Pitfalls of Assessing Extracted Hierarchies for Multi-Class Classification

Pablo del Moral[a,*], Sławomir Nowaczyk[a], Anita Sant'Anna[a], Sepideh Pashami[a]

[a] *CAISR, Halmstad University, Kristian IV:s väg 3, 301 18 Halmstad, Sweden*

Abstract

Using hierarchies of classes is one of the standard methods to solve multi-class classification problems. In the literature, selecting the right hierarchy is considered to play a key role in improving classification performance. Although different methods have been proposed, there is still a lack of understanding of what makes a hierarchy good and what makes a method to extract hierarchies perform better or worse.

To this effect, we analyze and compare some of the most popular approaches to extracting hierarchies. We identify some common pitfalls that may lead practitioners to make misleading conclusions about their methods. To address some of these problems, we demonstrate that using random hierarchies is an appropriate benchmark to assess how the hierarchy's quality affects the classification performance.

In particular, we show how the hierarchy's quality can become irrelevant depending on the experimental setup: when using powerful enough classifiers, the final performance is not affected by the quality of the hierarchy. We also show how comparing the effect of the hierarchies against non-hierarchical approaches might incorrectly indicate their superiority.

Our results confirm that datasets with a high number of classes generally present complex structures in how these classes relate to each other. In these datasets, the right hierarchy can dramatically improve classification performance.

Keywords:

Hierarchical Multi-class Classification, Multi-class Classification, Class Hierarchies

*Corresponding author

Email address: pablo.del_moral@hh.se (Pablo del Moral)

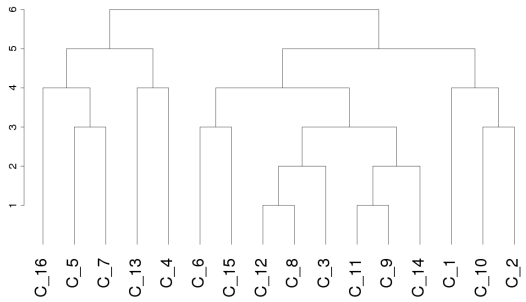


Figure 1: Example of a hierarchy of Classes

---

1. Introduction

Extracting hierarchies of classes to transform Multi-class Classification problems into Hierarchical Multi-class Classification (HMC) ones is a popular approach in the literature [1]. Compared to other approaches, it reduces training and especially testing running times while being competitive in classification performance [2]. However, quite surprisingly, a thorough discussion on the nature and importance of the extracted hierarchies and how to extract and evaluate them is still missing.

Intuitively, HMC consists of breaking the multi-class problem into a collection of binary classification problems defined by a particular hierarchy. The hierarchy is a binary dendrogram, where all classes are placed at the leaves and are iteratively merged until they are all clustered together at the root (see Figure 1). A hierarchical classifier is trained, building binary classifiers at each node to discriminate the classes in the left sub-tree from those in the right sub-tree. To test a new instance, we evaluate the classifier at the root node, selecting one of its two children. Recursively, we evaluate the selected node until we reach one of the leaves. The total number of evaluations equals the number of nodes in the path connecting the root and the selected leaf and is always smaller than the number of classes.

Hierarchical multi-class classification might suffer from error propagation. If an instance is incorrectly classified at the early nodes, the error will propagate downwards.

In this paper, we will focus on hard-class predictions, i.e., the output of every binary classifier is one of two classes. In this case, these errors will be unrecoverable.

There are many ways to define the goodness of a hierarchy: we can compare it with some ground truth, analyze the quality of its clusters, interpret the relationships between nodes and their meaning,... For hierarchical classification, the quality of a hierarchy is defined by the classification performance of a classifier trained on it.

The intuition is that in a good hierarchy, easy-to-separate classes are clustered close to the root, while hard-to-separate ones are clustered together farther from the root and closer to the leaves. This way, the classifiers trained at the higher nodes of the hierarchy need to solve easy decision boundaries, which are less prone to error, reducing the effects of the error propagation. On the other hand, the hard decision boundaries are solved at the bottom of the dendrogram by a classifier trained explicitly for them, improving the overall classification performance.

Clearly, this idea defines the first requirement for the existence of a good hierarchy: classes are not independent of each other, and there is some structure between them. The next requirement is that a hierarchical classifier trained on hierarchies that capture these relationships will significantly increase classification performance compared to a hierarchy that does not capture them.

The total number of possible hierarchies is $(2n-3)!!$ where n is the number of classes [3]. Finding the best hierarchy through an optimization scheme is costly, especially with many classes. However, suppose we assume that the hierarchy that best captures these underlying relationships between classes is the one that can be used to train a hierarchical classifier with the best performance. Then, we can focus on the process of extracting the hierarchy directly from the data in an unsupervised manner.

Extracting hierarchies from the data for HMC has been heavily studied in the machine learning literature [4, 5, 6, 7]. Typically, the main approach was presented together with a methodology to extract the hierarchy. The overall performance was then compared with other approaches that do not involve hierarchies, such as *single multi-class classifiers* or *One-vs-All* ensembles.

This type of analysis fails to answer the following questions: is there a structure in how classes relate to each other? Is this method discovering this structure? Is the

extracted hierarchy useful for the learning algorithm used? In other words, these works validate the goodness of the hierarchical classification approach, but not necessarily the quality of the method to extract the hierarchy and the quality of the hierarchy itself.

In this paper, we answer these questions by analyzing different methods of extracting hierarchies. Conceptually, extracting hierarchies from data points is a well-established field of research called hierarchical clustering. Any method in this area consists of two main components: measuring the dissimilarity between points and evaluating the quality of the hierarchy. In the specific context of HMC, extracting hierarchies presents two specific challenges. First, instead of measuring dissimilarities between points, one must measure dissimilarities between classes – which can be thought of as sets of data points, but that is not necessarily the best approach since it ignores important properties such as the shape or complexity of class boundaries. Second, the quality of the hierarchy is characterized externally, namely by the performance of the classifier trained on it, instead of the data's inherent structure.

We are going to critically evaluate different state-of-the-art approaches for obtaining hierarchies of classes, focusing on how to measure dissimilarity between classes and how to cluster them. In particular, we are going to compare two families of dissimilarities: Representative Based Dissimilarity (RBD) and Classifier Based Dissimilarity (CBD). RBD computes a representative for each class and measures dissimilarity between class representatives. On the other hand, CBD first trains a classifier on the whole dataset and infers dissimilarity from the resulting confusion matrix. In addition, we are going to compare two examples of clustering algorithms: Hierarchical Agglomerative Clustering (HAC) and Hierarchical K-means (HKM). The first one represents the family of bottom-up algorithms, whereas the latter represents the family of top-down. We demonstrate that the quality of the extracted hierarchy is not the only factor affecting the final classification performance and identify several considerations that must be taken into account when presenting evaluations. In particular, we analyze the effects of the HMC approach itself and how the effect of the quality of the hierarchy changes depending on how powerful the learning algorithms used for the hierarchical classifier are.

This paper contributes to the state-of-the-art with the following: 1) we identify several pitfalls in the process of extracting and evaluating methods to extract hier-

archies for HMC; 2) we propose using a random hierarchy as a necessary benchmark to evaluate the relevance of a hierarchy; in this way, one can establish whether a given extracted hierarchy is capturing an existing structure between classes or not; 3) we demonstrate how the effect on the classification performance of the hierarchy quality depends on the complexity of the classification problem and the complexity of the classifiers used; 4) we analyze different approaches for extracting hierarchies to showcase the previous contributions.

2. Literature Review

Traditionally, learning algorithms are designed to deal with binary classification problems. In this sense, multi-class classification approaches can be grouped into two categories: extended algorithms and binary adaptation.

Extended algorithms refer to those algorithms that are adapted for the multi-class case by solving it as a global optimization problem, overlooking relationships between classes. We will refer to them as single multi-class classifier or simply *single classifier*. Some learning algorithms are easily extended, like KNN, decision trees, or neural networks. Others, like SVMs ([8]; [9]) or logistic regressions [10], cannot be adapted so straightforwardly. In any case, the number of possible classes affects the computational complexity of training and testing at least linearly.

Binary adaptation decomposes the multi-class problem into several binary problems. Based on how the decomposition is done, we can distinguish between approaches that consider the relationships between classes and those that do not.

All-vs-all (AVA) and One-vs-all (OVA) are the most popular approaches among the latter. AVA [11] consists of training one classifier for each pair of classes. $\frac{N(N-1)}{2}$ classifiers must be trained. The final prediction is obtained after combining all classifiers through some form of ensemble aggregation method [12]. OVA [13] consists of training one classifier for each class, separating between the given class and the rest. The final prediction is again obtained via voting [12].

