [2.21]	32.55 [4.18]	33.37 [4.33]
gasch1_FUN	30.89 [3.28]	31.39 [4.11]	34.48 [3.64]↑
gasch2_FUN	24.00 [8.07]	22.64 [7.74]	28.00 [4.93]↑
pheno_FUN	20.60 [2.33]	20.99 [2.67]	22.66 [2.84]↑
seq_FUN	31.99 [6.19]	31.94 [6.62]	33.01 [6.84]
spo_FUN	23.87 [2.62]	22.56 [2.70]	28.62 [4.22]↑

(b) DAG datasets

Dataset	RSM	V	BK
cellcycle_GO	47.32 [2.79]	47.82 [4.73]	47.03 [2.70]
church_GO	42.76 [1.11]	43.40 [1.53]	43.23 [1.49]
derisi_GO	44.71 [1.73]	45.05 [3.42]	44.87 [2.09]
expr_GO	52.21 [2.50]	52.13 [2.22]	49.54 [2.76]
gasch1_GO	52.11 [4.56]	52.26 [2.82]	50.39 [3.49]
gasch2_GO	49.05 [3.02]	50.62 [3.29]	48.82 [2.51]
seq_GO	56.58 [3.46]	57.67 [1.91]	55.38 [4.23]
spo_GO	45.66 [2.75]	45.30 [1.48]	45.99 [1.30]

Table B.11: F1-macro L (median [IQR]) percentage using the proposed weighting scheme against other weighting schemes. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	RSM	V	BK
cellcycle_FUN	14.26 [1.76]	13.78 [2.03]	11.53 [1.66]↓
church_FUN	6.16 [5.14]	6.66 [5.66]	5.47 [3.56]
derisi_FUN	11.56 [3.08]	10.70 [1.51]	10.64 [0.78]
eisen_FUN	19.33 [3.41]	17.02 [5.37]	13.47 [7.30]↓
expr_FUN	17.68 [2.00]	17.17 [4.28]	15.18 [1.82]↓
gasch1_FUN	17.14 [2.67]	16.94 [2.51]	16.28 [3.45]
gasch2_FUN	7.56 [7.39]	6.54 [6.50]	8.21 [3.94]
pheno_FUN	12.62 [4.55]	12.77 [2.08]	10.54 [2.76]↓
seq_FUN	16.11 [3.05]	15.04 [3.98]	15.39 [1.81]↓
spo_FUN	12.14 [1.73]	12.38 [1.33]	11.41 [1.92]

(b) DAG datasets

Dataset	RSM	V	BK
cellcycle_GO	11.92 [1.39]	10.91 [1.82]	11.24 [3.11]
church_GO	8.75 [1.93]	9.47 [1.20]	9.67 [1.34]
derisi_GO	9.86 [1.36]	10.92 [1.44]	10.32 [1.95]
expr_GO	16.53 [2.81]	17.15 [2.88]	14.45 [4.73]
gasch1_GO	15.67 [3.43]	16.60 [2.53]	15.84 [3.47]
gasch2_GO	12.63 [1.93]	14.55 [2.73]	13.75 [0.89]
seq_GO	21.82 [3.61]	24.51 [3.15]	20.09 [3.18]
spo_GO	10.25 [1.84]	10.11 [2.22]	10.88 [2.70]

Table B.12: H-loss (median [IQR]) using the proposed weighting scheme against other weighting schemes. Statistically inferior results against CPE are marked with ↑ and statistically superior results are marked with ↓.

(a) Tree datasets

Dataset	RSM	V	BK
cellcycle_FUN	1.671 [0.1]	1.666 [0.09]	1.68 [0.03]
church_FUN	1.842 [0.2]	1.833 [0.2]	1.85 [0.12]
derisi_FUN	1.719 [0.06]	1.711 [0.08]	1.714 [0.04]
eisen_FUN	1.619 [0.2]	1.631 [0.2]	1.631 [0.15]
expr_FUN	1.533 [0.08]	1.533 [0.08]	1.564 [0.08]
gasch1_FUN	1.536 [0.08]	1.545 [0.05]	1.561 [0.1]
gasch2_FUN	1.691 [0.23]	1.712 [0.21]	1.665 [0.09]
pheno_FUN	1.698 [0.05]	1.69 [0.05]	1.707 [0.07]
seq_FUN	1.521 [0.17]	1.537 [0.13]	1.518 [0.19]
spo_FUN	1.700 [0.03]	1.709 [0.05]	1.678 [0.07]

(b) DAG datasets

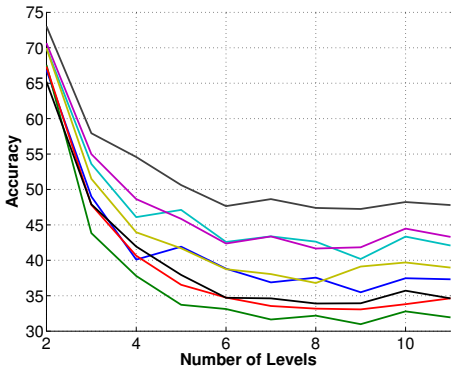
Dataset	RSM	V	BK
cellcycle_GO	1.634 [0.11]	1.635 [0.15]	1.721 [0.07]
church_GO	1.763 [0.05]	1.775 [0.08]	1.781 [0.06]
derisi_GO	1.693 [0.04]	1.703 [0.07]	1.771 [0.04]
expr_GO	1.483 [0.09]	1.494 [0.06]	1.625 [0.08]↓
gasch1_GO	1.513 [0.14]	1.494 [0.13]	1.599 [0.11]
gasch2_GO	1.602 [0.06]	1.57 [0.12]	1.660 [0.10]
seq_GO	1.386 [0.11]	1.376 [0.07]	1.471 [0.13]↓
spo_GO	1.68 [0.08]	1.701 [0.08]	1.775 [0.03]↓

Appendix C

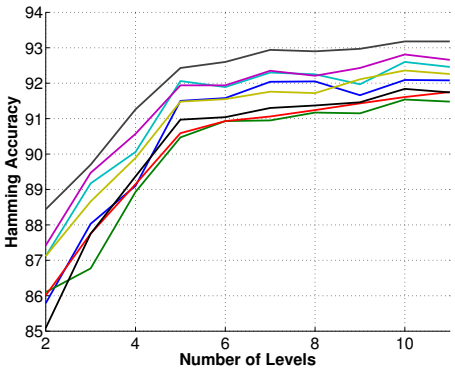
Hierarchy depth effect over classification performance

The individual behavior for each dataset along hierarchies with different levels evaluated by four different classifiers, are described in this section. We report the six accuracy measures used in this thesis, a ten-cross fold-validation for each dataset was performed per level, the mean per level is reported in the plots. Figure C.1 depicts the behavior of CPE, Figure C.2 of TD, Figure C.3 of TD-C and Figure C.4 of HIROM.

