# Hierarchical Classification

Tommaso Lombardi

February 2022

# Chapter 1

# Introduction

A very large amount of research in the datamining, machine learning, statistical pattern recognition and related research communities has focused on flat classification problems. By flat classification problem we are referring to standard binary or multi-class classification problems. On the other hand, many important real-world classification problems are naturally shaped as hierarchical classification problems, where the classes to be predicted are organized into a class hierarchy, typically a tree or a DAG(Direct Acyclic Graph). Text classification or text categorization, topic of this thesis work, is the process of automatically assigning one or more predefined categories to text documents. A wide range of supervised learning algorithms has been applied to this problem, using a training set of categorized documents to build a classifier that maps arbitrary documents to relevant categories. Most of the learning methods reported in the literature deal with classifying text into a set of categories without structural relationships among them (flat classification). The task of hierarchical classification, however, needs to be better defined, as it can be overlooked or confused with other tasks, which are often wrongly referred to by the same name.

## 1.1 What is hierarchical classification

Hierarchical classification can be seen as a particular type of structured classification problem, where the output of the classification algorithm is defined over a class taxonomy; whilst the term structured classification is broader and denotes a classification problem where there is some structure (hierarchical or not) among the classes. It is important then to define what exactly is a class taxonomy.

**Definition 1** (**Class taxonomy**)**.** A class taxonomy is a tree structured regular concept hierarchy defined over a partially ordered set $(C, \prec)$, where $C$ is a finite set that enumerates all class concepts in the application domain, and the relation $\prec$ represents the "IS-A" relationship and has the following properties: asymmetric, anti-reflexive and transitive.

- $\forall c_i, c_i \in C$, if $c_i \prec c_j$ then $c_j \not\prec c_i$
- $\forall c_i \in C, c_i \not\prec c_i$
- $\forall c_i, c_j, c_k \in C, c_i \prec c_j$ and $c_j \prec c_k$ imply $c_i \prec c_k$

A simple explanatory example of the asymmetric property can be the statement *all dogs are animals, but not all animals are dogs* while for transitive relations *all pines are evergreens, and all evergreens are trees; therefore all pines are trees.* Any classification problem with a class structure satisfying the aforementioned properties of the IS-A hierarchy can be considered as a hierarchical classification problem. This definition, although originally proposed for tree structured class taxonomies, can be used to define DAG structured class taxonomies as well, but, as it will be shown later, this study was done using only tree stuctured datasets.

## 1.2 Overview on hierarchical classification

According to [1] hierarchical classification methods differ in a number of criteria. The first criterion is the type of hierarchical structure used. This structure is based on the problem structure and it typically is either a tree or a DAG. The second criterion is related to how deep the classification in the hierarchy is performed. The hierarchical classification method can be implemented in a way that will always classify a leaf node, referred to as mandatory leaf-node prediction (MLNP) or the method can consider stopping the classification at any node in any level of the hierarchy called non-mandatory leaf node prediction (NMLNP). The third criterion is related to how the hierarchical structure is explored. The current literature often refers to top-down (or local) classifiers, when the system employs a set of local classifiers; big-bang (or global) classifiers, when a single classifier coping with the entire class hierarchy is used; or flat classifiers, which ignore the class relationships.

### 1.2.1 Flat classification approach

The flat classification approach, which is the simplest one to deal with hierarchical classification problems, consists of completely ignoring the class hierarchy. As shown in the simple example in the figure 1.1 and 1.2 the structure is flatten out and this approach behaves like a traditional classification algorithm during training and testing. However, it provides an indirect solution to the problem of hierarchical classification. When a prediction on an example is performed could happen that a certain class is predicted while its ancestor don't. This approach has the serious disadvantage of having to build a classifier to discriminate among a large number of classes, without exploring information about parent-child class relationships embedded in the class hierarchy.