In [13], the authors conclude that there are no significant differences in performance between OVA and AVA when the binary learners are well-tuned regularized

classifiers. Moreover, they reflect on the limitations of OVA in the cases where classes are not independent. They hypothesize that an approach that exploits these relationships can achieve better performance. This hypothesis was separately corroborated from a theoretical point of view in [14] and [15].

The main approaches focusing on exploiting the relationship between classes to improve performance are Chains of Classifiers (CC) and Hierarchical Multi-class Classification (HMC). CC [14] consists of training a sequence of binary classifiers. The first one distinguishes one class from the rest; the next one separates another class from the remaining rest. The process is repeated until all classes are classified. Dissimilarity between classes is used to establish the order of the classes so that easy-to-discern classes are classified first, and hard-to-discern ones are classified at the end of the chain.

2.1. Hierarchical Multi-class Classification

Hierarchical Multi-class Classification is based on using a hierarchy of classes encoded in a dendrogram structure. Each class corresponds to one leaf of the tree. At each node, a classifier is trained to distinguish among its children. Predictions start from the classifier at the root node and are propagated until a leaf is reached.

Already when using probabilistic predictions in the binary models trained at each node (i.e., when all nodes need to be evaluated), testing time is reduced compared to AVA or OVA [4]. This benefit significantly increases when hard predictions are propagated. In this setup, only the nodes connecting the final predicted class and the root need to be evaluated [2]. Reduction of testing time comes at the expense of reducing performance. In [16], the authors explore this trade-off by allowing the evaluation of the most probable paths instead of just one.

The term "hierarchical classification" has been used both for tasks where there is a pre-existing hierarchy of classes that can be exploited [17] or for tasks where such hierarchy needs to be extracted [18]. Even with a predefined hierarchy, improvements can be achieved by modifying it. In [19], hierarchies are modified by eliminating some of the intermediate nodes of the hierarchy. In [20], predefined hierarchies are modified by rewiring parent-child relations or deleting and creating intermediate nodes. In [21], the authors present a method to fusion hierarchies directly extracted from the

data and expert-made hierarchies. We will focus on the extraction of hierarchies for multi-class classification problems where there is no predefined one.

Extraction of hierarchies has also been used in the field of neural networks and language modeling. In cases with multiple outputs, hierarchical softmax provides significant speed-up for both training and testing [22]. Extraction of hierarchies of words was explored in [23], yielding significant improvements in performance.

Hierarchical Multi-Class classification with binary splitting is often referred to as nested dichotomies. In [24], they achieve improvements in predictive power by ensembling hierarchical classifiers trained on several randomly generated hierarchies. In [25], the authors study the effects of random hierarchies and relate them with the learning algorithm used for classification. They present a method to obtain hierarchies, selecting the best hierarchy from a pool of randomly generated ones.

Extracted hierarchies are sometimes allowed to have more than two children nodes per parent. In [26], the authors create a hierarchy of classes until a node contains less than a given number of classes (typically 100), then one-vs-all classifiers are trained. In [27], the authors propose to create shallow hierarchies to reduce error propagation by increasing the number of children allowed at each node.

In this paper, we will focus on extracted hierarchies with just binary splittings where only hard class propagation is considered. We choose to keep the term "Hierarchical Classification" because we believe that our contributions can be easily generalized to less constrained setups.

Focusing on the process of extracting the hierarchy, we distinguish two steps: selecting the dissimilarity measure to be used and a hierarchical clustering algorithm.

Dissimilarity measures between classes can be divided into two groups: Representative Based Dissimilarity (RBD) and Classifier Based Dissimilarity (CBD). RBD consists of obtaining a representative for each class and measuring distances between those representatives. In ([28]; [6]) authors select a centroid for each class and measure the cosine dissimilarity. A similar approach is used in [4], except for the usage of the euclidean distance between centroids. In [29], the authors use the Hausdorff distance to measure the dissimilarity between classes. The main advantage of using these types of dissimilarity is computational efficiency.

While RBD is simple and fast, it also presents some drawbacks. Namely, how *representative* is the class representative and how the distances between them describe the relations between classes. To answer these drawbacks, CBD builds on the concept of separability. If two classes are frequently misclassified into each other, they will be similar, whereas if the classes are not mixed, they must be dissimilar.

CBD involves training an auxiliary classifier and evaluating the confusion matrix between classes. In the earliest approach [7], the author calculates the confusion matrix using a single classifier, then uses the rows of the matrix as class representatives for the corresponding classes. In [2], the confusion matrix is calculated using the OVA approach; then, the confusion matrix is symmetrized. In [5], a single classifier is trained, then the dissimilarity between two classes is calculated based on the number of instances of each class wrongly classified as the other.

Once dissimilarities have been computed, a hierarchical clustering algorithm must be chosen. These algorithms can be divided into agglomerative and divisive. Hierarchical Agglomerative Clustering (HAC) consists of progressively merging the most similar objects [5, 7]. Divisive clustering algorithm consists of iteratively using flat cluster algorithms until each class is separated from the rest. K-means is used in [28] and [4], whereas [2], and [30] use spectral clustering algorithms. Other divisive methods aim to create balanced hierarchies in order to improve training and testing complexities (cf. [31]; [4]).

3. Methods

Formally, a hierarchy is defined as a binary dendrogram. Starting from the root, all classes are iteratively separated into two sub-trees until each class is placed in its own tree leaf (see Figure 1). Methods for extracting hierarchies from the data consist of two critical components: measuring the dissimilarity between classes and the hierarchical clustering algorithm.

3.1. Dissimilarity

A class is defined by a set of points annotated with the same label. Evaluating the dissimilarity between two classes c_1,c_2 means evaluating a function of the form

$d(c_1,c_2)=f(D_{c_1},D_{c_2})$, where $D_c=\{\mathbf{x_c^i}\}$ is the distribution of points defined by the feature vectors $\mathbf{x^i}$ annotated with label c.

Representative Based Distance (RBD). One way to measure the dissimilarity between classes consists of obtaining class representatives and evaluating their distance. This approach is frequently used, choosing the centroid as the class representative. Any distance metric can be used to calculate the dissimilarity between the class representatives. It is a straightforward approach and quite favorable in terms of computational complexity. Formally, the dissimilarity between two classes becomes $d(c_1,c_2)=d'(\overline{\mathbf{x}}_{c_1},\overline{\mathbf{x}}_{c_2})$, where d' is a distance metric (popular choices include euclidean or cosine distance).

Classifier Based Dissimilarity (CBD). Intuitively, if a classifier can not distinguish clearly between two classes, they must be similar. Conversely, if it can clearly differentiate between two classes, they must be dissimilar. The standard way of evaluating how well a classifier distinguishes between two classes is by evaluating its accuracy.

For a multi-class classification problem, evaluating all pairwise class dissimilarities is similar to using AVA. This task can become extremely time-consuming if the number of classes is high. Practitioners avoid this problem by obtaining a confusion matrix using faster approaches like OVA or single classifier. One option is to use the confusion matrix's rows as the corresponding class representatives [7, 2]. Another approach [5] is to calculate the dissimilarity between two classes as the accuracy measured in the corresponding subset of the confusion matrix M.

$$M=\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} => d(c_1,c_3)=\frac{m_{11}+m_{33}}{m_{13}+m_{31}+m_{11}+m_{33}} \tag{1}$$

Unlike AVA, these approaches not only evaluate how two classes relate to each other but also take into account the relation to the rest of the classes. The relationship between two classes can be overshadowed by the overlap with a third one.

All-vs-all testing. To avoid this problem and still use a faster approach, we propose to use the output of a probabilistic classifier as a proxy to evaluate the accuracy of distinguishing between two classes in the AVA fashion.