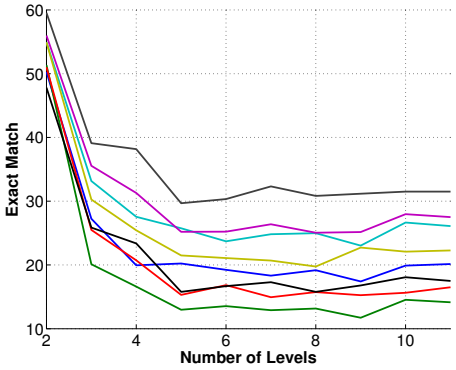
(a) Legend



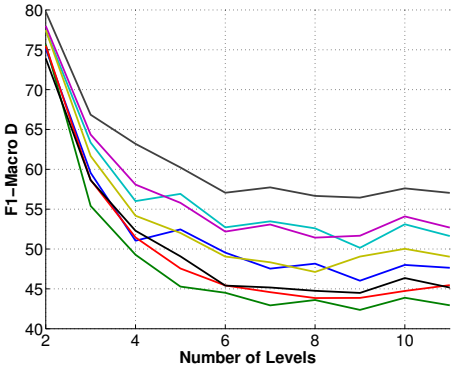
(b) Accuracy



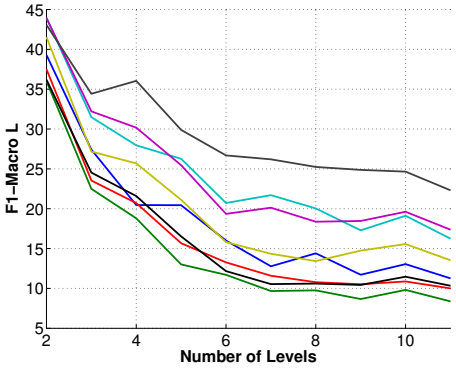
(c) Hamming Accuracy



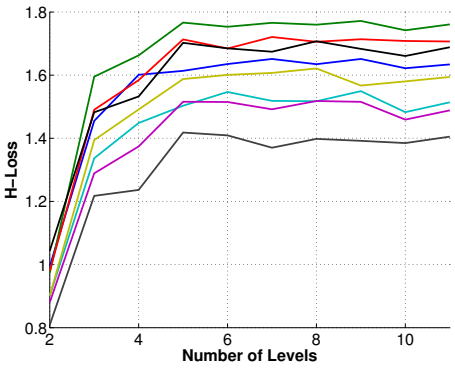
(d) Exact Match



(e) F1-macro D

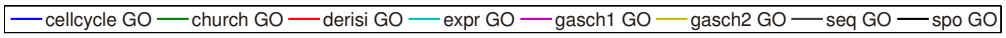


(f) F1-macro L

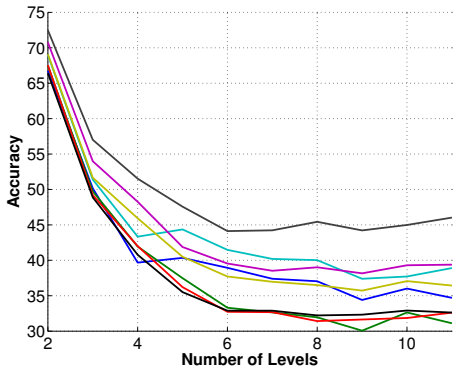


(g) H-loss

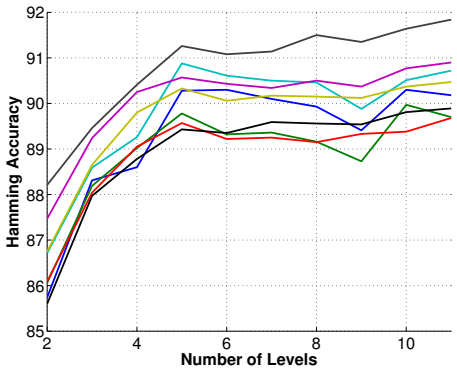
Figure C.1: Plots of the performance of CPE with hierarchies of different depths.



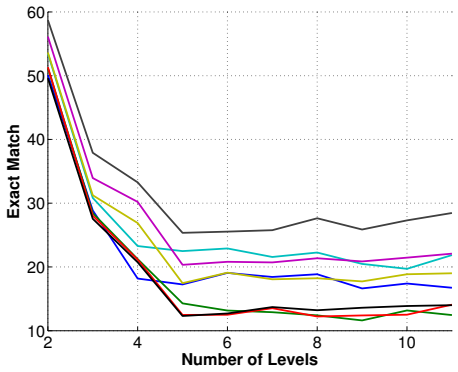
(a) Legend



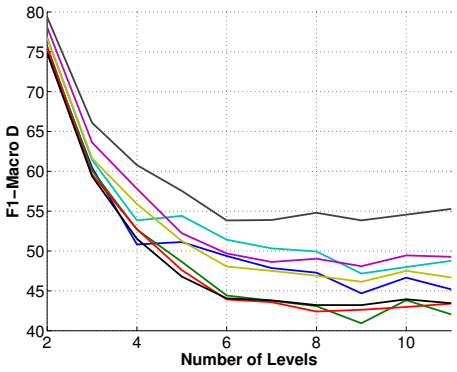
(b) Accuracy



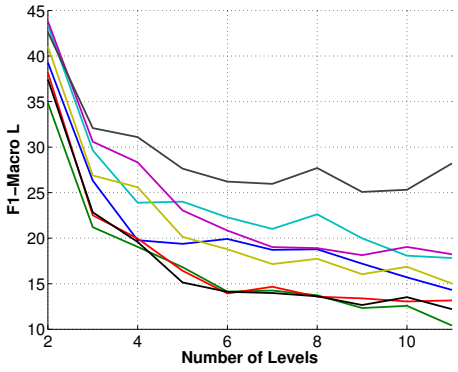
(c) Hamming Accuracy



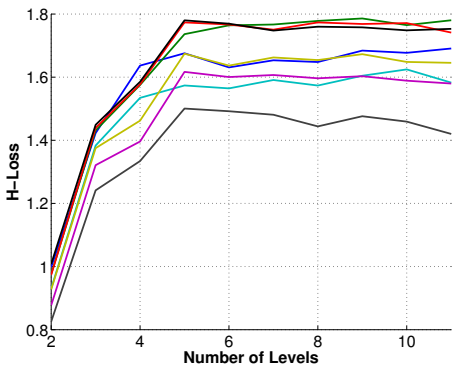
(d) Exact Match



(e) F1-macro D



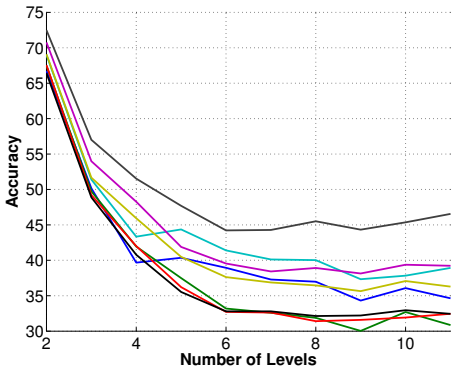
(f) F1-macro L



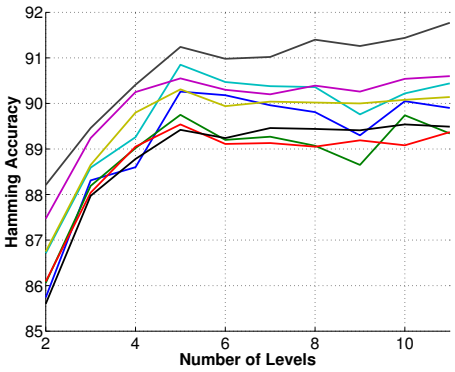
(g) H-loss

Figure C.2: Plots of the performance of TD with hierarchies of different depths.

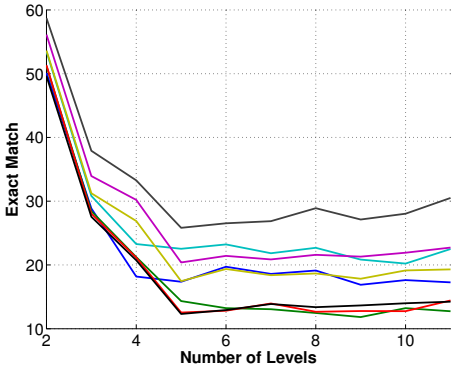
(a) Legend



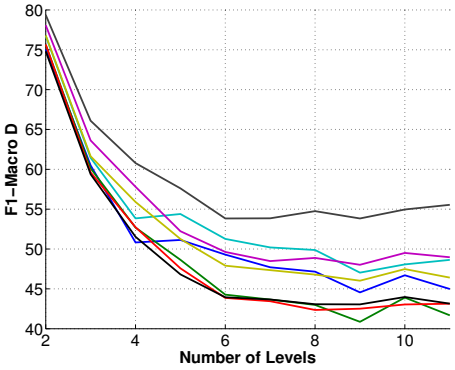
(b) Accuracy



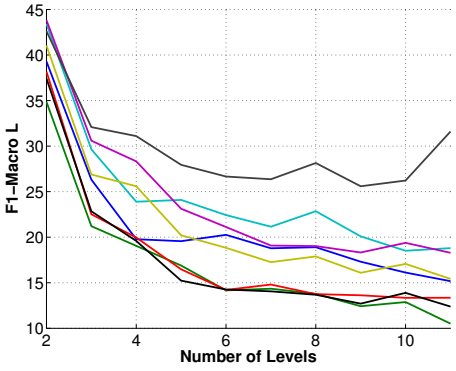
(c) Hamming Accuracy



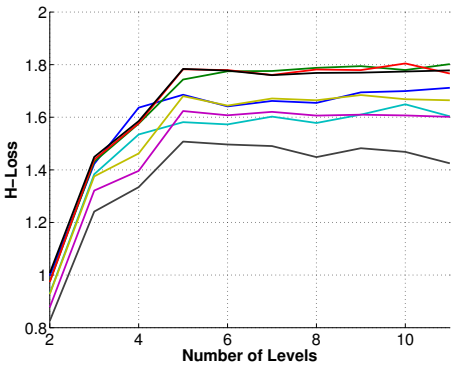
(d) Exact Match



(e) F1-macro D

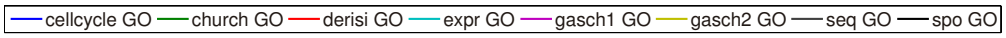


(f) F1-macro L

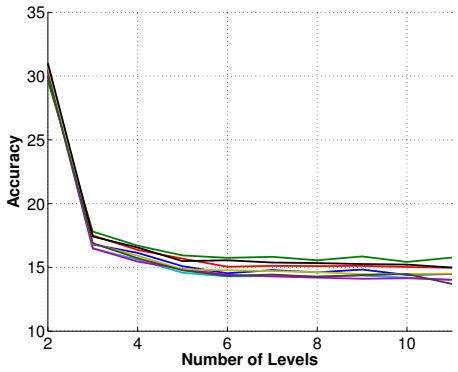


(g) H-loss

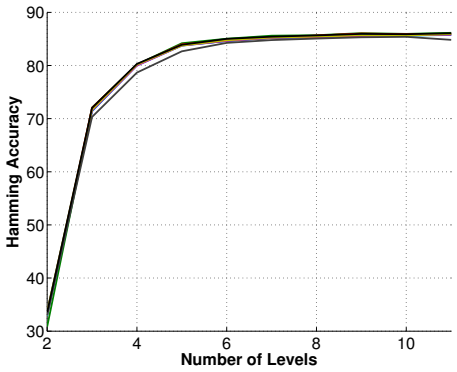
Figure C.3: Plots of the performance of TD-C with hierarchies of different depths.



