Semi-Supervised Hierarchical Classification

by

MSc. Jonathan Serrano Pérez

Dissertation submitted in partial
fulfillment of the requirements for the
degree of

PhD. in Computer Science

at the

Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla, México
April 30th, 2025

Advisor: **Dr. Luis Enrique Sucar Succar**

Computer Science Department, INAOE

To my mother Veronica
and my sisters, Nancy and Citlalli.

Abstract

Hierarchical classification is commonly seen as a special type of multi-label classification, where the instances can be associated to multiple labels, but labels are arranged in a predefined structure, a hierarchy. The predictions in hierarchical classification have to fulfill the hierarchical constraint that states if an instance is associated to a node, then it also has to be associated to the ancestors of that node.

Moreover, scarcity of labeled data is a common problem in supervised classification, because hand-labeling data is expensive, time-consuming or difficult to label; it is a problem also present in hierarchical classification. Even though, labels arranged in hierarchies are found in multiples domains, such as text categorization, image classification, biology and music, just a few works address the problem of scarcity of labeled data in a hierarchical classification scenario.

Therefore, the main goal of this research it to develop a semi-supervised hierarchical classifier that can be trained with labeled and unlabeled data, for the hierarchical problem where the hierarchies can be of directed acyclic graph type and the instances can be associated to multiple paths of labels of partial depth.

Semi-supervised hierarchical multi-label classifier based on local information (SSHMC-BLI) is the proposed classifier, which can be trained with labeled and unlabeled data to perform hierarchical classification tasks. The method mainly consists on building pseudo-labels for each unlabeled instance from the paths of labels of its labeled neighbors, while it considers whether the unlabeled instance is similar to its neighbors. Experiments on several artificial and real world datasets show that using unlabeled along with labeled data can help to improve the performance of a supervised hierarchical classifier trained only on labeled data, with statistical significance.

Finally, some extensions of SSHMC-BLI were proposed: SSHC-BLI which is variant that handles only hierarchies of tree type; SSHBMC which follows a Bayesian approach, that is, the hierarchy is modeled as a Bayesian network; and a variant of SSHMC-BLI that incorporates transfer learning among neighboring nodes.

Acknowledgment

Completing this PhD has been one of the most challenging and rewarding experiences of my life, and it would not have been possible without the support, guidance, and encouragement of many individuals.

First and foremost, I extend my deepest gratitude to my advisor, L. Enrique Sucar Succar, for their unwavering guidance, patience, and expertise. Their insightful feedback and encouragement pushed me to grow as a researcher. I am truly privileged to have had the opportunity to learn under his mentorship.

I am also immensely grateful to the members of my dissertation committee, Carlos N. Silla Junior, Eduardo F. Morales Manzanares, Jesús Ariel Carrasco Ochoa, José Martínez Carranza and María del Pilar Gómez Gil, for their invaluable time and constructive critiques that helped shape the direction of my research. Their support has been instrumental in refining my work and ensuring its academic rigor.

To my colleagues and fellow researchers at INAOE, thank you for the inspiring discussions, collaborations, and camaraderie that made this journey enjoyable and intellectually stimulating. Your encouragement and shared experiences helped me persevere through the toughest moments.

A special thank you to my friends and family, whose unwavering belief in me kept me motivated. To my parents, Veronica and Juan, and to my sisters, Nancy and Citlalli, for their patience, understanding, and endless support; also, I offer my deepest gratitude to God, whose grace and guidance have been my source of strength throughout this journey. I am forever grateful.

Lastly, I acknowledge the financial support from INAOE and CONAHCYT/ SECIHTI, under the doctoral scholarship grant 840675, for their support during this PhD research.

Thanks to all of you,
Jonathan Serrano Pérez.
Cholula, Puebla, México,
April, 2025

Contents

Acronyms

ANN Artificial Neural Network.

BCNN Bayesian and Convolutional Neural Networks.

CNN Convolutional Neural Networks.

DAG Directed Acyclic Graph.

EM Exact Match.

FD Full Depth.

FunCat Functional Catalogue.

GO Gene Ontology.

hF Hierarchical F-measure.

HMC-SSBR Hierarchical Multi-label Classification using Semi-Supervised Binary Relevance.

hP Hierarchical Precision.

hR Hierarchical Recall.

HSp Hierarchical Specialization.

LCL Local Classifier per Level.

LCN Local Classifier per Node.

LCPN Local Classifier per Parent Node.

MCC Matthews correlation coefficient.

MLP Multilayer Perceptron.

MPL Multiple Paths of Labels.

NMLNP Non-Mandatory Leaf Node Prediction.

PD Partial Depth.

PFS Parents First Search.

SC Supervised Classification.

SISI Similarity of an Instance with a Set of Instances.

SPL Single Path of Labels.

SPP Single Path Prediction.

SSHBMC Semi-Supervised Hierarchical Multi-label Classifier Based on Local Information.

SSHC Semi-Supervised Hierarchical Classifier.

SSHC-BLI Semi-Supervised Hierarchical Classifier Based on Local Information.

SSHMC-BLI Semi-Supervised Hierarchical Multi-label Classifier Based on Local Information.

SSL Semi-Supervised Learning.

STA Self Training A.

STHC Self-Training Hierarchical Classifier.

STML Self-Training for Multi-Label classification.

TD Top-Down.

TL Transfer Learning.

List of Symbols

X	Set of labeled instances.
Y	Set of labels to which the instances of X are associated.
U	Set of unlabeled instances.
L	Set of labels.
$\|L\|$	Cardinality of L.
$\mathbb{R}$	The set of real numbers.
x_i	i-th instance.
$x_{i,j}$	j-th element of the i-th instance.
$\{0, 1\}$	The set containing 0 and 1.
$[a, b]$	The real interval including a and b.
$(a, b]$	The real interval excluding a but including b.
$f : A \to B$	The function f with domain A and range B.

Chapter 1

Introduction

Hierarchies have been a fundamental part of human organization throughout history. From ancient civilizations to today's societies, hierarchies have played a crucial role in the structuring and functioning of communities and organizations. Moreover, hierarchies have allowed us to organize and classify abstractions. For instance, one of the most known hierarchies is the *taxonomic hierarchy* which organizes living beings into *classes*, that is, for each living being, in the first level a *kingdom* is assigned, in the second level a *phylum* (that inherits the characteristics from the corresponding kingdom) is assigned, and so on until a *specie* is assigned. As results of this, we can easily note that the amount of living beings, at each class, tend to decrease as one goes deeper in the hierarchy. Moreover, the increasing number of classes and instances requires automatic classifiers that allows us to classify an *unknown* instance into the available classes, for example, imagine manually classifying an animal into one of the $\sim$6 million animals species [Wiens, 2023, Stork, 2018], it would be a very difficult and time-consuming task.

Supervised classification (SC) is the branch of *machine learning* in charge of building classifiers from labeled instances, which are later used to predict the labels (classes) of *new/unknown* instances. Nevertheless, scarce data is a common problem of SC, this occurs when hand-labeling data is time-consuming, expensive or difficult to label [de Oliveira and Berton, 2023, van Engelen and Hoos, 2019, Gomes et al., 2022]. Consequently, the problem when a classifier is trained with few labeled data is that an *unreliable* classifier could be obtained. Furthermore, the problem of scarce data may be found in a scenario of multi-label classification, that is, when instances can be associated to multiple labels instead of a single.

Even more, hierarchical classification can be seen as a *special* type of multi-label classification, that is, an instance can be associated to multiple labels, but the labels are arranged into a predefined structure, that contains the relations among the labels, which is commonly a tree but in its general form is a directed acyclic

graph (DAG). As shown by Bielza et al. [2011] and Sucar et al. [2014], in multi-label classification the labels can be related, and by taking into account those relations when training a classifier the performance of it can be improved. Therefore, in hierarchical classification the relations among the label must be taken into account.

There are two main approaches that try to addressed the problem of scarce data. The first is generation of artificial data, for example the SMOTE method [Chawla et al., 2002]. The second consists on making use of unlabeled data along with labeled data, this approach is known as semi-supervised learning (SSL) [Gui et al., 2023, Yang et al., 2023, van Engelen and Hoos, 2019]. Hence, SSL has the advantage that large amounts of unlabeled information can be obtained from different sources, for instance the internet, such as video, text, images, etc.

Therefore, the aim of this research is to develop a *Semi-Supervised Hierarchical Classifier* (SSHC) which can be trained with labeled and unlabeled data, in a scenario of hierarchical classification where the hierarchy is any DAG and the instances can be associated to multiple paths of labels with partial depth. Furthermore, the SSHC has to make use of the relations among the labels (hierarchy), and it is expected that the SSHC can predict both single and multiple paths of labels.

First, *semi-supervised hierarchical classifier based on local information* (SSHC-BLI) was proposed as a hierarchical semi-supervise classifier, but it can only handle hierarchies of tree type, where the instances can be associated to a single path of labels of full depth. It consist on pseudo-labeling the unlabeled instances using the labels of their labeled neighbors while considers their similarity; later, a hierarchical classifier is trained with the labeled and pseudo-labeled instances.

Later, *Semi-supervised hierarchical multi-label classifier based on local information* (SSHMC-BLI) was built upon the principles of SSHC-BLI, which is able to handle any hierarchy of directed acyclic graph type where the instances can be associated to multiple paths of labels of partial depth. Furthermore, a couple of extensions of SSHMC-BLI were proposed, the first follows a Bayesian approach (SSHBMC) where the hierarchy is modeled as a Bayesian network, additionally, a supervised application on morphological classification of galaxies using the Bayesian network was carried out; and the second, which incorporates transfer learning among neighboring nodes.

Results on several real world datasets show that making use of unlabeled data along with labeled can help to improve the performance of a hierarchical classifier. As well as, the Bayesian and transfer learning extensions can help to improve the performance of the SSHMC-BLI classifer.

1.1 Motivation

A common problem in supervised classification is lack of data. This may be because hand-labeling data is time consuming and costly or just hard to label [de Oliveira and Berton, 2023, van Engelen and Hoos, 2019]. Hence, training a classifier with few labeled data could produce a unreliable classifier.

This is even more notorious in hierarchical classification, because the data of a node is split among its children, hence, nodes in deeper levels of the hierarchy only have a little fraction of data. So, an alternative way is to use semi-supervised learning, that is, use unlabeled data along with the labeled to train a classifier. Moreover, considering that upper nodes contain general information while lower nodes contain specific information, transfer learning may be applied, that is, upper nodes could share their learned information to the lower ones.

Furthermore, large amounts of information can be obtained from different sources of information, such as the internet. Information such as text, images, videos, etc., is commonly desired, nevertheless, most of that information is unlabeled. Moreover, unlabeled information could be required in scenarios where instances can have associated multiple labels, like hierarchical classification, which makes more challenging make use of unlabeled data. So strategies that take advantage of that information are required.

Hierarchical classification methods have been applied in multiple domains, showing better performance than flat classification (algorithms that do not consider the hierarchy in any way), some of them are functional genomics [Giunchiglia and Lukasiewicz, 2020, Serrano-Pérez and Sucar, 2019], text classification [Wu and Saito, 2018, Kowsari et al., 2017] and image classification [Sali et al., 2020, Kowsari et al., 2020, Murtaza et al., 2019]. So, we consider that a suitable Semi Supervised Hierarchical Classification algorithm, that consider the hierarchy, trained with labeled and unlabeled data, can produce a hierarchical classifier with better performance than using only the few labeled data.

1.2 Justification

This research will address the following problems related to hierarchical classification:

- **Few labeled data (Hierarchical classification)**: Training classifiers with few labeled data could produce an unreliable classifier, but making use of unlabeled data together with a suitable semi-supervised hierarchical classifier could help to improve the performance of the supervised classifier. Furthermore, it is well

known in hierarchical classification that the number of instances is split from each node to its children, which result in deep nodes with very few instances.

- **Single and multiple path prediction**: The unlabeled data is used in a scenario where the hierarchy can be of DAG type and the instances can be associated to multiple paths of labels which can be of partial depth.

1.3 Research Questions

- Will training a SSHC with unlabeled and few labeled data produce a classifier with better performance than the hierarchical classifier trained only in the few labeled data?
- Will a SSHC perform better when it uses transfer learning than when it does not?

1.4 Hypothesis

Training a SSHC, that uses as base hierarchical classifier to HC, with unlabeled and labeled data will produce a classifier with better performance than training the hierarchical classifier HC only on labeled data.

1.5 Problem Statement

When training a hierarchical classifier with few labeled data, it can result in an unreliable classifier. Using unlabeled data could help to improve the performance of the hierarchical classifier. In the literature (see section 2.2), several methods have been proposed that make use of labeled and unlabeled data to perform learning tasks. Nevertheless, most of them were proposed for *flat* classification and applying them directly on hierarchical classification means that the information provided by the hierarchy is ignored.

Hence, a suitable SSHC is required, that is, a SSHC that pseudo-labels unlabeled instances and selects the best of them to retrain itself, in order to get a better performance than the hierarchical classifier trained only in the labeled data.

Formally: let $X = \{x_1, x_2, ..., x_n\}$ and $U = \{x_{n+1}, x_{n+2}, ..., x_{n+m}\}$ be instances sets where $x_i \in \mathbb{R}^d$, that is, each instances x_i is described by a vector of d attributes, let $|L|$ be the number of labels, let $Y = \{y_1, y_2, ..., y_n\}$ be the set of labels for X,

where $y_i \in \{0,1\}^{|L|}$, that is, each $y_{i,j}$ indicates if the i-th instance is associated to the j-th label, in this way (X, Y) is a labeled set and U is an unlabeled set. Additionally, let $HS = (L, E)$ be the DAG that represents the hierarchy, where L is the set of nodes and E is the set of edges that link the labels.

Therefore, from (X, Y, U, HS) is required a classifier to predict labels for new instances: $f_{SSHC} : \mathbb{R}^d \to \{0,1\}^{|L|}$, and whose performance is better than the supervised classifier trained only with labeled data (X, Y, HS): $f_{HC} : \mathbb{R}^d \to \{0,1\}^{|L|}$, that is:

$$performance(f_{SSHC}) > performance(f_{HC}) \tag{1.1}$$

1.6 Objectives

The **general objective** of this research is to develop a semi-supervised hierarchical classifier for hierarchical multi-label classification, that can be trained with labeled and unlabeled data; whose performance, in tree hierarchies, is competitive with state-of-the-art methods, and in DAG[1] hierarchies its performance is better than the supervised classifier trained on labeled data.

Our **specific objectives** are:

- Propose a *methodology* for semi supervised hierarchical classifiers based on self training approach.
- Propose a strategy for labeling unlabeled data considering the hierarchy.
- Propose a SSHC based on the proposed methodology.
- Incorporate transfer learning between neighboring nodes.
- Extend the SSHC to Multiple Paths Prediction.

1.7 Scope and Limitations

In hierarchical classification there are different hierarchical classification problems (see section 2.1.1), however, the problems to cover in this research are the following:

- Problems with hierarchy of tree type, where instances are associated to a single path of labels, and the paths always reach a leaf node (full depth).

[1] We could not find any previous work on semi-supervised hierarchical classification for DAG hierarchies.

- Problems with hierarchy of DAG type, where instances are associated to multiple paths of labels, and the paths can finish in internal nodes (partial depth).

One notable limitation of this study is that the proposed semi-supervised methodology was not applied to image, video, and audio datasets. This restriction arises due to the inherent differences in data structure and processing requirements between these types of data and the datasets used in this work. Consequently, the findings and conclusions drawn from this study may not be directly transferable to multimedia data, and further research is needed to adapt and validate the approach for such datasets.

1.8 Contributions

The proposed semi-supervised hierarchical classifier can handle hierarchies of DAG type, where the instances can be associated to multiple paths of labels which can be of partial depth. Its main idea is to pseudo-label the unlabeled data by taking advantage of the smoothness assumption, which are later used to train a hierarchical multi-label classifier. It builds pseudo-labels using the labels of the nearest labeled neighbors to each unlabeled instance, then, the function *similarity of an instance with a set of instances* (SISI) is used to determine if the unlabeled instance is similar to its labeled neighbors, if they are similar, the unlabeled instance is pseudo-labeled, else it stays unlabeled. Experiments on several datasets where the hierarchy of tree type show that the proposed method (SSHC-BLI) is competitive with related methods, while in experiments on datasets where the hierarchy is of DAG type, the proposed method (SSHMC-BLI) showed outstanding performance when compared with its baseline, a supervised hierarchical classifier trained only on labeled instances.

The contributions of this research are next:

- The similarity function SISI to measure the similiarity of an instance with a set of instances. This function takes into account the distances among the set of instances, and the distances of an instances to the set of instances.
- The semi-supervised multi-label hierarchical classifier, SSHMC-BLI, for hierarchical problems where the hierarchy is of DAG type and the instances can be associated to a multiple paths of labels of partial depth. SSHMC-BLI builds pseudo-labels for each unlabeled instances using the labels of its neighbors, but if the unlabeled instance is not similar to its labeled neighbors, then it will stay unlabeled. Several experiments in real world datasets show that using unlabeled data along labeled data can help to improve the performance of a supervised classifier trained only on labeled data.

- The semi-supervised multi-label hierarchical classifier, SSHBMC, that models the hierarchy as a Bayesian network. In order to avoid truncating the probabilities of the nodes to obtain a prediction that fulfills the hierarchical probability constraint, as SSHMC-BLI does, SSHBMC uses a Bayesian network that models the data distribution and uses it as post-processing to obtain consistent predictions. Experiments in several real world datasets show that the inclusion of the Bayesian network can improve the performance of the semi-supervised hierarchical classifier.

- The supervised hierarchical classifier, BCNN, for hierarchical supervised classification of images. This method combines a CNN with a Bayesian network that models the hierarchy. This classifier was applied to morphological classification of galaxies, showing that the use of the Bayesian network can improve the performance of the CNN without the Bayesian network.

- The algorithm hierarchical specialization to transfer learning from parent to child in hierarchies of DAG type. This method help us to carry out transfer learning from parent to child in hierarchies of DAG type. Several experiments in real world datasets shows that transfer learning can improve the performance of a semi-supervised hierarchical classifier.

1.9 Publications

The following publications were derived from this PhD research.

JCR Journal:

- **J. Serrano-Pérez** and L. E. Sucar. *Semi-supervised hierarchical multi-label classifier based on local information.* International Journal of Approximate Reasoning, 2025, doi: 10.1016/j.ijar.2025.109411.

- **J. Serrano-Pérez** and L. E. Sucar. *A Semi-Supervised Hierarchical Classifier Based on Local Information.* Pattern Analysis and Applications, September 2024, doi: 10.1007/s10044-024-01345-1.

- **J. Serrano-Pérez**, R. D. Hernández, and L. E. Sucar. *Bayesian and convolutional networks for hierarchical morphological classification of galaxies.* Experimental Astronomy, August 2024. doi: 10.1007/s10686-024-09950-y.

Conference Proceedings:

- **J. Serrano-Pérez** and L. E. Sucar. *Semi-supervised hierarchical Bayesian classifier*. In Advances in Artificial Intelligence – IBERAMIA 2024. doi: 10.1007/978-3-031-80366-6_23.
- **J. Serrano-Pérez** and L. E. Sucar. *Semi-supervised hierarchical classification based on local information*. In Advances in Artificial Intelligence – IBERAMIA 2022, 2022. doi: 10.1007/978-3-031-22419-5_22.

1.10 Thesis Organization

The rest of the document is organized as follows. Chapter 2 summarizes the fundamentals of hierarchical classification, semisupervised learning and transfer learning. Chapter 3 presents the related work. Chapter 4 presents the datasets used for the experiments with hierarchies of tree and DAG type. Chapter 5 presents the proposed method for datasets with hierarchies of tree type. Chapter 6 presents the proposed method for datasets with hierarchies of DAG type. Chapter 7 presents an extension of the proposed method with Bayesian networks. Chapter 8 presents an extension of the proposed method with transfer learning. Finally, in Chapter 9, conclusions and some ideas for future work are given. A simple roadmap of the content of the thesis is shown in 1.1.

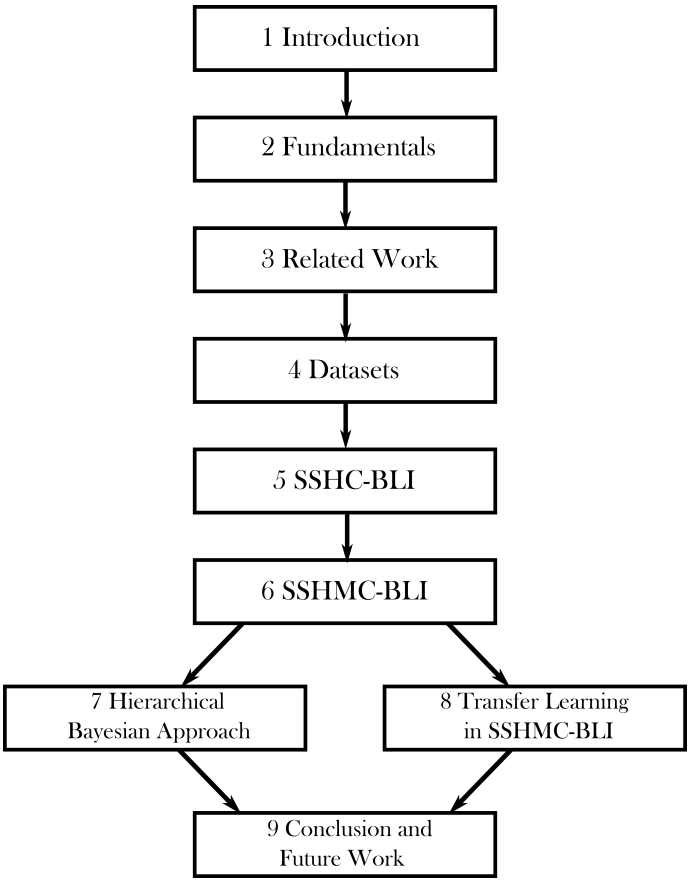


Figure 1.1: Simple roadmap of the thesis chapters.

Chapter 2

Fundamentals

This chapter presents the fundamentals required for this research. Hierarchical classification, semi-supervised learning and transfer learning are briefly introduced in sections 2.1, 2.2 and 2.4, respectively.

2.1 Hierarchical Classification

In machine learining, hierarchical classification can be seen as a special type of multilabel classification, where the labels are arranged in a predefined structure. The structure can be a *tree* or in its general form a *directed acyclic graph* (DAG).

Formally, we define hierarchical classification as a tuple $HC = < HS, X, Y >$, where:

- $HS = (L, E)$ is the hierarchical structure, a *directed acyclic graph* that represents the hierarchy, where L is the set of nodes and E is the set of edges that link the nodes.
- (X, Y) is the labeled set. $X = \{x_1, x_2, ..., x_n\}$ contains n instances, where $x_i \in \mathbb{R}^d$, that is, each instance x_i is described by a vector of d attributes. Let $Y = \{y_1, y_2, ..., y_n\}$ be the labels for X, where $y_i \in \{0, 1\}^{|L|}$, that is, each $y_{i,j}$ indicates whether the i-th instance is associated to the j-th label, while y_i satisfies the *hierarchical constraint*.

HC is composed by two main elements, a hierarchy and a labeled set.

Hence, the problem of *hierarchical classification* consists in assigning to a particular object described by d attributes, a subset of labels that comply the hierar-

chical constraint:

$$f_{HC} : \mathbb{R}^d \rightarrow \{0,1\}^{|L|} \tag{2.1}$$

In hierarchical classification the *hierarchical constraint* states that if an instance x is associated to the label $l \in L$ then x has to be associated to the ancestors of l, $Anc(l)$, given by the HS:

$$\forall x \in l \rightarrow x \in z, \forall z \in Anc(l) \tag{2.2}$$

Therefore, a *valid* or *consistent path* is a subset of the labels that complies the hierarchical constraint.

2.1.1 Hierarchical Classification Problems

In hierarchical classification there are different problems, thus, it is important to know the hierarchical classification problem, in order to choose a suitable method to train and predict. Silla and Freitas [2011] describe the different hierarchical problems as a 3-tuple $< \Upsilon, \Psi, \Phi >$ where:

- Υ: specifies the type of hierarchical structure in which the labels are arranged, so, it can take one of two values, T if it is a tree or DAG if it is a Direct Acyclic Graph.
- Ψ: specifies whether an instance can be associated either one or multiple paths. Thus the values that it can take are: Single Path of Labels (SPL) and Multiple Paths of Labels (MPL).
- Φ: describes the depth of the paths of the instances, two values are permitted: Full Depth (FD) if the path (or paths) of all instances reach a leaf node, and Partial Depth (PD) if at least one path of an instance does not reach a leaf node.

This implies that exist eight different hierarchical classification problems. Fig. 2.1 depicts examples of hierarchical problems when the hierarchy is a tree, while Fig. 2.2 depicts examples of hierarchical problems when the hierarchy is a DAG [Serrano-Pérez, 2019].

2.1.2 Hierarchical Classification Approaches

Some classical approaches for hierarchical classification are described in this section.

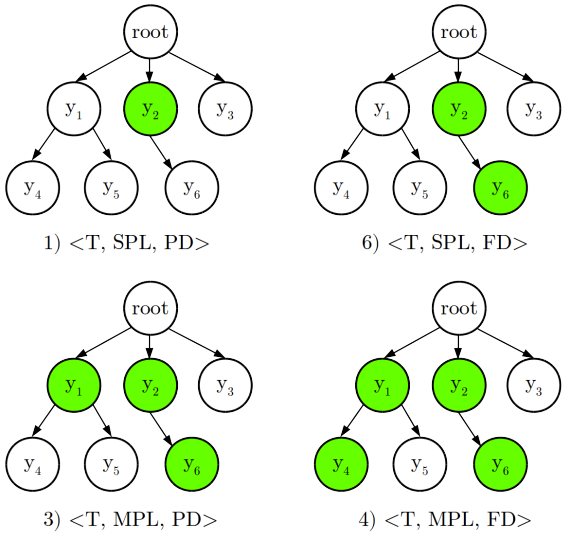


Figure 2.1: Examples of the four problems with hierarchy of tree type. The nodes shaded in green indicate the labels to which some instance is associated, for example, an instance x showed in the problem 3) is associated to the subset $\{y_1, y_2, y_6\}$.

2.1.2.1 Flat classification

Flat classification for hierarchical classification is perhaps the simplest method, because this method ignores the hierarchy and focus its training and predictions over the leaf nodes. Figure 2.3 shows examples of the nodes used to generate a *multiclass* classifier in each hierarchy, hence any multiclass classifier or strategy can be applied. Nevertheless, this approach ignores the useful information that the hierarchy provides.

2.1.2.2 Local Classifier per Parent Node Approach

This approach, Local Classifiers per Parent Node (LCPN), consist on training for each non-leaf node a multiclass classifier to predict its children nodes. An example of this approach is shown in Fig. 2.4, where the different multiclass problems are inside boxes. In DAG hierarchies it is not natural to use LCPN, because nodes with multiple parents obtain multiples predictions, one for each parent node, as can be seen on node y_6. However, strategies to combine multiple predictions can be used, such as averaging the scores obtained by the different classifiers [Ramírez-Corona et al., 2016].

In the prediction phase, the *Top-Down* (TD) procedure can be carried out to obtain a consistent prediction, that is, the instance is evaluated in the multiclass classifier of the root node, then the prediction advances toward the child node with

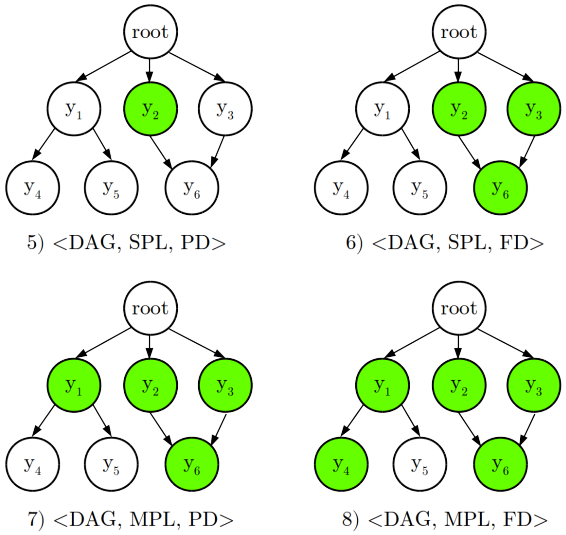


Figure 2.2: Examples of the four problems with hierarchy of DAG type. The nodes shaded in green indicate the labels to which some instance is associated, for example, the instance x showed in the problem 6) is associated to the subset $\{y_2, y_3, y_6\}$.

the highest score, and so on, until a leaf node is reached. Nevertheless, the TD procedure has a very well known problem, called *error-propagation*, which occurs when a prediction is wrong, this implies that all the next predictions will also be wrong. Furthermore, the TD procedure is unable to correct wrong predictions.

2.1.2.3 Local Classifier per Node Approach

In local classifier per node (LCN) approach, a binary classifier is trained for each node in the hierarchy, except for the root, an example is depicted in Fig. 2.5. These classifiers are in charge of predicting whether an instance is associated to the respective label or not. Several policies have been proposed to select the positive and negative instances to train each local classifier [Eisner et al., 2005, Fagni and Sebastiani, 2007, Feng et al., 2018], for instance:

- *Less inclusive policy*: for a node l, the positive instances are all the instances associated to l, and the negative instances are the rest.
- *Siblings policy*: for each node l, the positive instances are all the instances associated to l, while the negatives are those associated to the siblings of l.
- *Exclusive policy*: for each node l the positive instances are only those instances which its most specific label is l, and the negatives are those instances which its most specific label is some sibling of l.

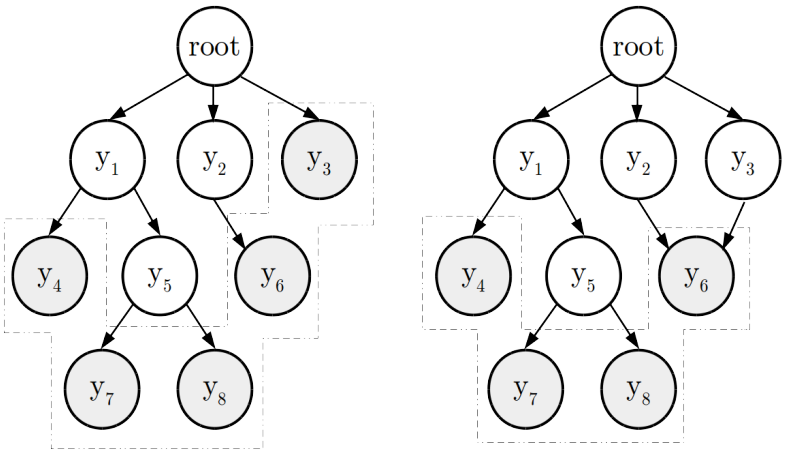


Figure 2.3: In flat classification the hierarchy is ignored. A multiclass classifier is trained considering only the leaf nodes (shaded in gray) in both hierarchies.

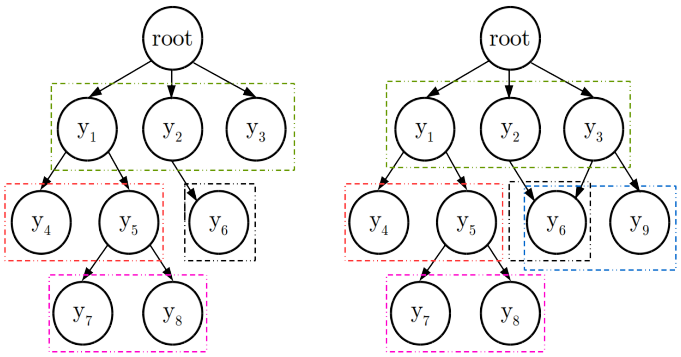


Figure 2.4: LCPN approach in hierarchies of tree (left) and DAG (right) type. Each box contains the nodes/labels for the multiclass problem of the node that all they are children. (Best seen in color.)

- *Balanced bottom-up*: for each node l, the positive instances are all the instances associated to l, while the negatives are at most equal to the amount of positives, taking them first from its siblings, then from uncles and so on.

Making use of these policies is not mandatory, that is, variants of them and new ones are allowed.

In order to obtain a consistent path, the procedure TD is also compatible with this approach, that is, the prediction starts in the root node advances toward the child node with the highest score, and so on until a leaf node is reached.

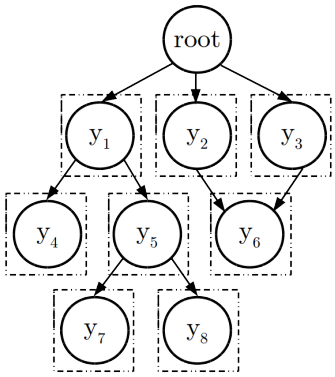


Figure 2.5: LCN approach. For each node (except the root) a binary classifier is trained.