Let's consider a test set $D=[(\mathbf{x^1},y^1)...(\mathbf{x^m},y^m)]$ where $\mathbf{x^i}$ is the feature vector, $y^i \in \{c_1,...c_n\}$ is the associated class. Let $\mathbf{h}(\mathbf{x})=[P(c_1|\mathbf{x}),...,P(c_n|\mathbf{x})]$ be the output of our probabilistic multi-class classifier. For each pair of classes c_j and c_k, we evaluate the classifier only on those instances with real labels c_j and c_k and only compare the probability predictions corresponding to these classes. We define the proxy of the binary classifier for classes c_j and c_k as:

$$h'(\mathbf{x}) = \underset{c_j,c_k}{\operatorname{argmax}}(P(c_j|\mathbf{x}),P(c_k|\mathbf{x})) \tag{2}$$

Finally, the distance between classes c_j and c_k is:

$$d(c_j,c_k) = \frac{\sum\limits_{i:y_i \in \{c_j,c_k\}} (h'(\mathbf{x_i}) = y_i)}{\sum_i (y_i = c_j) + \sum_i (y_i = c_k)} \tag{3}$$

3.2. Building the hierarchy

Hierarchical Agglomerative Clustering (HAC). works in a bottom-up fashion. Initially, each class is its own cluster. HAC iteratively merges the two most similar clusters until all of the data is combined in the root node. There are different approaches to calculating the distance between two clusters with several classes. The most popular is "Average Link", where the distance between two clusters is the average distance between all pairs of classes.

Hierarchical K-means. is an example of a divisive hierarchical clustering algorithm. It works in a top-down manner by iteratively separating the points into two groups. Starting from the root, we use k-means with $k=2$ to create two sub-clusters of classes. We recursively repeat this step on each sub-cluster until every class is separated from the rest and placed on one leaf.

	Dataset name	Instances	Attributes	Classes	Min	Max
1	Collins	1000	19	30	6	80
2	Pen digits	10992	16	10	1055	1144
3	Abalone	4168	9	21	6	689
4	Arrythmia	438	262	9	9	245
5	Image Segmentation	2310	18	7	330	330
6	Cartdiotocography	2126	35	10	53	579
7	Semeion	1593	256	10	155	162
8	Texture	5500	40	11	500	500
9	Statlog(Sat)	6430	36	6	625	1531
10	Walking	149332	4	22	911	21991
11	Usps	9298	256	10	708	1553
12	Bach	5571	90	65	6	503
13	Letters	20000	16	26	734	813
14	Helena	65196	27	100	111	4005
15	Plants(margin)	1600	64	100	16	16
16	Plants(shape)	1600	64	100	16	16
17	Plants(texture)	1599	64	100	15	16
18	Mnist	70000	719	10	6313	7877
19	Covertype	581012	54	7	2747	283301

Table 1: Information about dataset used: number of instances, number of features, number of classes, and number of instances for the most and lest frequent class.

4. Experiments and Results

All experimental results reported are the average accuracy measured over 20-fold Monte Carlo validation using 90% of the data for training and 10% for testing each fold. Variances in the accuracies are computed with the method presented in [32]. We will compare the classification performance of the hierarchical classifiers trained on hierarchies extracted with different methods on the same dataset. We will use the corrected resampled t-test presented in [32] for hypothesis testing. To compare methods to extract hierarchies over many datasets, we will use the Friedman test [33].

Each experiment consists of measuring the dissimilarities, extracting the hierarchy, training binary classifiers on each node, and evaluating the results. RBD is calculated directly since it is a deterministic measure, while CBD is estimated as an average over 10-fold Monte Carlo validation. For clarity, we will refer to the classifiers trained to measure CBD dissimilarities as **CBD classifiers**; we will refer to the classifiers trained on the hierarchy as **base classifiers**. The *base classifiers* trained at each node of the hierarchy use traditional cross-validation to tune the parameters.

To sample hierarchies uniformly, we follow the method proposed in [25]. The datasets used are publicly available in the UCI and Open ML repositories[1] and an overview is presented in Table 1.

4.1. Benchmarks vs random hierarchy

To evaluate the quality of an extracted hierarchy, one needs an appropriate benchmark to compare against. If we compare Hierarchical Multi-class Classification (HMC) against other approaches like One-vs-All (OVA) or a single classifier, we are evaluating the quality of HMC as an approach and the quality of the hierarchy simultaneously. We claim that comparing against a random hierarchy is a more suitable solution since it decouples the effects of using HMC and isolates the effects of the hierarchy's quality.

In this experiment, we compare the results of a classifier trained on a random hierarchy (it is worth highlighting that a new hierarchy is randomly drawn at each fold) with the two popular benchmarks used in the literature: single classifier and OVA. Our goal is to understand how the characteristics of HMC affect performance.

As the learning algorithm within the nodes of the hierarchy, we use CART (as implemented in the **rpart** R package). We select the best complexity parameter from a grid going from 10^0 to 10^{-6}.

In Table 2, the performance of the single classifier is significantly worse than HMC on 17 out of 19 datasets. In the remaining two, there is no significant difference between the two approaches. Clearly, this demonstrates that a single classifier cannot be used as a benchmark for evaluating the quality of the hierarchy; even HMC trained on a random hierarchy generally outperforms it.

On the other hand, OVA produces significantly better results than HMC trained on a random hierarchy in 5 out of 19 datasets. However, in 2 cases, HMC trained on a random hierarchy presents significantly better results, and for the remaining datasets, there are no significant differences. OVA is not expected to outperform HMC, even when trained on a random hierarchy, which makes OVA an inappropriate benchmark to evaluate the quality of the extracted hierarchies.

[1] `https://archive.ics.uci.edu/ml/datasets.html`, `https://www.openml.org`

	Single Classifier	Random	OVA
1	$\underline{0.07805 \pm 0.0068}^{\downarrow\downarrow\downarrow}$	0.1409±0.0096	0.166±0.01
2	$\underline{0.902 \pm 0.013}^{\downarrow\downarrow}$	0.9485±0.0023	0.9487±0.0019
3	0.254±0.0064	0.2417±0.0076	0.2462±0.0094
4	$\underline{0.6 \pm 0}^{\downarrow\downarrow\downarrow}$	0.731±0.016	0.718±0.022
5	$\underline{0.926 \pm 0.0054}^{\downarrow\downarrow}$	0.9537±0.0031	0.9576±0.0043
6	0.99928±0.00055	0.99904±0.00062	1±0
7	$\underline{0.523 \pm 0.014}^{\downarrow\downarrow\downarrow}$	0.731±0.016	0.762±0.011
8	$\underline{0.8433 \pm 0.0072}^{\downarrow\downarrow\downarrow}$	$\mathbf{0.9242 \pm 0.0037}^{\uparrow}$	0.9093±0.0033
9	$\underline{0.8383 \pm 0.0061}^{\downarrow}$	$\mathbf{0.8627 \pm 0.0033}^{\uparrow}$	0.8546±0.0039
10	$\underline{0.5099 \pm 0.0013}^{\downarrow\downarrow\downarrow}$	0.599±0.0017	$\mathbf{0.6209 \pm 0.0017}^{\uparrow\uparrow}$
11	$\underline{0.8059 \pm 0.0031}^{\downarrow\downarrow\downarrow}$	0.8551±0.006	$\mathbf{0.8758 \pm 0.0056}^{\uparrow}$
12	$\underline{0.6643 \pm 0.0051}^{\downarrow\downarrow\downarrow}$	0.7242±0.0044	0.7269±0.0052
13	$\underline{0.7208 \pm 0.0034}^{\downarrow\downarrow\downarrow}$	0.8541±0.0021	0.8547±0.0022
14	$\underline{0.20832 \pm 0.00091}^{\downarrow\downarrow\downarrow}$	0.2361±0.0038	$\mathbf{0.2859 \pm 0.0022}^{\uparrow\uparrow\uparrow}$
15	$\underline{0.021 \pm 0.0014}^{\downarrow\downarrow\downarrow}$	0.348±0.012	0.3555±0.0093
16	$\underline{0.02 \pm 0}^{\downarrow\downarrow\downarrow}$	0.323±0.01	0.322±0.013
17	$\underline{0.01 \pm 0}^{\downarrow\downarrow\downarrow}$	0.401±0.015	0.441±0.011
18	$\underline{0.88098 \pm 0.00047}^{\downarrow\downarrow\downarrow}$	0.93554±0.00091	$\mathbf{0.9394 \pm 0.00041}^{\uparrow\uparrow}$
19	$\underline{0.8184 \pm 0.0031}^{\downarrow\downarrow\downarrow}$	0.8595±0.0045	$\mathbf{0.8811 \pm 0.0014}^{\uparrow\uparrow}$

Table 2: Comparison of single classifier, HMC using a random hierarchy, OVA. Highlighted are the significantly best results, and underlined the significantly worst. One arrow indicates a p-value smaller than 0.05; two, 0.01; three, 0.001.