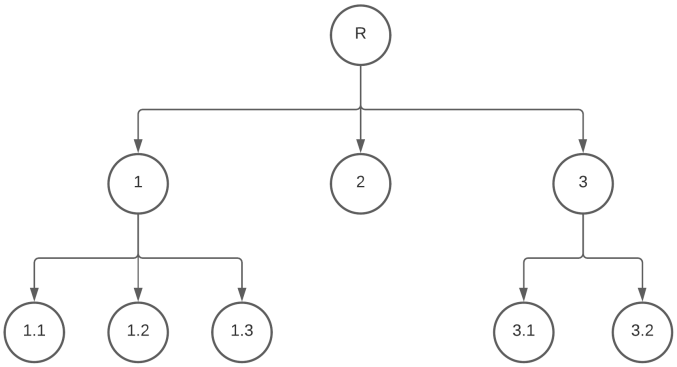



Figure 1.1: Tree hierarchy

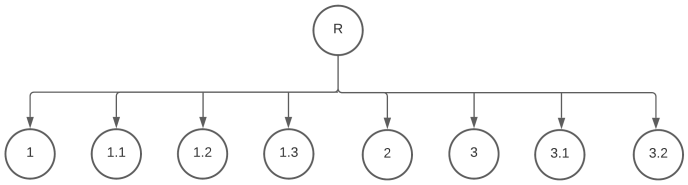



Figure 1.2: Tree hierarchy flatten out

### 1.2.2 Local classifier approach

Local classifier approach start to exploit the information that hierarchy provides. These approaches, therefore, can be grouped based on how they use this local information and how they build their classifiers around it. More precisely, there seems to exist three standard ways of using the local information: a local classifier per node (LCN), a local classifier per parent node (LCPN) and a local classifier per level (LCL). It should be noted that, although the three types of local hierarchical classification algorithms discussed in the next three sub-sections differ significantly in their training phase, they share a very similar top-down approach in their testing phase. As a result, a disadvantage of the top-down class-prediction approach (which is shared by all the three types of local classifiers discussed next) is that an error at a certain class level is going to be propagated downwards the hierarchy. Looking at the hierarchy in the figure 1.1:

- LCN need a classifier in node 1, 2, 3 ,1.1 ,1.2 1.3, 3.1, 3.2
- LCPN in node R, 1, 3
- LCL need a classifier in the first layer predicting among class 1, 2 ,3 and a classifier in the second layer predicting among class 1.1, 1.2, 1.3, 3.1, 3.2

### 1.2.3 Global classifier approach

Learning a single global model for all classes has the advantage that the total size of the global classification model is typically considerably smaller, in comparison with the total size of all the local models learned by any of the local classifier approaches. In addition, dependencies between different classes with respect to class membership can be taken into account in a natural, straightforward way. This kind of approach is known as the big-bang approach, also called "global" learning. In the global classifier approach, a single classification model is built taking into account the class hierarchy as a whole. Can be considered global classifiers even those classifiers (flat) provided with a strategy for avoiding class-prediction inconsistencies across class relationships. As pointed out in the section 1.2.1, when using a flat approach and a prediction on an example is performed, what could happen is that a certain class is predicted while its ancestors don't. Whether the hierarchical consistency requirement is mandatory is up to the specification of the problem.

**Definition 2** (**Hierarchical consistency**)**.** A label set $C_i \subseteq C$ assigned to an instance $d_i \in D$ is called consistent with a given hierarchy if $C_i$ includes complete ancestor sets for every label $c_k \in C_i$, i.e. if $c_k \in C_i$ and $c_j \in Ancestors(c_k)$, then $c_j \in C_i$.

**Definition 3** (**Hierarchical Consistency Requirement**)**.** Any label assignments produced by a hierarchical classification system on a given hierar-

chical categorization task should be consistent with the corresponding class hierarchy.

In case the hierarchical consistency requirement must be satisfied there is an exploitation of the hierarchy structure, exiting from the definition of merely flat scenario.