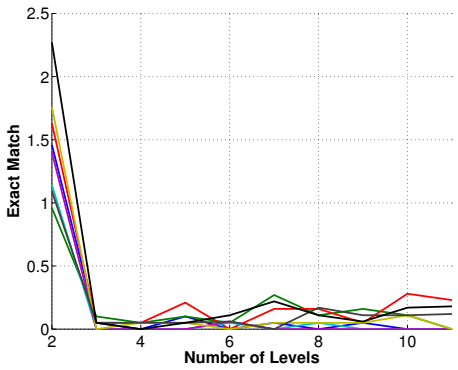
(a) Legend



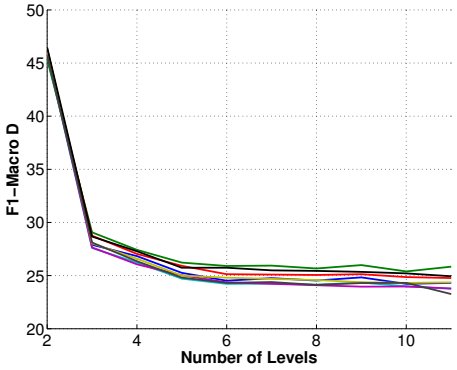
(b) Accuracy



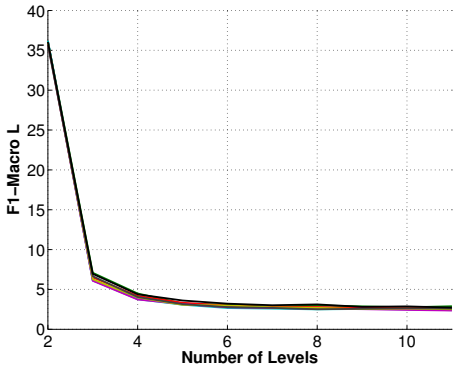
(c) Hamming Accuracy



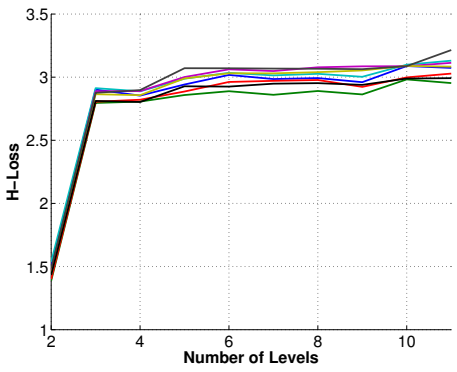
(d) Exact Match



(e) F1-macro D



(f) F1-macro L



(g) H-loss

Figure C.4: Plots of the performance of HIROM with hierarchies of different depths.

Appendix D

Comparison of MLNP approaches

The results are depicted in Sub-sections Section D.1 and D.2. The central mark of the plots is the median, the edges of the box are the 25th and 75th percentiles, the whiskers (lines) extend to the most extreme data points not considered outliers, and outliers are plotted individually and marked with a circle.

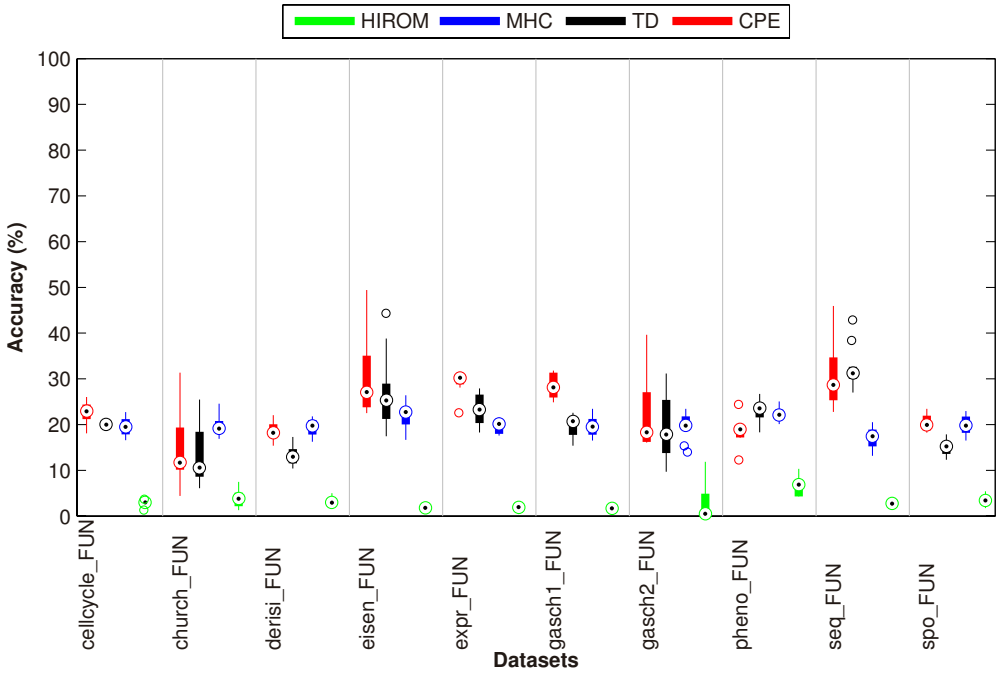
D.1 Tree Structured Datasets

For tree structured hierarchies, we used ten hierarchical datasets and compared CPE against four HMC methods:

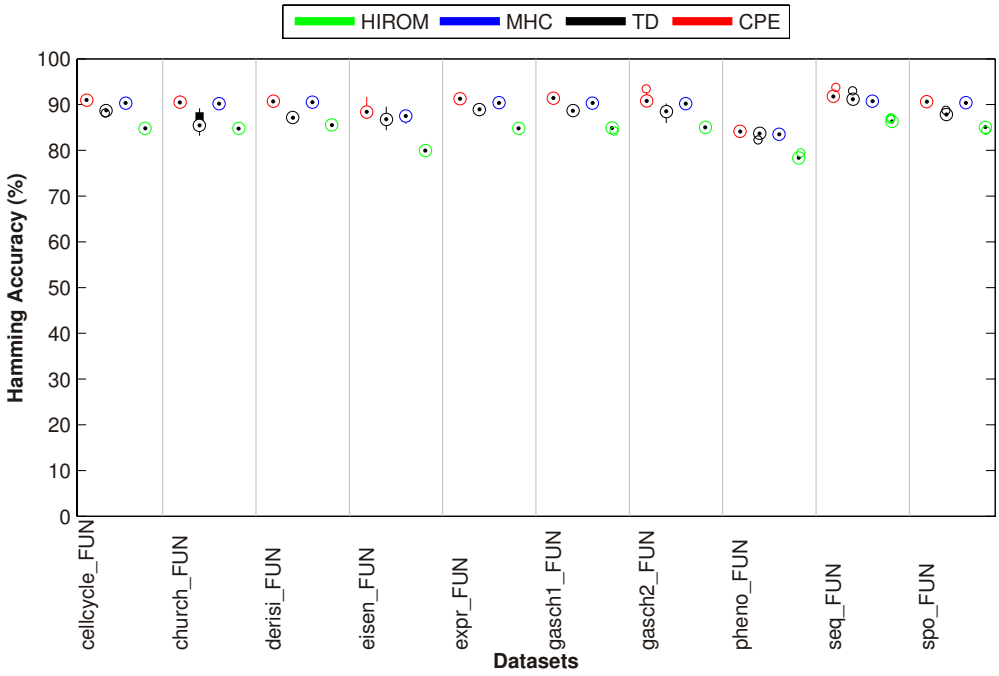
1. Top-Down LCPN (TD).