With this experiment, we have proven that Hierarchical Multi-class Classification has benefits independent of the quality of the hierarchy chosen. Using randomly generated hierarchies can result in performances significantly better than non-hierarchical approaches. If we want to evaluate methods of extracting hierarchies, comparing against methods that do not involve hierarchies gives an incomplete evaluation. Nevertheless, this type of analysis is still used in, for example, [4], [6], [7], [22], [30].

In [25], the authors propose a method to evaluate the quality of the hierarchy based on random hierarchies. The exceedance probability ranks the performance of a model trained on a given hierarchy against 10.000 random hierarchies. Training and testing 10000 hierarchies is costly. Also, there is the question of how many random hierarchies are needed to give a fair representation of all possible hierarchies. For example, with 100 classes, there are approximately 10^{184} possible hierarchies.

In the rest of the paper, we will compare different approaches to extracting hierarchies. Random hierarchies must be part of the process of evaluating their quality.

Instead of the exceedance probability, we will compare against the performance of a hierarchical classifier trained on randomly sampled hierarchies. This comparison does not allow us to determine how good a hierarchy is. However, it allows us to decouple the effects of the hierarchy's quality from the effects on the performance inherent in HMC approaches. If a given hierarchy obtains significantly better classification performance than a random hierarchy, we can conclude that this increase is due to the quality of the hierarchy and that this is capturing some relationship between the classes.

4.2. Quality of the hierarchy

Defining and quantifying the quality of a hierarchy of classes is not straightforward. Depending on the application, different metrics can be used. In the case of supervised classification, it is assumed that a hierarchical classifier trained on a good hierarchy should provide a better classification performance than on a bad one.

The majority of current HMC literature assumes that the performance of a hierarchical classifier is an (indirect) measure of the quality of the hierarchy. However, this performance is not only determined by the underlying quality of the hierarchy; but also by the relationship between the complexities of the classification problem and the learning algorithm used. In this section, we attempt to evaluate how these two factors interact. More specifically, to what degree can such a measure be used as an intrinsic property of a hierarchy, and to what degree is it specific to the experimental setup and the base learners used.

In this experiment, we illustrate the relationship between different learning algorithms and the quality of the hierarchy. For simplicity, we present the results for two datasets that showcase key findings. We draw 100 hierarchies uniformly at random

	Correlations			$\overline{\sigma_h}/\sigma(h)$		
	SVM-GLM	SVM-CART	GLM-CART	SVM	CART	GLM
5	-0.08	-0.18	0.06	1.02	0.85	0.20
7	-0.03	0.02	$0.52^{\uparrow\uparrow\uparrow}$	0.86	0.61	0.42

Table 3: On the left, pairwise correlations between different learning algorithms trained on the same hierarchies. Three arrows indicate correlations bigger than 0, with a p-value smaller than 0.001. No arrows indicates the lack of a significant difference. On the right, the ratio between the mean variance for each individual hierarchy and the variance of the performances across all hierarchies.

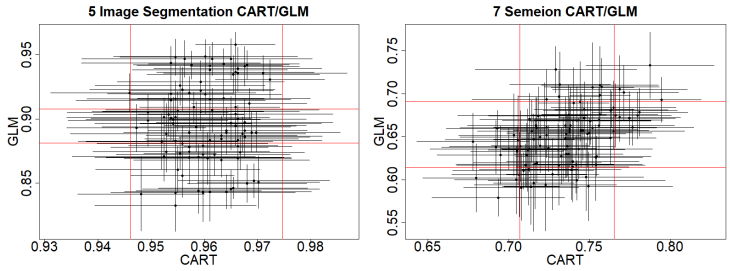


Figure 2: Pairwise comparison of the performance of CART and GLM trained on the same 100 randomly sampled hierarchies. Highlighted the variance of the hierarchy with the median performance.

and evaluate three different learning algorithms: CART, linear regression (GLM, as implemented in the R package **stats**), and SVM with gaussian kernel (as implemented in the package **liquidsvm**). For each random hierarchy, we calculate the performance over 20 montecarlo folds. In the case of SVM, we tune the width of the kernel and the regularization constant using an adaptive grid similar to the one used in [13].

In Table 3, we present the results of measuring the pairwise correlation between the results of the three different learning algorithms on the two selected datasets. We observe only one significant correlation, namely between GLM and CART on dataset 7 (*semeion*). This indicates an underlying quality of each hierarchy, where "good" hierarchies help both learning algorithms achieve better performance.

However, the same correlation does not happen on dataset 5 (*image segmentation*), nor between the other two pairs of classifiers. To understand these differences, we also measure the ratio between the variance of the results for a single hierarchy across the 20 montecarlo folds and the variance of the average accuracies of the 100 randomly sampled hierarchies (on the right-hand side of Table 3). This measure indicates how significant the differences are between the performances of hierarchical classifiers for each hierarchy. In dataset 5, there is no correlation between the performances of different hierarchies using GLM and CART because there are no significant differences in using one hierarchy or another for CART. This phenomenon is better understood by analyzing the corresponding plots in Figure 2, in particular the error bars.

In the left-hand subfigure, corresponding to dataset 5, all the evaluated hierarchies yield essentially the same performance for CART. On the other hand, there are

significant differences between hierarchies for GLM. This indicates that an underlying quality of the different hierarchies is not necessarily going to be visible in classification performance. For this dataset, CART is powerful enough not to be affected by it. It is worth contrasting with the right-hand subfigure, where the correlation between two algorithms is strong – both GLM and CART "agree" on some hierarchies being better than others.

When we compare the performances of GLM and CART with the performance of SVM on the same hierarchy (two leftmost columns in Table 3), every hint of correlation disappears. The reason is, again, the absolute insensitivity of a hierarchical classifier using SVM as a base learner for these two datasets.

These results hint at the existence of an underlying, objective quality of any given hierarchy. However, this quality of the hierarchy and the performance of a hierarchical classifier trained on it are not necessarily correlated. A powerful enough classifier will not be affected by the quality of the hierarchy: the quality of the hierarchy becomes irrelevant.

4.3. Random hierarchies vs informed hierarchies

Random hierarchies do not incorporate any information about the relationships between classes. The main hypothesis of HMC is that an informed hierarchy that exploits such information should yield better results in terms of predictive accuracy.

In this experiment, we evaluate one of the most popular methods to extract hierarchies, *Representative Based Dissimilarity*. RBD takes a representative for each class, computes their dissimilarity, and clusters them in a hierarchy. As a representative, we use the centroid; as dissimilarity, the euclidean distance; as clustering algorithm, *Hierarchical Agglomerative Clustering*.

The overall experimental setup is the same as the previous experiments. The comparison uses two base learners: CART and GLM. Both learning algorithms are trained on the same training set and on the same extracted hierarchy. In addition, we also analyze the differences between using hard-class predictions and probabilistic predictions.

Results are presented in Tables 4 and 5. For both CART and GLM, hierarchies obtained with RBD always yield similar or significantly better results than the

	CART			
	Hard-class		Probabilistic	
	Random	RBD	Random	RBD
1	0.1409±0.0096	0.171±0.011	0.1476±0.0087	0.18±0.011
2	0.9485±0.0023	**0.9551±0.002**$^{\uparrow}$	0.9485±0.0023	**0.9551±0.002**$^{\uparrow}$
3	0.2417±0.0076	0.2246±0.0084	0.2405±0.006	0.2417±0.0074
4	0.731±0.016	**0.766±0.017**$^{\uparrow}$	0.731±0.016	**0.766±0.017**$^{\uparrow}$
5	0.9537±0.0031	0.9617±0.0046	0.9537±0.0031	0.9617±0.0046
6	0.99904±0.00062	0.99952±0.00046	0.99904±0.00062	0.99952±0.00046
7	0.731±0.016	**0.787±0.011**$^{\uparrow\uparrow}$	0.731±0.016	**0.787±0.011**$^{\uparrow\uparrow}$
8	0.9242±0.0037	0.9333±0.0026	0.9242±0.0037	0.9333±0.0026
9	0.8627±0.0033	0.8644±0.0047	0.8627±0.0033	0.8644±0.0047
10	0.599±0.0017	0.6028±0.0011	0.6003±0.0017	**0.6045±0.0011**$^{\uparrow}$
11	0.8551±0.006	**0.8929±0.0032**$^{\uparrow\uparrow\uparrow}$	0.8551±0.006	**0.8929±0.0032**$^{\uparrow\uparrow\uparrow}$
12	0.7242±0.0044	**0.7473±0.0067**$^{\uparrow\uparrow}$	0.7244±0.0044	**0.7473±0.0067**$^{\uparrow\uparrow}$
13	0.8541±0.0021	**0.8633±0.0028**$^{\uparrow}$	0.8541±0.0021	**0.8633±0.0028**$^{\uparrow}$
14	0.2361±0.0038	**0.2685±0.0023**$^{\uparrow\uparrow\uparrow}$	0.2571±0.0028	**0.2856±0.0018**$^{\uparrow\uparrow\uparrow}$
15	0.348±0.012	**0.498±0.019**$^{\uparrow\uparrow\uparrow}$	0.348±0.012	**0.498±0.019**$^{\uparrow\uparrow\uparrow}$
16	0.323±0.01	**0.4015±0.0097**$^{\uparrow\uparrow\uparrow}$	0.323±0.01	**0.4025±0.0099**$^{\uparrow\uparrow\uparrow}$
17	0.401±0.015	**0.513±0.012**$^{\uparrow\uparrow\uparrow}$	0.401±0.015	**0.513±0.012**$^{\uparrow\uparrow\uparrow}$
18	0.93554±0.00091	**0.93853±0.00029**$^{\uparrow\uparrow}$	0.93554±0.00091	**0.93853±0.00029**$^{\uparrow\uparrow}$
19	0.8595±0.0045	**0.8784±0.0014**$^{\uparrow\uparrow}$	0.8595±0.0045	**0.8784±0.0014**$^{\uparrow\uparrow}$