2.1.2.4 Local classifier per level approach

Local classifier per level approach consists in training a multiclass classifier for each level of the hierarchy. This approach can be directly applied to hierarchies of tree type, since the level of each node can be calculated by adding the number of edges from the node to the root. However, determining the level of a node in a hierarchy of DAG type is not straightforward, because there are multiple paths to reach some nodes (nodes with multiple parents). In this work, the level of a node with multiple parents is the level of parent node with the greatest level plus one.

On the other hand, this approach is much less used than the previous approaches.

2.1.2.5 Hierarchical classification methods

Several methods have been proposed for the different hierarchical classification problems, some approaches that have been researched are: improve standard methods, modify the hierarchy, modeling probabilistically the hierarchy.

CLUS-HMC was proposed by Vens et al. [2008a], which creates a decision tree where each leaf contains the probability of each node in the hierarchy. Feature selection in a global and local way have been studied by Naik and Rangwala [2016], however, a TD procedure is still used in the prediction phase.

Some works addressed the problem of *error propagation* of the TD procedure, by proposing different ways of evaluation the different paths, such as P_{path}, descending order of probabilities (DOP), multiplication of probabilities (MP) and sum of probabilities (SP) [Kosmopoulos et al., 2015, Hernandez et al., 2013]

Other studies addressed the problem of hierarchical classification by consider-

ing that the hierarchies, usually designed by people, are not ideal/perfect; hence they modified the hierarchy by adding, *rewiring*, or deleting nodes [Naik and Rangwala, 2016, Perera et al., 2018]. Nakano et al. [2017] proposed a different way of modifying the hierarchy, which consists on replicating the internal nodes and adding them as subclasses of themselves. Two variants were proposed, *non-leaf local classifier per parent node* (nLLCPN) and *local classifier per parent node and branch* (LCPNB), both train LCPN for the modified hierarchy, however the first makes use of the TD procedure in the prediction phase, while the second use the measure SP to score paths. Later, Panta et al. [2019] analyzed the performances of LCPNB and nLLCPN using several base classifiers, where support vector machines (SVM) outperformed the rest of them.

Some works consider that the classifiers, in the local approaches of hierarchical classification, must not be independent and should share information among them. Hence, the local classifiers are related by using the strategy of Bayesian chained classifiers, Ramírez-Corona et al. [2016] applies it to LCPN, and Serrano-Pérez and Sucar [2019] to LCN.

Other works consider a Bayesian approach, so they focus on modeling the hierarchy as a Bayesian network which can be fed by independent classifiers or *related* classifiers [Serrano-Pérez and Sucar, 2019, Barutçuoglu et al., 2008, Decoro et al., 2007].

Recent applications of hierarchical classification include: the problem of diabetic retinopathy [Mukti et al., 2018], classification of sparkling wine [Yamashita et al., 2019], fashion image classification [Seo and shik Shin, 2019], identification of COVID-19 [Pereira et al., 2020], detection of arthropod species [Tresson et al., 2021], classification of insects [Bjerge et al., 2023], and classification of news articles [Petukhova and Fachada, 2023].

2.2 Semi Supervised Learning

Semi-Supervised Learning (SSL) can be seen as the branch of machine learning that aims to combine supervised and unsupervised learning [Chapelle et al., 2010, Zhu, 2008]. That is, SSL uses labeled and unlabeled data to perform learning tasks.

Semi-supervised classification methods are appropriated to scenarios where labeled data is scarce, and a reliable classifier could be hard to obtain. Scarce labeled data occurs when it is expensive or difficult to obtain, like computer-aided diagnosis, drug discovery and part-of-speech tagging [Gui et al., 2023, Yang et al., 2023, Gomes et al., 2022, van Engelen and Hoos, 2019].

2.2.1 Assumptions of Semi-Supervised Learning

In SSL there are some recognized assumptions, which are the foundation of most semi-supervised learning algorithms, which depend on one or more of them being satisfied (explicitly or implicitly) [Chapelle et al., 2010]. They are briefly described below:

- **Smoothness assumption**: It states that, for two input points $x_i, x_j \in X$ that are close by the input space, the corresponding labels y_i, y_j should be the same. This assumption can be applied transitively to unlabeled data. For example, let $x_0 \in X$ be a labeled point and let $x_1, x_2 \in U$ be unlabeled points, if x_1 is close to both x_0 and x_2, but x_0 is not close to x_2, x_2 can be labeled with the same label than x_0, that is, the label was transitively propagated through x_1.
- **Low-density assumption**: It states that the decision boundary of a classifier should preferably pass through low-density regions in the input space. In other words, the decision boundary should not pass through high-density areas.
- **Manifold assumption**: It states that the input space is composed of multiple lower dimension manifolds on which all data points lie. Data points on the same manifold have the same label.

An example of the smoothness and low-density assumption is shown in Fig. 2.6 a), another example related to manifold assumption is shown in Fig. 2.6 b).

2.2.2 Taxonomy of Semi-Supervised Learning Methods

van Engelen and Hoos [2019] proposed a taxonomy to group the SSL methods, the taxonomy is shown in Fig. 2.7. First, the SSL methods are divided into two main groups, inductive and transductive, the former produces a classification model (which can be used to label new instances), while the second is only focused with labeling the unlabeled data points.

The inductive methods are the most interesting for our research, because a classification model for labeling *new* data, different from the training set (labeled and unlabeled data), is required. Inductive methods can be grouped based on the way they incorporate the unlabeled data:

- **Wrapper methods**: They are the most well known algorithms for SSL. The procedure commonly consists of two alternating steps, *training*: one or more classifiers are trained with labeled and pseudo-labeled data (if available), and *pseudo-labeling*: the resulting classifiers are used to infer labels for previously

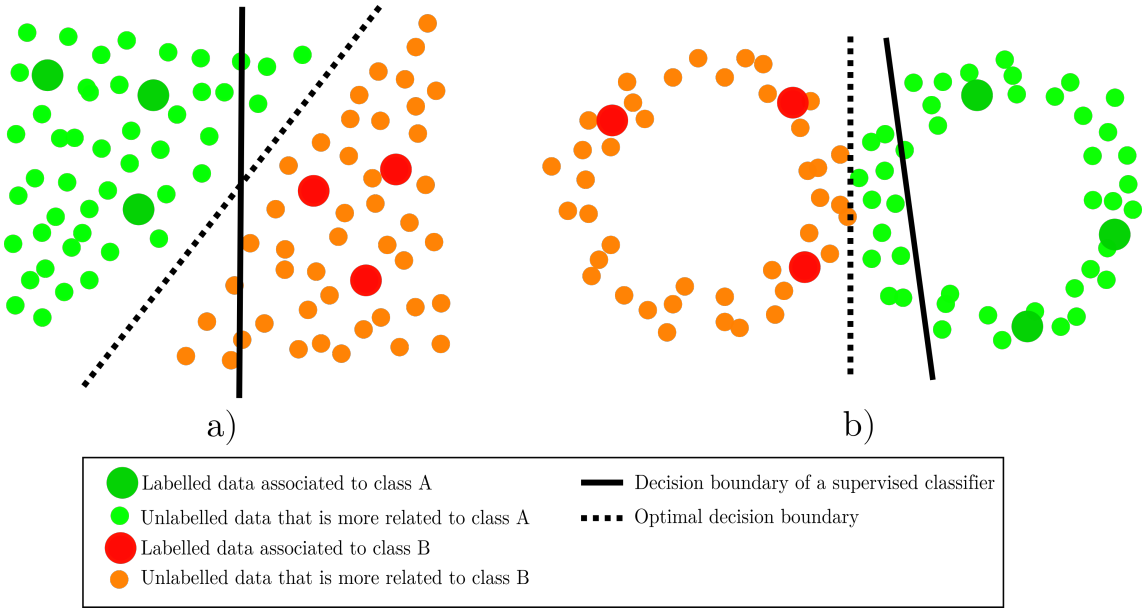


Figure 2.6: Semi-supervised learning assumptions [van Engelen and Hoos, 2019]. a) Smoothness assumption: close points by the input space must have the same label. a) Low-density assumption: the decision boundary must pass through low-density areas as the dotted line does. b) Manifold assumption: each *circle* represents a manifold, so points lying in the same manifold must have the same label. (Best seen in color.)

unlabeled data and those with the most confident predictions are pseudo-labeled and used in the next iteration. Wrapper methods can be divided into the following categories:

- *Self-training*: They consist of a single classifier iteratively trained with labeled and pseudo-labeled data.
- *Co-training*: It is an extension of self-training to multiple supervised classifiers, that is, two or more classifiers are trained on labeled data, and each one adds its most confident predictions to the labeled data of the other classifiers in each iteration. However, in co-training is important that the base classifiers are not strongly correlated in their predictions, in order to provide each other with useful information, this condition is called the *diversity criterion* [Wang and Zhou, 2010].
- *Boosting*: They are based on ensembles of multiple classifiers, following the boosting approach. Hence, pseudo-labeled data is added after each learinig step.

- **Unsupervised preprocessing**: They extract useful features from unlabeled data, pre-cluster the data or determine the initial parameters of a supervised method in a unsupervised manner. That is, the supervised classifier is only

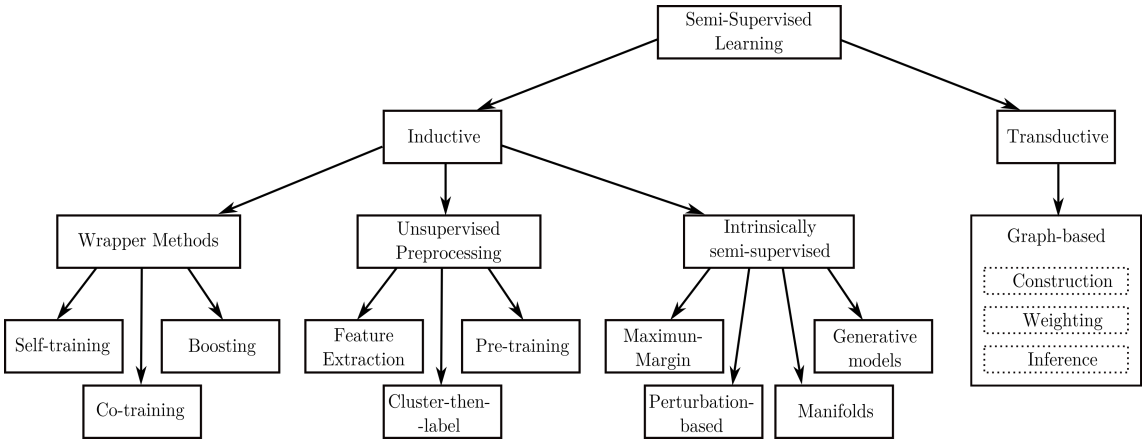


Figure 2.7: A taxonomy that groups the different approaches of semi-supervised learning [van Engelen and Hoos, 2019].

trained with the labeled data.

- **Intrinsically semi-supervised methods**: They directly optimize an objective function with elements for labeled and unlabeled instances. Intrinsically semi-supervised methods can be divided into the following categories:
  - *Maximum-margin methods*: This approach is focused in the low-density assumption. That is, these classifiers attempt to maximize the distance between the given data and the decision boundary.
  - *Perturbation based methods*: A predictive model should be robust to local perturbation in its input, because of the smoothness assumption. That is, if a data point is perturbed with a *small* amount of noise, the prediction for the noise and *clean* data points should be the same.
  - *Manifolds*: This approach is based on the manifold assumption. Two different techniques are presented, *manifold regularization techniques* that define a graph over data points and penalize differences in predictions for instances with small geodesic distance, and *manifold approximation techniques* that explicity estimate the manifolds where the data lie while optimize an objective function.
  - *Generative Models*: The main goal of these methods is to model the distribution that generated the data.

2.3 Semi-Supervised Hierarchical Classification

Formally, we define semi-supervised hierarchical classification as a tuple $SSHC = (HS, (X, Y), U)$, where:

- $HS = (L, E)$ is the DAG that represents the hierarchy, where L is the set of nodes and E is the set of edges that link the nodes.
- (X, Y) is the labeled set. $X = \{x_1, x_2, ..., x_n\}$ contains n instances, where $x_i \in \mathbb{R}^d$, that is, each instance x_i is described by a vector of d attributes. $Y = \{y_1, y_2, ..., y_n\}$ contains the labels for X, where $y_i \in \{0, 1\}^{|L|}$, that is, each y_{ij} indicates whether the i-th instance is associated to the j-th label, while y_i satisfies the hierarchical constraint.
- $U = \{x_{n+1}, x_{n+2}, ..., x_{n+m}\}$ is the unlabeled set, which contains m instances described by the same d attributes as in X.

As it can be seen, semi-supervised hierarchical classification is composed by three main elements, a hierarchy, a labeled set and a unlabeled set.

The problem of *semi-supervised hierarchical classification* consists in assigning to a particular object described by d attributes, a subset of labels that comply with the hierarchical constraint:

$$f_{SSHC} : \mathbb{R}^d \rightarrow \{0, 1\}^{|L|} \tag{2.3}$$

However, in hierarchical classification where the instances are associated to multiple paths of labels [Giunchiglia and Lukasiewicz, 2020, Cerri et al., 2016, Vens et al., 2008b], the problem is commonly *modified* to assign to a particular instance the probability of being associated to each node of the hierarchy:

$$f_{SSHC} : \mathbb{R}^d \rightarrow [0, 1]^{|L|} \tag{2.4}$$

Nevertheless, this prediction has to comply the *hierarchical probability constraint*, which is defined next:

Definition 1 ***Hierarchical probability constraint*** *states that the probability for an instance in the node l has to be equal or lower than the probabilities of all the parent nodes of the node l; let f be a model with one output per node, then:*

$$f_l \leq f_z,\ \forall z \in Parents(l);\ \forall l \in L$$

Finally, the research reported here is born from the hypothesis that training a semi-supervised hierarchical classifier, with labeled and unlabeled data in hierarchical problems of (DAG, MPL, PD) type, will perform better than training a hierarchical classifier only on labeled data.

$$performance(f_{SSHC}) \geq performance(f_{HC}) \tag{2.5}$$

2.4 Transfer Learning

Transfer learning (TL) is commonly used in cases when labeled and unlabeled data are difficult to collect. Hence, TL is focused on transferring the knowledge across domains. Zhuang et al. [2021] formally define TL as follows:

Given some/an observation(s) corresponding to m^s source domain(s) and task(s) (i.e, $\{(\mathcal{D}_{S_i}, \mathcal{T}_{S_i})|i = 1, ..., m^s\}$), and some/an observation(s) about m^T target domain(s) and task(s) (i.e., $\{(\mathcal{D}_{T_j}, \mathcal{T}_{T_j})|j = 1, ..., m^T\}$), transfer learning utilizes the knowledge implied in the source domain(s) to improve the performance of the learned decision functions $f^{T_j} = (j = 1, ..., m^T)$ on the target domain(s).

Note this definition covers both situations, single-source transfer learning and multi-source transfer learning.

When TL techniques are designed three issues should be considered [Aggarwal, 2014, Pan and Yang, 2010]: first, *what to transfer?*, that is, which part of the knowledge can be transferred from source domain to target domain; second, *how to transfer?*, once it is known which knowledge can be transferred, the algorithms for transferring the knowledge need to be developed; and third, *when to transfer?*, this last has to do with in which situations it is appropriate to use transfer learning.

It is worth to mention that in some situations where the source and target domains are no related to each other, the TL may result unsuccessful. In other words, the performance in the target domain could be worsened, this situation is known as *negative transfer*.

Transfer learning using pre-trained models offers several advantages over training one from scratch, for instance, in the context of classification of images using convolutional neural networks (CNN) [Li et al., 2022, Haridas and JyothiR, 2019, Khan et al., 2020]: (i) pre-trained CNNs have already been trained on large data sets, meaning they have learned feature-rich representations for a wide range of images; (ii) less training data required, especially useful when few training data is available for the new task; (iii) less computational resources, because training a network from scratch can be computationally expensive and time-consuming, especially if powerful hardware is not available.

2.5 Evaluation Measures

In order to evaluate the performance of the proposed and related methods, evaluation measures commonly used for hierarchical classification will be used. Several evaluation measures have been proposed [Nakano et al., 2017, Silla and Freitas, 2011]

which assess whether the predictions of the methods were either correct or partially correct.

First of all, let N be the number of instances in the test set, let Y be the real subset of labels to which an instance is associated, let $\widehat{Y}$ be the subset of predicted labels and let $|L|$ be the number of labels. The following evaluation measure are used to asses the performance of the methods in hierarchical problems where the hierarchy is of tree type:

- **Exact Match (EM)**: It is the most strict measure, because, the prediction of an instance has to be equal to real subset of labels. So, it returns the percentage of instances classified correctly.

$$EM = \frac{1}{N}\sum_{i=1}^{N} 1_{Y=\widehat{Y}} \tag{2.6}$$

- **Hierarchical F-measure (hF)**: F-measure for hierarchical classification.

$$hF = \frac{2 * hP * hR}{hP + hR} \tag{2.7}$$

$$hP = \frac{\sum_{i=1}^{N} \left|Y_i \cap \widehat{Y}_i\right|}{\sum_{i=1}^{N} \left|\widehat{Y}_i\right|} \tag{2.8}$$

$$hR = \frac{\sum_{i=1}^{N} \left|Y_i \cap \widehat{Y}_i\right|}{\sum_{i=1}^{N} |Y_i|} \tag{2.9}$$

  Where hP is the hierarchical Precision, which calculates the ratio of correct predictions over the number of predictions in the complete dataset, and hR is hierarchical Recall, which calculates the ratio of correct predictions over the number of real labels in the complete dataset.

- **Matthews correlation coefficient (MCC)**[Chicco and Jurman, 2020]: MCC has the advantage that it is unaffected by the unbalanced datasets issue. It ranges in the interval $[-1, 1]$, it reaches 1 in perfect classification and -1 in perfect missclasification. (Refer to Chicco and Jurman [2020] for special cases).

$$MCC = \frac{1}{\mid L \mid}\sum_{i \in L} MCC_i \tag{2.10}$$

$$MCC_i = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{2.11}$$

On the other hand, when the outputs of the methods are the probabilities for each node of the hierarchy, a common practice when the hierarchy is of DAG type and the instances are associated to multiple paths [Sun et al., 2017, Cerri et al., 2016, Robinson et al., 2015, Cerri et al., 2014, Bi and Kwok, 2011, Schietgat et al., 2010], other evaluation measures are used. That is, it is not suitable to use evaluation measures such as *hF, hR* and *hR* [Silla and Freitas, 2011, Nakano et al., 2017], since they require binary values (associated, not associated) for each node. Even though, a threshold could be applied to the output of the model to get binary values, it is not straightforward to set the threshold, moreover, different thresholds could produce different results.

In this case, the evaluation measure *area under the average precision and recall curve* $AU(\overline{PRC})$ also known as *average precision* (AP) [Zhu, 2004] is used to evaluate the performance of the models:

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{2.12}$$

where P_n and R_n are the precision (Eq. 2.13) and recall (Eq. 2.14) at the n-th threshold, respectively.

$$P = \frac{\sum_{i=1}^{|L|} TP_i}{\sum_{i=1}^{|L|} TP_i + \sum_{i=1}^{|L|} FP_i} \tag{2.13}$$

$$R = \frac{\sum_{i=1}^{|L|} TP_i}{\sum_{i=1}^{|L|} TP_i + \sum_{i=1}^{|L|} FN_i} \tag{2.14}$$

AP is an evaluation measure independent of a threshold to determine whether an instance is associated to a node, which makes it ideal in this kind of scenarios.

2.6 Summary

This chapter presented the fundamentals for this research. First, an introduction to hierarchical classification was given in section 2.1, where the different hierarchical classification problems and the main hierarchical classification approaches were presented. Second, an introductions to semi-supervised learning was given in section 2.2, where the assumptions on which most SSL methods are based on were shown, also, a taxonomy of the SSL was presented. Third, the general formulation of transfer learning was presented in section 2.4. Finally, the evaluation measures used in this work were presented in section 2.5.

Chapter 3

Related Work

This chapter presents the related work divided into two sections, *non-hierarchical* (section 3.1) and *hierarchical* (section 3.2) semi-supervised methods. Finally, a discussion and analysis about them is given in section 3.3, which includes a comparison between related works and ours.

3.1 Non-Hierarchical Semi-Supervised Methods

The self-training approach is one of the most popular semi-supervised methods. It was first proposed by Yarowsky [1995] for word sense disambiguation in text documents. It has also been applied to semantic segmentation [Yang et al., 2022], speech recognition [Xu et al., 2021], object detection [Yang et al., 2021], speech translation/recognition [Pino et al., 2020, Kahn et al., 2020], among others.

In co-training methods, we can find *multi-view co-training* methods [Liu et al., 2024, Shi et al., 2020, Xu et al., 2013, Blum and Mitchell, 1998] where multiple *views* (subsets of features) exist; *single-view co-training* methods [Du et al., 2011, Jiao Wang et al., 2008] where there is only a view, so methods split the data into multiple views; and *co-regularization* methods [Sindhwani and Rosenberg, 2008, Sindhwani et al., 2005] where the ensemble quality and the disagreement between base learners are simultaneously optimized. In similar fashion, Feng et al. [2022] proposed dynamic mutual training to address the noisy of pseudo instances by a re-weighted loss function based on the inter model-disagreement. Xu et al. [2023] proposed to use pseudo-negative labels, that are shared among submodels.

Several semi-supervised boosting methods have been proposed such as *SSM-Boost* [Grandvalet et al., 2001, d'Alché-Buc et al., 2002], *Adaptative Supervised Ensemble* [Bennett et al., 2002] and *SemiBoost* [Mallapragada et al., 2009].

There are some methods based on the low density assumption, which try to extend support vector machines to the semi-supervised: S3VM [Bennett and Demiriz, 1998, Ding et al., 2017] and S4VM [Li and Zhou, 2015]. Furthermore, SSL methods based on artificial neural networks have been proposed, where they added an additional term to its loss function to consider the unlabeled data: Han et al. [2021] applied it to COVID-19 detection in CT images, while MSleepNet [Liu et al., 2024] was proposed for sleep arousal and sleep stage detection.

There is a class of SSL methods known as graph-based semi-supervised learning methods [Song et al., 2023, Chong et al., 2020], which represent each sample as a node in an affinity graph, then the pseudo-labels for unlabeled instances can be inferred based on the structure of the constructed graph. However, most of this kind of methods are transductive [Kang et al., 2021, Wan et al., 2021, Feng et al., 2020]. Likewise, Li et al. [2023, 2019] proposed semi-supervised graph classification via cautious iteration (SEAL-CI), which is an iterative method that builds or updates two classifiers: one at *graph level* instance, where the instances are represented by graphs; and other at the *hierarchical level*, where the instances of the previous level are used as nodes, which form an *undirected* graph.

3.2 Hierarchical Semi-Supervised Methods

The first method for semi-supervised hierarchical classification was proposed by Metz and Freitas [2009]. It is a Top-Down (Tree, SPP, NMLNP) with LCN, where the positive instances of a label are those associated to the label or its descendants, and the negative instances are those associated to its siblings or its siblings descendants. Each local classifier is self-trained following one of three strategies: *self-train A*: all the instances in the labeled set are used, *self-train B*: the instances classified as positive in the parent class are used, and *self-train C*: only the instances classified as positive instances during the self-train step on the parent node are used as unlabeled set in the children nodes. Also, they state there is not statistically significant difference among their proposed strategies and a standard top-down classifier.

Santos and Canuto [2014a] proposed Hierarchical Multi-label classification using Semi-Supervised Label Powerset (HMC-SSLP). First, a HMC-Label Powerset (HMC-LP) [Cerri et al., 2009] is trained with labeled data, then it is used to label a (predefined) proportion of the unlabeled data which are added to the training set, this process is iterated until all the unlabeled data are labeled. HMC-LP [Cerri et al., 2009] combines all the classes of an example to generate a new hierarchy, nevertheless, examples of how to combine paths of different lengths are not given. On the other hand, the positive instances for each new class could be too few, which can result on unreliable classifiers. So, using the HMC-SSLP for semi-supervised hierarchical classification does not seem a good option in any way.

Furthermore, Santos and Canuto [2014a] proposed Hierarchical Multi-label using Semi-Supervised Random k-Labelsets (HMC-SSRAkEL) which is the semi-supervised version of HMC-RAkEL [Santos and Canuto, 2014a]. This method trains LCPN, that is, for each parent node a RAkEL (multi-label) classifier [Tsoumakas et al., 2011] is trained. Then, a Top-Down procedure is use to pseudo-label a (predefined) proportion of unlabeled data, which are added to the training set, this process is iterated until all the unlabeled data are labeled.

Hierarchical Multi-label Classification using Semi-Supervised Binary Relevance (HMC-SSBR) was proposed by Santos and Canuto [2014b], which is the semi-supervised version of HMC-BR [Cerri et al., 2009] (a Top-Down method with LCN, siblings policy). However, Santos and Canuto indicate that BR is replaced by SSBR [Santos and Canuto, 2012], a semi-supervised method for multi-label classification, that is, each node of the HMC-BR is self-trained and the prediction for each unlabeled data follows the Top-Down procedure.

The Top-Down procedure for labeling instances in HMC-SSRAkEL and HMC-SSBR can predict multiple labels in the *same* level, that is, it advances for different paths if the instance is classified as positive in the following label/node. On the other hand, for HMC-SSRAkEL, HMC-SSBR and HCM-SSLP methods a proportion of unlabeled data to be labeled in each iteration is predefined, the proportions used are 17%, 33% and 50%, which results in 6, 3 and 2 iterations, respectively. That is, when labels are predicted for unlabeled data, instead of choosing those with the most confident predictions, all of them are pseudo-labeled and added to the labeled data; so, these methods lack a way to select the instances with the most confident predictions.

Xiao et al. [2019] proposed Path Cost-Sentive Algorithm with Expectation-Maximization (PCEM) which consists in the following steps. First, the base classifier Path Cost-Sentive Naive Bayes Classifier (PCNB) [Xiao et al., 2019] is trained with the labeled data, then it pseudo-labels all the unlabeled instances and trains the PCNB with labeled and pseudo-labeled instances, this is iterated until the parameters of the PCNB converge. PCNB was proposed for hierarchical text classification, where the document representation is a vector called bag-of-word, that is, the number of attributes is equal to the number of words in the corpus, and each cell contains the frequency of the word in the document. A limitation of this method is that it is not straightforward to apply it in non-text domains, because PCNB is designed using the bag-of-word representation.

Levatić et al. [2024] proposed Semi-Supervised Predictive Clustering Trees (SSL-PCT) method, which is based on predictive clustering trees (PCT) [Breiman et al., 1984, Blockeel et al., 1998]. PCT's consist of a hierarchically organized set of clusters, where the root cluster is recursively divided into smaller cluster as one goes deeper to the leaves. They reported that the results of SSL-PCT were not so

successful on the functional genomics datasets, because the supervised hierarchical classifier was rarely outperformed. Moreover, this work carried out a transductive study, because data used for training was also used to evaluate the performance of the methods, i.e. the unlabeled data.

3.3 Analysis

Table 3.1 shows a comparison among the related methods and the proposed methods in this work. Non-hierarchical methods were summarized in the first row, since they were not proposed for hierarchical classification, therefore, they can not handle the hierarchy and they do not guarantee to comply the hierarchical constraint.

The proposed methods are different by the type of paths that they predict, the first predicts a single path of labels while the second is able to predict multiple path of labels. However, we highlight the following points:

- **Hierarchical Structure**: The related works were proposed for problems with hierarchy Tree type, which is a limitation of the methods. The proposed method will overcome this limitation and be able to work in any hierarchy (DAG type).
- **Number and depth of paths**: The related works are proposed for a predefined number of paths and depth. In this way, the proposed methods are able to predict either a single (full depth) or multiple (partial depth) paths of labels, SSHC-BLI and SSHMC-BLI respectively.
- **Labeling strategy**: a labeling strategy to pseudo-label the unlabeled data with the most confident predictions is required. Moreover, the labeling strategy has to consider the information provided by the hierarchy. Note that some related works lack of a way to pseudo-label only the data with the most confident prediction.
- **Transfer learning**: None of the related methods carries out transfer learning, while a variant of the proposed method is able to transfer learning from parent to child in hierarchies of DAG type.

As it can be seen, we address the hierarchical problem with hierarchy of DAG type and instances associated to multiple path of labels with partial depth following an inductive SSL approach, which has not been addressed before. Furthermore, we explore the case of transfer learning among local nodes in this semi-supervised hierarchical case.

Table 3.1: Comparative list of the related and proposed method. Υ: hierarchical structure, Ψ: number of paths, Φ: depth of paths, LS: labeling strategy to pseudo-label only instances with the most confident predictions, TL: transfer learning, N/A: not applicable.

Method	Υ	Ψ	Φ	**LS**	**SSL approach**	**TL**
Non-Hierarchical (section 3.1)	N/A	N/A	N/A	-	-	No
Metz and Freitas [2009]	Tree	SPL	FD	self-train {A,B,C}	Inductive	No
HMC-SSLP [Santos and Canuto, 2014a]	Tree	MPL	PD	No	Inductive	No
HMC-SSRAkEL [Santos and Canuto, 2014a]	Tree	MPL	PD	No	Inductive	No
HMC-SSBR [Santos and Canuto, 2014b]	Tree	MPL	PD	No	Inductive	No
PCEM [Xiao et al., 2019]	Tree	SPL	FD	No	Inductive	No
SSL-PCT [Levatić et al., 2024]	DAG	MPL	PD	No	Transductive	No
SSHC-BLI (proposed) [Serrano-Pérez and Sucar, 2024]	Tree	SPL	FD	SISI	Inductive	No
SSHMC-BLI (proposed) [?]	DAG	MPL	PD	SISI	Inductive	Yes

Chapter 4

Datasets

Several datasets were used to asses the performance of the proposed method and its variants. The datasets are split into two main groups: those with hierarchy of tree type and those with hierarchy of DAG type, sections 4.1 and 4.2, respectively. Description and results of the proposed methods are presented later in the corresponding chapters.

4.1 Trees

Artificial and real-world datasets from different fields were collected, all of them have associated a hierarchy of tree type, and the instances are associated to a single path of labels that always reach a leaf node, that is, they are hierarchical problems of (Tree, SPL, FD) type.

Real world datasets from the field of functional genomics were collected, the datasets are a subset of the *Functional Catalogue*[1] (FunCat) [Vens et al., 2008b], these datasets are divided into training, validation and test sets. The preprocessing applied to the datasets consisted on removing nodes with less than 40 instances in the training set, then the same nodes were also removed from validation and test sets, as *preprocessing 2* (Tree, SPL, FD) in Serrano-Pérez and Sucar [2019]. Description of FunCat datasets is shown in Table 4.1.

The *20 newsgroup*[2] dataset is a collection of approximately 20,000 documents, which is commonly used for text classification tasks. The hierarchy of this dataset is depicted in Fig. 4.1, which has 20 leaf nodes and 7 internal nodes, where related subjects are grouped together. Since the data is provided as raw text, the preprocessing

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

[2] `http://qwone.com/~jason/20Newsgroups/`

Table 4.1: Description of the FunCat and 20 newsgroup datasets. Attr.: number of attributes; Nodes: number of nodes in the hierarchy; MD: Maximum Depth.

Dataset	Training	Validation	Test	Attr.	Nodes	MD
cellcycle_FUN	1116	541	877	77	27	3
derisi_FUN	1141	556	905	63	26	3
eisen_FUN	806	387	631	79	22	3
gasch1_FUN	1120	540	881	173	27	3
gasch2_FUN	1121	542	887	52	27	3
20 newsgroup	9051	2263	7532	50	27	3

in Appendix A was applied to the dataset. The last row of Table 4.1 describes the 20 newsgroup dataset.