2. Multidimensional Hierarchical Classifier (MHC).

3. HIROM.

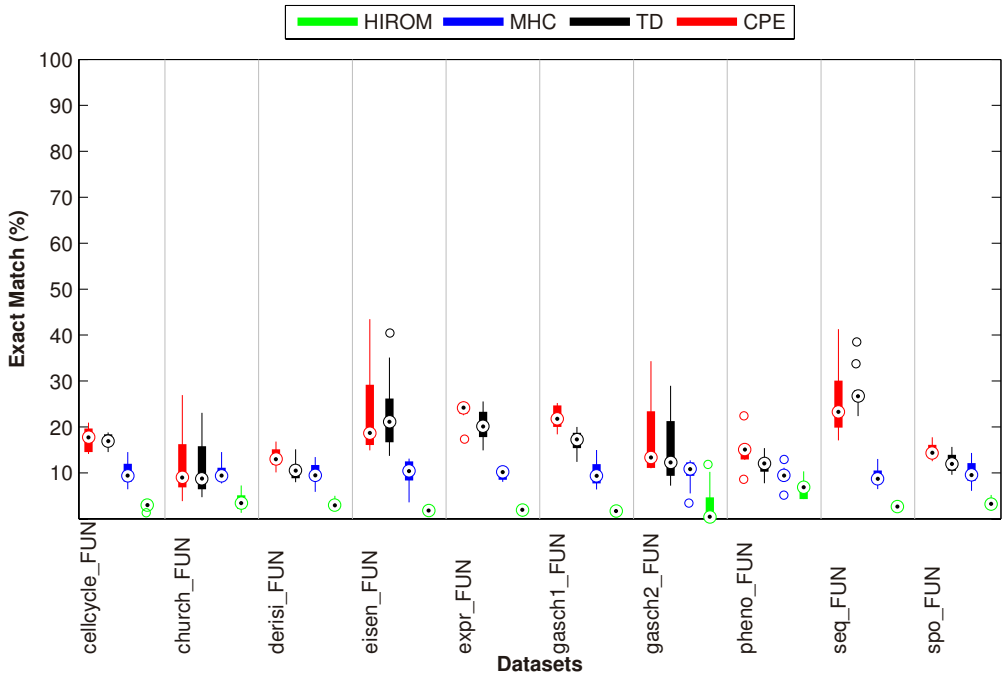


(a) Accuracy (%)

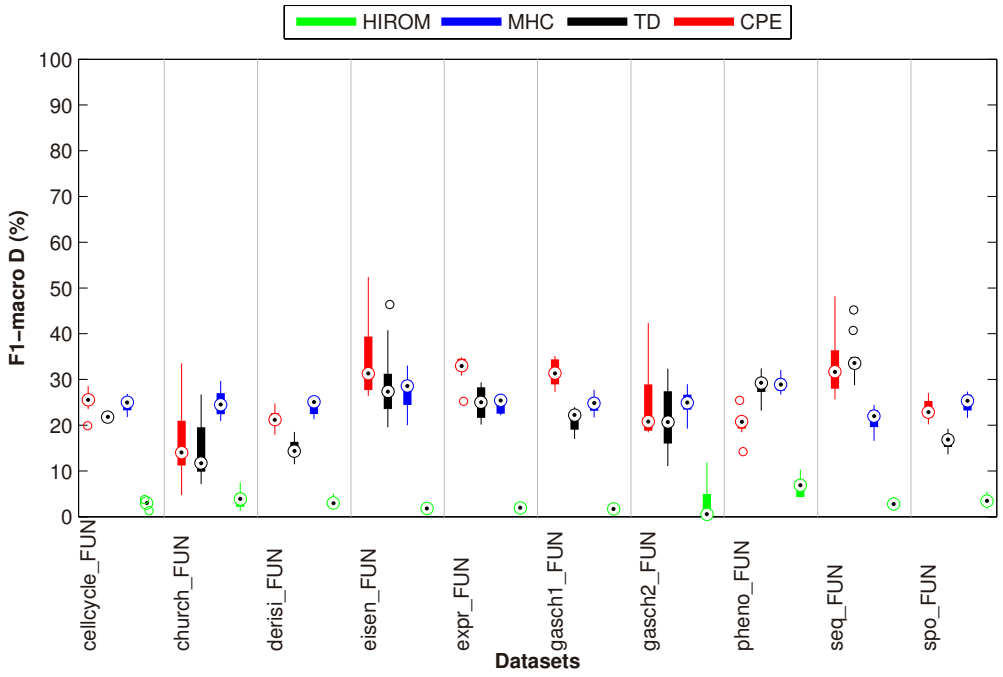


(b) Hamming Accuracy (%)

Figure D.1: Comparison of CPE against other MLNP methods for tree datasets.

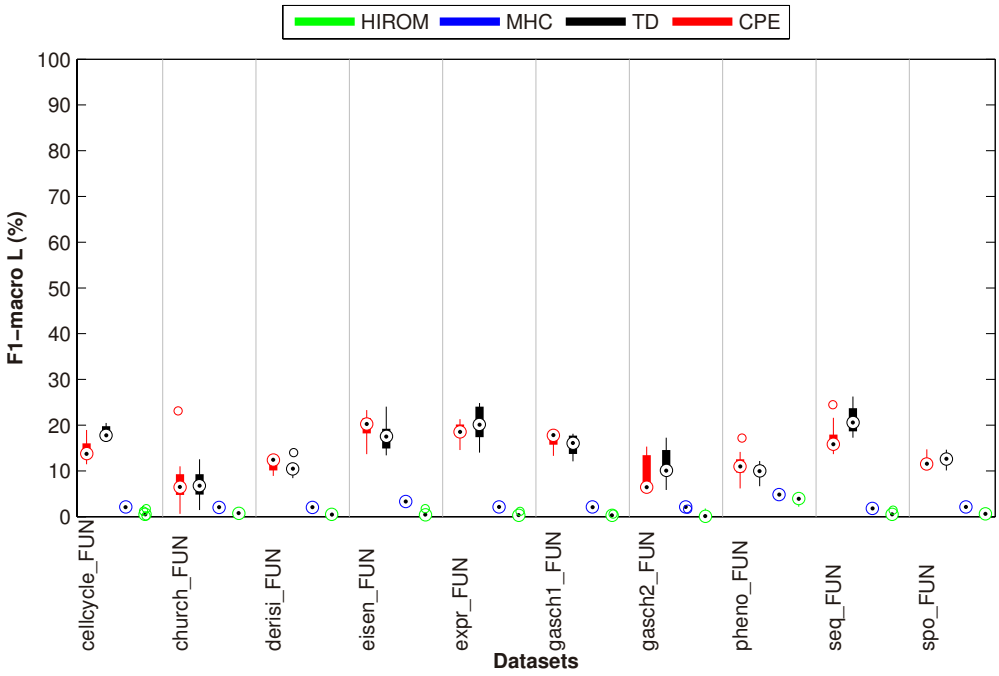


(c) Exact Match (%)

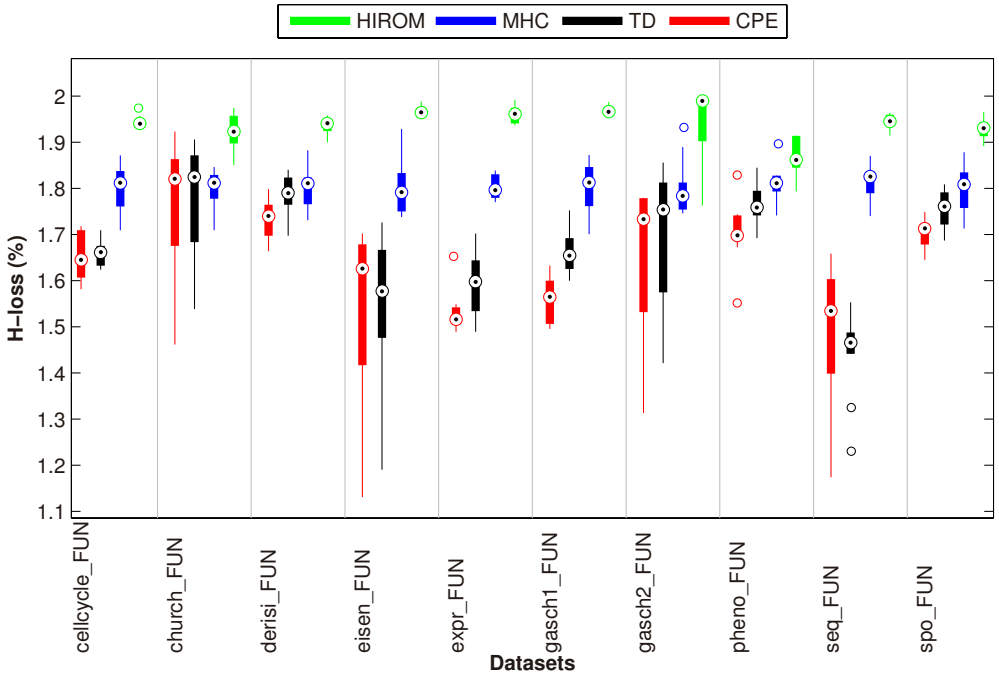


(d) F1-macro D (%)

Figure D.1: Comparison of CPE against other MLNP methods for tree datasets.



(e) F1-macro L (%)



(f) H-loss (%)

Figure D.1: Comparison of CPE against other MLNP methods for tree datasets.