## 1.3 Hirarchical training strategy

In order to train a hierarchical classification model in a supervised manner there are different techniques in the literature to retrieve positive and negative examples for each class. The notation that will be used is shown in the following table.

| Symbols and their meaning | |
|---|---|
| Symbol | Meaning |
| $T_r$ | set of training examples |
| $T_r^+(c_j)$ | set of positive training examples of class $c_j$ |
| $T_r^-(c_j)$ | set of negative training examples of $c_j$ |
| $\uparrow(c_j)$ | set of parent nodes of $c_j$ |
| $\downarrow(c_j)$ | set of children of $c_j$ |
| $\Uparrow(c_j)$ | set of anchestors of $c_j$ |
| $\Downarrow(c_j)$ | set of descendants of $c_j$ |
| $\leftrightarrow(c_j)$ | set of siblings of $c_j$ |
| $*(c_j)$ | set of examples whose most specific known class is $c_j$ |

- The "exclusive" policy [as defined by Eisner et al. (2005)]:
  - $T_r^+(c_j) = *(c_j)$
  - $T_r^-(c_j) = T_r \setminus *(c_j)$

This means that only examples explicitly labeled as $c_j$ as their most specific class are selected as positive examples, while everything else is used as negative examples. This approach has a few problems. First, it does not consider the hierarchy to create the local training sets. Second, it is limited to problems where for every class $c_j$ there is at least one sample whose most specific known class is $c_j$. Third, using the descendant nodes of $c_j$ as negative examples seems counter-intuitive considering that examples who belong to class $\Downarrow (c_j)$ also implicitly belong to class $c_j$ according to the "IS-A" hierarchy concept.

- The "less exclusive" policy [7]:
  - $T_r^+(c_j) = *(c_j)$
  - $T_r^-(c_j) = T_r \setminus *(c_j) \cup \Downarrow (c_j)$

  This approach avoids the first and third problems mentioned for the "exclusive" policy, but it is still limited to those problems where for every class $c_j$ there is at least one sample whose most specific known class is $c_j$

- The "less inclusive" policy [7]:
  - $T_r^+(c_j) = *(c_j) \cup \Downarrow (c_j)$
  - $T_r^-(c_j) = T_r \setminus *(c_j) \cup \Downarrow (c_j)$

  This policy, along with the "less exclusive", suffer from the problem of imbalance between the $T_r+$ and the $T_r^-$. In most cases, considering class $c_j$ $|T_r^-(c_j)| >> |T_r^+(c_j)|$, where $|\cdot|$ is the cardinality.

- The "inclusive" policy [7]:
  - $T_r^+(c_j) = *(c_j) \cup \Downarrow (c_j)$
  - $T_r^-(c_j) = T_r \setminus *(c_j) \cup \Downarrow (c_j) \cup \Uparrow (c_j)$

- The "siblings" policy [6]:
  - $T_r^+(c_j) = *(c_j) \cup \Downarrow (c_j)$
  - $T_r^-(c_j) = \leftrightarrow (c_j) \cup \Downarrow (\leftrightarrow (c_j))$
- The "exclusive siblings" policy [5]:
  - $T_r^+(c_j) = *(c_j)$
  - $T_r^-(c_j) = \leftrightarrow (c_j)$

## 1.4 A unifying framework for hierarchical classification

The paper [2] tried to define an unified framework for describing all the hierarchical classification tasks.
A hierarchical classification problem is described as a 3-tuple $< \Upsilon, \Psi, \Phi >$ where :

| | |
|---|---|
| $\Upsilon \in [T, DAG]$ | Describes the taxonomy's structure (Tree,DAG) |
| $\Psi \in [SPL, MPL]$ | Data instances have multiple path or single path labeling (single or multilabel) |
| $\Phi \in [FD, PD]$ | Describes the label depth of data instances (full and partial depth) |