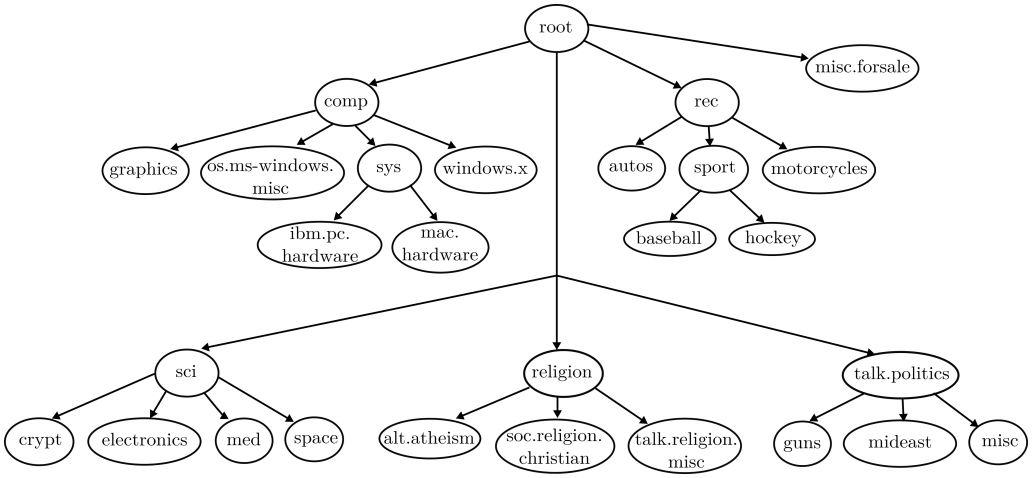


Figure 4.1: Hierarchy of the 20 newsgroup dataset.

Furthermore, the arficial datasets of Serrano-Pérez and Sucar [2021] [3] where the hierarchy is of tree type are considered. The artificial datasets are grouped by difficulty: easy, hard and very hard, that makes them useful to extend the analysis of the methods. In order to be consistent with previous datasets, the training set was divided into *training* and *validation* sets, with proportion 80% and 20%, respectively. The artificial datasets are described in Table 4.2.

Finally, in order to observe the behavior of the proposed method with different amounts of labeled and unlabeled data, the training sets were split into labeled and unlabeled sets:

- Labeled: {10, 30, 50, 70, 90}%

[3] `https://github.com/jona2510/ADforHC`

Table 4.2: Description of artificial datasets. Attr.: number of attributes; Nodes: number of nodes in the hierarchy; MD: Maximum Depth.

Dataset	Training	Validation	Test	Attr.	Nodes	MD
EA_01_FD_b	192	48	60	4	9	2
EA_01_FD_ub	832	208	260	4	9	2
EA_02_FD_b	192	48	60	10	9	2
EA_02_FD_ub	832	208	260	10	9	2
HA_01_FD_b	448	112	140	3	23	6
HA_01_FD_ub	1056	264	330	3	23	6
HA_02_FD_b	448	112	140	13	23	6
HA_02_FD_ub	1280	320	400	13	23	6
HA_03_FD_b	448	112	140	9	23	6
HA_03_FD_ub	1280	320	400	9	23	6
HA_09_FD_b	2816	704	1760	2	140	10
HA_09_FD_ub	8576	2144	5360	2	140	10
HA_10_FD_b	2624	656	1640	7	161	20
HA_10_FD_ub	7168	1792	4480	7	161	20
VH_01_FD_b	288	72	90	3	13	4
VH_01_FD_ub	528	132	165	3	13	4
VH_02_FD_b	224	56	70	10	10	3
VH_02_FD_ub	880	220	275	10	10	3
VH_03_FD_b	672	168	210	16	28	2
VH_03_FD_ub	1567	389	489	16	28	2
VH_08_FD_b	2016	504	1260	5	112	10
VH_08_FD_ub	5632	1408	3520	5	112	10

- Unlabeled: {90, 70, 50, 30, 10}%, complement with respect to labeled.

This split is shown properly in Fig. 4.2. The division of training set was randomly carried out 3 times, so results are the average of 3 executions, while the validation set was used for tuning the hyper-parameters of each method.

4.2 DAGs

Real world datasets from the field of functional genomics were collected, these datasets belong to the Gene Ontology[4] (GO) collection [Vens et al., 2008b]. The labels of each datasets are arranged in a hierarchy of DAG type, that is, some nodes

[4] `https://dtai.cs.kuleuven.be/clus/hmcdatasets/`

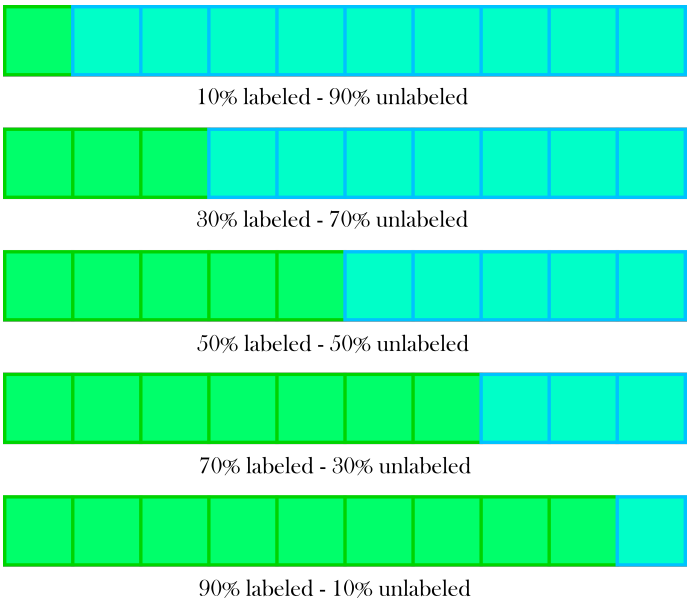


Figure 4.2: Division of the training set into labeled (green) and unlabeled (blue) sets. (Best seen in color)

have multiple parents. Furthermore, the instances can be associated to multiple paths of labels which can finish in internal nodes.

The datasets were preprocessed in similar fashion than Ramírez-Corona et al. [2016], that is, nodes with less than 50 instances associated in the training set were removed, then, the same nodes were removed from validation and test sets; however, in this case, all the paths to which instances are associated are kept. Description of GO datasets is shown in Table 4.3; the datasets are described as (*DAG, MPL, PD*).

Finally, in the same way than the datasets with hierarchy of tree type, the training sets were split into labeled and unlabeled sets in the proportions $\{10, ..., 90\}\%$. The split of training set was carried out 3 times, so results are averages.

4.3 Summary

In this chapter the datasets that are used to evaluate the performance of the different methods were described. The datasets are divided into two groups: those with hierarchy of tree type, where the instances are associated to a single path of labels of full depth, section 4.1; and those with hierarchy of DAG type, where the instances can be associated to multiple paths of labels of partial depth, section 4.2.

Table 4.3: Description of Gene Ontology datasets. *Attr.* shows the number of attributes; *Nodes* shows the number of nodes/labels in the hierarchy; and *MD* correspond to the maximum depth of the hierarchy.

Dataset	**Training**	**Validation**	**Test**	**Attr.**	**Nodes**	**MD**
cellcycle_GO	1625	848	1278	77	164	9
church_GO	1627	844	1278	31	164	9
derisi_GO	1605	842	1272	63	161	9
eisen_GO	1055	528	835	79	122	9
expr_GO	1636	849	1288	565	165	9
gasch1_GO	1631	846	1281	173	165	9
gasch2_GO	1636	849	1288	52	165	9
hom_GO	1661	867	1309	47034	166	9
pheno_GO	653	352	581	276	68	7
seq_GO	1692	876	1332	530	171	9
spo_GO	1597	837	1263	89	162	9
struc_GO	1659	859	1306	19628	169	9

Chapter 5

Semi-Supervised Hierarchical Classifier Based on Local Information

Semi-Supervised Hierarchical classifier Based on Local Information[1] (SSHC-BLI) [Serrano-Pérez and Sucar, 2024, Serrano-Pérez and Sucar, 2022] is the proposed method for semi-supervised hierarchical classification, which is based on the smoothness assumption, that is, neighboring instances must have the same or similar paths of labels. SSHC-BLI starts building the pseudo-label for each unlabeled instance using the labels of its neighbors, but if the unlabeled instance is not similar to its labeled neighbors, it will stay unlabeled; this process iterates until all the pseudo-labels do not change.

In semi-supervised hierarchical classification, pseudo-labeling the unlabeled instances using a flat approach is not the best way, because it will leave instances unlabeled that may be partially pseudo-labeled. For example, Fig. 5.1a depicts a hierarchy formed by nine nodes and Fig. 5.1b shows an unlabeled instance and three labeled instances, then, Fig. 5.1c calculates the probabilities for each label in a flat approach, since the hierarchy is ignored, the three labels got the same score which is not useful to build a pseudo-label, so the unlabeled point will stay unlabeled. In a hierarchical approach, the instance could be pseudo-labeled with a partial path, built from the labels of its neighbors, that fulfills the hierarchical constraint as shown in Fig 5.1d, where the number of times a label is seen is averaged for each label, in this way, the instance would be pseudo-labeled with the nodes 0 and 3, because they got the highest score.

The steps of SSHC-BLI are shown in Algorithm 1. It is an iterative method where the unlabeled instances are pseudo-labeled using the labels of their neighbors (lines 6 - 7), subsection 5.1 details how pseudo-labels are built. The similarity of

[1] Open source available at: https://github.com/jona2510/SSHC-BLI

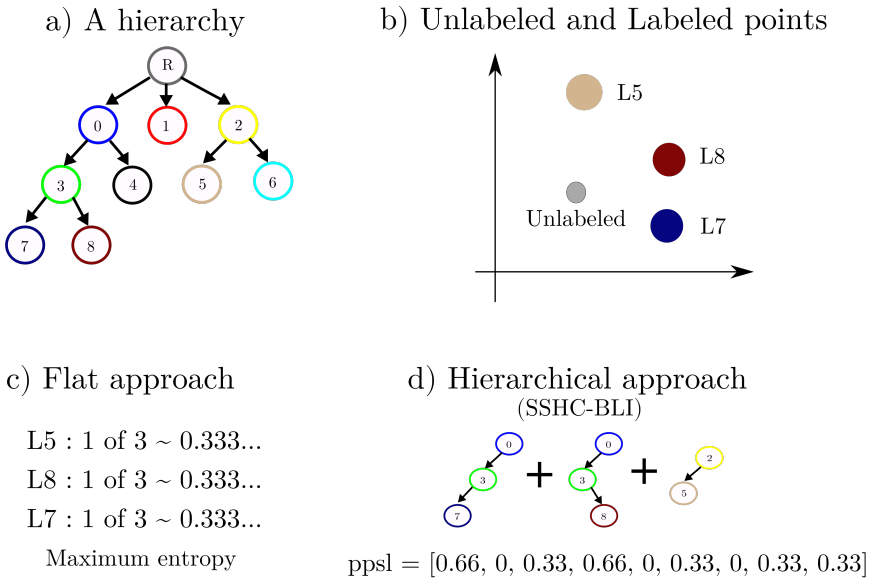


Figure 5.1: Example of pseudo-labeling an instance based on the smoothness assumption. (a) A hierarchy composed of nine labels. (b) an unlabeled point and 3 labeled points (colored with the most specific label with respect to the hierarchy in (a)) are shown. (c) In a flat approach, the unlabeled instance will stay unlabeled because the three instances are different since the hierarchy is ignored. (d) On the other hand, in the hierarchical approach, *ppsl* represents the probability for each label of the hierarchy, so, the instance could be pseudo-labeled with the nodes {0,3} that got the highest probability. (Best seen in color.

each unlabeled instance with its neighbors (line 9) is taken into account: if they are *similar*, then, the unlabeled instance is pseudo-labeled, otherwise it is not; section 5.2 details how the similarity is estimated. The loop finishes (line 11) either when the pseudo-labels of the unlabeled data do not change from an iteration to other, or when the maximum number of iterations is reached. Finally, with labeled and pseudo-labeled instances, a hierarchical classifier can be trained (line 17).

Three variants of the SSHC-BLI method were developed:

- **Variant 1** (V1): follows Algorithm 1 to the letter.
- **Variant 2** (V2): in V1, the pseudo-labels for the unlabeled set are re-estimated in each iteration, so, after the first iteration, an instance that was added to the training set will have itself as one of its labeled neighbors. In order to avoid this case, the function *getNLN* (line 6) is modified to guarantee that none of the labeled neighbors is the instance itself.
- **Variant 3** (V3): increases the value of k by one unit after a predefined number

Algorithm 1 SSHC-BLI

Require: (X, Y): labeled data, U: unlabeled data, k: number of labeled neighbors, THR: similarity threshold, $t2label$: threshold to pseudo-label an instance, HS hierarchy, $maxIterations$: maximum number of iterations.
Ensure: f_{sshc}: semi supervised hierarchical classifier

$T \leftarrow 1$ ▷ Iteration
$LD \leftarrow X$ ▷ LD: Labeled data
$CL \leftarrow Y$ ▷ Labels of labeled data
while $True$ **do**
  for each $u_j \in U$ **do**
    $IND_j \leftarrow getNLN(k, u_j, LD)$ ▷ Get nearest labeled neighbors
    $PSL_j \leftarrow getPseudoLabel(IND_j, LD, t2label)$ ▷ Pseudo label for u_j
  for each $u_j \in U$ with valid PSL_j **do**
    if $similarity(u_j, IND_j) < THR$ **then**
      $PSL_j = \emptyset$ ▷ Invalid pseudo-label
  if $(T > maxIterations)$ or $(PSL^T == PSL^{T-1})$ **then**
    break loop (while)
  else ▷ join labeled data with valid pseudo-labeled data
    $CL \leftarrow Y \cup valid(PSL)$
    $LD \leftarrow X \cup U[valid(PSL)]$
  $T \leftarrow T + 1$
$f_{SSHC} \leftarrow trainHC(LD, CL, HS)$ ▷ Train a hierarchical classifier

of iterations, since the number of instances in the training set could increase in each iteration.

5.1 Pseudo-labeling an instance

To pseudo-label an instance, the labels of instances *close* by the input space to it are required. Let $Y = [y_1, ..., y_k]$ be the labels of k instances close to the unlabeled instance x, then the probabilities for each individual label can be estimated with equation 5.1:

$$ppsl_j = \left\{ \frac{\sum_{i=1}^{k} y_{ij}}{k} \ , \ \forall j \in \{1, ..., \mid L \mid\} \right. \tag{5.1}$$

Then a threshold ($t2label$) is used to determine if an instance is associated to the label:

$$psl_j = \begin{cases} 1: & ppsl_j \geq t2label \\ 0: & ppsl_j < t2label \end{cases}, \ \forall j \in \{1, ..., \mid L \mid\} \tag{5.2}$$

$$0 \leq t2label \leq 1$$

In this way, *psl* is the pseudo-label generated for x. Different threshold may result in different pseudo-labels, for instance, a threshold of 1 will work as an AND function, that is, psl_i will be 1 only if all the k-instances are associated to the i-th label; in similar way, a threshold of 0 will work as an OR function, that is, psl_i will be 1 if at least one of the k-instances is associated to the i-th label. On the other hand, if *psl* is full of *zeros*, it means that x stays unlabeled.

5.2 SISI: Similarity of an instance with a set of instances

A similarity function to estimate how similar is an instance to a small set of instances is required; it is also required that the result is in interval $[0, 1]$, where 1 means that the instance is similar to the set of instances and 0 it is not. Nevertheless, it is not known a similarity function that complies the previous requisites to the best of our knowledge.

The heuristic function *Similarity of an Instance with a Set of Instances* (SISI) [Serrano-Pérez and Sucar, 2022] is proposed as a *local* measure, because it does not consider the complete data distribution, but only the instance of interest, $p \in \mathbb{R}^d$, and a set of instances, $X \subset \mathbb{R}^d$. In this way, it is said that the instance p is similar to the set of instances X, if the average of distances from p to each instance in X is equal or lower than the average of the distances among the set of instances in X. The distances among the set of instances X ($lavg(x)$) is calculated as shown in equation 5.3,

$$lavg(X) = \frac{\sum_{i=1}^{k} \sum_{j=i+1}^{k} d(x_i, x_j)}{\frac{k \cdot (k-1)}{2}} \tag{5.3}$$

and the distances of p with the set of instances X is calculated as shown in equation 5.4,

$$uavg(p, X) = \frac{\sum_{i=1}^{k} d(p, x_i)}{k} \tag{5.4}$$

where $x_i, x_j \in X$, k is the cardinality of X and $d(x_i, x_j)$ is any metric[2]. Additionally, two assumptions are made:

Assumption 1 *If $uavg(p, X)$ is lower or equal than $lavg(X)$, the instance p is similar to the set of instances X with score 1.*

Assumption 2 *If $uavg(p, X)$ is greater or equal than n times $lavg(X)$, with $n > 1$, the instance p is not similar to the set of instances X, that is, score 0.*

[2] Axioms that a metric must satisfy can be found in Appendix C.

From assumptions 1 and 2, the equation of the line that passes through points $(lavg, 1)$ and $(n \cdot lavg, 0)$ is defined as:

$$score = \frac{lavg - uavg}{(n-1) \cdot lavg} + 1 \tag{5.5}$$

Equation 5.5 scores the similarity of the point p with the set of instances X in interval $(lavg, n \cdot lavg)$. Finally, the function SISI is defined in equation 5.6 from assumptions 1 and 2, and equation 5.5.

$$SISI(p, X) = \begin{cases} 1 & uavg(p, X) \leq lavg(X) \\ 0 & uavg(p, X) \geq nlavg(X) \\ \frac{lavg(X) - uavg(p,X)}{(n-1) \cdot lavg(X)} + 1 & otherwise \end{cases} \tag{5.6}$$

The general behavior of SISI is shown in Fig. 5.2. Additionally, a comparison of SISI with existing similarity measures is given in Appendix B. On the other hand, for the experiments in this work, n was set to 3, and the *euclidean distance* was used as the metric.

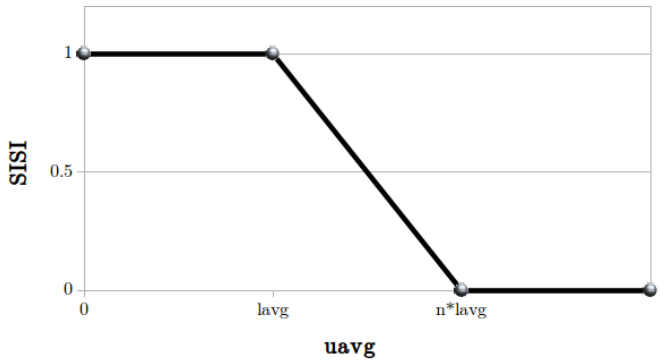


Figure 5.2: Behavior of the function SISI. If $uavg$ is lower or equal than $lavg$ then SISI returns 1; if $uavg$ is greater or equal than $n \cdot lavg$ then SISI returns 0; else, a score is calculated with equation 5.6.

5.3 Experiments and Results

The experiments are focused on showing that using unlabeled data could help to improve the performance of a hierarchical classifier trained only on labeled data. The performance of the proposed method is also compared against the next methods: supervised TD[Silla and Freitas, 2011] classifier (our base line), which is only trained with the labeled data; *self-train A* (STA) which is the strategy with the best performance as reported in Metz and Freitas [2009]; and *hierarchical multi-label classification using semi-supervised binary relevance* (HMC-SSBR) [Santos and Canuto, 2014b], which can be applied to hierarchies of tree type with depth greater than two.

5.3.1 Simple case: SSHC-BLI Behaviour

To illustrate the behavior of the SSHC-BLI variants, a simple artificial dataset was designed. The hierarchy is the one in Fig. 5.1a, that has 9 nodes; the dataset is two-dimensional: 20, 400 and 80 instances were generated[3] for labeled, unlabeled and test sets, respectively. The labeled and unlabeled instances are depicted in Fig. 5.3a, note that the labeled instances are colored with the most specific node of its path.

The SSHC-BLI variants were applied to this dataset with the following hyper-parameters: nearest labeled neighbors, $k = 3$; similarity threshold, $THR = 0.5$; threshold to positively label an instance, $t2label = 0.5$; and for variant 3, k increases by 1 every 5 iterations. Finally, a TD classifier was trained with labeled and pseudo-labeled data, which uses random forest classifiers[4] as LCN, and the *less inclusive* policy (for a node, the positive instances are those associated to the node, while the negatives are the rest) is used to select positive and negative instances in each node.

Fig. 5.3 shows how the SSHC-BLI variants pseudo-labeled the unlabeled set. Variant 1 pseudo-labeled the whole unlabeled set, but it wrongly pseudo-labeled some instances, most of them at the tips of the half moons (circled in blue) as shown in Fig. 5.3b. Variant 2 pseudo-labeled in a better way the tips of the half moons (circled in blue), as shown in Fig. 5.3c, however, there are unlabeled instances surrounded by labeled instances with the same labels, so, they should have the same path of labels, but they were not pseudo-labeled because the estimation of similarity with its neighbors. And variant 3 seems to smooth the results of the 2nd variant, as shown in Fig. 5.3d.

Later, for each variant, the labeled and pseudo-labeled instances were used to train the hierarchical classifier. Hence the results of the SSHC-BLI variants on the test set are shown in Table 5.1, the table also includes the results of the TD classifier. As it can be seen, the results obtained by the semi-supervised variants outperformed the results of the supervised classifier.

5.3.2 Artificial Datasets

The validation set was used for tuning the hyper-parameters of each method. The hyper-parameters and values that each method can take are the following:

- SSHC-BLI variants: similarity threshold (THR): {0.3, 0.5, 0.7}; number of labeled neighbors (k): {3, 4, 5}. Additionally, the threshold to pseudo-label

[3]The function *make moons* of `scikit-learn.org` was used to generate the instances.

[4]Implementation of `scikit-learn.org`, default parameters.

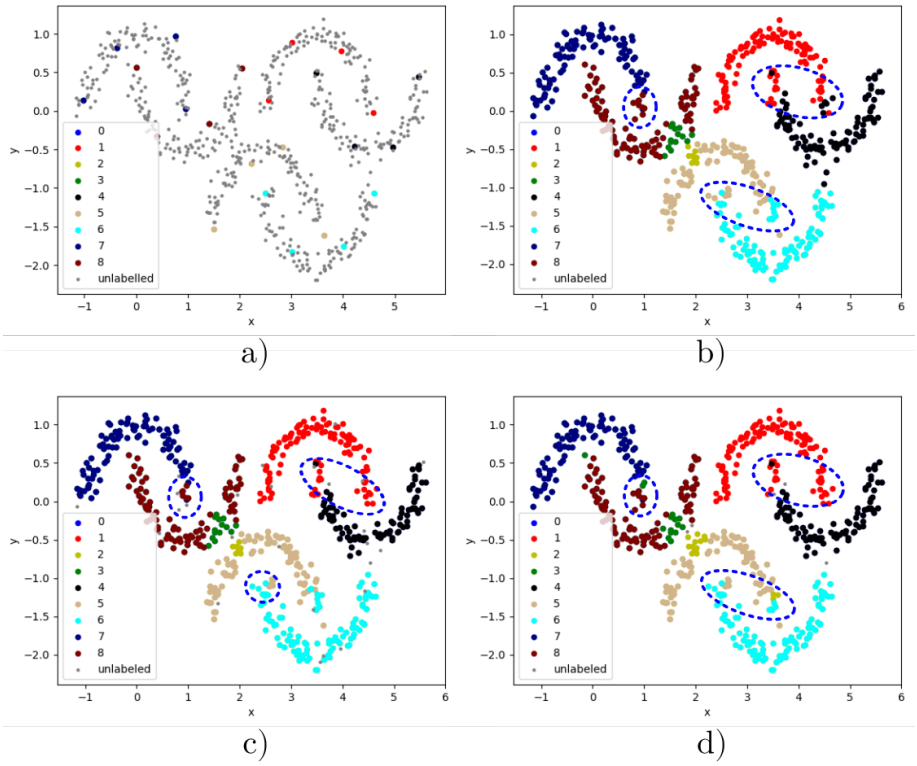


Figure 5.3: Initial configuration of labeled and unlabeled instances and final configuration obtained by the three variants of SSHC-BLI. The (pseudo-)labeled instances are colored with the most specific node of its path, while unlabeled instances are shown in gray. a) Initial labeled and unlabeled data. Pseudo-labels of variant 1 b), variant 2 c), and variant 3 d). Blue eclipses highlight the main pseudo-labels differences between the different variants. (Best seen in color.)

an instance (*t2label*) was set to 0.5, k increases by one unit each 10 iteration (only for variant 3) and the variants make use of a TD classifier that trains random forest classifiers[5] for each node.

- STA: the threshold α: {0.3, 0.5, 0.7}.
- HMC-SSBR: iterations: {2, 3, 6}; C: {1, 10, 100}; gamma: {'scale','auto',1}

The evaluation measure hF was used to determine the best configuration with grid search. Later, each method was trained with the best configuration but only with the labeled and unlabeled sets, the validation set was not used as labeled data. Finally, the methods were evaluated on the test set. Note that the supervised TD is the same than the used by the SSHC-BLI variants.

[5]Implementation of `scikit-learn.org`, default parameters except *n_jobs=5*.

Table 5.1: Results of SSHC-BLI variants (1, 2 and 3) and the TD classifier for the artificial dataset. EM: exact match, hR: hierarchical recall, hP: hierarchical precision, hF: hierarchical F-measure, MCC: Matthews correlation coefficient. In bold the best score.

Eval. Measure	**TD**	**V1**	**V2**	**V3**
EM	0.8367	0.9033	0.9033	**0.9167**
hR	0.9092	0.9415	0.9385	**0.9477**
hP	0.8821	0.9415	0.9428	**0.9521**
hF	0.8955	0.9415	0.9406	**0.9499**
MCC	0.8454	0.92	0.9114	**0.9318**

Fig. 5.4 shows the results obtained, in MCC and hF, by the different methods in a couple of datasets: *HA_02_FD_ub* and *VH_03_FD_ub*. The SSHC-BLI variants tend to outperform the TD classifier, which is only trained in the labeled data; however, the difference of performance is reduced as the amount of labeled information increases.

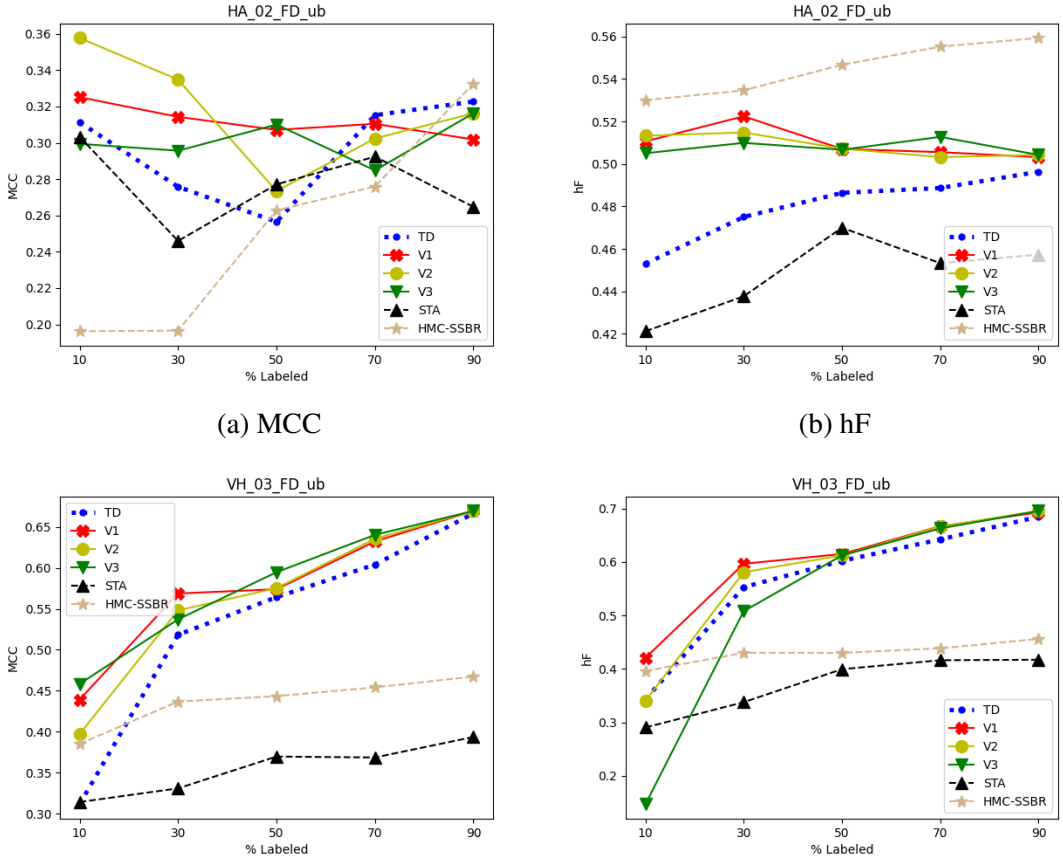
The complete tables of results for the artificial datasets (Table 4.2) on exact match, MCC and hF can be found in Appendix F. Table 5.2 shows the average rank of each classifier on the different evaluation measures. As it can be seen, the variants of the proposed method tend to get better performance than their supervised counterpart, the TD classifier, and the method STA throughout the different amounts of labeled and unlabeled data. Even though, HMC-SSBR performed very well in the easy datasets, it was unable to keep this performance in the hard and very hard datasets, being outperformed in the very hard datasets by V1 of the proposed method in all the evaluation measures but hR.

On the other hand, considering all the artificial datasets, the 22 datasets and their 5 splits, the SSHC-BLI variants got the best performance in the evaluation measures EM, hP and MCC. However, HMC-SSBR showed the best performance in the evaluation measure hR, hence also in hF. The SSHC-BLI variant 1 got the best performance of all the variants, which is the one that allows pseudo-labeled instances to contribute themselves when building a new pseudo-label, which seems beneficial in these artificial datasets.

5.3.3 20 Newsgroup

In the same way than in the previous section, the hyper-parameters of the methods and their results were obtained.

Results in the 20 newsgroup dataset are shown in Fig. 5.5, as it can be seen, the variants of SSHC-BLI tend to outperform their supervised counterpart and the STA



(a) MCC (b) hF

(c) MCC (d) hF

Figure 5.4: Results of artificial datasets. Results in HA_02_FD_ub dataset are shown in a) and b); results in VH_03_FD_ub dataset are shown in c) and d). (Best seen in color)

method when there is at least 30% of labeled data. Table 5.3 shows the average rank of each classifier, where variants 1 and 2 got the best performances in the evaluation measures EM and hR, being V1 the best. On the other hand, HMC-SSBR got the best performance in MCC and hP (hence also in hF).

Furthermore, the results of the SSHC-BLI variants converged with the obtained by the TD classifier as the amount of labeled data increases.

5.3.4 FunCat datasets

The same steps than in Section 5.3.2 were applied to obtain the hyper-parameters and results for the FunCat datasets.

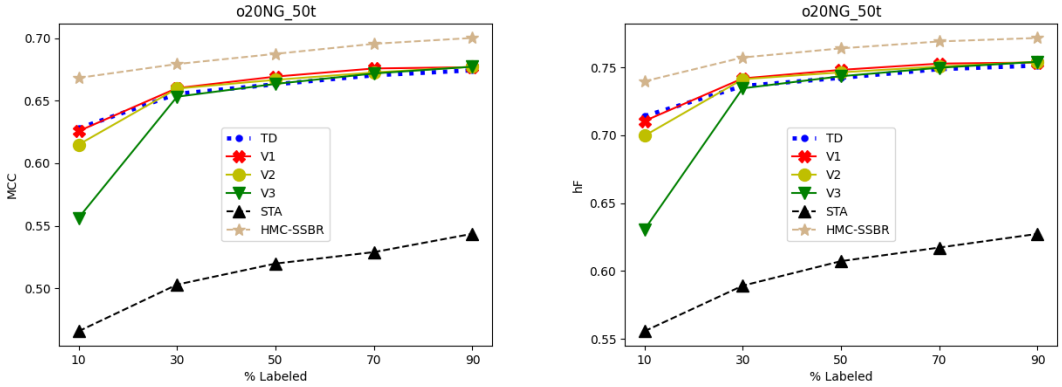
The results in terms of MCC and hF for the FunCat datasets are shown in Figs.

Table 5.2: Average rank of each classifier in different evaluation measures for the artificial datasets. EM: exact match, hR: hierarchical recall, hP: hierarchical precision, hF: hierarchical F-measure, MCC: Matthews correlation coefficient. In bold the best (lower is better).