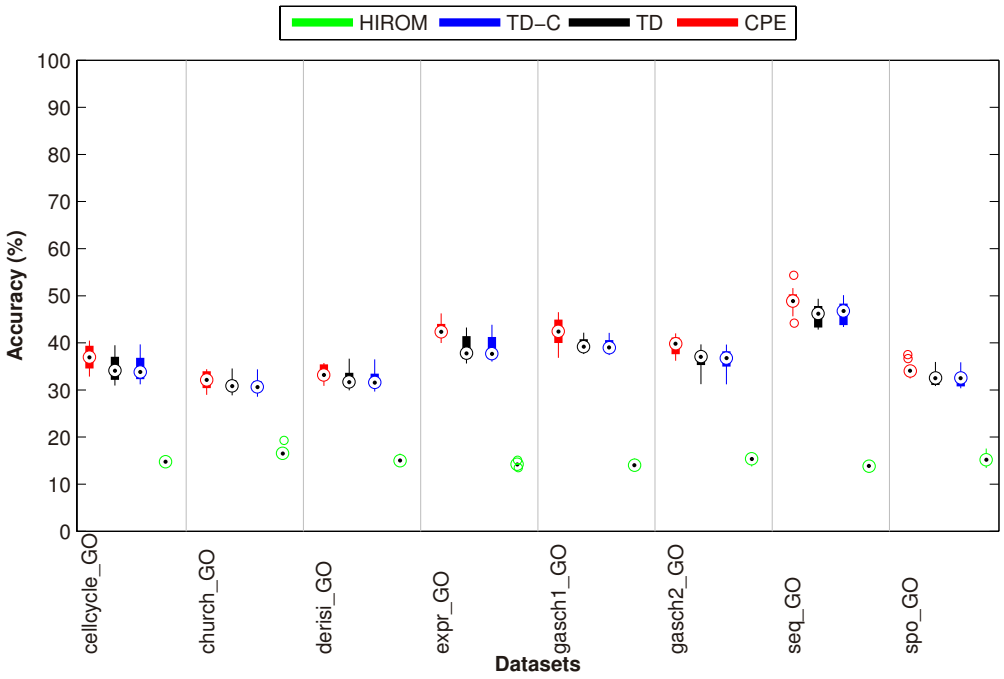
D.2 DAG Structured Datasets

For DAG structured datasets, we used eight hierarchical datasets and compared CPE against tree HMC methods:

1. Top-Down LCPN (TD).

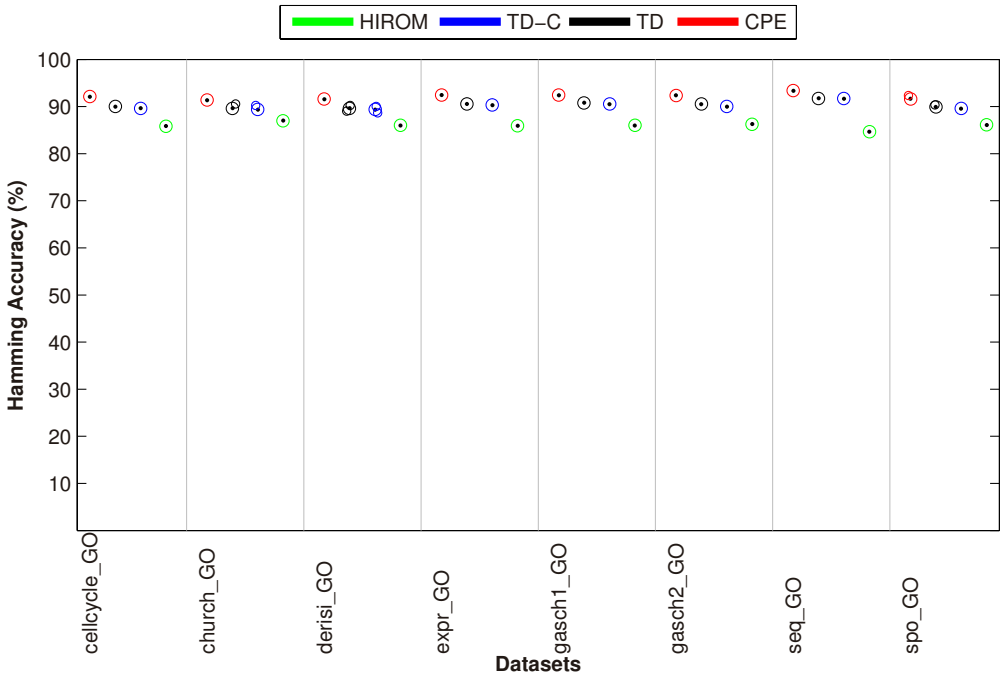
2. Top-Down LCPN Corrected (TD-C).

3. HIROM.

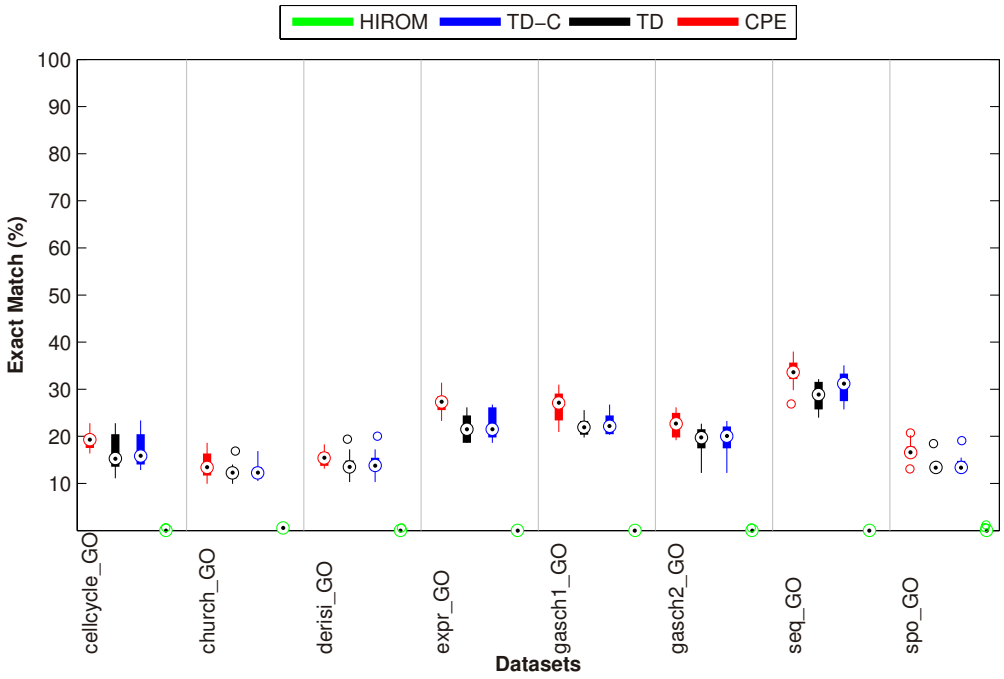


(g) Accuracy (%)

Figure D.2: Comparison of CPE against other MLNP methods for DAG datasets.

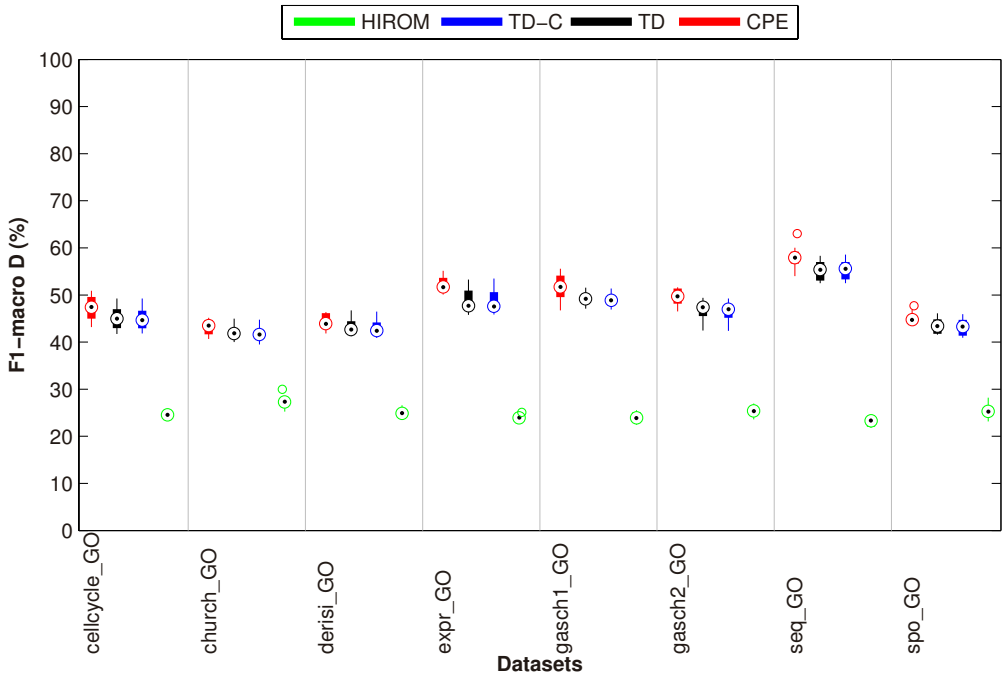


(h) Hamming Accuracy (%)

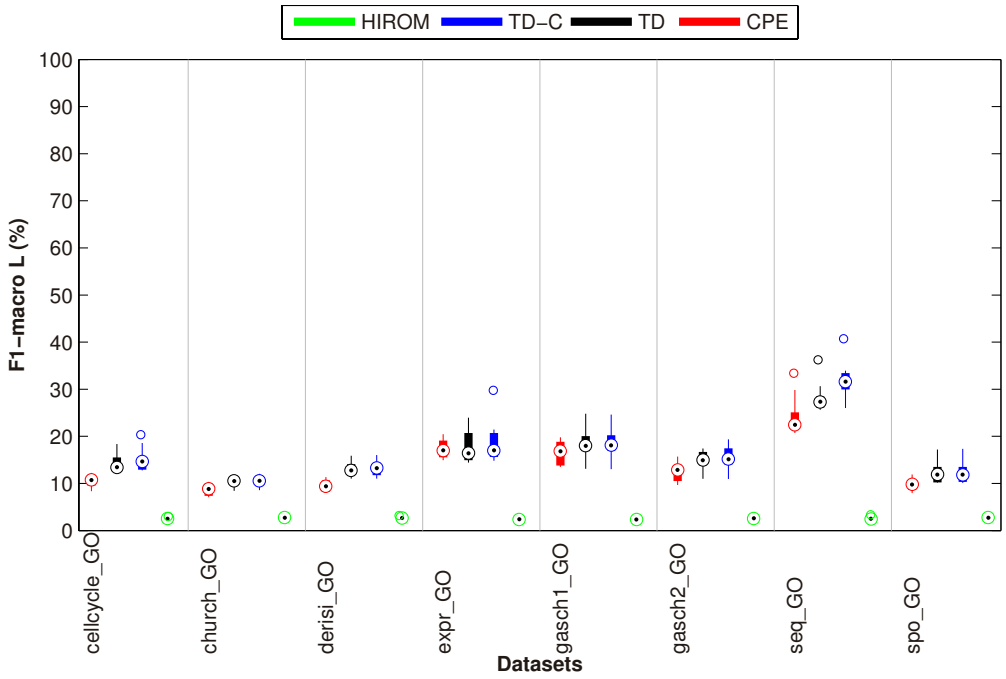


(i) Exact Match (%)

Figure D.2: Comparison of CPE against other MLNP methods for DAG datasets.