A hierarchical classification algorithm is described as a 4-tuple $< \Delta, \Xi, \Omega, \Theta >$where :

| | |
|---|---|
| $\Delta \in [SPP, MPP]$ | Indicates whether or not it can predict labels in just one or multiple different path (single and multiple path prediction) |
| $\Xi \in [MLNP, NMLNP]$ | Prediction depth ( mandatory and non-mandatory leaf node prediction) |
| $\Omega \in [T, DAG]$ | Taxonomy structure the algorithm can handle (Tree, DAG) |
| $\Theta \in [L, G, F]$ | Categorization of the algorithm (local, global, flat) |

For example an algorithm $< SPP, NMLNP, T, L >$, is an algorithm that can handle a tree structured taxonomy, gives a prediction that could be any node ( non mandatory leaf node), can predict labels in just one path (implicit in this case given the tree shaped hierarchy) and it is a local classifier (LCN or LCPN or LCL).

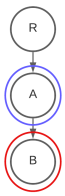
This notation will be used to summarize the models used in this work.

# Chapter 2

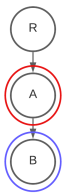
# Hierarchical evaluation metrics

## 2.1 Hierarchical evaluation metrics

The most commonly used measures for evaluating flat classification models such as precision, recall, F-measure, accuracy etc. are not appropriate for hierarchical classification, due to the fact that they do not take account of the relationship that exist among the classes. A hierarchical performance measure should use the class hierarchy in order to evaluate properly HC algorithms. For example, consider the tree hierarchy in 1.1. Assume that the true class for a test sample is 1.2 and that two different classification systems output 3 and 1.1 as the predicted classes. Using flat evaluation measures, both systems are punished equally, but the error of the first system is more severe as it makes a prediction in a different and unrelated sub-tree. Hierarchical classification open some new issues that need to be addressed. Starting with *specialization* error where the predicted class is a descendant of the true class, and *generalization* error, where an ancestor of the true class is selected. In both these cases the desired behavior of the measure would be to reduce the penalty of the error according to the distance between the true class and the predicted one and the relative level of the hierarchy where the error occurred.




(a) Specialization




(b) Generalization

Figure 2.1: Examples of specialization and generalization errors

Another common scenario in hierarchical multi-label classification is the one reported in figure 2.2 . Semantic or distance based metrics in the literature require to build pairs in order to evaluate the performance. In those cases is usual to deal with this situation by deciding, before even measuring the error, which pairs of true and predicted classes should be compared. For example, node 2 (true class) could be compared to 5 (predicted) and 7 to 1; or node 2 could be compared to both 1 and 5, and node 7 to none; other pairings are also possible. Depending on the pairings, the score assigned to the classifier will be different. Metrics that deals with this problem usually use the pairings that minimize the classification error.

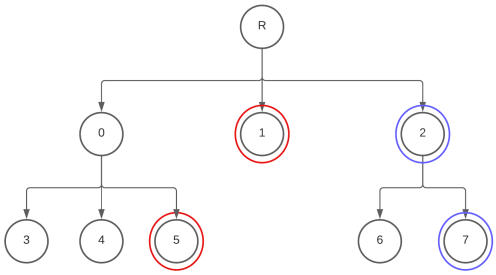



Figure 2.2: Pairing problem

Among all the hierarchical metrics, the most used are the metrics proposed by [3]: a hierarchical version of the classic precision, recall and f1 measure (hP, hR, hF1).

$$hP = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{P}_i|} \tag{2.1}$$

$$hR = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{T}_i|} \tag{2.2}$$

$$hF1 = \frac{2 * hP * hR}{hP + hR} \tag{2.3}$$

where:

$i \in [0, \ldots, m]$ is the sample index.