	Measure	TD	V1	V2	V3	STA	HMC-SSBR
Easy	EM	4.35	3.275	2.95	2.95	6	**1.475**
	hR	4.325	3.3	2.95	2.95	6	**1.475**
	hP	4.325	3.3	2.95	2.95	6	**1.475**
	hF	4.325	3.3	2.95	2.95	6	**1.475**
	MCC	3.775	2.875	2.975	2.975	5.75	**2.65**
Hard	EM	3.39	**2.5**	2.9	2.85	4.78	4.58
	hR	5.44	3.8	3.1	3.52	4.12	**1.02**
	hP	3.46	**2.7**	3.19	2.85	5.34	3.46
	hF	5.26	3.3	3.07	3.35	4.9	**1.12**
	MCC	**2.64**	3.19	3.38	3.12	4.12	4.55
Very Hard	EM	3.75	**2.038**	2.412	3.075	4.888	4.838
	hR	4.8	2.588	2.812	3.5	4.975	**2.325**
	hP	4.25	**2.025**	2.712	2.912	5.475	3.625
	hF	4.65	**2.45**	2.712	3.288	5.3	2.6
	MCC	3.325	2.725	2.762	**2.662**	4.325	5.2
ALL	EM	3.695	**2.473**	2.732	2.95	5.041	4.109
	hR	5.005	3.268	2.968	3.409	4.773	**1.577**
	hP	3.905	**2.564**	2.973	2.891	5.509	3.159
	hF	4.868	2.991	2.918	3.255	5.245	**1.723**
	MCC	3.095	2.964	3.082	**2.927**	4.491	4.441

5.6, 5.7 and 5.8. Each graph compares the results of the SSHC-BLI variants with the related methods and the TD classifier on different percentages of labeled-unlabeled data. As it can be seen, the variants of the proposed method tend to outperform the supervised TD and STA method in the hF measure, moreover, in the MCC measure the variants tend to outperform the same methods mainly when there is few labeled data.

Table 5.4 summarizes the results of the semi-supervised methods and the supervised classifier. The SSHC-BLI variants obtained the best performance in the evaluation measures EM and hR; nevertheless, HMC-SSBR got the best scores in MCC and hP (hence also in hF). In this case, the variant 3 got the best performance among the variants of the proposed method, which gradually increases the number of labeled neighbors when building new pseudo-labels, also, it does not allow to pseudo-labeled instances to make use of themselves when building the new



(a) MCC (b) hF

Figure 5.5: Results of 20 newsgroup dataset in a) MCC and b) hF. (Best seen in color)

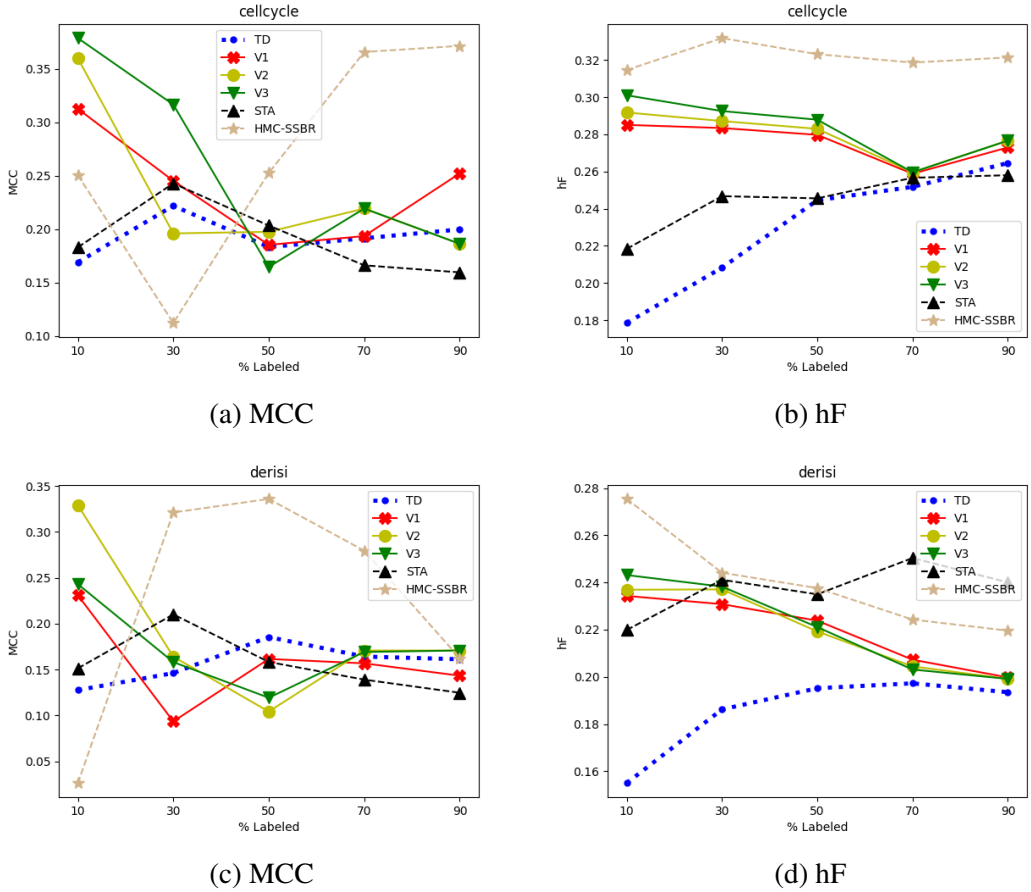
Table 5.3: Average rank of each classifier in the 20 newsgroup dataset. EM: exact match, hR: hierarchical recall, hP: hierarchical precision, hF: hierarchical F-measure, MCC: Matthews correlation coefficient. In bold the best (lower is better).

Measure	**TD**	**V1**	**V2**	**V3**	**STA**	**HMC-SSBR**
EM	3	**1.6**	2.1	3.5	6	4.8
hR	3.2	**1.6**	2.1	3.3	6	4.8
hP	4.2	2.6	3.1	4.1	6	**1**
hF	4.2	2.6	3.1	4.1	6	**1**
MCC	4.2	2.6	3.1	4.1	6	**1**

pseudo-labels; situations that were beneficial in these datasets.

5.4 Discussion

The performance of the different methods tend to variate throughout the amount of labeled-unlabeled data for each dataset, that is, the performances do not always increase as the amount of labeled data does (mainly in the Funcat, hard and very hard datasets). We attribute this behavior to the fact that those datasets are very hard and noisy. Vens et al. [2008b] indicate that the FunCat datasets suffer from non-unique feature representations, situation that would affect the performance of the classifiers as shown by Pliakos et al. [2015]. In similar way, the artificial datasets (hard and very hard) were generated from probability distributions where the intersection between them is high, and similar distributions are not necessarily grouped together [Serrano-Pérez and Sucar, 2021], making the learning task difficult.



(a) MCC (b) hF

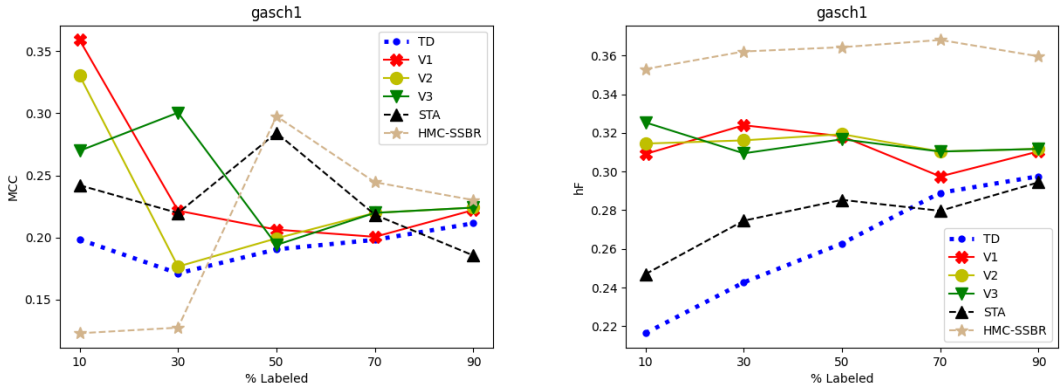
(c) MCC (d) hF

Figure 5.6: Results in the FunCat datasets: cellcycle a) and b); derisi c) and d). (Best seen in color.)

In order to know if there is statistical difference among the SSHC-BLI variants and the related methods (HMC-SSBR, STA, and TD), the Friedman test together with its post-hoc the Nemenyi test were used as recommended by Demšar [2006] when comparing multiple classifiers over multiple datasets.

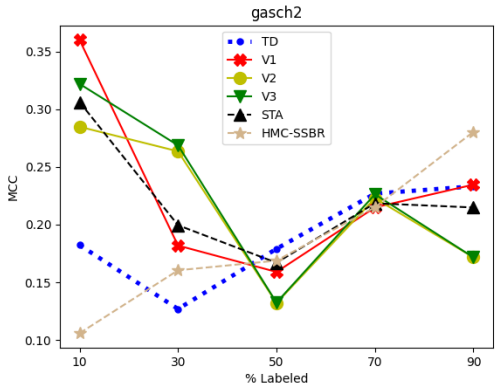
First, let r_i^j be the rank of the j-th of l algorithms on the i-th of M datasets, then $R_j = \frac{1}{M}\sum_{i=1}^{M} r_i^j$ is the average rank of the j-th algorithm. So, the null hypothesis of the Friedman test states that all the algorithms are equivalent, therefore their average ranks (R_j) should be equal, against the alternative which states that they are not.

Afterward, only if the null hypothesis is rejected, the Nemenyi test is used to compare all the classifiers against each other. Therefore, the performance of two classifiers is significantly different if their average ranks differ by at least the *critical*

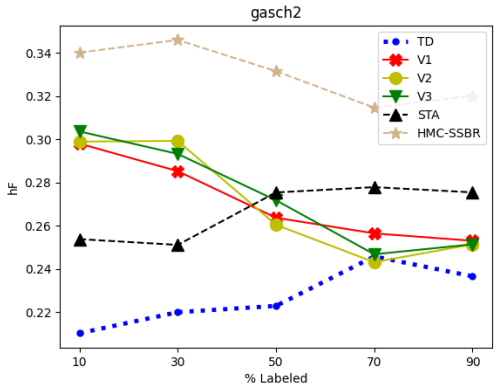


(a) MCC

(b) hF



(c) MCC



(d) hF

Figure 5.7: Results in the FunCat datasets: gasch1 a) and b); gasch2 c) and d). (Best seen in color.)

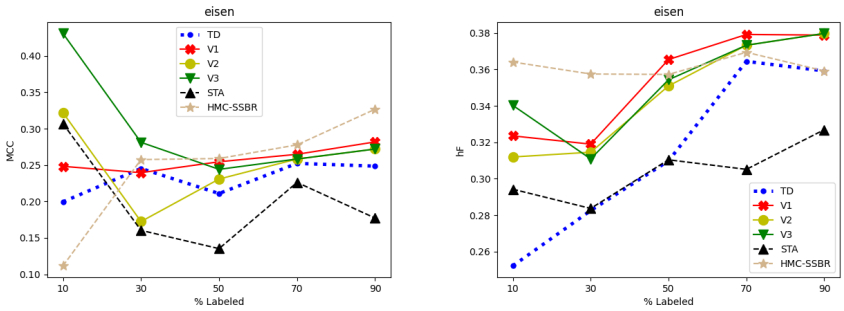
difference (CD), which can be estimated with equation 5.7,

$$CD = q_\alpha \sqrt{\frac{l(l+1)}{6M}} \tag{5.7}$$

where values q_α are based on the Studentized range statistic divided by $\sqrt{2}$ [Demšar, 2006].

The FunCat, 20 newsgroups and the artificial datasets (and their corresponding divisions) were considered. The results of the Friedman tests with significance $\alpha = 0.05$, for the different evaluations measures, are that null hypothesis can be rejected in favor of the alternatives, that is, the average ranks of the algorithms are not equal.

Since the null hypothesis were rejected, the Nemenyi tests can be applied. Fig 5.9 shows the graphical representation of the Nemenyi tests for EM and MCC, while Fig. 5.10 for hP, hR and hF. The performance of the related methods is



(a) MCC (b) hF

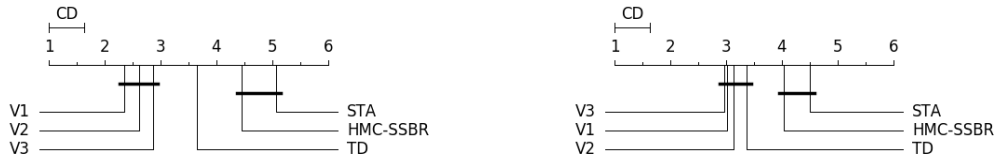
Figure 5.8: Results in the FunCat datasets: eisen: a) and b). (Best seen in color.)

Table 5.4: Average rank of each classifier in the FunCat datasets. In bold the best (lower is better).

Measure	TD	V1	V2	V3	STA	HMC-SSBR
EM	3.6	**1.92**	2.24	2.36	5	5.88
hR	5.48	2.68	2.58	**2.14**	3.94	4.18
hP	5.72	3.24	3.48	3.16	4.4	**1**
hF	5.76	3.32	3.32	2.96	4.28	**1.36**
MCC	4.36	3.32	3.36	2.88	4.28	**2.8**

significantly worst than the obtained by the SSHC-BLI variants on EM and MCC measures. Moreover, the performance of TD and STA is significantly worst than the performance of the SSHC-BLI variants on hF and its components. The variants of the proposed method are competitive with HMC-SSBR on hP, but HMC-SSBR performed better on hR, which calculates the ratio of correct prediction over the number of predictions in the datasets, and therefore on hF.

Throughout the experiments in the different datasets, the SSHC-BLI variants obtained competitive results. The variant 1 seems to be the best among the vari-



(a) EM (b) MCC

Figure 5.9: Graphical representations of the Nemenyi test for the evaluation measures (a) EM and (b) MCC. Classifiers that are not significantly different, with $p = 0.05$, are connected. CD: critical difference. (Lower is better)

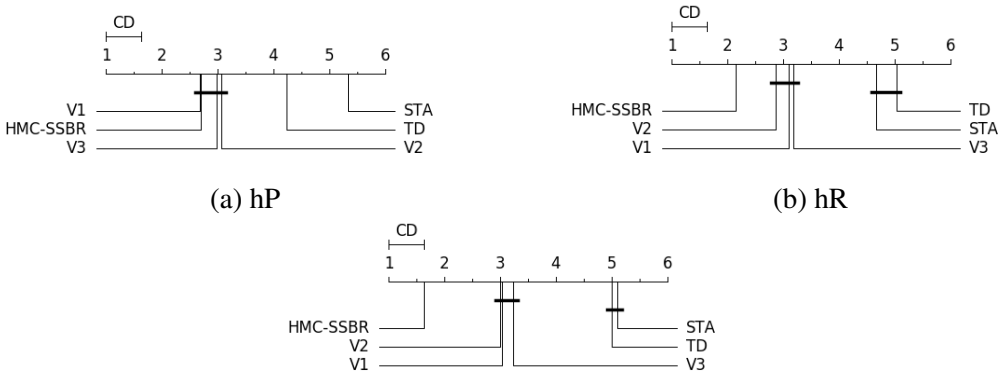


Figure 5.10: Graphical representations of the Nemenyi test for the evaluation measures (a) hP, (b) hR and (c) hF. Classifiers that are not significantly different, with $p = 0.05$, are connected. CD: critical difference. (Lower is better)

ants in general, nevertheless, the Nemenyi test indicates that there is no statistical difference among the variants on all the evaluation measures.

SSHC-BLI shows the best results on scenarios where the instances that are close in the input space share the same or similar paths of labels; that is, where the datasets fulfill the smoothness assumption, such as the easy artificial datasets. On the other hand, SSHC-BLI will have its performance degraded when the datasets do not fulfill the smoothness assumption, such as the very hard artificial datasets.

5.5 Summary

In this chapter, the similarity function SISI was proposed to measure the similarity of an instance (point) and a set of instances (points). Later, the semi-supervised classifier SSHC-BLI was proposed for hierarchies of tree type and instances associated to a single path of labels. SSHC-BLI builds pseudo-labels for each unlabeled instance using its labeled neighbors while considers whether the unlabeled instance is similar to its labeled neighbors. Experiments showed that SSHC-BLI is better than its supervised counterpart and it is competitive with related methods.

Chapter 6

Semi-Supervised Hierarchical Multi-label Classifier Based on Local Information

SSHC-BLI can only handle hierarchies of tree type and the instances must be associated to a single path of labels with full depth. In this chapter, a semi-supervised hierarchical classifier able to handle hierarchies of DAG type, while the instances can be associated to multiple paths of labels which can be of partial depth, is proposed.

Semi-Supervised Hierarchical Multi-label Classifier Based on Local Information[1] (SSHMC-BLI) [Serrano-Pérez and Sucar, 2025] is proposed to tackle the limitations of SSHC-BLI. The steps of SSHMC-BLI are similar to SSHC-BLI as shown in Algorithm 2. It is an iterative method that tries to pseudo-labeled the unlabeled data using its nearest labeled neighbors (lines 6-7), the way in which pseudo-labels are built is described in subsection 5.1. Fig. 6.1 shows an example of how a pseudo-label is built from the paths of labels of 3 instances, the labels of the instances are used to calculate *ppsl* (equation 5.1); later, a threshold is applied to *ppsl* which produces the pseudo-label as it is shown in 6.1d. This way of pseudo-labeling has the advantage that can be applied to any hierarchy of DAG type, furthermore, the instances can be associated to multiple paths of labels, that is, it is not limited to instances associated to a single path of labels.

This method considers the similitude of each unlabeled instances with its labeled neighbors (line 9), if they are not similar the unlabeled instance cannot be pseudo-labeled; the function SISI (equation 5.6) is used for estimating the similarity. The iteration process finishes when the pseudo-labels for the unlabeled data do not change from an iteration to another, or when the maximum number of iterations

[1] Open source available at: https://github.com/jona2510/SSHMC-BLI/tree/master

Algorithm 2 SSHMC-BLI algorithm

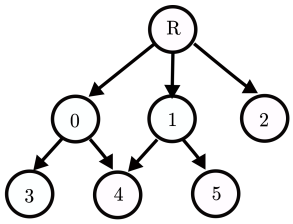
Require: (X, Y): labeled data, U: unlabeled data, k: number of nearest labeled neighbors, THR: similitude threshold, $t2label$: threshold to pseudo-label an instance, HS: hierarchy of DAG type, $maxIterations$: maximum number of iterations.

Ensure: f_{sshc}: trained SSHMC-BLI classifier

1: $T \leftarrow 1$ ▷ Iteration
2: $LD \leftarrow X$ ▷ LD: Labeled data
3: $CL \leftarrow L$ ▷ Labels of labeled data
4: **while** $True$ **do**
5: **for each** $u_j \in U$ **do**
6: $IND_j \leftarrow getNLN(k, u_j, LD)$ ▷ Get the nearest labeled neighbors
7: $PSL_j \leftarrow buildPseudoLabel(IND_j, LD, t2label)$ ▷ Pseudo label for u_j
8: **for each** $u_j \in U$ with valid PSL_j **do**
9: **if** $SISI(u_j, IND_j) < THR$ **then**
10: $PSL_j = \emptyset$ ▷ Invalid pseudo-label
11: **if** $(T > maxIterations)$ or $(PSL^T == PSL^{T-1})$ **then**
12: **break** loop (while)
13: **else** ▷ join labeled data with valid pseudo-labeled data
14: $CL \leftarrow Y \cup valid(PSL)$
15: $LD \leftarrow X \cup U[valid(PSL)]$
16: $T \leftarrow T + 1$
17: $f_{SSHC} \leftarrow trainHMC(LD, CL, HS)$ ▷ Train a hierarchical multi-label classifier

is reached. Finally, a *hierarchical multi-label classifier* is trained with the labeled and pseudo-labeled data, details can be found in section 6.1.

The three variants of SSHC-BLI are kept for SSHMC-BLI, the differences among them are briefly described next: *Variant 1* (V1) follows Algorithm 2 to the letter. *Variant 2* (V2), in each iteration all pseudo-labels for the unlabeled set are re-built, so, after the first iteration an instance, that was added to the labeled set, will have to itself as one of its nearest labeled neighbors; in order to avoid this, the function *getNLN* (line 6) is modified to guarantee that none of the nearest labeled neighbors is the instance itself. In *Variant 3* (V3), the value of k is increased after a predefined number of iterations.



(a) Hierarchy

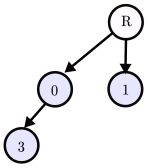
$y_1 = \{0,1,3,4\} = [1,1,0,1,1,0]$
$y_2 = \{0,1,3,5\} = [1,1,0,1,0,1]$
$y_3 = \{2\} = [0,0,1,0,0,0]$

(b) Labels of 3 instances

ppsl = [0.66, 0.66, 0.33, 0.66, 0.33, 0.33]

(c) Calculation of ppsl

psl(0.5) = [1,1,0,1,0,0]



(d) Pseudo-label after applying a threshold

Figure 6.1: Example of how a pseudo-label is built. (a) A hierarchy formed by 6 labels. (b)Shows the nodes to which three instances are associated, for example, the first instance is associated to the nodes $\{0, 1, 3, 4\}$, then its vector form is displayed, $[1, 1, 0, 1, 1, 0]$, that is, 1 if the node is associated, 0 otherwise. (c) *ppsl* is calculated with the labels in vector form of the instances in (b). Finally, the pseudo-label is obtained after applying a threshold (0.5) to *ppsl* as it is shown in (d).

6.1 Hierarchical Classifier for SSHMC-BLI

The hierarchical classifier is formed by LCNs, that is, for each node of the hierarchy (except the root node) a binary classifier[2] is trained; additionally, the policy *balanced bottom-up* (for a node, the positive instances are those associated to the node, while the negatives are taken from sibling, then uncles and so on, until the number of negatives equals the positives) is used to select the positive and negative instances at each node. In the prediction phase the probabilities of being associated to each LCN are obtained, then a post-processing is applied. The post-processing follows a top-down manner where the probabilities of each node are limited by the probabilities of their parents [Giunchiglia and Lukasiewicz, 2020]. Let a, b be nodes, let f^* be the post-processed output of the model f then:

$$f_a^* = min_{b \in S_a}(f_b) \tag{6.1}$$
$$S_a = Parents(a) \cup \{a\}$$

that is, a node gets the lowest probability among itself and its parents as shown in equation 6.1. In this way, the predictions of the model f^* are consistent with the

[2] *Random forest classifier*: default parameters except *n_jobs=5*. Implementation of `scikit-learn.org`.

hierarchy, because they comply the hierachical probability constraint.

An example of the result of post-processing is depicted in Fig. 6.2. The left hierarchy shows the probabilities of being positively associated to each node (output of the LCNs), which does not comply the hierarchical probability constraint, then, the post-processing is applied as it is shown in the hierarchy to the right, where the probabilities of a couple of nodes were limited in order to comply the hierarchical probability constraint.

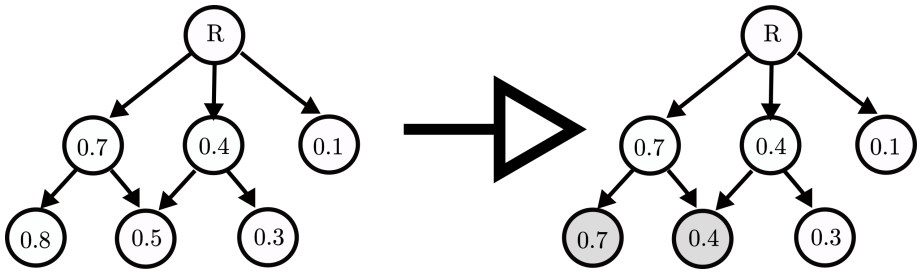


Figure 6.2: Example of post-processing. *Left*: the hierarchy has the probability of being associated to each node. *Right*: the hierarchy shows the post-processed output; grey nodes were limited by the probabilities of their parents.

6.2 Experiments and Results

The experiments are focused on showing if using unlabeled data may help to improve the performance of a hierarchical classifier trained only on labeled data, in the case where the hierarchy is a DAG, and the instances can be associated to multiple paths of labels which can be of partial depth.

The results of the proposed method are compared against standard methods (STML and STHC, both described in Appendix E) and a supervised hierarchical (LCN) classifier, which can be seen as the base line.

In order to carry out the experiments, the training set of each dataset was split into labeled and unlabeled sets. Then, the best configuration of each method is obtained by tuning their hyper-parameters with grid search, while they are evaluated on the validation set. Results of the methods on the test set are reported, also, they were compared with the Friedman test to identify statistical difference among the methods.

6.2.1 SSHMC-BLI Behaviour

A small artificial dataset was designed to show if the SSHMC-BLI variants pseudo-label properly the unlabeled data. The hierarchy is shown in Fig. 6.1a, which is composed by 6 nodes; the dataset is two-dimensional: 12, 330 and 300 instances were generated from normal distributions for labeled, unlabeled and test sets, respectively. Figs. 6.3a and 6.4a show unlabeled instances and the instances associated to nodes 1 and 4, respectively.

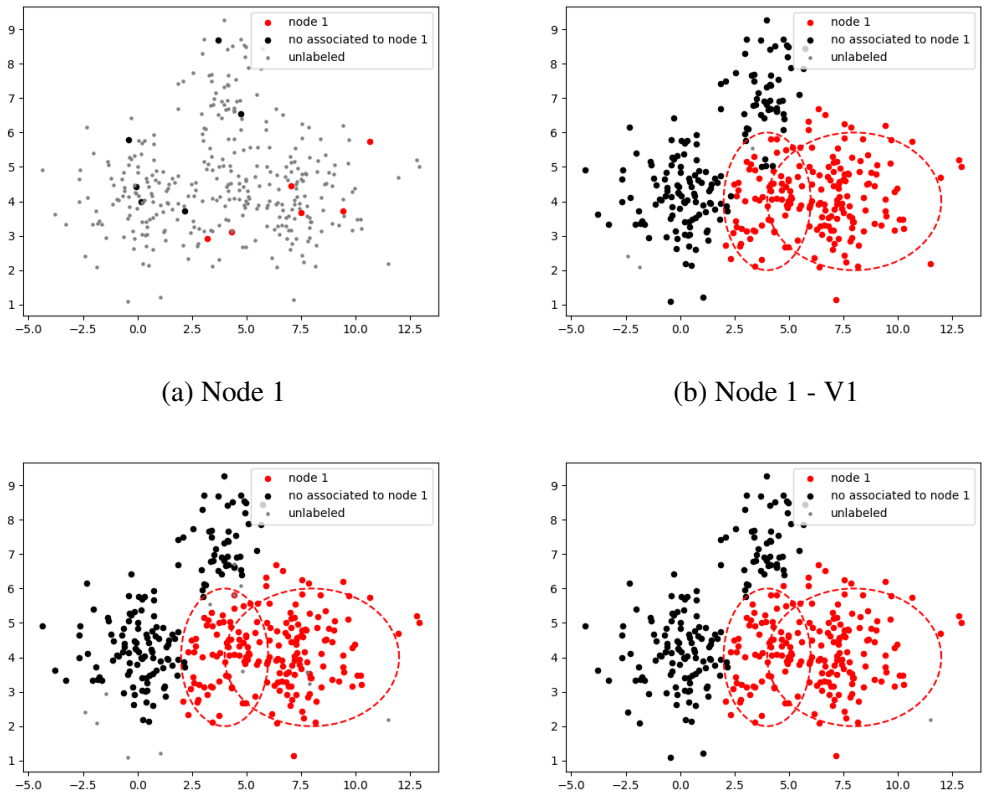
The SSHMC-BLI variants were applied to this dataset with the following configuration: nearest labeled neighbors, $k = 3$; similitude threshold, $THR = 0.5$; threshold to positively label an instance, $t2label = 0.5$; for variant 3, k increases by 1 each 10 iterations. Finally, a hierarchical multi-label classifier (section 6.1) was trained with labeled and pseudo-labeled data.

Figs. 6.3 and 6.4 show how SSHMC-BLI variants pseudo-labeled the unlabeled data with respect to nodes 1 and 4; inside the red circles is approximately 95% of the data that should be associated to the corresponding node. Variant 1 pseudo-labeled almost the whole unlabeled set, but it wrongly pseudo-labeled some instances as it can be seen in Fig. 6.4b for node 4; however, most of them were properly pseudo-labeled by its parent node, node 1, as it can be seen in Fig. 6.3b. Variant 2 pseudo-labeled in a better way the instances associated to nodes 1 and 4, as it can be seen in Figs. 6.3c and 6.4c; however, it was the variant where more instances stayed unlabeled due to the estimation of the similitude with its nearest neighbors. Finally, variant 3 seems to smooth the results obtained by the variant 2 as shown in Figs. 6.3d and 6.4d, since it was able to pseudo-label most of the instances that were previously unlabeled.

Later, for each variant the labeled and pseudo-labeled instances are used to train the hierarchical multi-label classifier. Results obtained by the SSHMC-BLI variants and the supervised classifier (LCN) are shown in Table 6.1. As it can be seen, the results of the SSHMC-BLI variants outperformed the supervised classifier.

Table 6.1: Results of SSHMC-BLI variants (1, 2 and 3) and the supervised classifier, LCN, for the artificial dataset. In bold the best score.

Measure	**LCN**	**V1**	**V2**	**V3**
Avg. precision	0.4492	**0.4817**	0.4526	0.4573



(a) Node 1

(b) Node 1 - V1

(c) Node 1 - V2

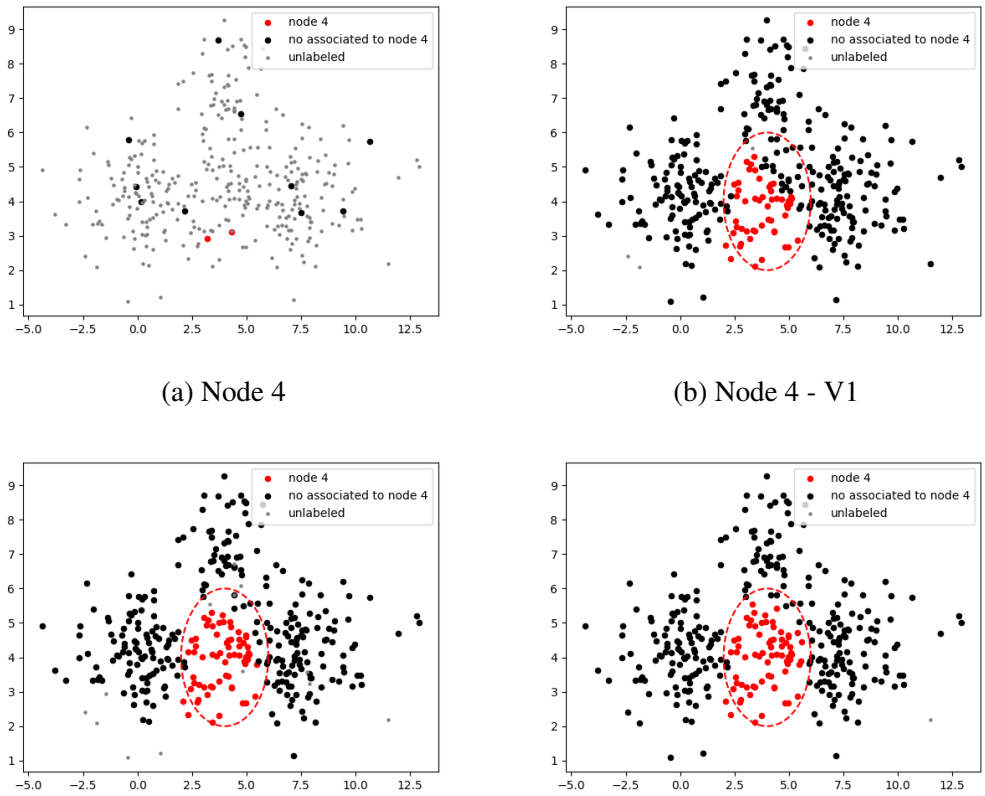
(d) Node 1 - V3

Figure 6.3: Pseudo-labels for the artificial dataset. (a) Initial data for node 1. (b,c,d) show the pseudo-labeled data at the end of each variant, V1, V2, V3, respectively. Approximately 95% of the data that should be associated to node 1 is inside the red circles. (Best seen in color.)

6.2.2 Results in the GO collection

First, the validation set was used for tuning the hyper-parameters of the SSHMC-BLI variants for each run. The parameters and values that could take are the following: similitude threshold (THR): {0.3, 0.5, 0.7}; number of labeled neighbors (k): {3, 4, 5}. The evaluation measure AP was used to determine the best configuration. Later, the semi-supervised classifier is trained with the best configuration but only with the labeled and unlabeled sets; that is, the validation set is not used for training. Finally, the SSHMC-BLI classifier is evaluated in the test set.