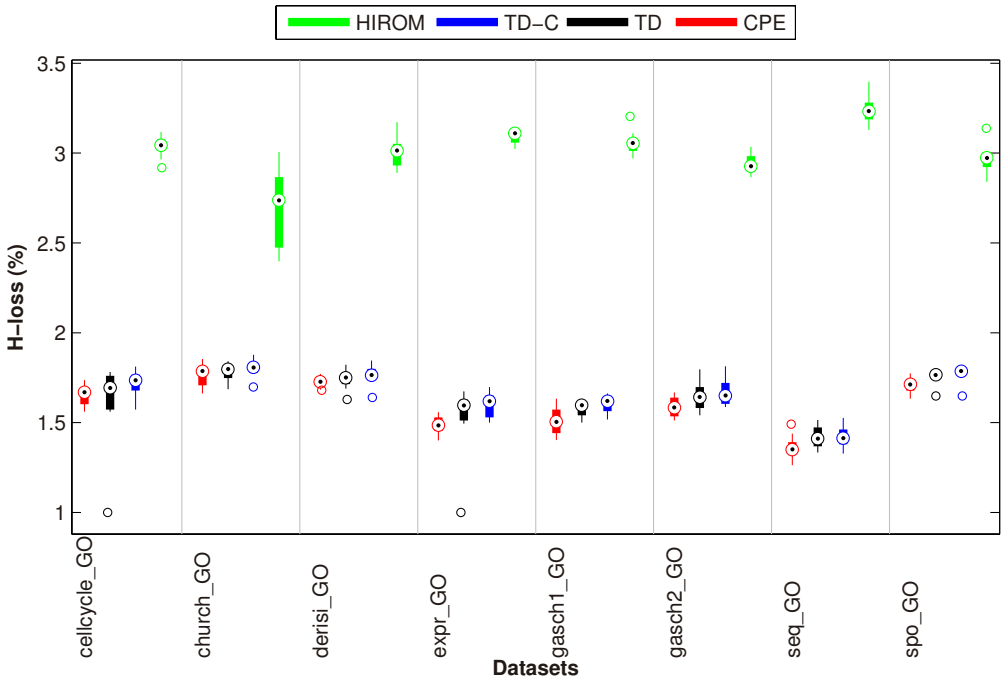


(j) F1-macro D (%)



(k) F1-macro L (%)

Figure D.2: Comparison of CPE against other MLNP methods for DAG datasets.



(l) H-loss (%)

Figure D.2: Comparison of CPE against other MLNP methods for DAG datasets.

Appendix E

Selection of the best NMLNP approach

The pruning approaches described in Section 4.4 for the NMLNP version of CPE were tested to select the best. In the case of DAG structured datasets the root of the hierarchy has only one child and this child (l_0) is parent of the rest of the nodes of the hierarchy. The problem in this kind of hierarchies is that most measures score the conservative classifications as the better ones. In this case, the method *"Top-Down, Select & Prune, IG"* predict just l_0 node for every new instance, this classification is useless due to the fact that every instance belongs to l_0 and nevertheless is the one that is better scored.

Gain-Loose Balance deals with this problem and gives better scores to other methods that return relevant predictions. For that reason, and the fact that the other measures were inconsistent along the databases and the different structures, the methods were compared using Gain-Loose Balance. The results for the NMLNP approaches are depicted on Table E.1.

Discussion

We observe that *"Top-Down, Prune & Select, BEST"* method obtains in most of the cases the better score in both, tree and DAG structures.

The datasets have approximately 16% percent of instances which real label set is just the label l_0 . Since the average number of labels per instance is three, when a method predict only l_0 it already has $1/3$ of the correct answer. This can be one of the reasons why the rest of the metrics give the best scores to the *"Top-Down, Select & Prune, IG"* method, even when this method predict just the l_0 label in every single instance it classifies.

Table E.1: NGLB (%) of the different approaches for pruning NMLNP in tree and DAG structured datasets.

Dataset	Top-Down						Bottom-Up					
	Prune & Select			Select & Prune			Prune & Select			Select & Prune		
	SUM	BEST	IG	SUM	BEST	IG	SUM	BEST	IG	SUM	BEST	IG
cellcycle_FUN	60.53	**60.91**	59.26	59.38	59.91	58.86	59.59	60.19	58.77	59.30	59.43	58.92
church_FUN	59.52	59.53	56.51	57.39	57.63	56.34	59.63	**59.83**	56.42	57.55	57.75	56.48
derisi_FUN	60.29	**60.70**	57.43	59.02	59.04	56.90	59.33	59.73	57.08	58.34	58.52	57.70
eisen_FUN	62.17	62.66	61.44	61.61	62.02	60.06	62.05	**62.87**	61.34	61.78	61.67	60.80
expr_FUN	60.66	**61.62**	60.77	61.05	61.45	60.34	60.98	60.68	61.26	60.89	60.78	60.56
gasch1_FUN	61.20	**62.45**	60.63	61.07	61.48	59.98	60.62	61.09	60.46	61.22	61.12	60.68
gasch2_FUN	61.29	**61.30**	59.43	60.05	60.11	58.76	60.74	61.16	59.14	60.19	59.93	58.91
pheno_FUN	63.43	**63.85**	59.57	60.60	60.46	59.72	63.72	63.68	58.87	60.55	60.69	59.41
seq_FUN	62.42	**62.81**	62.02	62.69	62.75	62.16	62.43	62.19	62.05	62.74	62.42	62.05
spo_FUN	60.46	**61.05**	57.96	59.12	59.48	57.60	59.36	60.05	58.26	58.77	59.19	57.86
cellcycle_GO	83.34	**83.83**	79.91	81.63	82.25	82.15	82.06	82.42	80.01	81.43	81.76	80.07
church_GO	83.48	**83.57**	79.08	81.67	81.63	82.14	82.19	82.27	79.15	81.56	81.78	78.94
derisi_GO	83.56	**83.73**	78.94	81.67	82.07	82.08	81.96	82.12	79.16	80.94	81.49	78.95
expr_GO	82.35	**83.55**	81.10	81.30	82.05	82.16	81.83	82.21	81.21	81.50	81.61	81.13
gasch1_GO	83.03	**83.96**	80.96	81.75	82.57	82.14	82.18	82.50	81.07	81.55	82.09	80.79
gasch2_GO	83.92	**84.12**	80.00	82.16	82.17	82.16	82.56	82.76	80.03	81.53	82.00	79.83
seq_GO	83.32	**84.01**	82.25	82.75	83.34	81.91	82.90	83.09	82.31	82.77	83.09	82.08
spo_GO	83.42	**83.84**	79.49	81.27	81.67	82.11	82.06	82.17	79.61	81.19	81.65	79.24

Appendix F

Comparison of NMLNP approaches

F.1 Tree Structured Datasets

The results are depicted in the following figures, the central mark of the plots is the median, the edges of the box are the 25th and 75th percentiles, the whiskers (lines) extend to the most extreme data points not considered outliers, and outliers are plotted individually and marked with a circle.