$P_i$ is the predicted set for the example $i$

$\hat{P}_i$ is the augmented predicted set. Is the set consisting of the most specific class(es) predicted for test example $i$ and all its(their) ancestor classes

$T_i$ is the target set for the example $i$

$\hat{T}_i$ is the augmented target set. Is the set consisting of the true most specific class(es) of test example $i$ and all its(their) ancestor classes

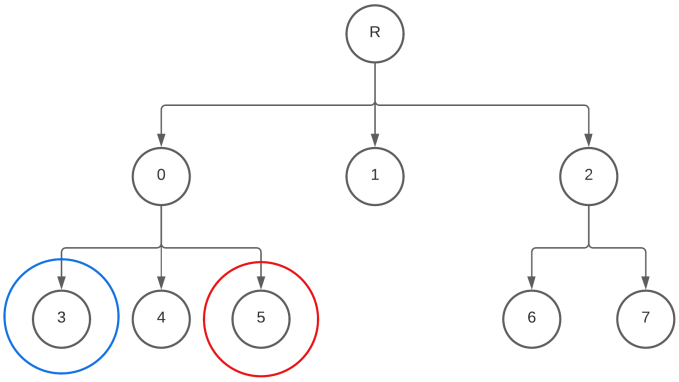



Figure 2.3: target in blue, prediction in red

The figure 2.3 represent a possible outcome of a prediction on a test example over that taxonomy, and in this case the sets involved for the calculation of the hP, hR, hF1 are $P = (5)$ $\hat{P} = (0, 5)$ $T = (3)$ $\hat{T} = (0, 3)$. Given the nature of the "IS-A" relationship between classes, the predicted

set and the target set should always be represented by the augmented version of themselves: if the target, as well as the prediction, is *pine* is also *tree*. This statement comes from the fact that a hierarchical prediction, and so a hierarchical target, should always satisfy the hierarchical consistency requirement (definition 3). So for now on, given an example $i$ the augmented target set $\hat{T}_i$ and the augmented predicted set $\hat{P}_i$ will be called respectively *hierarchical target set* and *hierarchical predicted set*. Once both the targets and the predictions are in the hierarchical form the problem with those metrics is that they yeld the same result as the conventional micro-averaged metrics :

$$micro\_avg\_P = \frac{\sum_j TP_j}{\sum_j TP_j + FP_j} \tag{2.4}$$

$$micro\_avg\_R = \frac{\sum_j TP_j}{\sum_j TP_j + FN_j} \tag{2.5}$$

$$micro\_avg\_f1 = \frac{2 * micro\_avg\_P * micro\_avg}{micro\_avg\_P + micro\_avg\_R} \tag{2.6}$$

where:

$j \in [0, \ldots, n]$ is the class/node index.

TP, FP, FN are the well known true positive, false positive, false negative

To demonstrate this statement let $i$ be the index of a generic sample. Supposing that the output of the hierarchical classifier to that sample is the one showed in figure 2.3, as we said before, $\hat{P}_i = (0, 5)$ $\hat{T}_i = (0, 3)$ yielding for node 0 $TP = 1$ $FP = 0$, for node 3 $TP = 0$ $FP = 0$, for node 4 $TP = 0$ $FP = 1$, hence:

$$|\hat{P}_i \cap \hat{T}_i| = \sum_j TP_j(i) \tag{2.7}$$

$$|\hat{P}_i| = \sum_j (TP_j(i) + FP_j(i)) \tag{2.8}$$

$$j \in [0, \ldots, n] \text{is the class/node index}$$

summing over all the example:

$$\frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{P}_i|} = \frac{\sum_j TP_j}{\sum_j TP_j + FP_j} \tag{2.9}$$

The same reasoning can be applied on the other metrics.

Even if the HP, HR and HF1 present some problems in the paper by [3] there is listed a series of desired properties that a hierarchical evaluation measures should satisfy :

- Give credit to partially correct classification
- Given a definition of distance, punish distant errors more heavily
- Punish errors at higher levels of the hierarchy more heavily

# Chapter 3

# Imbalanceness

## 3.1 Imbalanced datasets

Nowadays, a lot of researchers face imbalanced class distribution issues, mostly when working with real world datasets. Those problems have often skewed class distributions and, due to this, they present training datasets where several classes are represented by an extremely large number of examples, while some others are represented by only a few. This particular situation is known as the class-imbalance problem, a.k.a. learning from unbalanced data, and it is considered in the literature as a major obstacle to building precise classifiers. Considering that in general, the learning classifiers are usually focused in the minimization of the global error rate, the solutions obtained for problems showing class-imbalance through the traditional learning techniques are usually biased towards the most probable classes showing a poor prediction power for the least probable classes.