The results in average precision of the SSHMC-BLI variants, the standard methods (STML, STHC) and the supervised (LCN) classifier are shown in Figures 6.6 and 6.7. STML got the worse performance, since it is a multi-label method



(a) Node 4

(b) Node 4 - V1

(c) Node 4 - V2

(d) Node 4 - V3

Figure 6.4: Pseudo-labels for the artificial dataset. (a) Initial data for node 4. (b,c,d) show the pseudo-labeled data at the end of each variant, V1, V2, V3, respectively. Approximately 95% of the data that should be associated to node 4 is inside the red circle. (Best seen in color.)

which does not consider the hierarchy, therefore, the predictions do not guarantee to comply the hierarchical probability constraint. STHC improved the performance of STML just by adding the post-processing to comply the hierarchical probability constraint, nevertheless, its performance is most of times lower than the supervised, showing that the self-training strategy does not help to improve the performance of the supervised classifier. Finally, the different variants of the SSHMC-BLI got the best performances among the different semi-supervised and the supervised methods. The greatest improvement is found when there is just few labeled instances, then the performance of the SSHMC-BLI variants became closer to the supervised classifier as the amount of labeled data increase.

6.3 Discussion

Table 6.2 summarizes the results of the semi-supervised methods and the supervised classifier, that is, it presents the average rank of each classifier, where the SSHMC-BLI variants got the best performance. Variant 2 showed the best performance, this variant does not allow to pseudo-labeled instances to make use of themselves when building the new pseudo-labels; situation that was beneficial in these datasets.

Table 6.2: Average rank of each classifier in the GO collection. In bold the best (lower is better).

	LCN	**V1**	**V2**	**V3**	**STML**	**STHC**
Avg. Precision	4.133	2.0	**1.492**	2.708	5.958	4.708

The SSHMC-BLI variants got their best performance when there is few labeled data in most of the GO collection, then, their performance become closer to the performance of the supervised classifier as the amount of labeled data increases.

On the other hand, in order to know if there is statistical difference among the SSHMC-BLI variants, standard methods and the supervised classifier, the Friedman test together with its post-hoc the Nemenyi test were used [Demšar, 2006].

The result of the Friedman test with $p = 0.05$ for evaluation measure *average precision* is that the null hypothesis can be rejected in favor of the alternative, that is, the average ranks of the algorithms are not equal. Since the null hypothesis was rejected, the Nemenyi test can be applied. Fig. 6.5 shows the graphical representation of the Nemenyi test for average precision. The three variants of SSHMC-BLI are significantly better than STML, STHC and the supervised classifier, LCN.

Throughout the experiments, the SSHMC-BLI variants obtained the best performances. Variant 2 got the best performance among the variants in general, although for some datasets the variant 1 was sightly better. However, the Nemenyi test indicates that the performances of variants 1 and 2 are not significantly different.

Finally, it is expected that the SSHMC-BLI variants work properly on scenarios were close instances share the same or similar paths of labels, that is, when the datasets fulfill the smoothness assumption. On the other hand, the SSHMC-BLI will have its performance limited when the datasets do not fulfill the smoothness assumption, that is, when close instances do not necessarily share the same or similar path of labels.

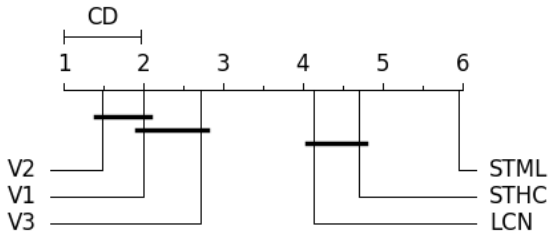


Figure 6.5: Graphical representations of Nemenyi test for the evaluation measures average precision. Classifiers that are not significantly different, with $p = 0.05$, are connected. CD: critical difference. (Lower is better)

6.4 Summary

In this chapter, the semi-supervised classifier SSHMC-BLI was proposed, which can handle hierarchies of DAG and the instances can be associated to multiple paths of labels of partial depth. In similar way than SSHC-BLI, SSHMC-BLI builds pseudo-labels for the unalabeled instances using the labels of its neighbors while considers the similitude among them. Results showed that SSHMC-BLI is statistically better than its supervised counterpart and standard methods.

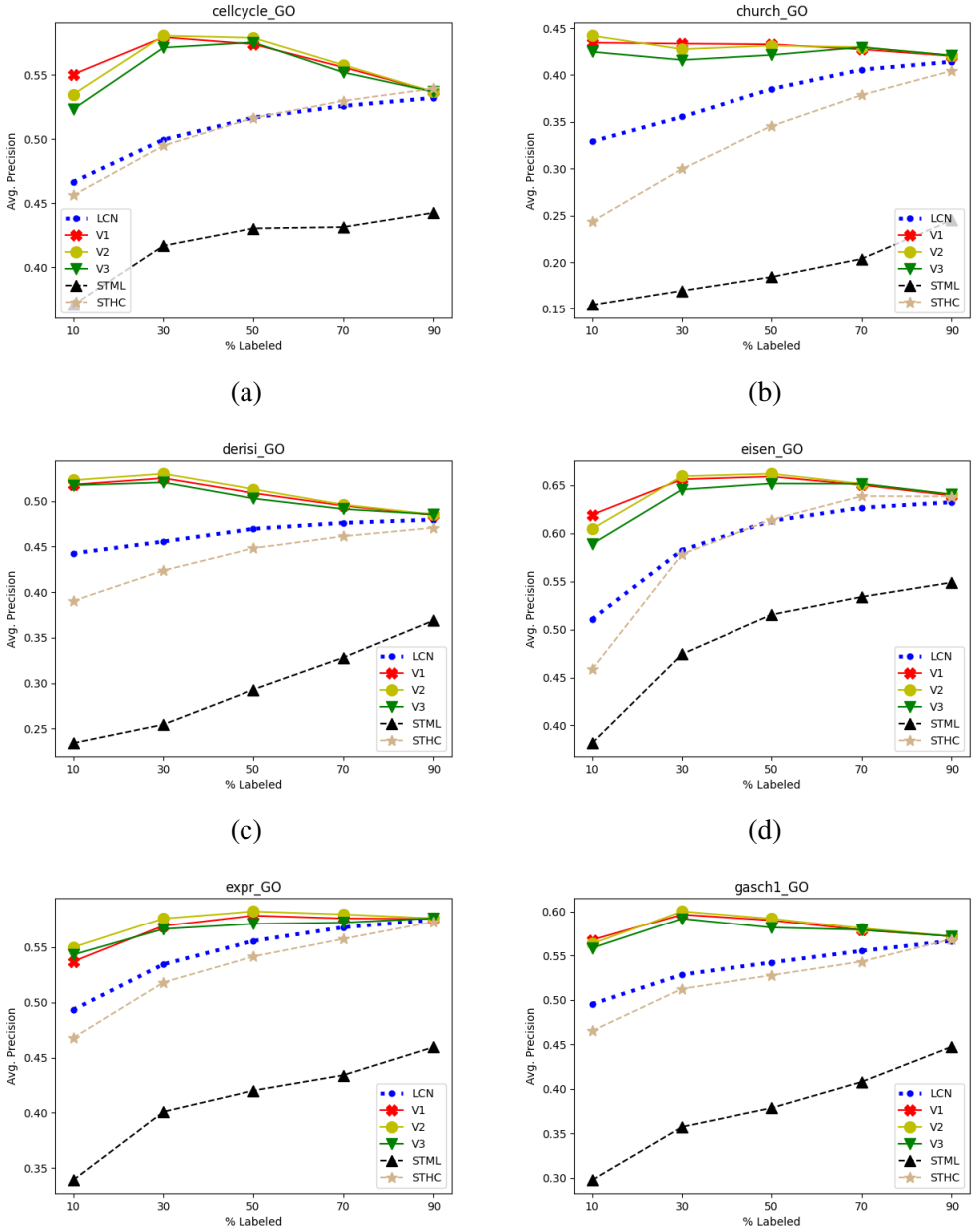


Figure 6.6: Results in GO datasets with the evaluation measure average precision: a) cellcycle, b) church, c) derisi, d) eisen, e) expr and f) gasch1. The x-axis correspond to the amount of labeled data (while its complement is the unlabeled data). (Best seen in color.)

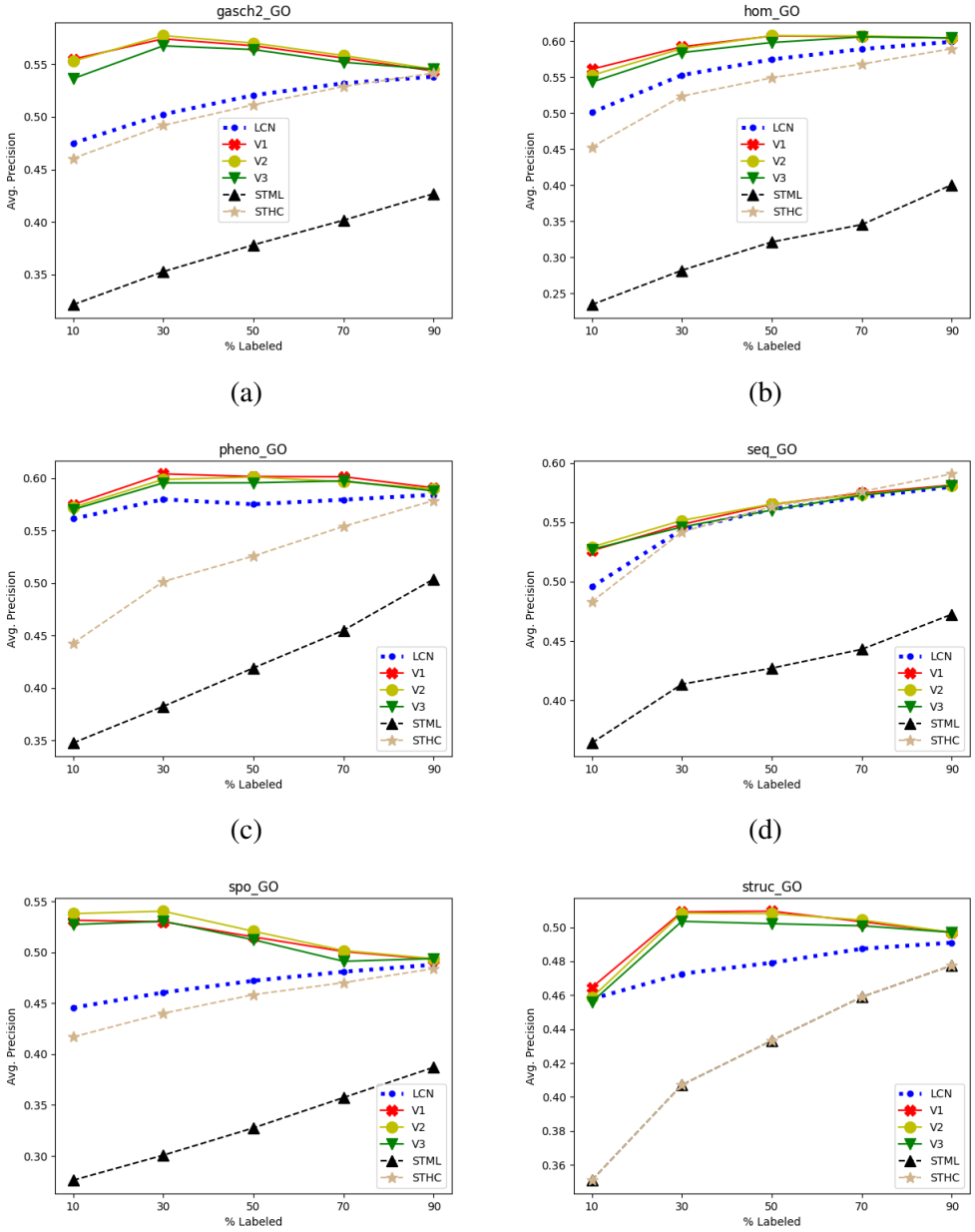


Figure 6.7: Results in GO datasets with the evaluation measure average precision: a) gasch2, b) hom, c) pheno, d) seq, (removed e) spo and f) struc). The x-axis correspond to the amount of labeled data (while its complement is the unlabeled data). (Best seen in color.)

Chapter 7

Hierarchical Bayesian Approach

Even tough, the results of SSHMC-BLI showed that using unlabeled data along labeled can help to improve the performance of a supervised classifier, the hierarchical classifier makes use of a post-processing that may affect the general performance of the classifier, since it is truncating the probabilities of some nodes to obtain predictions that comply the hierarchical probability constraint.

Therefore, we propose to use a Bayesian network, as an alternative post-processing, to obtain predictions that comply the hierarchical probability constraint without being truncated. First, the hierarchical Bayesian classifier is introduced in section 7.1, followed by a supervised application to morphological classification of galaxies in section 7.2. At last, the hierarchical Bayesian classifier is integrated to the semi-supervised hierarchical classifier in section 7.3.

7.1 Hierarchical Bayesian Classifier

The hierarchical Bayesian classifier [Serrano-Pérez and Sucar, 2019, Barutçuoglu et al., 2008] consist in modeling the hierarchy as a Bayesian network, which represents the data distribution as well as models the behavior of the *base classifier*. Hence the Bayesian network receives the initial probability estimates for each label in the hierarchy, then these are updated according to the hierarchical relations via probabilistic inference. Furthermore, through this post-processing step, the Bayesian network guarantees to comply the hierarchical probability constraint.

Two types of random nodes compose the Bayesian network, as shown in Fig. 7.1, y and q nodes. y nodes represents the data distribution and are also in charge of maintaining the hierarchical constraint in the Bayesian network, while q nodes model the initial estimations from the local classifiers.

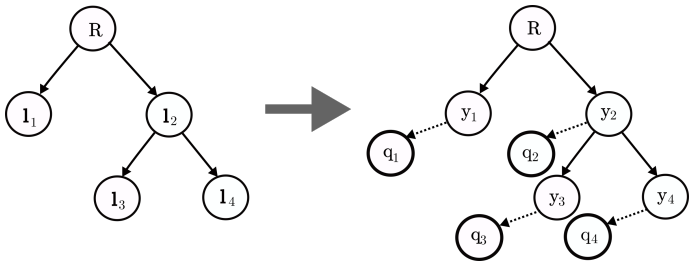


Figure 7.1: The hierarchy on the left is transformed into a Bayesian network (right). The y_i nodes in in the Bayesian network correspond to the l_i nodes in the hierarchy, and the q_i nodes in the Bayesian network correspond to the initial estimations from the local classifiers.

For each node of the hierarchy, there is a y_i node in the Bayesian network. The parameters for each node y_i are estimated by maximum likelihood from the training set, as shown in Table 7.1, where $pa(y_i)$ is the set of parents of y_i given by the hierarchy. Note that if an instance is not associated to all the parents of $pa(y_i)$, then it will be not associated to y_i with probability one, $P(y_i{=}0|pa(y_i){\neq}\mathbf{1}){=}1$, this is what maintains the hierarchical probability constraint in the Bayesian network.

Table 7.1: Conditional probability table of $P(y_i|pa(y_i))$. a is the number of instances associated to both y_i and its parents, $pa(y_i)$; b is the number of instances not associated to y_i but associated to all its parents, $pa(y_i)$; **1** represents the case when the instances are associated to all nodes of $pa(y_i)$, while $\neq$**1** all the other cases. Laplace smoothing is applied only when $pa(y_i){=}\mathbf{1}$.

		$pa(y_i)$	
		1	$\neq$**1**
y_i	1	$\frac{a+1}{a+b+2}$	0
	0	$\frac{b+1}{a+b+2}$	1

On the other hand, q nodes are the second type of random nodes in the Bayesian network. Each y_i node has a child node, q_i, except the root node. q nodes model the behavior of the base classifier for predicting instances that were not used in training. Furthermore, q nodes will receive the predictions of the base classifier, which will be propagated in the Bayesian network.

Considering that the base classifier provides as prediction the probabilities of being associated to each node, the distribution for each node q_i, $P(q_i|y_i)$, is modeled parametrically with Gaussian distributions [Barutçuoglu et al., 2008]. That is,

$$P(q_i{\in}\mathbb{R}|y_i{=}0){\simeq}N(\mu_0, \sigma_0^2),$$

where μ_0 is the mean and σ_0^2 is the variance of the predictions of the base classifier

in the instances not associated to y_i in the validation set; and the same for

$$P(q_i \in \mathbb{R} | y_i = 1) \simeq N(\mu_1, \sigma_1^2),$$

but considering the instances associated to y_i in the validation set.

The base classifier can be any classifier able to provide the probability for each node of the hierarchy (except the root).

To sum up:

- *Training phase*: the base classifier is trained with the training set. Then, the parameters of the Bayesian network are estimated, $P(y_i|pa(y_i))$ can be calculated from the training set, while $P(q_i|y_i)$ can be estimated from the predictions of the base classifier in a validation set.
- *Prediction phase*: the base classifier is fed with *new* data, the base classifier's predictions are sent to the Bayesian network, and the evidence is propagated. The posterior probabilities of the y nodes is the output, which guarantee to comply the hierarchical probability constraint.

7.2 Morphological Classification of Galaxies

Galaxy morphological classification is essential for understanding galaxy evolution and studying stellar populations and their physical properties. Despite recent advances, galaxy classification continues to face several difficulties and challenges. Some current issues and challenges include subjectivity, which can lead to inconsistencies and variations in classification results, mainly due to subjective decisions made by experts based on visual inspection.

The great utility of deep learning in astronomy is straightforward, specifically for galaxy classification. However, previous works do not take advantage of the hierarchy, which could help to improve accuracy. Therefore, Bayesian and Convolutional Neural Networks [Serrano-Pérez et al., 2024] is proposed for supervised morphological galaxy classification, which models the hierarchy as a Bayesian network as described in section 7.1, while is fed by a CNN as detailed in section 7.2.1; in this case, the hierarchy is of tree type and the instances are associated to a single path of labels of partial depth.

7.2.1 Bayesian and Convolutional Neural Networks

Bayesian and convolutional neural networks[1] (BCNN) [Serrano-Pérez et al., 2024] is the proposed classifier for hierarchical image classification. BCNN uses the Bayesian network described previously in section 7.1, but the base classifier that feeds the Bayesian network is a pretrained convolutional neural network (CNN) [Li et al., 2022, Haridas and JyothiR, 2019, Khan et al., 2020]. CNNs have the advantage that can be trained with raw images because they can automate the feature extraction process from the images [Li et al., 2022, Altenberger and Lenz, 2018].

EfficientNet V2-xl-21k [Tan and Le, 2021] is the CNN used by BCNN, which was joined with a dense layer (with sigmoid activation function) that has one output for each node of the hierarchy, except the root node. This way, the training can be carried out only in this last layer, but the whole CNN can be retrained, too; the CNN is optimized with the gradient descent (with momentum) optimizer and binary cross-entropy as the loss function. Later, the predictions of the CNN are used to feed the Bayesian network through the q nodes, as shown in Fig. 7.2.

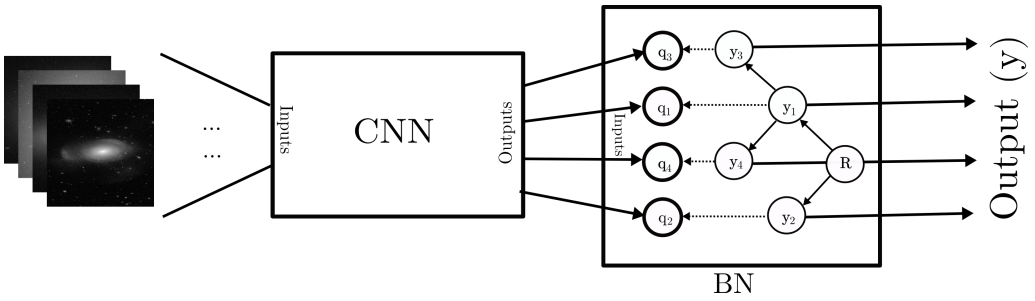


Figure 7.2: Model of BCNN [Serrano-Pérez et al., 2024]. It comprises two main modules: a CNN that feeds a Bayesian network. The CNN classifier is fed directly with the images, which outputs the probability for each class (in this example, four classes). These probabilities are sent to the Bayesian network (BN) that optimizes the classification via probabilistic inference. In this simple example, the hierarchy consists of 4 classes, where y_1 and y_2 are sub-classes of the root, R; and y_3 and y_4 are sub-classes of y_1.

Finally, the prediction for new instances is obtained from the posterior probabilities of the y nodes, where the TD procedure is applied to get a single path of labels.

[1] Open source available at: https://github.com/jona2510/BCNN

7.2.2 Galaxies Dataset

A collection of galaxies[2] sourced from the Principal Galaxies Catalog (PGC) and the APM Equatorial Catalogue of Galaxies were employed. Both compilations provide details regarding the morphological and numerical characteristics of each galaxy. All the images were standardized to size 300×300 pixels and monochrome. The labels of galaxies are arranged in the hierarchy shown in Fig. 7.3. As it can be seen, the different labels and the hierarchy are based on the Hubble sequence [Hubble, 1926, 1927], that is, galaxies with labels *Sa, Sb, Sc* and *Sd* are grouped as *Spiral* galaxies; *Elliptical, Lenticular* and *Spiral* labels are grouped as *Regular* galaxies; and galaxies of labels *Regular, Irregular* and *Others* are grouped in the *root* node. Each image is associated with a single path of labels of full depth. Table 7.2 presents the number of galaxies per label.

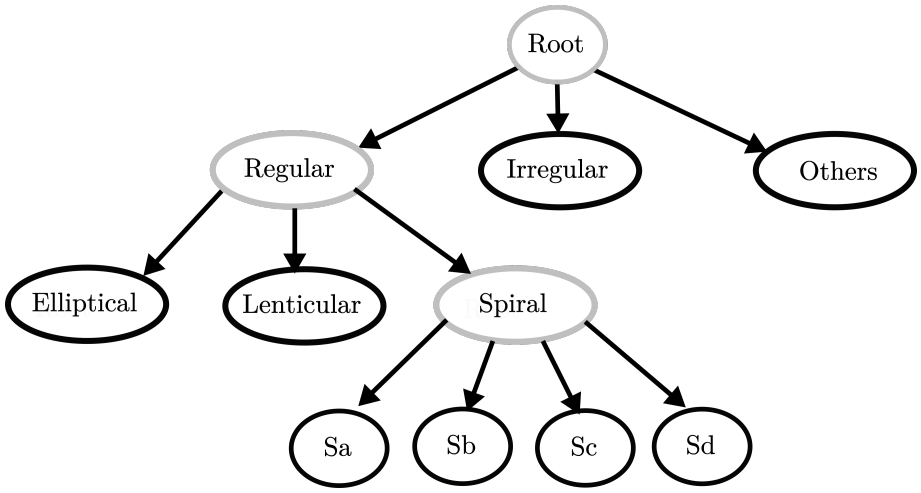


Figure 7.3: Galaxy hierarchy [Serrano-Pérez et al., 2024].

7.2.3 Experiments and Results

The first experiment consisted on showing if making use of the Bayesian network helps to improve the performance of the CNN than when it is not used. Table 7.3 shows that the incorporation of the Bayesian network helped to improve the performance of the CNN by ~ 7 points in exact match and ~ 4 points in hF.

This application requires a classifier that predicts a single path of labels. Hence some strategies were compared to select one of the paths of labels. Table 7.4 shows the results of the strategies TD procedure, sum of probabilities (SP) [Hernandez

[2]The dataset is available at `https://www.kaggle.com/datasets/jonsperez/galaxies-dataset-for-hierarchical-classification`

Table 7.2: Labels of galaxies and the number of images associated to them in training, test, and validation sets. *Images associated with multiple labels are counted only once.

Class	Total	Training	Test	Validation
Sa	277	177	56	44
Sb	281	180	56	45
Sc	222	142	45	35
Sd	62	40	12	10
Spiral	842	539	169	134
Elliptical	471	301	94	76
Lenticular	387	247	78	62
Regular	1700	1087	341	272
Irregular	181	116	36	29
Others	50	32	10	8
*Total no. of images:	1931	1235	387	309

Table 7.3: Results (percentage) of the classifier with and without the Bayesian network, BCNN and CNN, respectively. EM: exact match, hR: hierarchical recall, hP: hierarchical precision, hF: hierarchical F-measure.

Classifier	EM	hR	hP	hF
CNN	57.62	80.38	75.03	77.61
BCNN	64.34	81.94	80.15	81.04

et al., 2013] and score gain-loose balance (scoreGLB), where TD procedure showed the best performance, so TD is used for the rest of experiments.

In the previous experiments, the weights of the pre-trained CNN were frozen, so they can not be updated, that is, the layers of the pre-trained CNN worked only as feature extractors. Hence in the next experiment, all the weights of the CNN layers can be updated (retraining). Additionally, an experiment that carries out image augmentation was carried out, details of image augmentation can be found in Appendix D. Table 7.5 shows the results, where retraining all of the layers helped to improve the performance of the classifier; furthermore, the performance of the classifier was further improved by carrying out image augmentation.

Finally, the results obtained by BCNN where compared against Deep Galaxy 2 [Khalifa et al., 2018] and different CNN models: Inception v3 [Palacio et al., 2018], Inception ResNet v2 [Szegedy et al., 2017], BiT m-r152x4 [Kolesnikov et al., 2020] and EfficientNet V2-xl-21k [Tan and Le, 2021]. In the same way than BCNN, all of the CNN models were joined with a dense layer with one output per label and sigmoid activation functions, also, the models were optimized with the gradient

Table 7.4: Results (percentage) of the BCNN classifier with different procedures to obtain the path of labels. EM: exact match, hR: hierarchical recall, hP: hierarchical precision, hF: hierarchical F-measure. In bold the best.

Procedure	**EM**	**hR**	**hP**	**hF**
TD	**64.34**	**81.94**	**80.15**	**81.04**
SP [Hernandez et al., 2013]	62.53	79.71	79.8	79.75
scoreGLB [Ramírez-Corona et al., 2016]	63.31	80.04	80.13	80.09

Table 7.5: Results (percentage) of BCNN classifier retraining or not all the CNN's layers. ImgA: plus image augmentation. EM: exact match, hR: hierarchical recall, hP: hierarchical precision, hF: hierarchical F-measure. In bold the best.

Retraining	**EM**	**hR**	**hP**	**hF**
No	64.34	81.94	80.15	81.04
Yes	65.89	82.16	82.25	82.21
Yes+ImgA	**67.18**	**84.17**	**82.33**	**83.24**

descent (with momentum) optimizer and binary cross-entropy as the loss function; the framework *tensorflow*[3] was used for training the CNNs. Additionally, the TD procedure is applied to their predictions to obtain consistent predictions.

Table 7.6 shows the results of BCNN and the different models, where EfficientNet can be seen as the baseline. BCNN outperformed the rest of the classifiers in all the evaluation measures. Furthermore, when BCNN was trained with image augmentation and retraining of all the layers, the performance increased by ~ 3 points in exact match and ~ 2 in hF.

Table 7.6: Results (percentage) of the BCNN classifier compared with several CNN models. ImgA: plus image augmentation; FN: plus retraining of all the layers. EM: exact match, hR: hierarchical recall, hP: hierarchical precision, hF: hierarchical F-measure. In bold the best.

Classifier	**EM**	**hR**	**hP**	**hF**
Deep Galaxy v2[Khalifa et al., 2018]	16.54	41.34	49.61	55.89
Inception v3 [Palacio et al., 2018]	48.32	76.14	69.27	72.54
Inception ResNet v2 [Szegedy et al., 2017]	42.12	73.91	64.62	68.95
BiT m-r152x4 [Kolesnikov et al., 2020]	52.97	72.69	75.29	73.96
EfficientNet V2-xl-21k [Tan and Le, 2021]	57.62	80.38	75.03	77.61
BCNN	64.34	81.94	80.15	81.04
BCNN(ImgA,FN)	**67.18**	**84.17**	**82.33**	**83.24**

[3] `https://www.tensorflow.org/`

7.2.4 Discussion

From the experiments, we conclude that all the elements (Bayesian network hierarchical classifier, data augmentation, and retraining) contribute to improve the performance of the basic CNN classifier in all measures. There is an improvement of ~ 10 points on exact match and ~ 7 points for the hierarchical F-measure over the baseline (EfficientNet). However, the Bayesian network provides the most significant impact on the performance of the different elements, ~ 7 points (from 57% to 64%) in exact match and ~ 3 points (77% to 81%) in hF.

7.3 Semi-Supervised Hierarchical Bayesian Multi-label Classifier

Section 7.2 showed that the incorporation of the Bayesian network to post-process the output of a base classifier could help to improve its performance. Therefore, Semi-Supervised Hierarchical Bayesian Multi-label Classifier (SSHBMC) [Serrano-Pérez and Sucar, 2025] proposes to incorporate the Bayesian network as an alternative post-processing to truncating the probabilities as SSHMC-BLI does.

Algorithm 3 shows the steps of SSHBMC, which is based on SSHMC-BLI. It is an iterative method that tries to build pseudo-paths of labels (line 10) to pseudo-label the unlabeled data using its nearest labeled neighbors (line 6), details of how pseudo-paths of labels are built can be found in section 5.1. Also, the method considers the similarity of each unlabeled instances with its neighbors (line 7), if they are not *similar* the unlabeled instance stays unlabeled; SISI is the function used to estimate the similarity. Furthermore, the hierarchical Bayesian classifier described in section 7.1 is used as the hierarchical classifier, which uses LCN to feed the Bayesian network.

7.3.1 Experiments and Results

Results in the GO datasets (section 4.2) and a comparison of SSHBMC (which incorporates the Bayesian network) against SSHMC-BLI (variant 1), standard methods (STML and STHC, both described in Appendix E) and the supervised classifier (LCN) are shown in Figures 7.5 and 7.6. The proposed method, SSHBMC, tends to outperform most of times the rest of methods, even SSHMC-BLI; showing that making use of the Bayesian network as post-processing is helping to improve the performance of the classifier compared to when the post-processing consists on truncating the probabilities.

Algorithm 3 SSHBMC algorithm

Require: (X, Y): labeled data, U: unlabeled data, k: number of nearest labeled neighbors, THR: similarity threshold, $thLabel$: threshold to pseudo-label an instance, HS: hierarchy of DAG type, $maxIterations$: maximum number of iterations.
Ensure: f_{SSHC}: trained SSHBMC classifier

$T \leftarrow 1$
$LD \leftarrow X$
$CL \leftarrow Y$
while $True$ **do**
  for each $u_j \in U$ **do**
    $IND_j \leftarrow getNLN(k, u_j, LD)$ ▷ Nearest labeled neighbors
    if $SISI(u_j, IND_j) < THR$ **then**
      $PSL_j = \emptyset$
    else
      $PSL_j \leftarrow buildPPL(IND_j, LD, thLabel)$ ▷ Pseudo-path-of-labels for u_j
  if $(T > maxIterations)$ or $(PSL^T == PSL^{T-1})$ **then**
    break loop (while)
  else
    $CL \leftarrow Y \cup valid(PSL)$
    $LD \leftarrow X \cup U[valid(PSL)]$
  $T \leftarrow T + 1$
$f_{SSHC} \leftarrow trainHBC(LD, CL, HS)$ ▷ Train the hierarchical Bayesian classifier

7.3.2 Discussion

The Friedman test together with its post-hoc, Nemenyi test, were carried out to verify if there is statistical difference among the methods [Demšar, 2006].

The result of the Friedman test with $\alpha = 0.05$ is that the null hypothesis can be rejected in favor of the alternative, in other words, the average ranks of the algorithms are not equal. Then, the Nemenyi test was applied, Fig. 7.4 shows its graphical representation. As it can be seen, SSHBMC is significantly better from the rest of methods.

7.4 Summary

In this chapter, the semi-supevised hierarchical classifier SSHBMC was proposed, which extends SSHMC-BLI by using a Bayesian network to post-process the output of the local classifiers instead of truncating the probabilities. Section 7.1 presents the

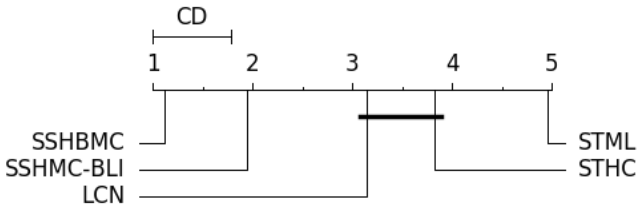


Figure 7.4: Graphical representation of the Nemenyi test. Classifiers that are not significantly different, with $\alpha = 0.05$, are connected. CD: critical difference. (Lower is better.)

hierarchical Bayesian classifier, which models the hierarchy as a Bayesian network while considers the behavior of the base classifier. Additionally, the supervised classifier BCNN was proposed for hierarchical classification of images, which is based on the hierarchical classifier and it was applied to morphological classification of galaxies.