Table 4: Comparison of HMC trained on a random hierarchy and on a hierarchy using RBD, and CART as base learner. Both deterministic and probability predictions are shown. Highlighted are the significantly best results. One arrow indicates a p-value smaller than 0.05; two, 0.01; three, 0.001.

random hierarchy. This indicates that the informed hierarchies have indeed captured some existing relationships between classes that are then exploited by the classifier.

For CART, when compared with the random hierarchy, we observe significant improvements in 12 out of 19 datasets using the RBD hierarchy in the hard-class case, and 13 in the probabilistic case. In the rest of the cases, we can not observe significant differences between any of the approaches. For GLM, we observe significant improvements in 14 out of 19 datasets for the RBD hierarchy with hard-class and probabilistic prediction. In 4 datasets, we can not observe significant differences between the approaches.

The first conclusion is that informed hierarchies are always better or as good as random hierarchies. When the difference in performance is significant, we can confirm that there is a structure between classes and that this method to extract hierarchies is capturing (at least partially) this structure. In the absence of significant

	GLM			
	Hard-class		Probabilistic	
	Random	RBD	Random	RBD
1	0.1945±0.0098	**0.248±0.013**$^{\uparrow\uparrow}$	0.231±0.017	0.264±0.014
2	0.86±0.013	**0.9184±0.0021**$^{\uparrow\uparrow}$	0.865±0.012	**0.919±0.0021**$^{\uparrow\uparrow}$
3	0.2248±0.0088	0.2448±0.0056	0.2558±0.0076	0.2694±0.0063
4	0.509±0.022	0.507±0.027	0.51±0.023	0.507±0.027
5	0.888±0.016	**0.9429±0.0045**$^{\uparrow\uparrow}$	0.891±0.015	**0.9426±0.0045**$^{\uparrow\uparrow}$
6	1±0	1±0	1±0	1±0
7	0.719±0.018	**0.776±0.0088**$^{\uparrow}$	0.72±0.018	**0.776±0.0088**$^{\uparrow}$
8	0.9824±0.0042	**0.99582±0.00078**$^{\uparrow}$	0.9825±0.0041	**0.99582±0.00078**$^{\uparrow}$
9	0.8249±0.0074	**0.8488±0.004**$^{\uparrow\uparrow}$	0.8263±0.007	**0.8503±0.0039**$^{\uparrow\uparrow}$
10	0.2058±0.0096	0.2114±0.0011	0.2433±0.0069	**0.2732±0.001**$^{\uparrow\uparrow}$
11	0.85±0.013	**0.9084±0.0031**$^{\uparrow\uparrow\uparrow}$	0.853±0.012	**0.9083±0.0031**$^{\uparrow\uparrow}$
12	0.6679±0.0061	**0.6949±0.006**$^{\uparrow\uparrow}$	0.674±0.005	**0.6991±0.0064**$^{\uparrow\uparrow}$
13	0.483±0.017	**0.7359±0.0027**$^{\uparrow\uparrow\uparrow}$	0.525±0.015	**0.7536±0.003**$^{\uparrow\uparrow\uparrow}$
14	0.2003±0.0051	**0.2661±0.0023**$^{\uparrow\uparrow\uparrow}$	0.2754±0.0034	**0.3185±0.0016**$^{\uparrow\uparrow\uparrow}$
15	0.262±0.015	**0.371±0.012**$^{\uparrow\uparrow\uparrow}$	0.272±0.016	**0.371±0.012**$^{\uparrow\uparrow\uparrow}$
16	0.122±0.0093	**0.268±0.014**$^{\uparrow\uparrow\uparrow}$	0.128±0.012	**0.268±0.014**$^{\uparrow\uparrow\uparrow}$
17	0.336±0.016	0.362±0.011	0.339±0.017	0.362±0.011
18	0.6977±0.0057	**0.7194±0.0017**$^{\uparrow\uparrow}$	0.6995±0.0055	**0.7201±0.0018**$^{\uparrow\uparrow}$
19	0.79±0.01	**0.8549±0.0012**$^{\uparrow\uparrow\uparrow}$	0.8059±0.008	**0.8612±0.0013**$^{\uparrow\uparrow\uparrow}$

Table 5: Comparison of HMC trained on a random hierarchy and on a hierarchy using RBD and GLM as base learner. Both deterministic and probability predictions are shown. Highlighted are the significantly best results. One arrow indicates a p-value smaller than 0.05; two, 0.01; three, 0.001.

differences, we can neither confirm nor refute that there exists a structure in how classes relate to each other. This can mean two different things: either there does not exist a hierarchy that produces significant improvements, or we have not found it yet.

It is worth noting that if we had compared with the single classifiers from the previous subsection, we would have concluded that the RBD hierarchies for datasets 1, 5, 9, and 10 were useful using CART. This conclusion would be wrong: the results of HMC trained on them are significantly better than the single classifier; however, they are not significantly better than HMC trained on a random hierarchy.

In all the datasets where there were significant differences using CART, we found significant differences using GLM. However, there are multiple examples where we found significant differences using GLM, but not when using CART. This reinforces our conclusion from the previous section: powerful enough classifiers will not be affected by the quality of the hierarchy. The absence of a significant difference using

	XGBoost		SVM	
	Random	RBD	Random	RBD
1	0.257±0.016	0.267±0.018	0.304±0.061	0.305±0.049
2	0.9895±0.0014	0.9908±0.0011	0.9940 ± 0.0020	0.9940 ± 0.0020
3	0.262±0.011	0.253±0.011	0.270 ± 0.018	0.274±0.021
4	0.762±0.018	0.767±0.017	0.726±0.053	0.737±0.048
5	0.9784±0.0031	0.9823±0.0038	0.974±0.0074	0.978±0.0087
6	0.99952±0.00058	0.99976±0.00042	1±0	1±0
7	0.9224±0.0067	0.9328±0.0071	0.953±0.12	0.953±0.015
8	0.9814±0.0017	0.9835±0.0017	0.998±0.002	0.999±0.0015
9	0.9186±0.0029	0.9192±0.0032	0.9220 ±0.0066	0.9170±0.0094
11	0.9603±0.0024	**0.9653±0.003**	0.970±0.0076	0.976±0.0053
12	0.7844±0.0064	0.7907±0.0062	0.811±0.014	0.816±0.014
13	0.9562±0.0016	0.9568±0.0018	0.962±0.0048	0.968±0.0029
14	0.3582±0.0017	**0.3673±0.0017**	0.34±0.0051	**0.356±0.0044**
15	0.64±0.025	**0.716±0.012**	0.814±0.035	0.836±0.037
16	0.509±0.019	0.544±0.016	0.713±0.040	0.716±0.038
17	0.668±0.018	**0.715±0.015**	0.844±0.030	0.850±0.028

Table 6: Comparison of HMC trained on a random hierarchy and on a hierarchy using RBD using Extreme Gradient Boosting and SVM using gaussian kernel. Highlighted are the significantly best results. One arrow indicates a p-value smaller than 0.05; two, 0.01; three, 0.001.

CART does not necessarily mean that the extracted hierarchy is not capturing interesting relationships between classes. Rather, CART can discover these relationships by itself.

As a natural extension, we hypothesize that for an arbitrarily complex problem, there is an arbitrarily complex learning algorithm for which the different possible decompositions will be irrelevant.