## 3.2 Imbalance Ratio

In the literature the imbalance-ratio is the most frequently used summary of the class imbalance extent due to its simplicity. It reflects the number of

instances of the most probable class for each instance of the least probable class. However whilst it is a very informative summary of the class-imbalance extent for binary problems, it is not capable of completely and honestly describing the disparity among the frequencies of more than two classes. In the multi-class scenario, there exists other classes rather than the most and least probable classes and they are not taken into account for the calculation of this summary. This may lead to the undesired situation of characterising multi-class problems with disparate class-imbalance extents using the same imbalance-ratio. In order to clarify this drawback, let's consider the following toy example: Imagine that a 3-class problem with an imbalance-ratio of 20 (100 : 5) is provided. This means that there are 20 examples of the most probable class (c1) for each example of the least probable class (c3). However, by means of just imbalance-ratio, little knowledge can be extracted regarding the remaining class c2, i.e. the number of examples of c2 can vary from 5 to 100, and all these 95 different possible scenarios share an imbalance-ratio equal to 20. To solve the IR calculation issue in the multi label scenario, [8] proposed metrics such as Imbalance Ratio per Label (IRLbl) and Mean Imbalance Ratio (MeanIR). While the IRLbl measure calculates the IR for each label as a ratio between the frequency of the given label and the most common label in the labelsets, MeanIR is the average level of imbalance in a multi-label dataset as the average IRLbl.

$$IRLbl(y) = \frac{\max\limits_{y' \in L} \left( \sum\limits_{i=1}^{|D|} h(y', Y_i) \right)}{\sum_{i=1}^{|D|} h(y, Y_i)} \tag{3.1}$$

where:

$$h(y, Y_i) = \begin{cases} 1 & y \in Y_i \\ 0 & yY_i \end{cases}$$

$$MeanIR = \frac{1}{|L|} \sum_{y \in L} IRLbl(y) \tag{3.2}$$

## 3.3 Hierarchical imbalance ratio

The measures 3.1 and 3.2 proposed by [8] are fitted to work with multi-label dataset, but not hierarchically organized ones. metric will not be able to detect the imbalance in a local perspective.

$$IR(n,p) = \frac{1}{|C_n|^2} \sum_{i=1}^{|C_n|} \frac{\sum_{j=1}^{D} h(S_j, C_n, p)}{\sum_{j=1}^{D} h(S_j, C_{n_i}, p)} \tag{3.3}$$

where:

$$h(S_j, C_n, p) = \begin{cases} 1 & \text{if } S_j \in Tr(n) \text{ given policy } p \\ 0 & \text{otherwise} \end{cases}$$

$$p \in [siblings, inclusive, ...]$$

# Bibliography

[1] Alex A. Freitas, André de Carvalho (2007) A Tutorial on Hierarchical Classification with Applications in Bioinformatics

[2] Carlos N. Silla, Jr. Alex A. Freitas (2010) A survey of hierarchical classification across different application domains Carlos N. Silla Jr. Alex A. Freitas

[3] Svetlana Kiritchenko et al. (2006) Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization

[4] Rodolfo M. Pereira et al. (2021) Handling imbalance in hierarchical classification problems using local classifiers approaches

[5] Michelangelo Ceci, Donato Malerba (2007) Classifying web documents in a hierarchy of categories: a comprehensive study

[6] Fagni Tiziano, Sebastiani Fabrizio (2007) On the selection of negative examples for hierarchical text categorization

[7] Eisner R, Poulin B, Szafron D, Lu P, Greiner R (2005) Improving Protein Function Prediction using the Hierarchical Structure of the Gene Ontology

[8] Francisco Charte et al. (2013) A First Approach to Deal with Imbalance in Multi-label Datasets