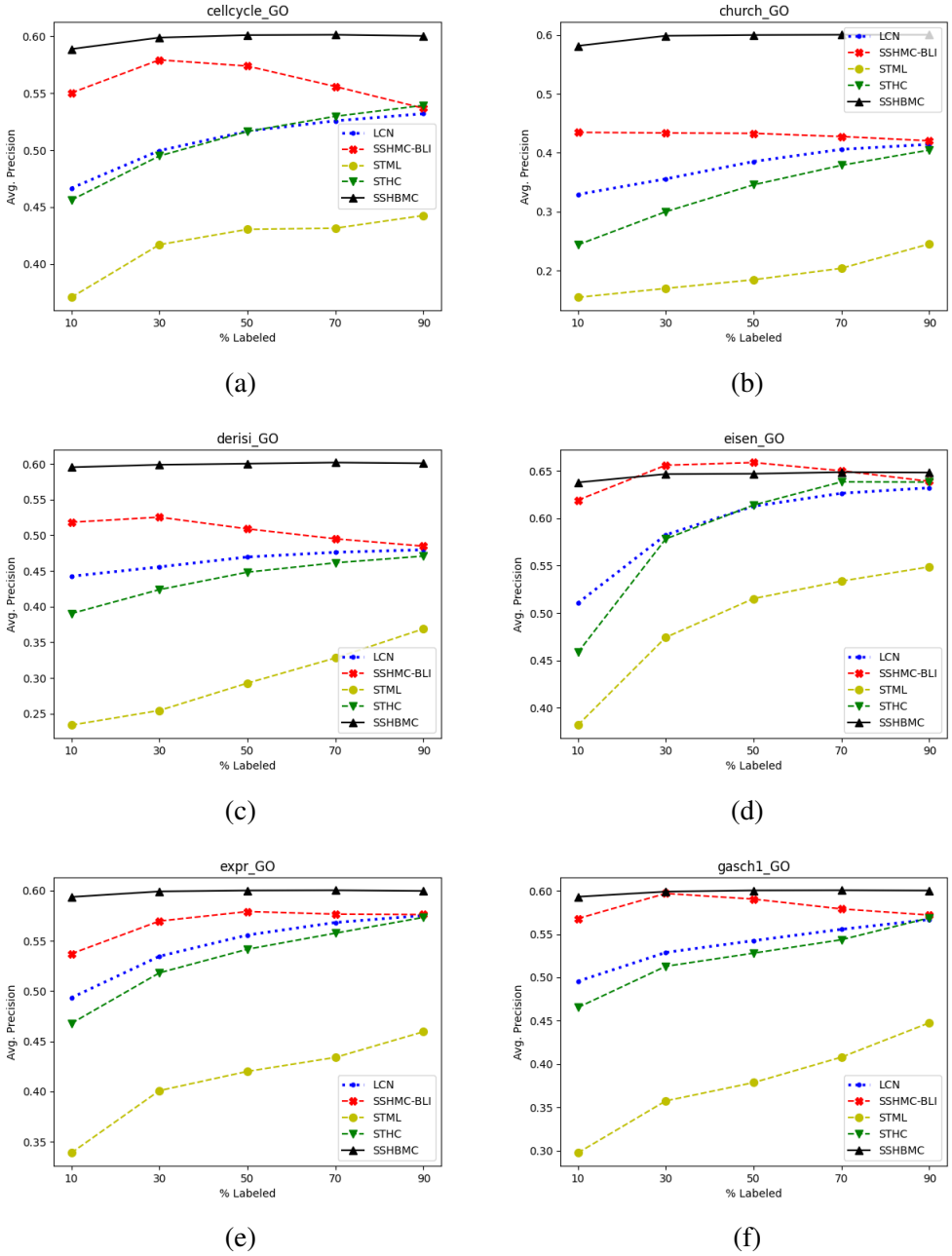


Figure 7.5: Results in terms of average precision (AP) in the following datasets: a) cellcycle, b) church, c) derisi, d) eisen, e) expr and f) gasch1. The x-axis corresponds to the amount of labeled data, while its complement is the unlabeled data. (Best seen in color.)

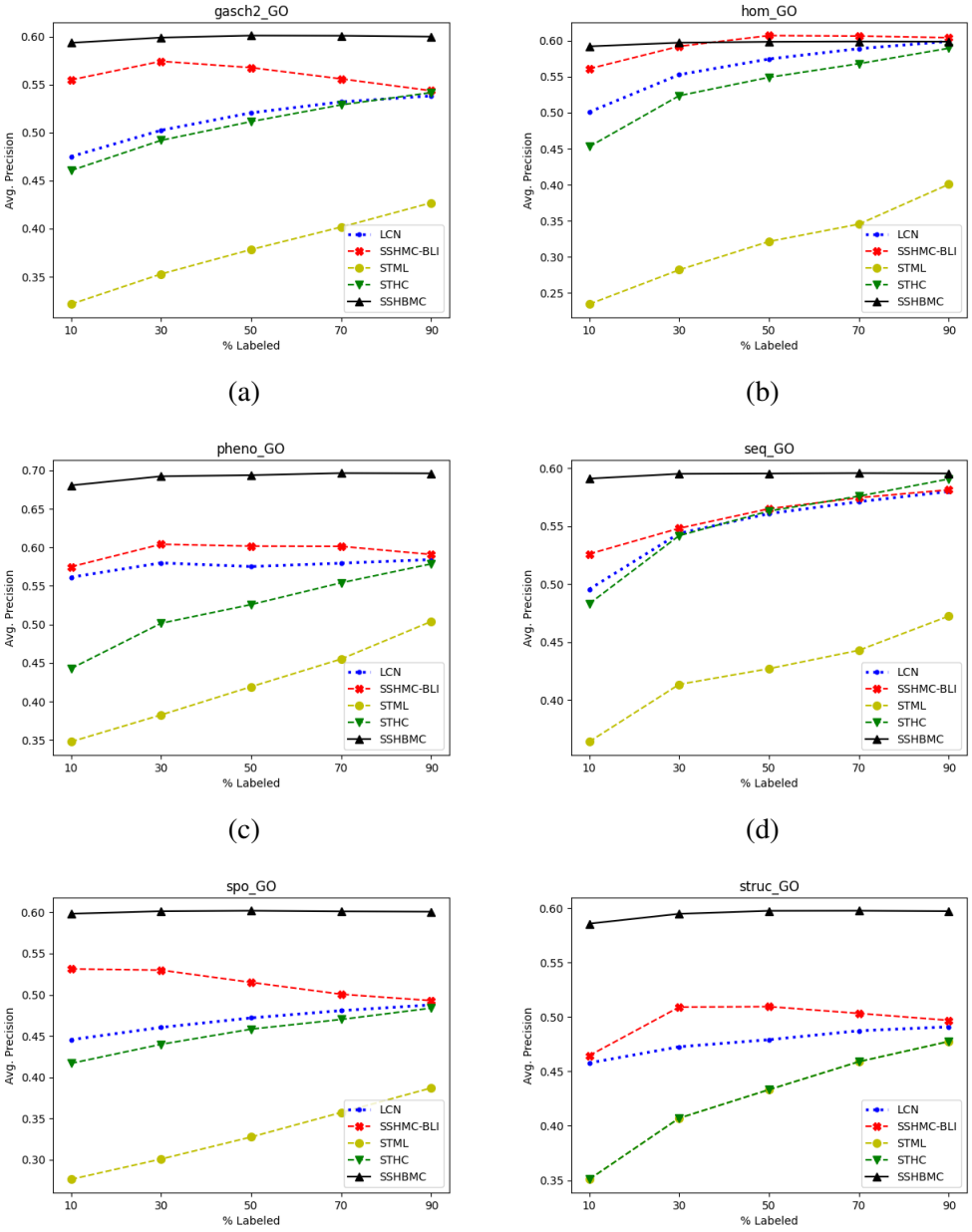


Figure 7.6: Results in terms of average precision in the following datasets: a) gasch2, b) hom, c) pheno, d) seq, e) spo and f) struc. The x-axis corresponds to the amount of labeled data, while its complement is the unlabeled data. (Best seen in color.)

Chapter 8

Transfer Learning in SSHMC-BLI

Since the scarcity of labeled data is a mayor problem in this field, the available information has to be exploited as much as possible. One way that we proposed to achieve this is by carrying out *transfer learning* (TL) between neighboring nodes.

In hierarchical classification, the LCN approach implies that upper nodes learn general information while depth nodes learn specific information. So, instead of training independent local classifiers as classical LCN does, we propose to transfer the knowledge of each parent node to its children, in this way, the child nodes will not be trained from scratch.

8.1 Hierarchical Specialization

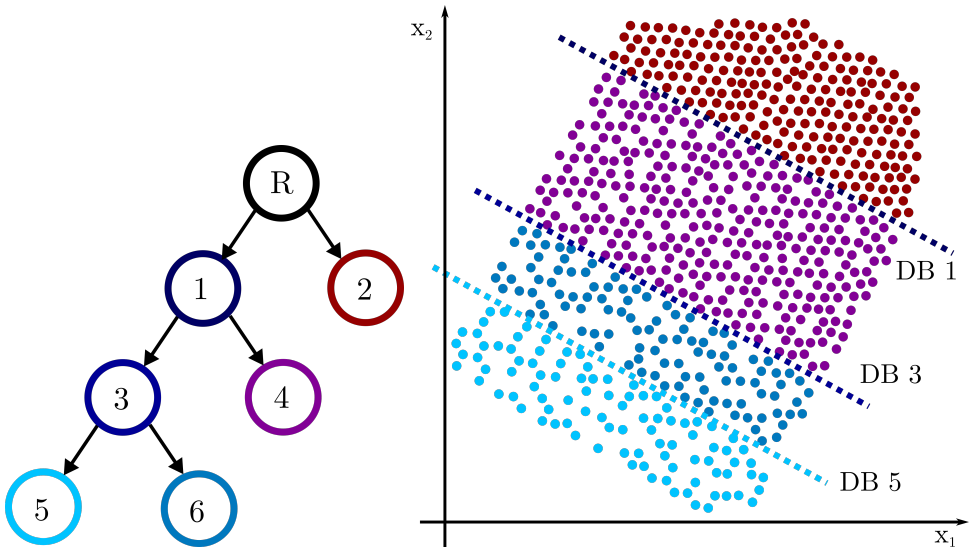
Hierarchical specialization (HSp) is the proposed method to train a classifier based on LCN, which takes into account that upper nodes learn general information, so, this information can be shared to their children to be specialized by them.

The idea of HSp can be illustrated with an example, but first, a *perceptron* [Rosenblatt, 1958] will be used as the local classifier, whose function is shown in equation 8.1:

$$y = f\left(b + \sum_{i=1}^{d} w_i x_i\right) \tag{8.1}$$

where $x \in \mathbb{R}^d$ is the input space, $w \in \mathbb{R}^d$ and b are the parameters of the perceptron, d is the number of attributes and $f(\cdot)$ is the activation function.

The example consist on training a binary classifier from scratch, in this example a perceptron, so later, the parameters learned by the classifier can be copied for its children, so they are not trained from scratch. The hierarchy and labeled instances



(a) (b) hF

Figure 8.1: Example of hierarchical specialization. a) A hierarchy. (b) Some labeled points, $x \in \mathbb{R}^2$, are shown (colored with the most specific label). The decision boundaries (DB) of nodes 1, 3 and 5 are shown, instances below the line are positively classified, negatively otherwise. TL may be applied in a TD manner, for instance, the DB of node 1 may be used as the staring point of node 3 instead of being trained from scratch, in the same way, the DB of node 3 may be used as starting point of node 5. (Best seen in color)

for the example are shown in Fig. 8.1, hence the training for node 1 will start by giving random values to the parameters of the perceptron, in this example the number of attributes is $d = 2$, which are later fit to the labeled set, so, at the end of the training, it will find[1] that $w_2 \approx 0$, and some values for w_1 and b, hence the decision boundary (DB 1) can be plot as shown in Fig. 8.1b, where instances below the line will be positively classified, while instances over the line will be negatively classified. Later, there are two alternatives for training node 3: the first is training from scratch, i.e. the parameters of the perceptron would be randomly initialized; or the second, use as starting point the learned parameters of its parent, that is, copy the parameters of node 1's perceptron, this means that an *specialization* would be carry out. In this example, it is clear that the second option should be preferred because the parameters learned by node 1's perceptron will be pretty much similar than those learned by node 3's perceptron but b; decision boundary of node 3 (DB 3) is plotted in Fig 8.1b, where instances below the line are positively classified, negatively otherwise. Following the same idea of specialization, the classifier of node 5 (DB 5) can be obtained using as starting points the parameters of node 3's perceptron. As it can be seen, the classifier of node 3 is an specialization of the node 1, while the classifier of node 5 is an specialization of node 3.

[1] In this example is also possible that $w_1 \approx 0$.

HSp works only on the training phase of the hierarchical classifier based on LCN. Algorithm 4 shows the general steps of HSp, it initializes an *artificial neural network* (ANN) from scratch for children of the root node (line 5); for the rest of nodes, it selects one parent and copy its parameters (line 8), that is, transfer learning from parent to child, in other words, an specialization. After selecting the positive and negative instances with the corresponding policy (line 9; the balanced bottom-up policy is used in the experiments of this chapter) the local classifier can be (re-)trained.

Even tough, in the toy example a perceptron was used, any ANN [Haykin, 2009, Goodfellow et al., 2016], whose parameters can be copied to be retrained later, can be used as the binary classifier, for instance the *multilayer perceptron*.

On the other hand, the order determined by *parents first search* (PFS) is followed to walk through all nodes, which guarantees that a node can be trained only when its parents have already been trained; PFS is a search for DAGs similar to *breadth first search*[Needham and Hodler, 2019], however, PFS adds a node to its queue only if all its parents have already been visited. Algorithm 5 shows the steps of PFS, it requires an empty queue and an empty list of visited nodes (lines 1 and 2, respectively); the search always starts at the root node, so, it adds the root node at the list and adds root's children to the queue (lines 4-6); later, the search gets the first node of the queue and adds it to the list (lines 9 and 10, respectively), then only its children whose parents have already been visited, can be added to the queue (lines 12-16); the search iterates until the queue is empty. Finally, the root node is removed from the list of visited nodes, because it is not useful for LCN, and the list of visited nodes is returned (lines 18-19).

Algorithm 4 HSp algorithm; *fit* function of the hierarchical classifier.

Require: (X, Y): (pseudo-)labeled data, H: hierarchy of DAG type, n number of nodes.
Ensure: trained classifier

1: $lcns \leftarrow array(n)$ ▷ Array of local classifiers
2:
3: **for each** $j \in PFS(H)$ **do** ▷ PFS: parents first search
4: **if** $isRootChild(j, H)$ **then**
5: $lcns[j] \leftarrow ANN(hyperparameters)$ ▷ Artificial neural networks
6: **else**
7: $p \leftarrow getParent(j)$ ▷ index of one parent
8: $lcns[j] \leftarrow copy(lcns[p])$ ▷ copy the parameters of its parent
9: $(X', Y') \leftarrow policy(j, H, X, Y)$
10: $lcns[j].fit(X', Y')$ ▷ training of the binary ANN

Algorithm 5 Parents first seach: $PFS(H)$

Require: H: hierarchy of DAG type.
Ensure: list with ordered nodes.

```
queue ← Queue()                                  ▷ Initialize a queue
visited ← List()                                 ▷ Empty list

children ← getChildren(0, H)                     ▷ Get children of the root node: 0
visited.append(0)
queue.append(children)

while queue.length > 0 do                        ▷ While there are items in the queue
    node ← queue.pop()
    visited.append(node)

    children ← getChildren(node, H)
    for each j ∈ children do
        parents ← getParents(j, H)               ▷ Get j's parents
        if all parents in visited then
            queue.append(j)

visited.remove(0)                                ▷ Remove root node
return visited
```

8.2 Experiments and Results

The experiments in this section are carried out to show if TL helps to improve the performance of a semi-supervised hierarchical classifier than when it is not used, in the case when the hierarchy is of DAG type and the instances can be associated to multiple path of labels of partial depth.

Multilayer perceptron (MLP) is the ANN used as the local classifier, which is formed of: input layer, 3 hidden layers with 100 neurons each (activation relu) and output layer (activation sigmoid). The MLP is optimized with the Adam optimizer and binary cross-entropy as the loss by 100 epochs.

Figs. 8.3 and 8.4 show the results, on the GO datasets, of SSHMC-BLI (v1) with and without TL as well as the results of the supervised classifier (LCN[2]). SSHMC-BLI keeps outperforming most of times the supervised classifier as in chapter 6, furthermore, when TL is carried out, the performance of the classifier tend to be improved most of times.

[2]It is trained with the same MLP.

8.3 Discussion

Table 8.1 presents the average rank of the results of the classifiers SSHMC-BLI, with and without TL, and the supervised classifier, where SSHMC-BLI with TL shows the best general performance.

Table 8.1: Average rank of each classifier in the GO collection. In bold the best (lower is better).

	LCN	**SSHMC-BLI**	**SSHMC-BLI+TL**
Avg. Precision	2.94	1.8	**1.26**

To know if there is statistical difference among the SSHMC-BLI the supervised classifier, the Friedman test together with its post-hoc the Nemenyi test were applied [Demšar, 2006].

The result of the Friedman test with $p = 0.05$ for evaluation measure average precision is that the null hypothesis can be rejected in favor of the alternative, that is, the average ranks of the algorithms are not equal. Since the null hypothesis was rejected, the Nemenyi test can be applied. Fig. 8.2 shows the graphical representation of the Nemenyi test for average precision. As it can seen, SSHMC-BLI with TL is statistically better than when TL is not used. Furthermore, SSHMC-BLI with and without TL are statistically better than the supervised classifier.

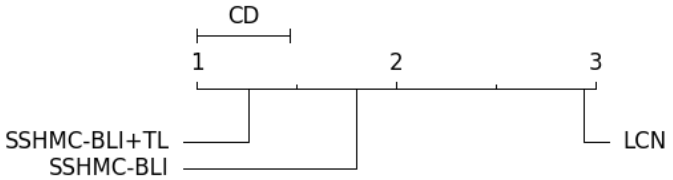


Figure 8.2: Graphical representations of Nemenyi test for the evaluation measures average precision. Classifiers that are not significantly different, with $p = 0.05$, are connected. CD: critical difference. (Lower is better)

8.4 Summary

In this chapter, the semi-supervised hierarchical classifier SSHMC-BLI was extended by including TL among neighboring nodes. The algorithm HSp was proposed to train a hierarchical classifier based on LCN to transfer learning from a parent to its children. Furthermore, it was shown that using TL can help to improve the performance of the semi-supervised hierarchical classifier than when it is not used.

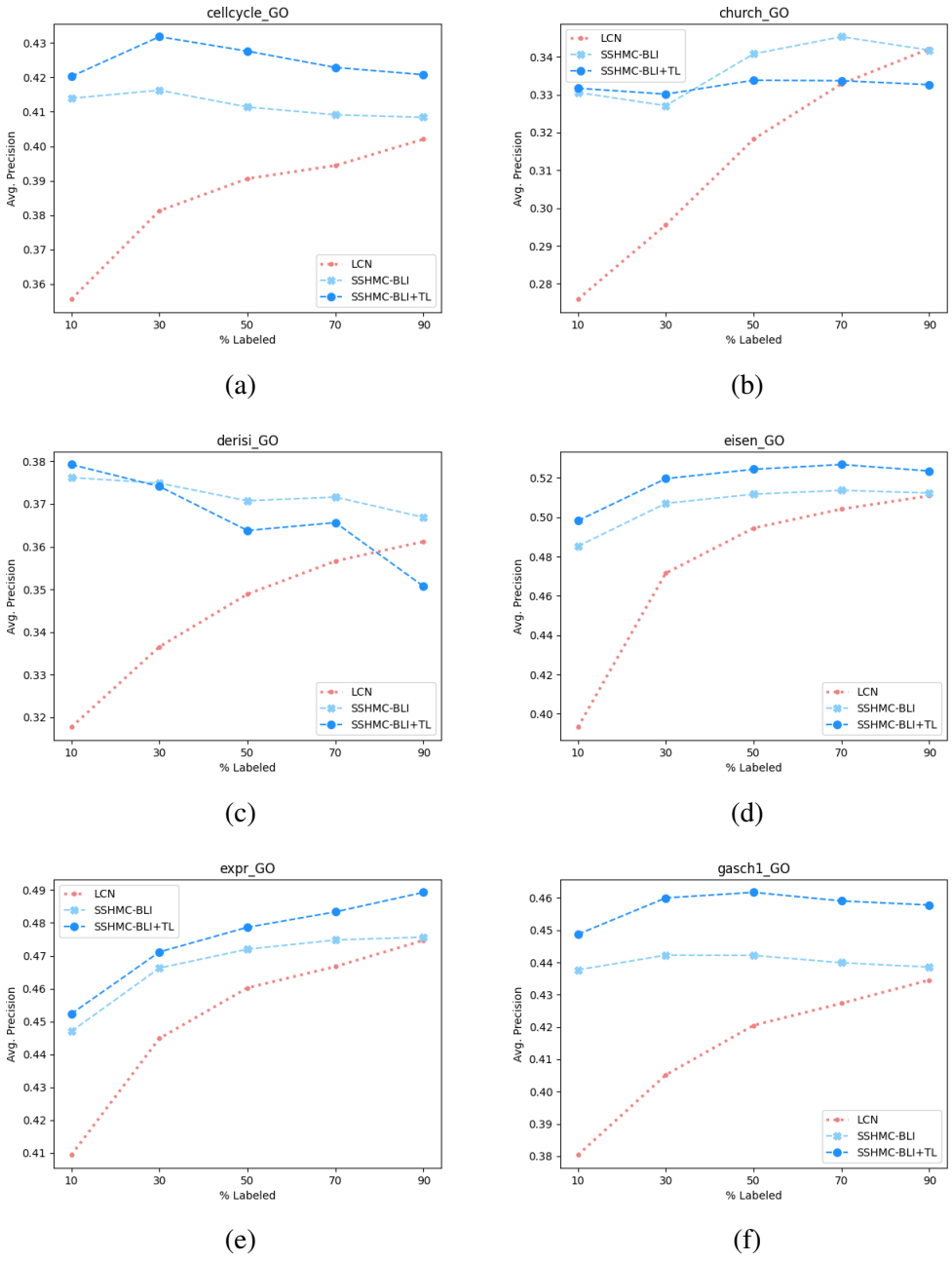


Figure 8.3: Results in terms of average precision (AP) in the following datasets: a) cellcycle, b) church, c) derisi, d) eisen, e) expr and f) gasch1. The x-axis corresponds to the amount of labeled data, while its complement is the unlabeled data. (Best seen in color.)

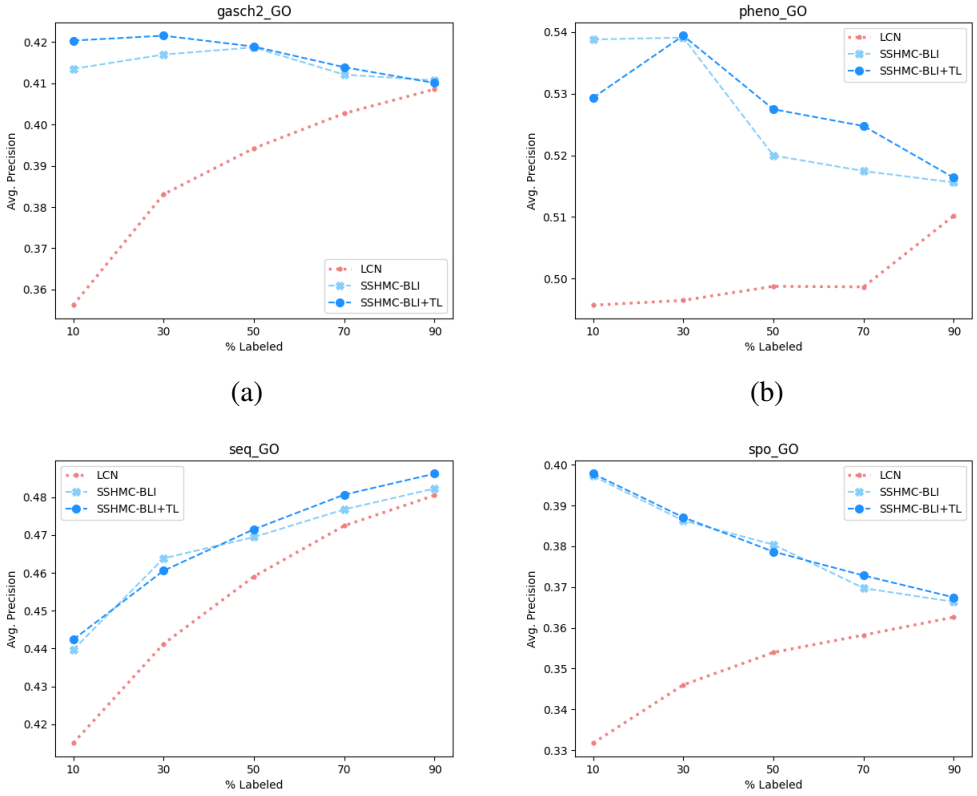


Figure 8.4: Results in terms of average precision in the following datasets: a) gasch2, b) pheno, c) seq and d) spo. The x-axis corresponds to the amount of labeled data, while its complement is the unlabeled data. (Best seen in color.)

Chapter 9

Conclusion and Future Work

Supervised classification often suffers from a lack of labeled data due to the time-consuming and costly nature of hand-labeling. This issue is exacerbated in hierarchical classification, where data is split among nodes, leaving deeper nodes with even less data. To address this, semi-supervised learning can be employed, utilizing both labeled and unlabeled data to train classifiers. Additionally, transfer learning can be applied, allowing upper nodes to share their general information with lower nodes. Therefore, our objective was to develop a semi-supervised hierarchical classifier that can be trained with labeled and unlabeled data to perform classification on datasets where the hierarchy is of DAG type and the instances can be associated to multiple paths of labels of partial depth.

Regarding our proposed method on semi-supervised hierachical classification where hierarchies are of tree type, we can conclude that:

- Use unlabeled data along labeled data can help to improve the performance of a hierarchical classifier trained only on labeled data.
- Our proposed method method, SSHC-BLI, is statistically better than its supervised counterpart and it is competitive with related methods.

Regarding our study on hierarchical classification where the hierarchies are of DAG type and the instances can be associated to multiple paths of labels of partial depth, we can conclude that:

- Use unlabeled data along labeled data can help to improve the performance of a hierarchical classifier trained only on labeled data.
- Our proposed method, SSHMC-BLI, is statistically better than its supervised counterpart and standard semi-supervised methods.

- Making use of a Bayesian network to post-process the output of a hierarchical classifier can boost its performance.

Regarding our study of incorporating transfer learning between neighboring nodes, we can conclude that:

- Transfer learning between neighboring nodes can help to improved the performance of a hierarchical classifier.
- Our proposed method, SSHMC-BLI, is statistically better when TL is carried out than when it is not.

Finally, the contributions of this research are summarized below:

- The similarity function SISI.
- The semi-supervised hierarchical classifier SSHC-BLI for hierarchies of tree type.
- The semi-supervised hierarchical classifiers SSHMC-BLI for hierarchies of DAG type; and its extension with Bayesian networks, SSHBMC.
- The supervised hierarchical classifier BCNN for hierarchical classification of images.
- The algorithm HSp for transfer learning among neighboring nodes.

9.1 Future Work

Different lines of research can be followed:

- Constructing datasets with hierarchies of DAG type to organize data in a structure where each node can have multiple parents, allowing for more flexible and complex relationships compared to tree structures; for various sources, such as text, images, and videos.
- Extension of the semi-supervised methods to handle images, videos and audios by solving the pre-processing requirements of these types of data. This extension could significantly enhance the versatility and applicability of the semi-supervised hierarchical classifier in various domains.
- With respect to transfer learning in hierarchical classification:

- In hierarchies of DAG type, nodes may have multiple parents, hence strategies to transfer learning from multiples parents in a LCN approach may be explored; also strategies that choose the *best* parent to transfer knowledge may be explored.
- Transfer learning in the LCPN approach: in similar way than HSp, TL may be carried out from parent to child, but using LCPN classifiers.
- Transfer learning in the LCL approach: in similar way that HSp, TL may be carried out from upper to bottom levels.

Bibliography

C. C. Aggarwal. *Data Classification: Algorithms and Applications*. Chapman & Hall/CRC, 1st edition, 2014. ISBN 1466586745.

F. Altenberger and C. Lenz. A non-technical survey on deep convolutional neural network architectures. *CoRR*, abs/1803.02129, 2018. URL http://arxiv.org/abs/1803.02129.

Z. Barutçuoglu, R. E. Schapire, O. G. Troyanskaya, and C. DeCoro. Bayesian aggregation for hierarchical classification. Technical report, Princeton University, 2008.

K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Proceedings of the 11th International Conference on Neural Information Processing Systems*, NIPS'98, page 368–374, Cambridge, MA, USA, 1998. MIT Press.

K. P. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 289–296, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 158113567X. doi: 10.1145/775047.775090. URL https://doi.org/10.1145/775047.775090.

W. Bi and J. T. Kwok. Multi-label classification on tree- and dag-structured hierarchies. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 17–24, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL http://dl.acm.org/citation.cfm?id=3104482.3104485.

C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705 – 727, 2011. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2011.01.007. URL http://www.sciencedirect.com/science/article/pii/S0888613X11000314.

K. Bjerge, Q. Geissmann, J. Alison, H. M. Mann, T. T. Høye, M. Dyrmann, and H. Karstoft. Hierarchical classification of insects with multitask learning and anomaly detection. *Ecological Informatics*, 77:102278, 2023. ISSN 1574-9541. doi:

https://doi.org/10.1016/j.ecoinf.2023.102278. URL https://www.sciencedirect.com/science/article/pii/S1574954123003072.

H. Blockeel, L. D. Raedt, and J. Ramon. Top-down induction of clustering trees. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 55–63, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 92–100, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 1581130570. doi: 10.1145/279943.279962. URL https://doi.org/10.1145/279943.279962.

L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984. ISBN 9780412048418. URL https://books.google.com.mx/books?id=JwQx-WOmSyQC.

R. Cerri, A. de Carvalho, and A. F. Comparing local and global hierarchical multilabel classification methods using decision trees. In *Anais Do V Workshop em Algoritmos e Aplicaes de Minerao de Dados*, January 2009.

R. Cerri, R. C. Barros, and A. C. de Carvalho. Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, 80(1):39 – 56, 2014. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2013.03.007. URL http://www.sciencedirect.com/science/article/pii/S0022000013000718.

R. Cerri, R. C. Barros, A. C. P. L. F. de Carvalho, and Y. Jin. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinformatics*, 17(1):373, Sep 2016. ISSN 1471-2105. doi: 10.1186/s12859-016-1232-1. URL https://doi.org/10.1186/s12859-016-1232-1.

O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16 (1):321–357, June 2002. ISSN 1076-9757.

D. Chicco and G. Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 2020. ISSN 1471-2164. doi: 10.1186/s12864-019-6413-7. URL https://doi.org/10.1186/s12864-019-6413-7.

Y. Chong, Y. Ding, Q. Yan, and S. Pan. Graph-based semi-supervised learning: A review. *Neurocomputing*, 408:216–230, 2020. ISSN 0925-2312. doi: https://

doi.org/10.1016/j.neucom.2019.12.130. URL https://www.sciencedirect.com/science/article/pii/S0925231220304938.

W. D. G. de Oliveira and L. Berton. A systematic review for class-imbalance in semi-supervised learning. *Artificial Intelligence Review*, 56(Suppl 2):2349–2382, 2023. doi: 10.1007/s10462-023-10579-0. URL https://doi.org/10.1007/s10462-023-10579-0.

C. Decoro, Z. Barutçuoglu, and R. Fiebrink. Bayesian aggregation for hierarchical genre classification. In *International Symposium on Music Information Retrieval 2007*, pages 77–80, Jan. 2007.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, Dec. 2006. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1248547.1248548.

S. Ding, Z. Zhu, and X. Zhang. An overview on semi-supervised support vector machine. *Neural Comput. Appl.*, 28(5):969–978, May 2017. ISSN 0941-0643. doi: 10.1007/s00521-015-2113-7. URL https://doi.org/10.1007/s00521-015-2113-7.

F. d'Alché-Buc, Y. Grandvalet, and C. Ambroise. Semi-supervised marginboost. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, volume 14, page 553–560, Cambridge, MA, USA, 2002. MIT Press. URL https://proceedings.neurips.cc/paper/2001/file/931af583573227f0220bc568c65ce104-Paper.pdf.

J. Du, C. X. Ling, and Z. Zhou. When does cotraining work in real data? *IEEE Transactions on Knowledge and Data Engineering*, 23(5):788–799, 2011. doi: 10.1109/TKDE.2010.158.