For tree structured hierarchies, we used ten hierarchical datasets and compared CPE against three HMC-NMLNP methods:

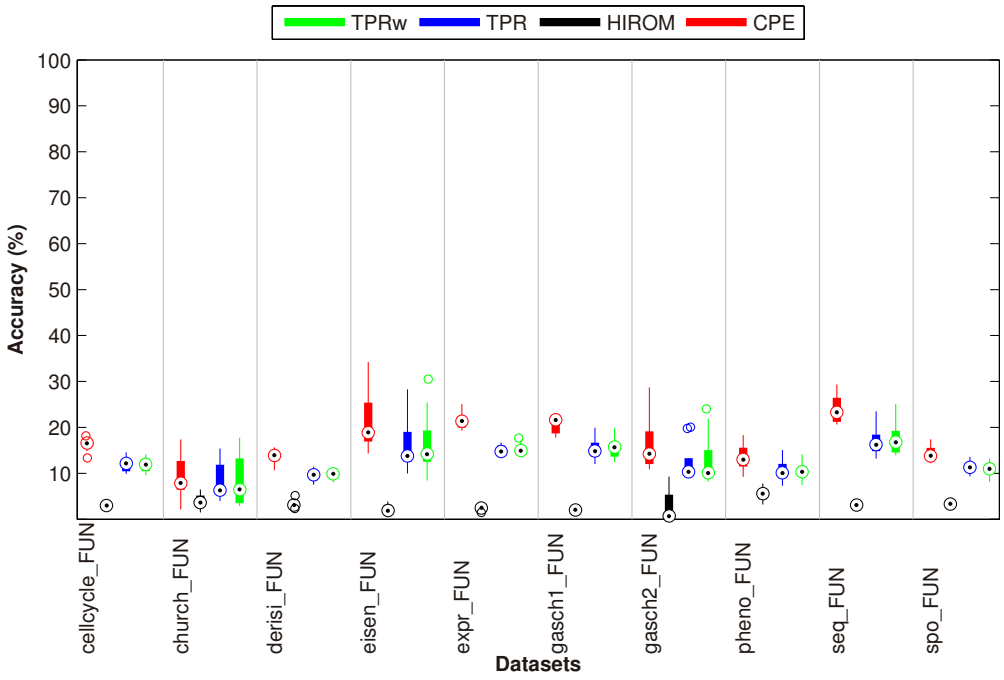
1. HIROM.

2. True Path Rule (TPR).

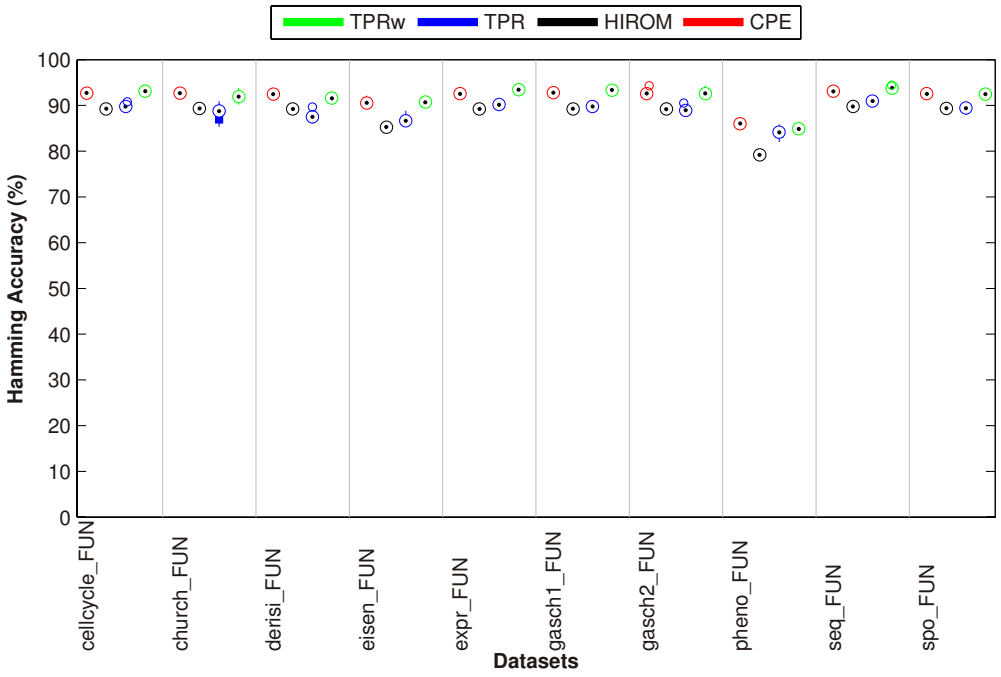
3. Weighted True Path Rule (TPRw).

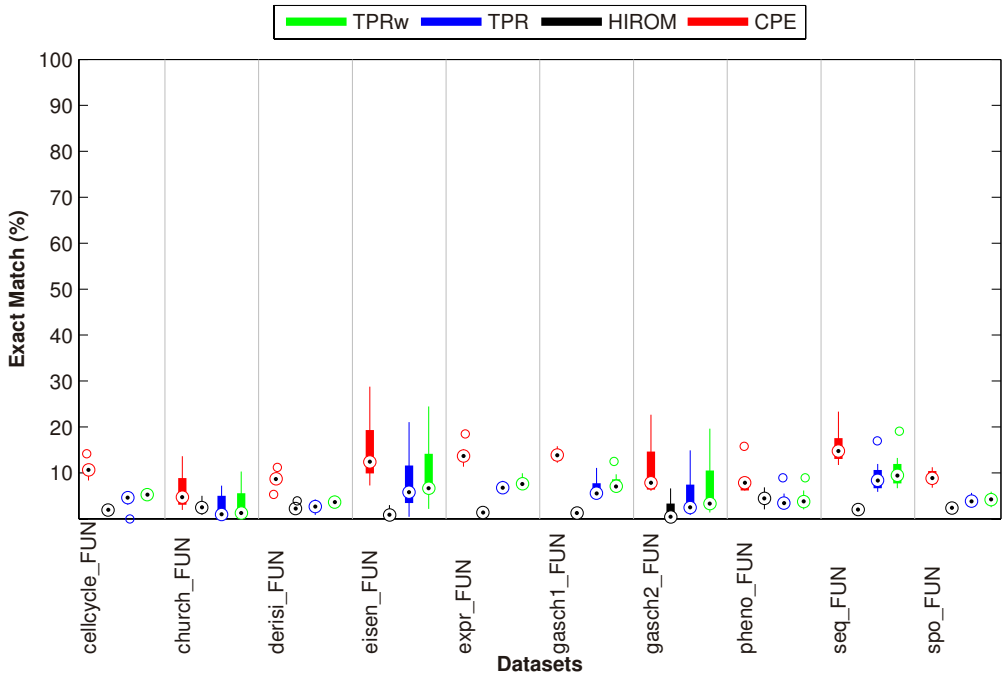
Figure F.1: Comparison of CPE against other NMLNP methods for tree datasets.

(a) Accuracy (%)

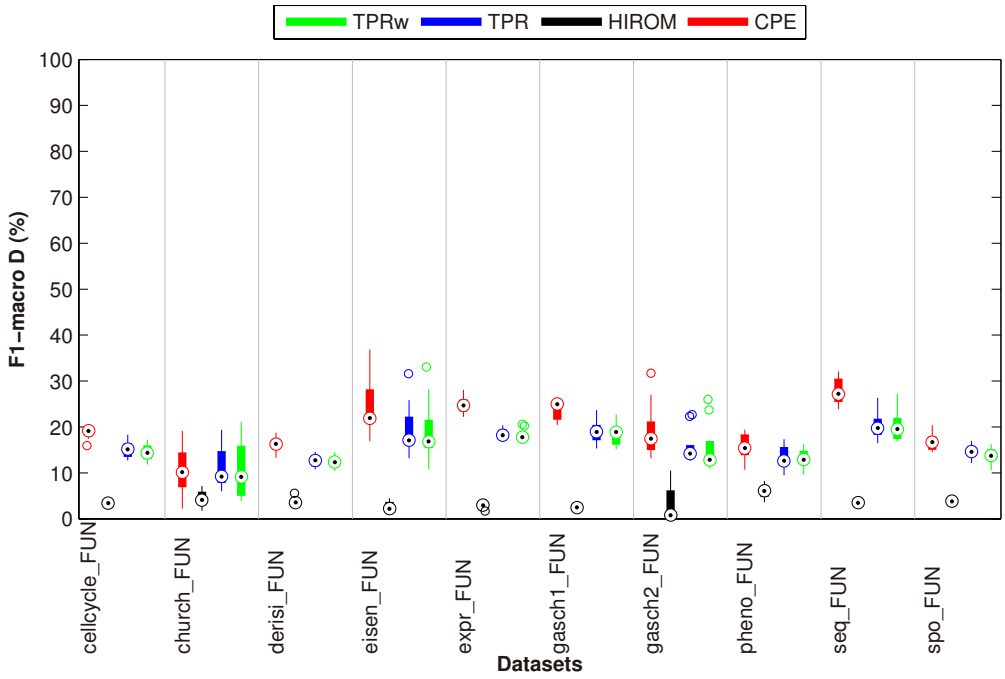


(b) Hamming Accuracy (%)