In Table 6, we can see the results of running a boosting algorithm based on trees and a support vector machine using gaussian kernels implemented as in Section 4.2. For the boosting algorithm, we have used the **xgboost** package and simply tuned the trees' depth. Although we still find significant differences in some datasets (only one in the case of SVM), those differences are comparatively smaller and less significant when using CART or logistic regression. It is of special interest the result of SVM on dataset 14: a hierarchy extracted using the distance between centroids of classes provides a significant improvement, even when the classifier trained on it uses a kernelized SVM that does not use the original feature space.

If the practitioner is interested in finding the best possible classification perfor-

mance without regard for the time it may take to train and evaluate the models, the effort should be directed into finding the most powerful classifier and tuning its hyper-parameters. In these cases, there is probably no great added value from choosing one decomposition over another. It is interesting to find good hierarchies when there are restrictions on how much resources we can afford for the training and evaluation phase.

At each fold of our validation scheme, we extract a hierarchy and train our models. In this scenario, our results' variation source is not just the sampling of training and testing sets, like in traditional classification tasks. The method to extract the hierarchy also contributes to the variance. This depends not only on the validation scheme but also on the intrinsic complexity of the relationship between classes and how we extract the hierarchy. This is evident by comparing the variances using a random hierarchy or an RBD hierarchy using GLM as a base classifier in Table 5.

Therefore, to establish a fair comparison between two methods to extract hierarchies over several datasets, we need first to make sure that there is an underlying structure that provides statistically significant improvements for each dataset. First, we need to analyze if the proposed method provides significant improvements over a random hierarchy, then we need to provide a sound statistical analysis of each dataset. This type of analysis is missing, for example, in [5] and [25].

4.4. Representative Based Dissimilarity vs Best of 50 heuristic

RBD using centroids and euclidean distance as metric is a standard procedure to measure dissimilarities. There are some concerns, though: how representative is the centroid for the complete class distribution (which can be arbitrarily complex) and how useful is the distance between centers to measure dissimilarity between classes.

Best-of-50 (Bo50) method was introduced in [25], it samples 50 hierarchies at random and selects the one with the best performance using 3-fold cross-validation.

In Table 7, using CART, we observe 8 datasets without significant improvement over the random hierarchy (Table 4). In 5 out of the other 11 datasets, we do not observe significant differences between Bo50 and RBD approaches. We can see significant differences in favor of RBD in 5 datasets and in favor of Bo50 in 1 dataset. The most significant differences happen in datasets 14, 15, 16, and 17, those that have

	CART		Logistic	
	RBD	Bo50	RBD	Bo50
1	0.171±0.011	0.185±0.013	0.248±0.013	0.276±0.015
2	0.9551±0.002	0.9599±0.0021	0.9184±0.0021	**0.9347±0.0044**$^{\uparrow\uparrow\uparrow}$
3	0.2246±0.0084	**0.269±0.0057**$^{\uparrow\uparrow\uparrow}$	0.2448±0.0056	**0.2726±0.0057**$^{\uparrow\uparrow}$
4	0.766±0.017	0.769±0.018	0.507±0.027	**0.576±0.031**$^{\uparrow}$
5	0.9617±0.0046	0.9693±0.0044	0.9429±0.0045	**0.9587±0.0053**$^{\uparrow\uparrow\uparrow}$
6	0.99952±0.00046	1±0	1±0	1±0
7	0.787±0.011	0.784±0.013	0.776±0.0088	0.796±0.014
8	0.9333±0.0026	0.9361±0.0029	0.99582±0.00078	0.99691±0.00084
9	0.8644±0.0047	0.8662±0.0056	0.8488±0.004	**0.8584±0.004**$^{\uparrow\uparrow}$
10	0.6028±0.0011	0.6031±0.0022	0.2114±0.0011	**0.2873±0.0056**$^{\uparrow\uparrow\uparrow}$
11	**0.8929±0.0032**$^{\uparrow}$	0.8855±0.0041	**0.9084±0.0031**$^{\uparrow}$	0.8995±0.0053
12	0.7473±0.0067	0.7454±0.0064	0.6949±0.006	**0.711±0.0063**$^{\uparrow}$
13	0.8633±0.0028	0.8614±0.003	**0.7359±0.0027**$^{\uparrow\uparrow\uparrow}$	0.573±0.006
14	**0.2685±0.0023**$^{\uparrow\uparrow\uparrow}$	0.2587±0.0022	**0.2661±0.0023**$^{\uparrow\uparrow\uparrow}$	0.2317±0.0028
15	**0.498±0.019**$^{\uparrow\uparrow\uparrow}$	0.372±0.019	**0.371±0.012**$^{\uparrow\uparrow}$	0.312±0.015
16	**0.4015±0.0097**$^{\uparrow\uparrow}$	0.34±0.018	**0.268±0.014**$^{\uparrow\uparrow\uparrow}$	0.1495±0.0096
17	**0.513±0.012**$^{\uparrow\uparrow\uparrow}$	0.422±0.012	0.362±0.011	0.394±0.015
18	0.93853±0.00029	0.93778±0.00073	0.7194±0.0017	0.7174±0.004
19	0.8784±0.0014	0.8807±0.0021	0.8549±0.0012	0.8514±0.0033

Table 7: Comparison of HMC trained on a hierarchy obtained with the Best of 50 heuristic and on a hierarchy using RBD. CART and GLM are used as learning algorithm. Highlighted are the significantly best results. One arrow indicates a p-value smaller than 0.05; two, 0.01; three, 0.001.

100 classes. Bo50 provides competitive results, especially if the number of classes is not very high. However, with many classes, it significantly under-performs compared to RBD, and in some cases (15, 16, and 17), it does not provide results significantly better than a random hierarchy. As the number of classes increases, so does the number of possible hierarchies – and Bo50 becomes less likely to sample a good hierarchy.

In Table 7 using GLM, we observe just one dataset where there is no significant improvement over the random hierarchy (Table 5). We observe significant differences in favor of Bo50 in 9 datasets and in 5 datasets in favor of RBD. These 5 datasets are the same 5 datasets where we observed the same phenomenon using CART. In 8 of the 9 datasets where Bo50 significantly outperforms RBD using GLM, we did not observe any difference using CART. Hierarchies that are useful for GLM are irrelevant for CART, arguably a more powerful classifier [34].

	RBD	CBD1	CBD2
1	0.248±0.013	0.252±0.014	0.252±0.014
2	0.9184±0.0021	0.912±0.0026	0.912±0.0026
3	0.2448±0.0056	0.243±0.0062	0.243±0.0062
4	0.507±0.027	0.537±0.016	0.537±0.016
5	0.9429±0.0045	**0.9584±0.0041**	**0.9584±0.0041**
6	**1±0**	**1±0**	**1±0**
7	**0.776±0.0088**	0.7523±0.0092	0.7523±0.0092
8	0.99582±0.00078	0.9893±0.0013	0.9893±0.0013
9	0.8488±0.004	0.8337±0.0036	0.8337±0.0036
10	0.2114±0.0011	0.20537±0.00072	0.20537±0.00072
11	**0.9084±0.0031**	0.8697±0.0037	0.8697±0.0037
12	0.6949±0.006	0.6505±0.0049	0.6505±0.0049
13	**0.7359±0.0027**	0.4004±0.0043	0.4004±0.0043
14	**0.2661±0.0023**	0.19±0.0013	0.19±0.0013
15	0.371±0.012	0.189±0.011	0.189±0.011
16	**0.268±0.014**	0.09±0.0087	0.09±0.0087
17	0.362±0.011	0.258±0.014	0.258±0.014
18	**0.7194±0.0017**	0.694±0.02	0.694±0.02
19	0.8549±0.0012	0.8162±0.0011	0.8162±0.0011
	CBD3	OVA CART	OVA GLM
1	0.252±0.014	**0.253±0.015**	0.243±0.01
2	0.912±0.0026	0.9407±0.0016	**0.9446±0.0024**
3	0.243±0.0062	**0.2681±0.0058**	0.2649±0.0061
4	0.537±0.016	0.544±0.019	**0.556±0.021**
5	**0.9584±0.0041**	0.955±0.0041	0.9574±0.0041
6	**1±0**	**1±0**	**1±0**
7	0.7523±0.0092	0.7731±0.0088	0.775±0.0096
8	0.9893±0.0013	0.99664±0.00046	**0.99745±0.00075**
9	0.8337±0.0036	0.8566±0.0032	**0.8577±0.0029**
10	0.20537±0.00072	0.2129±0.001	**0.2326±0.001**
11	0.8697±0.0037	0.899±0.0081	0.9013±0.0035
12	0.6505±0.0049	0.6993±0.0057	**0.7189±0.0058**
13	0.4004±0.0043	0.6764±0.0062	0.6986±0.004
14	0.19±0.0013	0.2476±0.0057	0.2615±0.0045
15	0.189±0.011	0.326±0.014	**0.436±0.015**
16	0.09±0.0087	0.199±0.011	0.263±0.012
17	0.258±0.014	0.365±0.015	**0.536±0.011**
18	0.694±0.02	0.694±0.02	0.7178±0.0014
19	0.8162±0.0011	0.8597±0.0014	**0.8605±0.0013**

Table 8: Comparison of HMC trained on different hierarchies: random hierarchy, RBD hierarchy, Flat Classifier CBD (increasing complexity),OVA CBD (using CART and GLM) as learning algorithms). Highlighted are the highest accuracy for each dataset, which does not necessarily significantly better.