R. Eisner, B. Poulin, D. Szafron, P. Lu, and R. Greiner. Improving protein function prediction using the hierarchical structure of the gene ontology. In *2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–10, Nov 2005. doi: 10.1109/CIBCB.2005.1594940.

T. Fagni and F. Sebastiani. On the selection of negative examples for hierarchical text categorization. *Proceedings 3rd Lang Technology Conference*, January 2007.

S. Feng, P. Fu, and W. Zheng. A hierarchical multi-label classification method based on neural networks for gene function prediction. *Biotechnology & Biotechnological Equipment*, 32(6):1613–1621, 2018. doi: 10.1080/13102818.2018.1521302. URL https://doi.org/10.1080/13102818.2018.1521302.

W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang. Graph random neural networks for semi-supervised learning on graphs. In

H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22092–22103. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb4c835feb0a65cc39739320d7a51c02-Paper.pdf.

Z. Feng, Q. Zhou, Q. Gu, X. Tan, G. Cheng, X. Lu, J. Shi, and L. Ma. Dmt: Dynamic mutual training for semi-supervised learning. *Pattern Recognition*, 130:108777, 2022. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2022.108777. URL https://www.sciencedirect.com/science/article/pii/S0031320322002588.

E. Giunchiglia and T. Lukasiewicz. Coherent hierarchical multi-label classification networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9662–9673. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/6dd4e10e3296fa63738371ec0d5df818-Paper.pdf.

H. M. Gomes, M. Grzenda, R. Mello, J. Read, M. H. Le Nguyen, and A. Bifet. A survey on semi-supervised learning for delayed partially labelled data streams. *ACM Comput. Surv.*, 55(4), Nov. 2022. ISSN 0360-0300. doi: 10.1145/3523055. URL https://doi.org/10.1145/3523055.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618.

Y. Grandvalet, F. d'Alché Buc, and C. Ambroise. Boosting mixture models for semi-supervised learning. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Artificial Neural Networks — ICANN 2001*, pages 41–48, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44668-2.

Q. Gui, H. Zhou, N. Guo, and B. Niu. A survey of class-imbalanced semi-supervised learning. *Machine Learning*, pages 1–30, 2023. doi: 10.1007/s10994-023-06344-7.

C. H. Han, M. Kim, and J. T. Kwak. Semi-supervised learning for an improved diagnosis of covid-19 in ct images. *PLOS ONE*, 16(4):1–13, 04 2021. doi: 10.1371/journal.pone.0249450. URL https://doi.org/10.1371/journal.pone.0249450.

R. Haridas and L. JyothiR. Convolutional neural networks: A comprehensive survey. *International Journal of Applied Engineering Research*, 2019. URL https://api.semanticscholar.org/CorpusID:221223881.

S. Haykin. *Neural Networks and Learning Machines*. Pearson International Edition. Pearson, 2009. ISBN 9780131293762. URL https://books.google.com.mx/books?id=KCwWOAAACAAJ.

J. Hernandez, L. Sucar, and E. Morales. A hybrid global-local approach for hierarchical classification. *FLAIRS 2013 - Proceedings of the 26th International Florida Artificial Intelligence Research Society Conference*, pages 432–437, 01 2013.

E. P. Hubble. Extragalactic nebulae. *The Astrophysical Journal*, 64:321–369, Dec. 1926. doi: 10.1086/143018.

E. P. Hubble. The classification of spiral nebulae. *The Observatory*, 50:276–281, Sept. 1927.

Jiao Wang, Si-wei Luo, and Xian-hua Zeng. A random subspace method for co-training. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 195–200, 2008. doi: 10.1109/IJCNN.2008.4633789.

J. Kahn, A. Lee, and A. Hannun. Self-training for end-to-end speech recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7084–7088, 2020. doi: 10.1109/ICASSP40776.2020.9054295. URL https://doi.org/10.1109/ICASSP40776.2020.9054295.

Z. Kang, C. Peng, Q. Cheng, X. Liu, X. Peng, Z. Xu, and L. Tian. Structured graph learning for clustering and semi-supervised classification. *Pattern Recognition*, 110:107627, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107627. URL https://www.sciencedirect.com/science/article/pii/S0031320320304301.

N. E. Khalifa, M. Hamed Taha, A. E. Hassanien, and I. Selim. Deep galaxy v2: Robust deep convolutional neural networks for galaxy morphology classifications. In *2018 International Conference on Computing Sciences and Engineering (ICCSE)*, pages 1–6, 2018. doi: 10.1109/ICCSE1.2018.8374210.

A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artif. Intell. Rev.*, 53(8):5455–5516, 2020. doi: 10.1007/s10462-020-09825-6. URL https://doi.org/10.1007/s10462-020-09825-6.

A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby. Big transfer (bit): General visual representation learning. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part V*, volume 12350 of *Lecture Notes in Computer Science*, pages 491–507. Springer, 2020. doi: 10.1007/978-3-030-58558-7_29. URL https://doi.org/10.1007/978-3-030-58558-7_29.

A. Kosmopoulos, G. Paliouras, and I. Androutsopoulos. Probabilistic Cascading for Large Scale Hierarchical Classification. *arXiv e-prints*, art. arXiv:1505.02251, May 2015.

K. Kowsari, D. E. Brown, M. Heidarysafa, K. Jafari Meimandi, M. S. Gerber, and L. E. Barnes. Hdltex: Hierarchical deep learning for text classification. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2017. doi: 10.1109/icmla.2017.0-134. URL http://dx.doi.org/10.1109/ICMLA.2017.0-134.

K. Kowsari, R. Sali, L. Ehsan, W. Adorno, A. Ali, S. Moore, B. Amadi, P. Kelly, S. Syed, and D. Brown. Hmic: Hierarchical medical image classification, a deep learning approach. *Information*, 11(6):318, Jun 2020. ISSN 2078-2489. doi: 10.3390/info11060318. URL http://dx.doi.org/10.3390/info11060318.

J. Levatić, M. Ceci, D. Kocev, and S. Džeroski. Semi-supervised predictive clustering trees for (hierarchical) multi-label classification. *International Journal of Intelligent Systems*, 2024(1):5610291, 2024. doi: https://doi.org/10.1155/2024/5610291. URL https://onlinelibrary.wiley.com/doi/abs/10.1155/2024/5610291.

J. Li, Y. Rong, H. Cheng, H. Meng, W. Huang, and J. Huang. Semi-supervised graph classification: A hierarchical graph perspective. In *The World Wide Web Conference*, WWW '19, page 972–982, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313461. URL https://doi.org/10.1145/3308558.3313461.

J. Li, Y. Huang, H. Chang, and Y. Rong. Semi-supervised hierarchical graph classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (5):6265–6276, 2023. doi: 10.1109/TPAMI.2022.3203703.

Y. Li and Z. Zhou. Towards making unlabeled data never hurt. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):175–188, 2015. doi: 10.1109/TPAMI.2014.2299812.

Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Trans. Neural Networks Learn. Syst.*, 33(12):6999–7019, 2022. doi: 10.1109/TNNLS.2021.3084827. URL https://doi.org/10.1109/TNNLS.2021.3084827.

H. Liu, H. Zhang, B. Li, X. Yu, Y. Zhang, and T. Penzel. Msleepnet: A semi-supervision-based multiview hybrid neural network for simultaneous sleep arousal and sleep stage detection. *IEEE Transactions on Instrumentation and Measurement*, 73:1–9, 2024. doi: 10.1109/TIM.2023.3348898.

P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu. Semiboost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2000–2014, 2009. doi: 10.1109/TPAMI.2008.235.

J. Metz and A. A. Freitas. Extending hierarchical classification with semi-supervised learning. In *Proceedings of the UK Workshop on Computational Intelligence*, pages 1–6, 2009.

F. Mukti, C. Eswaran, N. Hashim, C. C. Ho, and M. U. A. Ayoobkhan. An automated grading system for diabetic retinopathy using curvelet transform and hierarchical classification. *International Journal of Engineering and Technology(UAE)*, 7:154–157, January 2018. doi: 10.14419/ijet.v7i2.15.11375.

G. Murtaza, L. Shuib, G. Mujtaba, and G. Raza. Breast cancer multi-classification through deep neural network and hierarchical classification approach. *Multimedia Tools and Applications*, 79:15481–15511, 2019.

A. Naik and H. Rangwala. Embedding feature selection for large-scale hierarchical classification. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1212–1221, Dec 2016. doi: 10.1109/BigData.2016.7840725.

A. Naik and H. Rangwala. Filter based taxonomy modification for improving hierarchical classification. *CoRR*, abs/1603.00772, 2016. URL `http://arxiv.org/abs/1603.00772`.

F. K. Nakano, W. J. Pinto, G. L. Pappa, and R. Cerri. Top-down strategies for hierarchical classification of transposable elements with neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2539–2546, May 2017. doi: 10.1109/IJCNN.2017.7966165.

M. Needham and A. Hodler. *Graph Algorithms: Practical Examples in Apache Spark and Neo4j*. O'Reilly Media, 2019. ISBN 9781492047681. URL `https://books.google.com.mx/books?id=UwIevgEACAAJ`.

S. Palacio, J. Folz, J. Hees, F. Raue, D. Borth, and A. Dengel. What do deep networks like to see? In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3108–3117. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00328. URL `http://openaccess.thecvf.com/content_cvpr_2018/html/Palacio_What_Do_Deep_CVPR_2018_paper.html`.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

M. Panta, A. Mishra, M. T. Hoque, and J. Atallah. Machine learning based prediction of hierarchical classification of transposable elements, 2019. URL `https://arxiv.org/abs/1907.01674`.

R. M. Pereira, D. Bertolini, L. O. Teixeira, C. N. Silla, and Y. M. Costa. Covid-19 identification in chest x-ray images on flat and hierarchical classification scenarios. *Computer Methods and Programs in Biomedicine*, 194:105532, 2020. ISSN 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2020.105532. URL `https://www.sciencedirect.com/science/article/pii/S0169260720309664`.

S. Perera, O. Raz, R. Routray, S. Bao, and M. Zalmanovici. Optimizing hierarchical classification with adaptive node collapses. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

A. Petukhova and N. Fachada. Mn-ds: A multilabeled news dataset for news articles hierarchical classification. *Data*, 8(5), 2023. ISSN 2306-5729. doi: 10.3390/data8050074. URL https://www.mdpi.com/2306-5729/8/5/74.

J. Pino, Q. Xu, X. Ma, M. J. Dousti, and Y. Tang. Self-Training for End-to-End Speech Translation. In *Proc. Interspeech 2020*, pages 1476–1480, 2020. doi: 10.21437/Interspeech.2020-2938. URL https://doi.org/10.21437/Interspeech.2020-2938.

K. Pliakos, I. Triguero, D. Kocev, and C. Vens. Representational power of gene features for function prediction, 2015. URL https://lirias.kuleuven.be/1776146&lang=en.

M. Ramírez-Corona, L. E. Sucar, and E. F. Morales. Hierarchical multilabel classification based on path evaluation. *International Journal of Approximate Reasoning*, 68:179–193, 2016. doi: 10.1016/j.ijar.2015.07.008. URL https://doi.org/10.1016/j.ijar.2015.07.008.

P. N. Robinson, M. Frasca, S. Köhler, M. Notaro, M. Re, and G. Valentini. A hierarchical ensemble method for dag-structured taxonomies. In F. Schwenker, F. Roli, and J. Kittler, editors, *Multiple Classifier Systems*, pages 15–26, Cham, 2015. Springer International Publishing. ISBN 978-3-319-20248-8.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 0033-295X. doi: 10.1037/h0042519. URL http://dx.doi.org/10.1037/h0042519.

R. Sali, S. Adewole, L. Ehsan, L. A. Denson, P. Kelly, B. C. Amadi, L. Holtz, S. A. Ali, S. R. Moore, S. Syed, and D. E. Brown. Hierarchical deep convolutional neural networks for multi-category diagnosis of gastrointestinal disorders on histopathological images, 2020.

A. Santos and A. Canuto. Applying semi-supervised learning in hierarchical multi-label classification. *Expert Systems with Applications*, 41(14):6075–6085, 2014a. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2014.03.052. URL https://www.sciencedirect.com/science/article/pii/S0957417414001900.

A. Santos and A. Canuto. Applying the self-training semi-supervised learning in hierarchical multi-label methods. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 872–879, 2014b. doi: 10.1109/IJCNN.2014.6889565.

A. M. Santos and A. M. P. Canuto. Using semi-supervised learning in multi-label classification problems. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2012. doi: 10.1109/IJCNN.2012.6252800.

L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, and S. Džeroski. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11, 2010. doi: https://doi.org/10.1186/1471-2105-11-2.

Y. Seo and K. shik Shin. Hierarchical convolutional neural networks for fashion image classification. *Expert Systems with Applications*, 116:328–339, 2019. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.09.022. URL https://www.sciencedirect.com/science/article/pii/S0957417418305992.

J. Serrano-Pérez. Hierarchical classification with bayesian networks and chained classifiers, 2019. URL https://inaoe.repositorioinstitucional.mx/jspui/handle/1009/1948.

J. Serrano-Pérez and L. E. Sucar. Hierarchical classification with bayesian networks and chained classifiers. In *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, Sarasota, Florida, USA, May 19-22 2019.*, pages 488–493, 2019.

J. Serrano-Pérez and L. E. Sucar. Semi-supervised hierarchical classification based on local information. In A. C. Bicharra Garcia, M. Ferro, and J. C. Rodríguez Ribón, editors, *Advances in Artificial Intelligence – IBERAMIA 2022*, pages 255–266, Cham, 2022. Springer International Publishing. ISBN 978-3-031-22419-5. doi: 10.1007/978-3-031-22419-5_22. URL https://doi.org/10.1007/978-3-031-22419-5_22.

J. Serrano-Pérez and L. E. Sucar. Semi-supervised hierarchical bayesian multi-label classification. In L. Correia, A. Rosá, and F. Garijo, editors, *Advances in Artificial Intelligence – IBERAMIA 2024*, pages 275–286, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-80366-6. doi: 10.1007/978-3-031-80366-6_23. URL https://doi.org/10.1007/978-3-031-80366-6_23.

J. Serrano-Pérez and L. E. Sucar. Artificial datasets for hierarchical classification. *Expert Systems with Applications*, 182:115218, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.115218. URL https://www.sciencedirect.com/science/article/pii/S0957417421006515.

J. Serrano-Pérez and L. E. Sucar. A semi-supervised hierarchical classifier based on local information. *Pattern Analysis and Applications*, September 2024. doi: 10.1007/s10044-024-01345-1. URL https://doi.org/10.1007/s10044-024-01345-1.

J. Serrano-Pérez and L. E. Sucar. Semi-supervised hierarchical multi-label classifier based on local information. *International Journal of Approximate Reasoning*, 181:109411, 2025. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2025.109411. URL https://www.sciencedirect.com/science/article/pii/S0888613X25000520.

J. Serrano-Pérez, R. D. Hernández, and L. E. Sucar. Bayesian and convolutional networks for hierarchical morphological classification of galaxies. *Experimental Astronomy*, August 2024. doi: 10.1007/s10686-024-09950-y. URL https://doi.org/10.1007/s10686-024-09950-y.

C. Shi, Z. Lv, X. Yang, P. Xu, and I. Bibi. Hierarchical multi-view semi-supervised learning for very high-resolution remote sensing image classification. *Remote Sensing*, 12(6), 2020. ISSN 2072-4292. doi: 10.3390/rs12061012. URL https://www.mdpi.com/2072-4292/12/6/1012.

C. N. Silla and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1):31–72, January 2011. ISSN 1573-756X. doi: 10.1007/s10618-010-0175-9. URL https://doi.org/10.1007/s10618-010-0175-9.

V. Sindhwani and D. S. Rosenberg. An rkhs for multi-view learning and manifold co-regularization. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 976–983, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390279. URL https://doi.org/10.1145/1390156.1390279.

V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning with multiple views. In *Proceedings of the 22nd ICML Workshop on Learning with Multiple Views*, 2005.

Z. Song, X. Yang, Z. Xu, and I. King. Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8174–8194, 2023. doi: 10.1109/TNNLS.2022.3155478.

N. E. Stork. How many species of insects and other terrestrial arthropods are there on earth? *Annual Review of Entomology*, 63(Volume 63, 2018):31–45, 2018. ISSN 1545-4487. doi: https://doi.org/10.1146/annurev-ento-020117-043348. URL https://www.annualreviews.org/content/journals/10.1146/annurev-ento-020117-043348.

L. E. Sucar, C. Bielza, E. F. Morales, P. Hernandez-Leal, J. H. Zaragoza, and P. Larrañaga. Multi-label classification with bayesian network-based chain classifiers. *Pattern Recognition Letters*, 41:14 – 22, 2014. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2013.11.007. URL http://www.sciencedirect.

com/science/article/pii/S0167865513004303. Supervised and Unsupervised Classification Techniques and their Applications.

Z. Sun, Y. Zhao, D. Cao, and H. Hao. Hierarchical multilabel classification with optimal path prediction. *Neural Processing Letters*, 45(1):263–277, Feb 2017. ISSN 1573-773X. doi: 10.1007/s11063-016-9526-x. URL https://doi.org/10.1007/s11063-016-9526-x.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In S. Singh and S. Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4278–4284. AAAI Press, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806.

M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10096–10106. PMLR, 2021. URL http://proceedings.mlr.press/v139/tan21a.html.

P. Tresson, D. Carval, P. Tixier, and W. Puech. Hierarchical classification of very small objects: Application to the detection of arthropod species. *IEEE Access*, 9:63925–63932, 2021. doi: 10.1109/ACCESS.2021.3075293. URL https://doi.org/10.1109/ACCESS.2021.3075293.

G. Tsoumakas, I. Katakis, and I. Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2011. doi: 10.1109/TKDE.2010.164.

J. E. van Engelen and H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109:373–440, 2019. doi: 10.1007/s10994-019-05855-6. URL https://doi.org/10.1007/s10994-019-05855-6.

C. Vens, J. Struyf, L. Schietgat, S. Dzeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. *Machine learning*, 73(2):185–214, 2008a. ISSN 0885-6125.

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

S. Wan, S. Pan, J. Yang, and C. Gong. Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):10049–10057, May 2021. doi: 10.1609/aaai.v35i11.17206. URL https://ojs.aaai.org/index.php/AAAI/article/view/17206.

W. Wang and Z.-H. Zhou. A new analysis of co-training. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1135–1142, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

J. J. Wiens. How many species are there on earth? progress and problems. *PLOS Biology*, 21(11):1–4, 11 2023. doi: 10.1371/journal.pbio.3002388. URL https://doi.org/10.1371/journal.pbio.3002388.

Z. Wu and S. Saito. Hinet: Hierarchical classification with neural network, 2018. URL https://arxiv.org/abs/1705.11105.

H. Xiao, X. Liu, and Y. Song. Efficient path prediction for semi-supervised and weakly supervised hierarchical text classification. In *The World Wide Web Conference*, WWW '19, page 3370–3376, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313658. URL https://doi.org/10.1145/3308558.3313658.

C. Xu, D. Tao, and C. Xu. A survey on multi-view learning, 2013. URL https://arxiv.org/abs/1304.5634.

H. Xu, H. Xiao, H. Hao, L. Dong, X. Qiu, and C. Peng. Semi-supervised learning with pseudo-negative labels for image classification. *Knowledge-Based Systems*, 260:110166, 2023. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2022.110166. URL https://www.sciencedirect.com/science/article/pii/S095070512201262X.

Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, and M. Auli. Self-training and pre-training are complementary for speech recognition. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3030–3034, 2021. doi: 10.1109/ICASSP39728.2021.9414641.

G. H. Yamashita, M. J. Anzanello, F. Soares, M. K. Rocha, F. S. Fogliatto, N. P. Rodrigues, E. Rodrigues, P. G. Celso, V. Manfroi, and P. F. Hertz. Hierarchical classification of sparkling wine samples according to the country of origin based on the most informative chemical elements. *Food Control*, 106:106737, 2019. ISSN 0956-7135. doi: https://doi.org/10.1016/j.foodcont.2019.106737. URL http://www.sciencedirect.com/science/article/pii/S0956713519303184.

J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10368–10378, June 2021.

L. Yang, W. Zhuo, L. Qi, Y. Shi, and Y. Gao. St++: Make self-training work better for semi-supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4268–4277, June 2022.

X. Yang, Z. Song, I. King, and Z. Xu. A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):8934–8954, 2023. doi: 10.1109/TKDE.2022.3220219.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, page 189–196, USA, 1995. Association for Computational Linguistics. doi: 10.3115/981658.981684. URL `https://doi.org/10.3115/981658.981684`.

M. Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2(30):6, 2004.

X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, July 2008.

F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021. doi: 10.1109/JPROC.2020.3004555.

Appendices

Appendix A

Text preprocessing

The preprocessing applied to the 20 newsgroup is described next:

1. Punctuation marks were removed: period, comma, parentheses...
2. Lemmatization of the words.
3. Stop words were removed: *a, an, do, in, for, by...*
4. Words with a frequency lower than 2 were removed.
5. The representation *Term Frequency * Inverse Document Frequency* (Tf-Idf) was calculated, where rare features in the corpus have their value increased.
6. The Tf-Idf representation was transformed into a latent space of lower dimensionality. Latent semantic indexing (LSI) was applied with 50 latent dimensions.

The model for preprocessing was built from the training set, which was later applied to the test set. Finally, the training set was divided into *training* and *validation* sets, with proportions of 80% and 20%, respectively.

Appendix B

Similarity measures

In this appendix, several similarity measures are compared against SISI.

Table B.1 describes several similarity measures. Measures from Manhattan to Canberra are used to measure the similarity between two points. Cosine is used to measure the cosine of the angle between two points, however, it is not a metric since it does not comply the triangle inequality. Nevertheless, none of the previous measures is able to measure the similarity of a point with a set of points.

On the other hand, Mahalanobis is able to measure the distances between a point and a distribution, which requires a *significant* amount of data to estimate the probability distribution D, i.e. the mean μ and the positive semi-definite covariance matrix S, amount of data that is not available in the present work, considering that number of labeled neighbors is at most five. Moreover, none of the above similarity measures return a value in the interval $[0, 1]$.

Therefore, SISI is proposed as an option to measure the similarity of a point with a small set of points, which takes into account the distances among the set of points, and the distances of the point with the points of the set. Furthermore, SISI returns a score in $[0, 1]$.

Table B.1: List of similarity measures, where $a, b \in \mathbb{R}^n$ are two points, D is a probability distribution on $\mathbb{R}^n$ with mean $\mu = (\mu_1, \mu_2, ..., \mu_n)$ and S is a positive semi-definite covariance matrix.

Name	Equation	Description
Manhattan	$d(a,b) = \sum_{i=1}^{n} \mid a_i - b_i \mid$	Measures the sum of absolute differences between two points.
Euclidean	$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$	Measures the straight-line distance between two points in a feature space.
Minkowski	$d(a,b) = \sqrt[p]{\sum_{i=1}^{n} \mid a_i - b_i \mid^p}$	Generalization of Manhattan and Euclidean distances.
Chebyshev	$d(a,b) = \max_i \mid a_i - b_i \mid$	Special case of the Minkowski distance where p goes to infinity
Canberra	$d(a,b) = \sum_{i=1}^{n} \frac{\mid a_i - b_i \mid}{\mid a_i \mid + \mid b_i \mid}$	Weighted version of the Manhattan distance.
Cosine	$d(a,b) = \frac{a \cdot b}{\|a\|\|b\|}$	Measures the cosine of the angle between two vectors.
Mahalanobis	$d(a,D) = \sqrt{(a-\mu)S^{-1}(a-\mu)^T}$	Measure of the distance between a point and a distribution
SISI	Equation 5.6	Similarity of an instance with a set of instances

Appendix C

Metric

Let X be a set, and let d be a function defined on pair of elements of X. d is a metric of space X if the following axioms are satisfied for all $x_1, x_2, x_3 \in X$:

- $d(x_1, x_2) \geq 0$
- $d(x_1, x_2) = 0$, if and only if $x_1 = x_2$, *Identity.*
- $d(x_1, x_2) = d(x_2, x_1)$, *Symmetry.*
- $d(x_1, x_2) \leq d(x_1, x_3) + d(x_3, x_2)$, *triangle inequality.*

Appendix D

Image augmentation

Data augmentation is carried out to balance the leaf nodes. It consists on applying some operations to the available images, such as flip, scale, rotation, etc., to obtain different images.

First, let $LMAX$ be the largest number of associated images to the leaf nodes. Then, for each l leaf node with m_l instances associated: generate n images, where $n = LMAX * 0.95 - m_l$; the n generated images will be associated to l and all its ancestors given by the hierarchy. Image generation follows the next pipeline (iterating over the images associated with the l node):

1. Flip (either horizontal or vertical) with a probability of 0.5.
2. Shift both x and y axis in the range [-10, 10] (percent) with probability one.
3. Scale in the range [-30, 30] (percent) with probability one.
4. Rotate in the range [0, 360] (degrees) with probability one.
5. To fill the voids that may appear in the images due to the different transformation, the strategy *border reflect*[1] is used to fill those pixels.
6. Resize to 300x300 pixels.

An example is shown in Fig. D.1. Two images are generated from the image on the left.

[1] BORDER_REFLECT_101 from opencv.org

Figure D.1: example of generation of artificial images. The image on the left is the available one; the two on the left are generated images after applying some operations to the image on the left.

Appendix E

DAGs Standard Methods for Semi-Supervised Hierarchical Classification

Beyond comparing the proposed semi-supervised methods (for DAG hierarchies) with a supervised classifier, a couple of semi-supervised methods were developed. They are described next:

- *Self-Training for Multi-Label classification* (STML): for each label a binary classifier is self-trained using the whole unlabeled set; the prediction for a new instance is the union of the predictions of the binary classifiers (which can also be probabilities). This method (as any multi-label classifier) ignores the hierarchy, and its predictions do not guarantee to comply neither the hierarchical constraint nor the hierarchical probability constraint.
- *Self-Training Hierarchical Classifier* (STHC): each node of the hierarchy is self-trained like STML, but in the prediction phase, the predictions of the local classifiers are post-processed such that they comply the hierarchical probability constraint, that is, if the probability in a node is greater than the probability of its parent with the lowest probability, its probability is reduced down to the probability of that parent. STHC is introduced as a *standard* semi supervised classifier for hierarchical classification where the hierarchy is a DAG and the instances are associated to multiple paths of labels. Furthermore, it can be seen as an extension of self-train A [Metz and Freitas, 2009] where the predictions of the binary classifiers are post-processed to comply the hierarchical probability constraint instead of applying the top-down procedure.

Appendix F

Tables of Results on Datasets with Hierarchies of Tree Type

Table F.1 shows the results of the SSHC-BLI variants (V1, V2, V3) and the related methods with the evaluation measure exact match. Table F.2 shows the results with hF. Finally, Table F.3 shows the results with the evaluation measure MCC.

Table F.1: Results in the datasets with the evaluation measure exact match of the SSHC-BLI variants (V1, V2, V3) and related methods.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
EA_01_FD_b_10	100	98.333	98.333	98.333	93.889	100
EA_01_FD_b_30	98.889	99.444	99.444	99.444	96.111	100
EA_01_FD_b_50	99.444	98.889	98.889	98.889	95.556	100
EA_01_FD_b_70	98.889	98.889	98.889	98.889	95.556	100
EA_01_FD_b_90	98.889	98.333	98.333	98.333	96.667	100
EA_01_FD_ub_10	100	100	100	100	95.256	100
EA_01_FD_ub_30	99.872	100	100	100	96.795	100
EA_01_FD_ub_50	99.872	100	100	100	97.308	100
EA_01_FD_ub_70	99.872	100	100	100	98.333	100
EA_01_FD_ub_90	99.872	100	100	100	98.205	100
EA_02_FD_b_10	96.667	97.778	98.333	98.333	72.222	100
EA_02_FD_b_30	95.556	97.778	97.222	97.222	77.222	100
EA_02_FD_b_50	96.111	98.333	98.333	98.333	81.111	100
EA_02_FD_b_70	95.556	97.222	97.222	97.222	83.889	100
EA_02_FD_b_90	96.111	97.222	97.222	97.222	85.556	100
EA_02_FD_ub_10	95.641	99.231	99.487	99.487	77.051	99.487
EA_02_FD_ub_30	97.949	99.103	99.231	99.231	83.333	99.615

(Continues in the next page)

Table F.1: Results in the datasets with the evaluation measure exact match of the SSHC-BLI variants (V1, V2, V3) and related methods, TD: Top-Down, STA: self-train A, HMC-SSBR: Hierarchical Multi-label Classification using Semi-Supervised Binary Relevance.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
EA_02_FD_ub_50	98.718	99.103	99.231	99.231	86.026	99.615
EA_02_FD_ub_70	98.59	99.231	99.359	99.359	87.436	99.615
EA_02_FD_ub_90	98.846	98.974	98.974	98.974	87.179	99.615
HA_01_FD_b_10	85.476	98.333	99.048	99.048	86.19	99.048
HA_01_FD_b_30	95.714	98.571	99.286	99.286	89.048	100
HA_01_FD_b_50	98.571	99.048	99.286	99.286	90.714	100
HA_01_FD_b_70	99.286	98.81	99.048	99.048	92.857	100
HA_01_FD_b_90	98.81	98.571	98.571	98.571	91.905	100
HA_01_FD_ub_10	96.465	99.394	99.394	99.394	87.778	99.697
HA_01_FD_ub_30	98.182	99.293	99.192	99.192	92.323	99.596
HA_01_FD_ub_50	98.586	99.192	99.495	99.495	93.636	99.697
HA_01_FD_ub_70	98.788	98.99	99.091	99.091	95.354	99.899
HA_01_FD_ub_90	99.091	99.192	99.192	99.192	95.455	100
HA_02_FD_b_10	10	11.905	8.571	7.857	8.333	14.048
HA_02_FD_b_30	10.238	10	12.381	11.667	12.143	13.333
HA_02_FD_b_50	13.571	12.143	14.286	14.286	12.381	11.429
HA_02_FD_b_70	15	16.667	14.286	14.286	14.286	9.524
HA_02_FD_b_90	15	17.381	16.667	16.667	15.714	7.619
HA_02_FD_ub_10	13.583	13	13.25	12.583	13.333	14.667
HA_02_FD_ub_30	15.75	15.25	15	15.667	14.833	13.75
HA_02_FD_ub_50	16.667	15.5	15.167	15.583	16.917	12.75
HA_02_FD_ub_70	16.833	17.667	15.667	17.333	16.833	11.833
HA_02_FD_ub_90	18.25	18.667	18.75	18.75	18.5	9.833
HA_03_FD_b_10	9.762	12.857	10.952	11.429	9.286	9.048
HA_03_FD_b_30	12.857	14.524	13.333	12.143	13.333	9.286
HA_03_FD_b_50	14.524	13.81	16.19	15	13.571	10
HA_03_FD_b_70	14.762	15.714	16.19	16.19	12.619	11.429
HA_03_FD_b_90	16.19	15.714	15.476	15.476	13.333	10.714
HA_03_FD_ub_10	15.5	14.917	12	10.917	13.25	14.583
HA_03_FD_ub_30	16.333	16.5	15.833	14.25	14.75	14.5
HA_03_FD_ub_50	19.833	20.417	18.833	19.583	16.583	15.583
HA_03_FD_ub_70	18.417	20.417	19	19	18.25	15.083
HA_03_FD_ub_90	20	20.583	20.417	20.417	19.083	15.917
HA_09_FD_b_10	4.091	3.807	3.939	3.788	3.655	0.303
HA_09_FD_b_30	4.943	4.867	5.076	4.886	4.223	0.057
HA_09_FD_b_50	5.095	5.284	5.871	5.701	4.413	0

(Continues in the next page)

Table F.1: Results in the datasets with the evaluation measure exact match of the SSHC-BLI variants (V1, V2, V3) and related methods, TD: Top-Down, STA: self-train A, HMC-SSBR: Hierarchical Multi-label Classification using Semi-Supervised Binary Relevance.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
HA_09_FD_b_70	5.568	4.962	4.981	5.284	4.886	0.038
HA_09_FD_b_90	5.246	5.511	5.417	5.417	5.398	0.038
HA_09_FD_ub_10	5.143	4.789	4.745	4.59	5	2.345
HA_09_FD_ub_30	5.821	5.933	5.914	5.715	4.956	1.748
HA_09_FD_ub_50	5.74	6.119	5.927	6.082	5.429	1.275
HA_09_FD_ub_70	6.057	6.412	6.231	6.256	5.628	0.964
HA_09_FD_ub_90	6.287	6.604	6.598	6.598	5.902	0.759
HA_10_FD_b_10	7.073	7.297	7.093	7.886	6.077	3.089
HA_10_FD_b_30	9.634	10.061	9.329	9.756	7.215	4.167
HA_10_FD_b_50	10.671	11.138	10.285	10.63	8.415	2.419
HA_10_FD_b_70	11.098	11.585	11.89	11.89	9.126	2.256
HA_10_FD_b_90	11.484	11.809	11.809	11.809	9.37	2.033
HA_10_FD_ub_10	9.903	10.603	9.546	10.149	8.601	9.866
HA_10_FD_ub_30	12.567	13.08	12.969	13.266	10.789	10.878
HA_10_FD_ub_50	14.04	14.442	13.772	14.673	11.235	10.737
HA_10_FD_ub_70	14.576	15.26	15.149	14.94	12.054	10.625
HA_10_FD_ub_90	16.012	16.287	16.369	16.369	11.897	9.643
VH_01_FD_b_10	10.37	12.593	12.593	12.593	11.852	10.37
VH_01_FD_b_30	13.704	11.852	14.074	13.704	11.852	10.741
VH_01_FD_b_50	13.704	13.333	13.333	14.074	14.444	11.852
VH_01_FD_b_70	10.741	13.333	12.593	12.593	10.37	10
VH_01_FD_b_90	11.111	11.481	11.481	11.481	10.37	5.185
VH_01_FD_ub_10	8.081	12.323	13.535	14.343	11.515	14.141
VH_01_FD_ub_30	8.687	10.505	10.303	10.707	10.909	14.343
VH_01_FD_ub_50	7.071	9.293	9.293	9.293	10.101	11.717
VH_01_FD_ub_70	6.465	7.677	7.879	7.879	9.899	4.848
VH_01_FD_ub_90	5.859	6.667	6.263	6.263	8.283	3.838
VH_02_FD_b_10	30.476	32.381	30.952	20.476	30	27.143
VH_02_FD_b_30	45.714	52.381	47.619	45.238	34.762	38.571
VH_02_FD_b_50	55.714	55.714	56.667	55.714	40.952	37.619
VH_02_FD_b_70	59.048	60.952	60.476	61.429	40.952	42.381
VH_02_FD_b_90	64.762	65.238	66.19	66.19	45.238	49.048
VH_02_FD_ub_10	47.03	50.061	49.939	25.333	38.424	47.273
VH_02_FD_ub_30	62.182	63.879	62.182	55.152	43.758	47.273
VH_02_FD_ub_50	68.364	69.333	68.606	65.818	48.242	54.303
VH_02_FD_ub_70	71.03	73.212	71.273	72.364	50.667	60.242

(Continues in the next page)

Table F.1: Results in the datasets with the evaluation measure exact match of the SSHC-BLI variants (V1, V2, V3) and related methods, TD: Top-Down, STA: self-train A, HMC-SSBR: Hierarchical Multi-label Classification using Semi-Supervised Binary Relevance.