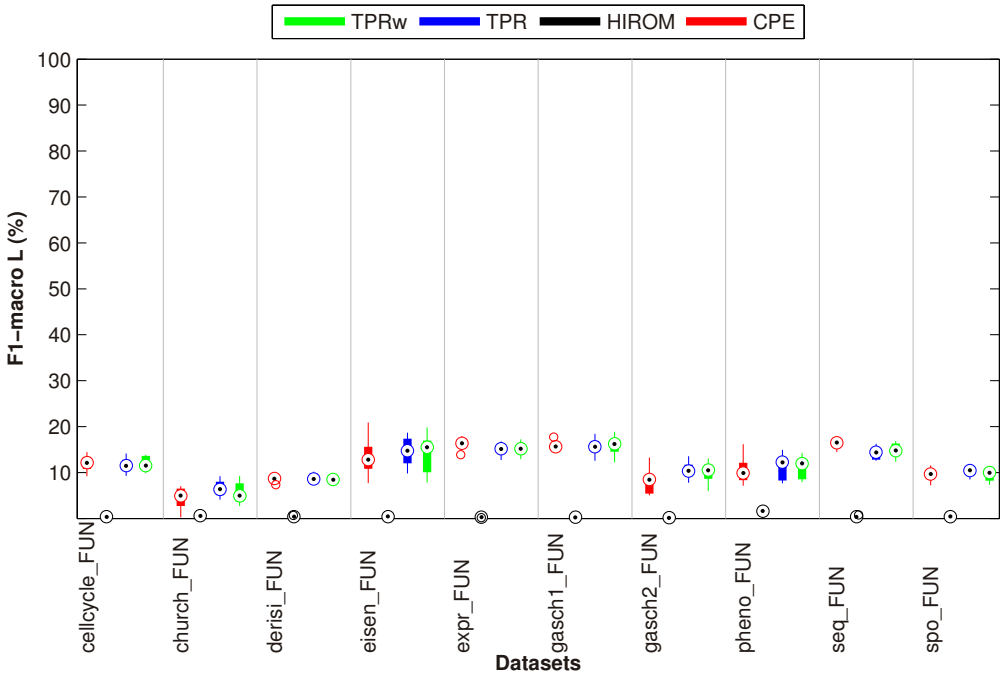




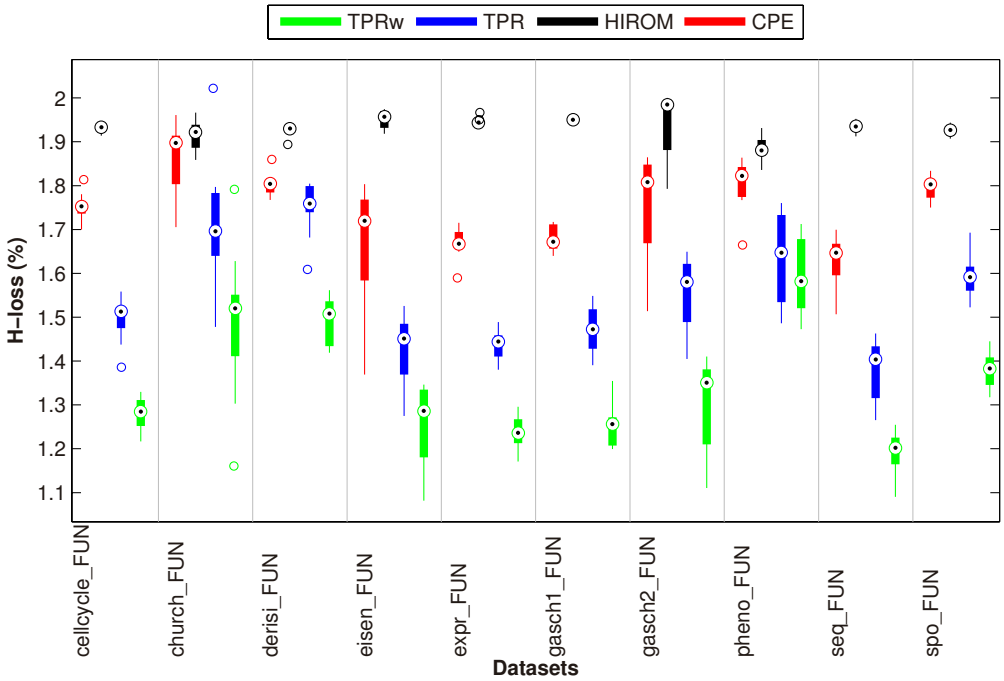
(c) Exact Match (%)



(d) F1-macro D (%)



(e) F1-macro L (%)



(f) H-loss (%)

F.2 DAG Structured Datasets

For DAG structured hierarchies, we used ten hierarchical datasets and compared CPE against three HMC-NMLNP methods:

1. HIROM. Proposed by Bi and Kwok (2012).

Table F.1: Accuracy percentage (mean [std]) comparison of MLNP methods.

Dataset	CPE	HIROM
cellcycle_GO	**41.08 [1.04]**	14.82 [0.46]↓
church_GO	**40.41 [2.90]**	15.41 [1.14]↓
derisi_GO	**40.98 [1.24]**	15.30 [0.79]↓
expr_GO	**39.15 [2.04]**	14.69 [0.51]↓
gasch1_GO	**40.82 [1.55]**	14.73 [0.56]↓
gasch2_GO	**42.14 [4.55]**	15.03 [1.05]↓
seq_GO	**40.90 [1.91]**	14.04 [0.86]↓
spo_GO	**41.19 [1.25]**	15.31 [0.49]↓

Table F.2: Hamming Accuracy percentage (mean [std]) comparison of MLNP methods.

Dataset	CPE	HIROM
cellcycle_GO	**95.46 [0.15]**	84.66 [0.34]↓
church_GO	**95.44 [0.30]**	84.99 [0.87]↓
derisi_GO	**95.49 [0.20]**	85.17 [0.39]↓
expr_GO	**94.68 [0.36]**	84.83 [0.42]↓
gasch1_GO	**95.16 [0.29]**	84.85 [0.39]↓
gasch2_GO	**95.56 [0.54]**	85.20 [0.93]↓
seq_GO	**95.13 [0.16]**	85.37 [0.65]↓
spo_GO	**95.43 [0.22]**	84.85 [0.33]↓

Table F.3: Exact Match percentage (mean [std]) comparison of NMLNP methods.

Dataset	CPE	HIROM
cellcycle_GO	**3.61 [1.39]**	0.00 [0.00]↓
church_GO	**3.10 [3.45]**	0.00 [0.00]↓
derisi_GO	**4.10 [0.72]**	0.06 [0.12]↓
expr_GO	**8.76 [1.92]**	0.00 [0.00]↓
gasch1_GO	**7.01 [0.90]**	0.00 [0.00]↓
gasch2_GO	**5.00 [5.61]**	0.00 [0.00]↓
seq_GO	**9.65 [3.52]**	0.00 [0.00]↓
spo_GO	**4.53 [0.97]**	0.06 [0.12]↓

Table F.4: F1-macro D percentage (mean [std])comparison of NMLNP methods.

Dataset	CPE	HIROM
cellcycle_GO	**55.81 [0.98]**	24.66 [0.58]↓
church_GO	**55.27 [2.40]**	25.47 [1.53]↓
derisi_GO	**55.61 [1.23]**	25.28 [0.97]↓
expr_GO	**52.53 [1.99]**	24.53 [0.68]↓
gasch1_GO	**54.80 [1.60]**	24.53 [0.65]↓
gasch2_GO	**56.69 [3.77]**	25.00 [1.43]↓
seq_GO	**54.28 [1.46]**	23.67 [1.14]↓
spo_GO	**55.76 [1.26]**	25.31 [0.63]↓

Table F.5: F1-macro L percentage (mean [std]) comparison of NMLNP methods.

Dataset	CPE	HIROM
cellcycle_GO	**5.09 [0.87]**	3.37 [0.20]↓
church_GO	**3.77 [0.89]**	3.52 [0.25]
derisi_GO	**5.16 [0.41]**	3.47 [0.15]↓
expr_GO	**10.34 [1.36]**	3.27 [0.11]↓
gasch1_GO	**8.87 [1.18]**	3.27 [0.14]↓
gasch2_GO	**4.64 [1.41]**	3.32 [0.29]↓
seq_GO	**11.44 [2.48]**	2.88 [0.18]↓
spo_GO	**5.37 [0.38]**	3.54 [0.20]↓

Table F.6: H-loss (mean [std]) comparison of MLNP methods.

Dataset	CPE	HIROM
cellcycle_GO	**1.396 [0.04]**	3.065 [0.07]↓
church_GO	**1.400 [0.07]**	2.949 [0.14]↓
derisi_GO	**1.389 [0.03]**	2.987 [0.06]↓
expr_GO	**1.478 [0.05]**	3.049 [0.10]↓
gasch1_GO	**1.430 [0.05]**	3.054 [0.08]↓
gasch2_GO	**1.362 [0.10]**	2.970 [0.17]↓
seq_GO	**1.461 [0.05]**	3.104 [0.15]↓
spo_GO	**1.392 [0.03]**	2.988 [0.09]↓

Table F.7: Normalized Gain-Loose Balance percentage (mean [std]) comparison of NMLNP methods.

Dataset	CPE	HIROM
cellcycle_GO	**83.83 [0.43]**	63.65 [0.73]↓
church_GO	**83.57 [0.67]**	64.68 [1.96]↓
derisi_GO	**83.73 [0.53]**	64.10 [0.91]↓
expr_GO	**83.55 [0.65]**	63.87 [0.96]↓
gasch1_GO	**83.96 [0.59]**	63.79 [0.80]↓
gasch2_GO	**84.12 [1.56]**	64.50 [2.48]↓
seq_GO	**84.01 [0.53]**	63.49 [1.48]↓
spo_GO	**83.84 [0.53]**	64.13 [0.63]↓