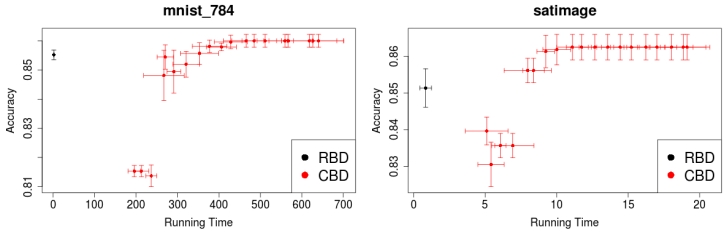


Figure 3: Evolution of the performance of HMC trained on different hierarchies as the complexity of the CBD classifier increases, compared against RBD.

4.5. Representative Based Dissimilarity vs Classifier Based Dissimilarity

Another popular approach to measure dissimilarities between classes is using an initial classifier to evaluate how similar classes are to each other. *Classifier Based Dissimilarity* builds on the concept of separability and measures how well a classifier can separate between the different classes. In these experiments, we are going to compare CBD and RBD. While RBD is a deterministic metric, CBD will depend on how we train the classifier. We will measure how the quality of the CBD classifier affects the quality of the hierarchy and, therefore, the classification performance.

To obtain the CBD, we are going to run single classifiers and OVA (using CART and GLM). For the single classifiers we use CART with three different sets of complexity parameters: (0.5,0.1), (0.5,0.1,0.05,0.01), (0.5,0.1,0.05,0.01,0.005,0.001). For OVA using CART we select the complexity parameter from the values (0.5,0.1,0.05, 0.01,0.005,0.001) using 10 fold cross-validation for each binary classifier. Once we have the CBD classifiers, we compute the dissimilarity as explained in 3.1.

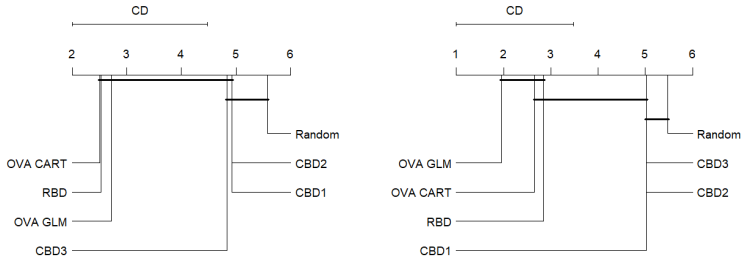
Table 8 shows the results of using different methods to extract hierarchies using GLM as base classifier. In 9 out of 19 datasets, at least one CBD variant presents significantly better results than RBD. The counterexample only happens once.

In some cases, the hierarchies using the simplest CBD classifier outperforms their RBD counterpart; in others, hierarchies with simple CBD classifiers perform worse than RBD, but this difference disappears as the CBD classifier becomes more complex. OVA using GLM as CBD classifier produces significantly better results than OVA using CART as CBD classifier in 6 datasets, while the opposite happens only once. This is apparent in those datasets with more classes. The learning algorithm used on

the hierarchy is also GLM; this opens up interesting questions about the relationship between the CBD classifier and the base classifier trained on the hierarchy.

In general, we observe an improvement as the complexity of the CBD classifier increases. In Figure 3, we can see a finer detail on this effect. Similar to the CBD columns in Table 8, we have chosen different configurations of the CART algorithm's complexity parameter. As a proxy for the complexity, we have calculated the running time it takes to train the CBD classifier. Clearly, the range of classifiers complexities explored in this experiment is limited. Intuitively, however, it is obvious that CBD cannot be thought of as a single measure and, as such, compared directly against RBD. For any given dataset, it is possible to find a classifier that will create a worse, equal, or better hierarchy than RBD (if RBD has not found the best possible hierarchy). CBD is a family of dissimilarities and needs to be treated as such.

In [5], a direct comparison is performed between a version of CBD and RBD without acknowledging all the different possibilities to obtain CBD dissimilarities. In [7], [30], single versions of CBD are used to extract hierarchies. The results are compared with non-hierarchical approaches. In [2] and [23], single versions of CBD are introduced to extract hierarchies. The results are compared with random hierarchies. To the best of our knowledge, there has not been a deep study of the influence on how CBD is obtained and the final classification performance of the hierarchical classification problem.



(a) CART (b) GLM

Figure 4: Critical difference diagrams among different methods to extract hierarchies using CART and GLM as base classifiers.

In Figure 4, we have the diagrams of critical differences after running the Friedman

test using CART and GLM as base classifiers. There is no single method that significantly outperforms the rest. With GLM as base learner, hierarchies extracted using RBD, and CBD with OVA GLM and OVA CART are equivalent as the top method. With CART as base learner, all methods except the random hierarchy are equivalent.

We are trying to evaluate methods to extract hierarchies using the performance of a hierarchical classifier trained on it. This is an indirect measure of the quality of the hierarchies that depends critically, as we have already discussed, on the learning algorithm used as base classifier. In our example, all methods of extracting hierarchies (except the random) are equally good using CART as base classifier. However, if we use GLM as base learner, it is apparent that some methods are better than others.

4.6. Relationship between the base classifier and the CBD classifier

In the previous experiment, we established that CBD using OVA and GLM performs better than CBD using CART when we also use GLM as base classifier. The intention behind CBD is to measure how easy two classes are to separate. More specifically, we want to know how easy it will be for the learning algorithm trained on the hierarchy to separate between these two classes. However, the results of training the CBD classifier give us only an approximation of how well the base classifier will perform.

With this motivation in mind, in this experiment, we want to explore the relationship between the power of the CBD classifier used to measure dissimilarities between classes and the power of the base classifier trained on the hierarchy. The same experimental setup from the previous experiments applies. As CBD classifier, we use a single CART as in the previous section. As the base classifier, we also choose CART.

Figure 5 shows how the classification accuracy across four different datasets changes choosing different configurations of the complexity parameters for both the CBD and the base classifier. The most interesting results can be seen in Figure 5a: for the simpler base classifier, the accuracy drops as the CBD classifier gets more complex. The same behavior can be seen in 5d, but not in 5b and 5c. This shows that the relationships between classes found by a powerful CBD classifier will not necessarily be useful when we have a weak base classifier.

Finally, in Figure 5c, one can see very small differences in overall classification

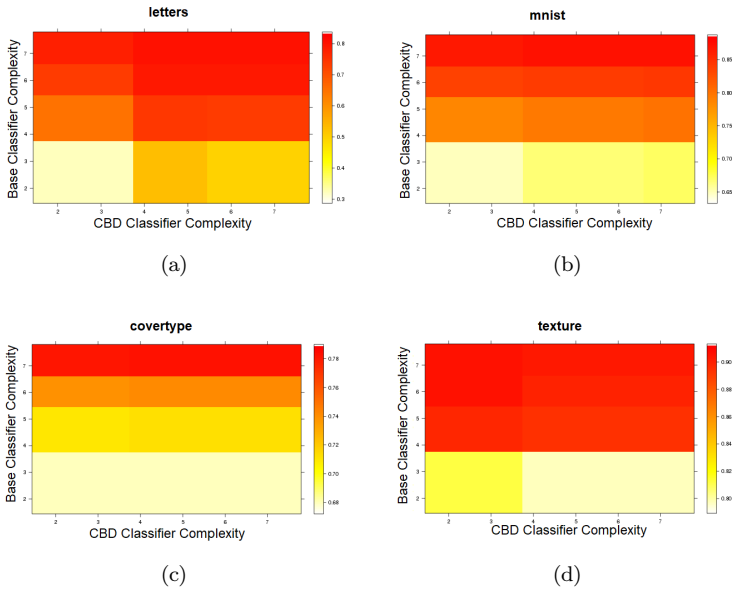


Figure 5: Comparison of the complexity of the CBD classifier and the base classifier

performance as the quality of the hierarchy increases for any configuration. This suggests that for this dataset, there is no inherent structure between classes relevant for any of the classifiers.