Dataset	**TD**	**V1**	**V2**	**V3**	**STA**	**HMC-SSBR**
VH_02_FD_ub_90	74.303	75.879	75.515	75.515	51.758	63.758
VH_03_FD_b_10	28.571	38.413	36.19	17.619	23.016	27.302
VH_03_FD_b_30	46.667	48.095	47.143	40.476	28.73	28.254
VH_03_FD_b_50	50.952	54.762	53.492	53.492	34.444	27.778
VH_03_FD_b_70	57.143	60.952	60.317	60.317	37.778	28.889
VH_03_FD_b_90	62.222	63.492	63.651	63.651	39.683	31.27
VH_03_FD_ub_10	34.288	42.059	34.015	14.656	28.903	29.039
VH_03_FD_ub_30	55.283	59.646	58.078	50.784	33.674	32.515
VH_03_FD_ub_50	60.191	61.486	61.213	61.213	39.877	32.993
VH_03_FD_ub_70	64.213	66.667	66.599	66.258	41.581	33.061
VH_03_FD_ub_90	68.439	69.257	69.53	69.53	41.65	34.697
VH_08_FD_b_10	7.249	7.381	7.434	7.249	5.132	7.116
VH_08_FD_b_30	9.365	9.788	9.735	8.386	6.376	8.148
VH_08_FD_b_50	10.026	11.19	10.556	11.19	7.46	8.466
VH_08_FD_b_70	11.243	11.931	11.481	11.481	8.148	8.677
VH_08_FD_b_90	11.587	11.481	11.693	11.693	8.704	8.651
VH_08_FD_ub_10	8.92	9.091	9.451	8.068	7.519	10.199
VH_08_FD_ub_30	11.809	12.358	12.311	10.994	9.555	12.434
VH_08_FD_ub_50	13.475	14.148	14.46	14.025	10.994	12.898
VH_08_FD_ub_70	15.246	15.095	14.962	15.36	11.117	13.153
VH_08_FD_ub_90	16.335	16.496	16.241	16.241	12.121	13.144
o20NG_50t_10	60.67	59.904	58.559	51.779	41.591	56.103
o20NG_50t_30	63.75	64.007	63.98	63.193	45.512	58.147
o20NG_50t_50	64.511	65.1	64.795	64.467	47.57	59.112
o20NG_50t_70	65.312	65.817	65.538	65.401	48.628	59.785
o20NG_50t_90	65.711	65.959	66.016	66.016	50.212	60.139
cellcycle_10	12.657	14.709	15.583	16.914	11.288	9.806
cellcycle_30	15.279	16.914	16.724	16.154	13.569	5.511
cellcycle_50	18.206	19.08	19.156	18.662	12.961	14.253
cellcycle_70	19.574	19.46	19.916	19.916	13.873	13.227
cellcycle_90	20.601	21.171	21.817	21.817	15.013	13.759
derisi_10	10.46	11.786	11.418	10.755	11.565	3.757
derisi_30	13.37	13.076	12.634	12.67	11.823	6.298
derisi_50	13.996	14.659	14.401	14.328	12.265	7.035
derisi_70	14.659	15.101	15.359	15.028	13.517	7.182
derisi_90	14.401	14.659	14.954	14.954	12.634	7.477

(Continues in the next page)

Table F.1: Results in the datasets with the evaluation measure exact match of the SSHC-BLI variants (V1, V2, V3) and related methods, TD: Top-Down, STA: self-train A, HMC-SSBR: Hierarchical Multi-label Classification using Semi-Supervised Binary Relevance.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
eisen_10	19.176	18.965	17.961	18.806	15.689	6.339
eisen_30	21.659	22.081	20.285	21.712	15.214	9.667
eisen_50	23.719	28.473	26.36	25.885	17.908	16.27
eisen_70	28.209	29.847	29.477	29.477	18.014	17.433
eisen_90	28.473	30.005	30.375	30.375	19.915	16.852
gasch1_10	15.929	18.842	18.426	19.485	13.583	7.491
gasch1_30	18.313	22.777	21.302	20.621	15.778	9.118
gasch1_50	20.507	24.064	23.761	23.345	16.799	17.745
gasch1_70	23.042	23.004	24.518	24.518	16.875	18.35
gasch1_90	23.61	25.123	25.161	25.161	18.464	17.707
gasch2_10	14.393	16.911	16.084	16.272	13.867	8.643
gasch2_30	15.859	18.076	18.489	18.113	13.303	7.516
gasch2_50	16.46	18.452	17.926	17.738	16.234	7.967
gasch2_70	19.015	20.068	18.414	18.715	16.761	12.815
gasch2_90	18.527	19.805	19.654	19.654	15.483	15.145

(End exact match results)

Table F.2: Results in the datasets with the evaluation measure hF of the SSHC-BLI variants (V1, V2, V3) and related methods, TD: Top-Down, STA: self-train A, HMC-SSBR: Hierarchical Multi-label Classification using Semi-Supervised Binary Relevance.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
EA_01_FD_b_10	100	99.167	99.167	99.167	95.278	100
EA_01_FD_b_30	99.444	99.722	99.722	99.722	96.389	100
EA_01_FD_b_50	99.722	99.444	99.444	99.444	95.556	100
EA_01_FD_b_70	99.444	99.444	99.444	99.444	96.111	100
EA_01_FD_b_90	99.444	99.167	99.167	99.167	96.667	100
EA_01_FD_ub_10	100	100	100	100	96.731	100
EA_01_FD_ub_30	99.936	100	100	100	97.628	100
EA_01_FD_ub_50	99.936	100	100	100	98.077	100
EA_01_FD_ub_70	99.936	100	100	100	98.782	100
EA_01_FD_ub_90	99.936	100	100	100	98.718	100
EA_02_FD_b_10	98.333	98.333	98.889	98.889	80.278	100
EA_02_FD_b_30	97.222	98.333	98.056	98.056	83.611	100
EA_02_FD_b_50	97.5	98.889	98.889	98.889	87.5	100

(Continues in the next page)

Table F.2: Results in the datasets with the evaluation measure hF of the SSHC-BLI variants (V1, V2, V3) and related methods.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
EA_02_FD_b_70	97.5	98.056	98.056	98.056	88.889	100
EA_02_FD_b_90	97.222	98.056	98.056	98.056	90	100
EA_02_FD_ub_10	96.987	99.615	99.744	99.744	83.141	99.744
EA_02_FD_ub_30	98.718	99.487	99.551	99.551	88.718	99.808
EA_02_FD_ub_50	99.167	99.487	99.615	99.615	90.705	99.808
EA_02_FD_ub_70	99.038	99.615	99.679	99.679	91.282	99.808
EA_02_FD_ub_90	99.295	99.359	99.359	99.359	91.41	99.808
HA_01_FD_b_10	93.464	99.326	99.678	99.678	94.677	99.766
HA_01_FD_b_30	98.236	99.356	99.737	99.737	96.034	100
HA_01_FD_b_50	99.531	99.649	99.737	99.737	96.327	100
HA_01_FD_b_70	99.766	99.502	99.59	99.59	97.088	100
HA_01_FD_b_90	99.444	99.356	99.356	99.356	96.676	100
HA_01_FD_ub_10	98.438	99.785	99.811	99.811	95.338	99.962
HA_01_FD_ub_30	99.19	99.76	99.748	99.748	97.094	99.937
HA_01_FD_ub_50	99.38	99.684	99.811	99.811	97.823	99.962
HA_01_FD_ub_70	99.443	99.621	99.672	99.672	98.291	99.987
HA_01_FD_ub_90	99.57	99.608	99.608	99.608	98.391	100
HA_02_FD_b_10	45.58	53.147	48.322	46.809	40.275	56.1
HA_02_FD_b_30	48.316	52.056	52.361	51.378	44.382	55.589
HA_02_FD_b_50	48.925	51.705	52.607	52.971	45.432	54.981
HA_02_FD_b_70	49.965	50.902	51.013	51.013	46.638	55.498
HA_02_FD_b_90	49.956	51.391	51.117	51.117	46.12	54.829
HA_02_FD_ub_10	45.299	51.063	51.333	50.507	42.124	53.006
HA_02_FD_ub_30	47.506	52.246	51.482	50.994	43.767	53.467
HA_02_FD_ub_50	48.64	50.708	50.722	50.67	46.999	54.67
HA_02_FD_ub_70	48.869	50.558	50.321	51.275	45.331	55.525
HA_02_FD_ub_90	49.633	50.32	50.428	50.428	45.733	55.935
HA_03_FD_b_10	28.994	33.031	33.901	33.789	32.784	36.95
HA_03_FD_b_30	29.725	34.445	33.462	32.179	34.161	37.835
HA_03_FD_b_50	31.114	33.594	33.803	33.853	34.214	37.502
HA_03_FD_b_70	31.031	33.33	32.425	32.425	32.224	36.681
HA_03_FD_b_90	31.97	31.379	31.098	31.098	33.056	36.769
HA_03_FD_ub_10	29.06	33.051	34.098	33.227	32.496	37.496
HA_03_FD_ub_30	28.282	31.801	32.699	33.12	32.933	37.281
HA_03_FD_ub_50	30.566	31.868	31.44	30.836	33.013	37.694
HA_03_FD_ub_70	28.493	30.127	29.493	29.098	33.836	38.376
HA_03_FD_ub_90	28.525	29.825	29.823	29.823	32.789	37.283

(Continues in the next page)

Table F.2: Results in the datasets with the evaluation measure hF of the SSHC-BLI variants (V1, V2, V3) and related methods.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
HA_09_FD_b_10	62.797	66.918	66.911	66.874	64.056	71.405
HA_09_FD_b_30	62.238	65.43	65.612	65.685	63.97	72.993
HA_09_FD_b_50	62.365	64.765	65.148	65.054	63.861	73.643
HA_09_FD_b_70	62.548	63.439	63.637	63.642	64.181	74.256
HA_09_FD_b_90	61.665	62.289	62.313	62.313	64.529	74.631
HA_09_FD_ub_10	63.239	66.11	66.138	66.154	64.755	71.19
HA_09_FD_ub_30	62.823	65.465	65.407	65.321	64.631	72.731
HA_09_FD_ub_50	63.158	64.881	64.908	65.037	64.941	73.462
HA_09_FD_ub_70	63.078	64.276	64.231	64.201	64.743	73.884
HA_09_FD_ub_90	63.61	64.221	64.242	64.242	64.878	74.306
HA_10_FD_b_10	69.744	75.021	74.756	74.893	70.101	74.969
HA_10_FD_b_30	72.467	75.212	75.261	75.111	70.512	75.875
HA_10_FD_b_50	72.675	74.877	74.868	74.69	71.976	76.776
HA_10_FD_b_70	73.312	74.436	74.589	74.589	72.243	77.187
HA_10_FD_b_90	73.554	73.87	73.948	73.948	72.501	77.919
HA_10_FD_ub_10	71.908	76.066	75.846	75.692	71.017	75.284
HA_10_FD_ub_30	73.509	76.255	76.368	75.962	72.516	76.106
HA_10_FD_ub_50	74.211	75.949	75.867	75.826	72.784	76.504
HA_10_FD_ub_70	74.403	75.43	75.392	75.223	73.15	76.86
HA_10_FD_ub_90	74.987	75.301	75.246	75.246	73.484	77.411
VH_01_FD_b_10	24.915	39.894	41.301	40.764	35.502	44.384
VH_01_FD_b_30	29.126	33.775	34.7	34.51	31.065	44.061
VH_01_FD_b_50	24.233	29.035	28.287	29.279	27.835	44.006
VH_01_FD_b_70	20.042	24.33	23.539	23.539	23.074	44.153
VH_01_FD_b_90	18.542	19.256	19.354	19.354	19.124	41.314
VH_01_FD_ub_10	31.972	43.498	44.178	44.63	36.09	49.212
VH_01_FD_ub_30	29.764	39.133	38.912	38.964	34.139	50.338
VH_01_FD_ub_50	25.941	32.477	32.208	32.208	30.426	50.921
VH_01_FD_ub_70	23.018	26.86	27.133	27.133	28.941	52.213
VH_01_FD_ub_90	19.11	20.595	20.251	20.251	23.435	51.853
VH_02_FD_b_10	35.501	36.776	34.192	20.32	38.548	49.408
VH_02_FD_b_30	49.672	53.13	50.13	49.517	43.556	55.147
VH_02_FD_b_50	61.03	57.39	58.258	58.064	48.791	55.073
VH_02_FD_b_70	62.59	63.86	61.704	62.844	46.276	58.794
VH_02_FD_b_90	67.123	67.682	70.234	70.234	49.95	62.662
VH_02_FD_ub_10	58.751	61.875	61.782	35.002	53.656	60.994
VH_02_FD_ub_30	70.226	71.397	70.393	63.552	54.34	63.669

(Continues in the next page)

Table F.2: Results in the datasets with the evaluation measure hF of the SSHC-BLI variants (V1, V2, V3) and related methods.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
VH_02_FD_ub_50	74.673	75.974	74.937	72.893	57.937	67.874
VH_02_FD_ub_70	77.507	79.171	77.791	78.818	59.92	71.647
VH_02_FD_ub_90	80.34	81.533	81.597	81.597	60.438	73.642
VH_03_FD_b_10	28.81	38.81	36.19	17.619	23.175	35.496
VH_03_FD_b_30	46.667	48.095	47.143	40.476	28.81	39.568
VH_03_FD_b_50	50.952	54.762	53.492	53.492	34.524	40.861
VH_03_FD_b_70	57.143	60.952	60.317	60.317	37.778	43.05
VH_03_FD_b_90	62.222	63.492	63.651	63.651	39.683	42.578
VH_03_FD_ub_10	34.288	42.059	34.015	14.69	29.039	39.618
VH_03_FD_ub_30	55.283	59.646	58.078	50.784	33.776	42.999
VH_03_FD_ub_50	60.191	61.486	61.213	61.213	39.945	42.986
VH_03_FD_ub_70	64.213	66.667	66.599	66.258	41.616	43.844
VH_03_FD_ub_90	68.439	69.257	69.53	69.53	41.718	45.608
VH_08_FD_b_10	37.025	46.596	47.123	46.416	38.904	48.723
VH_08_FD_b_30	39.792	47.062	47.465	46.414	40.925	49.378
VH_08_FD_b_50	40.83	45.852	46.228	46.274	41.183	50.364
VH_08_FD_b_70	41.991	45.029	44.819	44.819	41.515	51.467
VH_08_FD_b_90	42.271	43.208	43.184	43.184	42.343	52.239
VH_08_FD_ub_10	41.393	49.169	49.612	49.193	43.388	51.093
VH_08_FD_ub_30	45.089	50.363	50.566	50.384	44.532	52.504
VH_08_FD_ub_50	45.949	49.9	50.177	50.116	45.454	53.2
VH_08_FD_ub_70	47.046	48.887	48.868	49.045	45.565	53.947
VH_08_FD_ub_90	47.605	48.4	48.397	48.397	45.74	54.53
o20NG_50t_10	71.409	71.049	69.961	63.048	55.568	73.974
o20NG_50t_30	73.626	74.187	74.129	73.458	58.923	75.729
o20NG_50t_50	74.231	74.818	74.634	74.347	60.73	76.401
o20NG_50t_70	74.868	75.278	75.065	74.973	61.732	76.91
o20NG_50t_90	75.134	75.36	75.41	75.41	62.74	77.167
cellcycle_10	17.845	28.508	29.176	30.103	21.834	31.461
cellcycle_30	20.819	28.34	28.715	29.253	24.672	33.181
cellcycle_50	24.454	27.973	28.296	28.789	24.555	32.314
cellcycle_70	25.172	25.879	25.957	25.957	25.662	31.863
cellcycle_90	26.456	27.295	27.661	27.661	25.799	32.139
derisi_10	15.497	23.431	23.694	24.316	21.989	27.542
derisi_30	18.621	23.082	23.707	23.827	24.116	24.414
derisi_50	19.519	22.383	21.925	22.11	23.5	23.764
derisi_70	19.727	20.729	20.444	20.312	25.038	22.429

(Continues in the next page)

Table F.2: Results in the datasets with the evaluation measure hF of the SSHC-BLI variants (V1, V2, V3) and related methods.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
derisi_90	19.351	19.977	19.915	19.915	24.006	21.96
eisen_10	25.224	32.352	31.194	34.024	29.428	36.401
eisen_30	28.255	31.898	31.456	31.089	28.372	35.745
eisen_50	30.981	36.537	35.089	35.428	31.04	35.722
eisen_70	36.438	37.912	37.322	37.322	30.516	36.935
eisen_90	35.913	37.877	37.967	37.967	32.69	35.914
gasch1_10	21.638	30.907	31.445	32.539	24.692	35.298
gasch1_30	24.264	32.392	31.611	30.94	27.458	36.207
gasch1_50	26.279	31.826	31.931	31.668	28.535	36.428
gasch1_70	28.9	29.746	31.033	31.033	27.969	36.804
gasch1_90	29.755	31.04	31.172	31.172	29.442	35.957
gasch2_10	21.023	29.807	29.897	30.368	25.381	34.019
gasch2_30	21.998	28.529	29.931	29.33	25.113	34.605
gasch2_50	22.288	26.372	26.046	27.181	27.543	33.16
gasch2_70	24.579	25.647	24.313	24.68	27.788	31.468
gasch2_90	23.659	25.305	25.142	25.142	27.548	32.012

(End hF results)

Table F.3: Results in the datasets with the evaluation measure MCC of the SSHC-BLI variants (V1, V2, V3) and related methods, TD: Top-Down, STA: self-train A, HMC-SSBR: Hierarchical Multi-label Classification using Semi-Supervised Binary Relevance.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
EA_01_FD_b_10	100	100	100	100	99.111	100
EA_01_FD_b_30	100	100	100	100	100	100
EA_01_FD_b_50	100	100	100	100	99.181	100
EA_01_FD_b_70	100	100	100	100	99.311	100
EA_01_FD_b_90	100	100	100	100	100	100
EA_01_FD_ub_10	100	100	100	100	95.368	100
EA_01_FD_ub_30	100	100	100	100	96.062	100
EA_01_FD_ub_50	100	100	100	100	96.06	100
EA_01_FD_ub_70	100	100	100	100	97.129	100
EA_01_FD_ub_90	100	100	100	100	96.93	100
EA_02_FD_b_10	98.676	100	100	100	75.307	100
EA_02_FD_b_30	99.034	100	99.556	99.556	83.805	100
EA_02_FD_b_50	99.556	100	100	100	86.593	100

(Continues in the next page)

Table F.3: Results in the datasets with the evaluation measure MCC of the SSHC-BLI variants (V1, V2, V3) and related methods.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
EA_02_FD_b_70	100	99.556	99.556	99.556	88.223	100
EA_02_FD_b_90	99.111	99.556	99.556	99.556	91.044	100
EA_02_FD_ub_10	96.555	100	100	100	71.294	100
EA_02_FD_ub_30	98.018	100	100	100	79.989	100
EA_02_FD_ub_50	99.475	100	100	100	83.499	100
EA_02_FD_ub_70	99.506	100	100	100	83.645	100
EA_02_FD_ub_90	100	100	100	100	85.804	100
HA_01_FD_b_10	97.679	100	100	100	94.401	100
HA_01_FD_b_30	99.844	100	100	100	95.245	100
HA_01_FD_b_50	100	100	100	100	95.808	100
HA_01_FD_b_70	100	100	100	100	97.403	100
HA_01_FD_b_90	100	100	100	100	96.306	100
HA_01_FD_ub_10	98.797	100	100	100	89.199	100
HA_01_FD_ub_30	99.333	99.966	100	100	94.027	100
HA_01_FD_ub_50	99.707	99.83	100	100	96.038	100
HA_01_FD_ub_70	99.865	99.83	99.865	99.865	97.757	100
HA_01_FD_ub_90	99.854	99.865	99.865	99.865	98.051	100
HA_02_FD_b_10	51.762	43.052	38.913	36.086	41.78	27.714
HA_02_FD_b_30	43.625	40.434	38.024	38.686	39.047	27.816
HA_02_FD_b_50	46.299	39.765	44.263	44.658	41.432	22.586
HA_02_FD_b_70	48.735	43.362	46.32	46.32	41.142	27.824
HA_02_FD_b_90	48.096	47.401	36.907	36.907	31.868	30.473
HA_02_FD_ub_10	31.133	32.514	35.777	29.944	30.338	19.621
HA_02_FD_ub_30	27.579	31.431	33.481	29.565	24.596	19.658
HA_02_FD_ub_50	25.647	30.712	27.343	31.006	27.711	26.276
HA_02_FD_ub_70	31.524	31.048	30.238	28.482	29.239	27.594
HA_02_FD_ub_90	32.272	30.18	31.626	31.626	26.473	33.259
HA_03_FD_b_10	35.003	30.182	24.92	32.276	31.418	35.3
HA_03_FD_b_30	35.395	31.912	27.097	32.164	27.352	32.837
HA_03_FD_b_50	33.875	33.733	27.669	31.711	26.99	26.8
HA_03_FD_b_70	33.814	39.702	29.873	29.873	31.374	17.399
HA_03_FD_b_90	30.766	27.13	35.36	35.36	37.131	20.472
HA_03_FD_ub_10	23.359	25.316	18.855	22.49	18.618	10.767
HA_03_FD_ub_30	24.433	20.428	17.521	25.188	19.298	7.805
HA_03_FD_ub_50	28.761	21.722	23.134	27.855	18.303	12.629
HA_03_FD_ub_70	30.906	23.265	31.813	33.132	20.444	9.813
HA_03_FD_ub_90	28.604	28.41	26.677	26.677	23.423	8.322

(Continues in the next page)

Table F.3: Results in the datasets with the evaluation measure MCC of the SSHC-BLI variants (V1, V2, V3) and related methods.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
HA_09_FD_b_10	48.638	45.67	41.561	45.579	44.374	5.099
HA_09_FD_b_30	43.051	45.344	42.717	43.18	41.58	6.187
HA_09_FD_b_50	43.99	43.337	42.452	41.139	44.987	4.583
HA_09_FD_b_70	42.658	46.46	44.459	43.426	41.686	6.178
HA_09_FD_b_90	45.441	41.312	43.261	43.261	36.11	6.267
HA_09_FD_ub_10	28.135	29.395	29.948	29.606	24.612	4.383
HA_09_FD_ub_30	28.778	26.012	25.701	27.505	25.798	5.38
HA_09_FD_ub_50	27.636	24.526	26.582	24.294	28.78	5.479
HA_09_FD_ub_70	29.267	26.523	26.934	25.729	25.72	6.21
HA_09_FD_ub_90	30.224	29.492	28.538	28.538	25.876	5.875
HA_10_FD_b_10	39.187	38.505	41.392	38.755	40.064	22.59
HA_10_FD_b_30	36.829	33.507	35.431	37.689	37.001	18.022
HA_10_FD_b_50	35.514	32.819	33.161	33.963	35.427	16.464
HA_10_FD_b_70	35.827	33.507	32.487	32.487	34.839	18.774
HA_10_FD_b_90	33.513	33.279	32.651	32.651	34.479	19.923
HA_10_FD_ub_10	30.04	27.787	30.592	30.463	29.974	26.741
HA_10_FD_ub_30	27.803	27.991	27.705	26.824	30.497	30.774
HA_10_FD_ub_50	27.47	26.624	26.901	28.084	29.957	30.883
HA_10_FD_ub_70	28.39	26.902	26.882	27.387	30.066	31.403
HA_10_FD_ub_90	27.84	28.828	28.093	28.093	28.537	30.855
VH_01_FD_b_10	30.892	19.567	24.196	26.452	31.289	2.913
VH_01_FD_b_30	21.912	17.314	24.162	26.206	32.014	-0.183
VH_01_FD_b_50	32.117	21.209	23.836	26.826	33.383	0.284
VH_01_FD_b_70	36.878	33.641	34.278	34.278	38.913	-0.348
VH_01_FD_b_90	35.605	40.92	34.064	34.064	48.427	-2.748
VH_01_FD_ub_10	16.778	14.279	22.141	20.503	17.697	-0.382
VH_01_FD_ub_30	10.406	18.828	20.401	20.984	19.252	1.19
VH_01_FD_ub_50	21.985	17.164	16.511	16.511	21.025	0.864
VH_01_FD_ub_70	35.311	21.74	16.436	16.436	30.335	3.549
VH_01_FD_ub_90	40.329	37.913	42.313	42.313	27.598	0.277
VH_02_FD_b_10	31.317	37.824	26.024	47.556	32.12	45.995
VH_02_FD_b_30	37.9	45.147	40.164	36.982	26.802	51.441
VH_02_FD_b_50	49.636	53.641	46.358	45.942	34.448	46.489
VH_02_FD_b_70	53.292	57.75	51.562	54.935	32.172	49.336
VH_02_FD_b_90	60.973	59.125	64.081	64.081	42.916	55.136
VH_02_FD_ub_10	40.464	45.481	42.797	51.953	30.475	35.468
VH_02_FD_ub_30	49.737	52.823	51.109	47.214	32.88	46.589

(Continues in the next page)

Table F.3: Results in the datasets with the evaluation measure MCC of the SSHC-BLI variants (V1, V2, V3) and related methods.

Dataset	TD	V1	V2	V3	STA	HMC-SSBR
VH_02_FD_ub_50	56.467	58.849	59.283	56.775	35.584	54.428
VH_02_FD_ub_70	59.513	64.48	60.644	63.449	38.455	55.77
VH_02_FD_ub_90	64.497	68.101	67.853	67.853	37.665	59.189
VH_03_FD_b_10	33.043	38.642	45.756	38.208	46.891	45.206
VH_03_FD_b_30	47.099	50.171	49.684	44.547	31.621	41.358
VH_03_FD_b_50	47.75	52.843	52.618	52.733	35.135	42.35
VH_03_FD_b_70	55.156	58.679	58.027	58.027	37.648	39.979
VH_03_FD_b_90	59.838	60.997	61.047	61.047	41.77	42.775
VH_03_FD_ub_10	31.142	43.92	39.711	45.835	31.411	38.503
VH_03_FD_ub_30	51.833	56.882	54.813	53.714	33.082	43.687
VH_03_FD_ub_50	56.429	57.39	57.526	59.474	36.963	44.339
VH_03_FD_ub_70	60.407	63.229	63.547	64.038	36.855	45.416
VH_03_FD_ub_90	66.64	66.945	66.955	66.955	39.353	46.73
VH_08_FD_b_10	32.661	38.35	35.157	36.319	37.941	12.857
VH_08_FD_b_30	30.705	28.091	30.827	34.793	32.601	12.118
VH_08_FD_b_50	31.955	27.864	30.077	31.486	25.14	16.337
VH_08_FD_b_70	30.477	30.771	31.948	31.948	25.094	17.245
VH_08_FD_b_90	29.322	27.375	28.957	28.957	28.962	22.249
VH_08_FD_ub_10	27.246	22.268	21.414	22.869	21.218	15.35
VH_08_FD_ub_30	24.456	27.139	24.892	23.118	21.967	15.255
VH_08_FD_ub_50	26.014	25.395	26.751	25.052	21.254	13.219
VH_08_FD_ub_70	25.047	26.134	26.201	25.921	21.676	10.322
VH_08_FD_ub_90	25.721	26.405	24.922	24.922	24.629	18.316
o20NG_50t_10	62.788	62.568	61.492	55.629	46.562	66.827
o20NG_50t_30	65.55	66.01	65.974	65.321	50.294	67.936
o20NG_50t_50	66.315	66.93	66.673	66.347	51.975	68.749
o20NG_50t_70	67.034	67.581	67.263	67.182	52.893	69.555
o20NG_50t_90	67.434	67.688	67.722	67.722	54.359	70.022
cellcycle_10	16.896	31.258	35.983	37.875	18.299	25.053
cellcycle_30	22.184	24.52	19.603	31.654	24.278	11.22
cellcycle_50	18.326	18.528	19.75	16.466	20.339	25.265
cellcycle_70	19.145	19.352	21.924	21.924	16.615	36.575
cellcycle_90	19.973	25.214	18.63	18.63	15.956	37.146
derisi_10	12.757	23.094	32.912	24.303	15.125	2.618
derisi_30	14.608	9.343	16.4	15.808	21.006	32.109
derisi_50	18.536	16.149	10.418	11.923	15.82	33.588
derisi_70	16.411	15.672	17.071	16.908	13.894	27.946

(Continues in the next page)

Table F.3: Results in the datasets with the evaluation measure MCC of the SSHC-BLI variants (V1, V2, V3) and related methods.

Dataset	**TD**	**V1**	**V2**	**V3**	**STA**	**HMC-SSBR**
derisi_90	16.114	14.326	17.074	17.074	12.465	16.248
eisen_10	19.949	24.826	32.204	43.079	30.691	11.155
eisen_30	24.536	23.955	17.243	28.123	16.035	25.759
eisen_50	21.091	25.453	23.08	24.412	13.523	25.898
eisen_70	25.235	26.482	25.833	25.833	22.618	27.779
eisen_90	24.854	28.182	27.203	27.203	17.718	32.67
gasch1_10	19.84	35.904	33.046	26.976	24.206	12.286
gasch1_30	17.107	22.155	17.657	30.049	21.968	12.739
gasch1_50	19.042	20.632	19.929	19.398	28.424	29.76
gasch1_70	19.81	20.051	21.98	21.98	21.827	24.445
gasch1_90	21.141	22.21	22.405	22.405	18.548	23.008
gasch2_10	18.228	35.997	28.477	32.19	30.619	10.592
gasch2_30	12.695	18.189	26.349	26.867	19.95	16.045
gasch2_50	17.906	15.904	13.211	13.251	16.699	16.88
gasch2_70	22.689	21.49	22.282	22.624	21.849	21.501
gasch2_90	23.315	23.463	17.166	17.166	21.485	28.037

(End MCC results)