4.7. Hierarchical Agglomerative Clustering vs. Hierarchical K-Means

So far, we have only discussed differences in the hierarchies depending on how we measure dissimilarity. In this experiment, we explore the differences created by different clustering algorithms. In particular, Hierarchical Agglomerative Clustering (HAC) and Hierarchical K-Means (HKM).

The experimental setup is similar to the previous sections. In Tables 9 and 10, we can see the results of using CART and GLM as learning algorithms. In both cases, we will compare the clustering algorithms using RBD and CBD using OVA as dissimilarity metrics. When we use CART as a base learner, we will also use CART as a classifier for OVA; similarly, we will use GLM for OVA when using GLM as a base classifier.

Using CART as a base classifier, we cannot observe many significant differences between HAC and HKM. The main improvement in the hierarchies comes from the

	CART			
	RBD-HAC	RBD-HKM	OVA-HAC	OVA-HKM
1	0.171±0.016	0.173±0.014	0.151±0.018	0.16±0.017
2	0.9558±0.0024	0.9506±0.0019	0.9585±0.0026	0.9588±0.0032
3	0.229±0.011	0.236±0.011	0.2467±0.0092	0.257±0.01
4	0.767±0.019	0.749±0.015	0.72±0.025	0.684±0.029
5	0.9606±0.0058	0.9621±0.0043	0.9673±0.0045	0.9658±0.0042
6	0.99952±0.00058	0.9971±0.0017	1±0	1±0
7	0.792±0.015	0.786±0.018	0.781±0.013	0.769±0.015
8	0.9346±0.0037	0.9365±0.0035	0.935±0.0026	0.9309±0.0028
9	0.8659±0.0053	0.8631±0.0056	0.8613±0.0048	0.8662±0.0054
10	0.6±0.0018	0.6047±0.0019	0.605±0.0013	0.6023±0.0015
11	0.8944±0.0037	0.8838±0.0041	0.8834±0.0074	0.8876±0.0048
12	0.747±0.0085	0.7435±0.0061	0.73±0.0071	0.7397±0.0069
13	0.8628±0.0036	0.8666±0.003	0.8673±0.0037	0.8651±0.0029
14	0.2688±0.0029	0.2704±0.0019	0.2558±0.0042	**0.2754±0.0023**†††
15	0.496±0.017	0.466±0.018	0.42±0.017	0.381±0.02
16	0.392±0.015	0.411±0.018	0.367±0.023	0.365±0.017
17	0.517±0.017	0.484±0.016	**0.474±0.015**††	0.418±0.017
18	0.93855±0.00035	0.93854±0.00035	0.93851±0.00037	0.93852±3e-04
19	0.8784±0.0018	**0.8856±0.0017**†††	**0.8857±0.0017**††	0.8808±0.002
d	9.2	6.6	13.5	6.5

Table 9: Comparison HAC and HKM using CART as learning algorithm. In the last row, average depth of the extracted hierarchies

information shown in the dissimilarities and not from the clustering algorithm.

On the other hand, if we use GLM as a base classifier, we find several significant differences in the pairwise comparisons in favor of HMC. This can be explained by the shape of the final hierarchy and the choice of GLM as a learner. As suggested by the average depths of the hierarchies created, HAC tends to create hierarchies in a chain shape, while HKM tends to generate more balanced hierarchies (see Figure 6).

Depending on the learning algorithm chosen, the final results might vary not only based on the information about how similar the classes are but also on the particular shape of the hierarchy.

5. Conclusions and Discussions

The most popular approaches to solving multi-class problems (AVA, OVA, extended algorithms) become prohibitively time-consuming as the number of classes in-

	Logistic Regression			
	RBD-HAC	RBD-HKM	OVA-HAC	OVA-HKM
1	0.248±0.016	0.25±0.016	0.243±0.013	0.265±0.016
2	**0.9184±0.0027**†††	0.887±0.0042	**0.9446±0.003**†††	0.9167±0.0034
3	0.2448±0.007	0.243±0.01	0.2649±0.0076	0.2712±0.0079
4	0.507±0.034	0.458±0.03	0.556±0.026	0.536±0.037
5	0.9429±0.0056	0.9429±0.0061	**0.9574±0.0052**†††	0.8645±0.0074
6	1±0	1±0	1±0	1±0
7	0.776±0.011	0.786±0.013	0.775±0.012	0.791±0.011
8	0.99582±0.00098	0.99555±0.00078	**0.99745±0.00093**†††	0.9884±0.0021
9	0.8488±0.0051	0.8488±0.0051	0.8577±0.0036	0.8593±0.0036
10	0.2114±0.0013	**0.2194±0.0013**†††	0.2326±0.0013	**0.2444±0.002**†††
11	**0.9084±0.0039**†††	0.8887±0.0041	**0.9013±0.0044**†††	0.8693±0.0035
12	0.6949±0.0076	0.7108±0.0095	0.7189±0.0072	0.7111±0.0066
13	**0.7359±0.0034**†††	0.6916±0.0045	**0.6986±0.005**†††	0.6638±0.0058
14	0.2661±0.0029	**0.2767±0.0024**†††	0.2615±0.0057	0.2628±0.0023
15	**0.371±0.015**††	0.327±0.014	**0.436±0.018**†††	0.236±0.019
16	**0.269±0.018**††	0.228±0.017	**0.262±0.015**†††	0.14±0.012
17	**0.361±0.014**††	0.298±0.016	**0.536±0.014**†††	0.267±0.011
18	0.7186±0.0022	0.697±0.021	0.7154±0.0024	0.698±0.02
19	0.8549±0.0015	**0.8661±0.0019**†††	**0.8605±0.0016**†††	0.8457±0.0033
d	9.2	6.6	18.1	6.7

Table 10: Comparison HAC and HKM using logistic regression as learning algorithm. In the last row, average depth of the extracted hierarchies

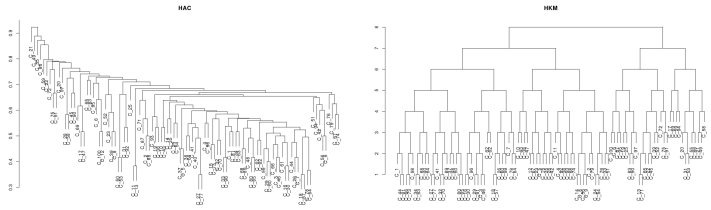


Figure 6: On the left, hierarchy obtained with HAC with average link. On the right, hierarchy obtained with Hierarchical K-means.

creases. Hierarchical Multi-class Classification (HMC) significantly reduces prediction time while maintaining reasonable training time and classification performance [2].

Shortening the prediction time is achieved by reducing the number of classifiers that need to be evaluated. There are several solutions proposed in the literature, but this paper is the first comprehensive discussion on what is a good hierarchy and how to measure how its quality can affect the classification performance of a hierarchical

classifier trained on it. We demonstrate that the relevance of the hierarchy's quality depends on the trade-off between the complexity of the classification problem and the complexity of the learning algorithm used.

We have observed that good hierarchies provide measurable increments in classification performance only when the classifier is sensitive to the quality of the hierarchy. We have not found a single case of a "strictly inconsistent" hierarchy performance: where one hierarchy would be significantly better for one learning algorithm and significantly worse for another. The differences across different algorithms are either unmeasurable or consistent. This supports the argument that hierarchy quality can be evaluated objectively.

We have compared the state-of-the-art practices to extract hierarchies and evaluated them. In the process, we have identified some of the common pitfalls of extracting hierarchies and how to avoid them for HMC.

The quality of the hierarchy does not always affect the performance of the classifier. A hierarchy will only be useful if it can exploit the existing relationships between classes. This requires that there is an actual structure in how classes relate to each other and that the extraction method can find it – and neither of those assumptions necessarily holds for all datasets. In addition, it requires that the extracted hierarchy is useful for the learning algorithm. We have shown that the weaker the base classifier is, the more relevant the hierarchy's quality becomes. We have observed that for some datasets, a hierarchy can be useful using a simple classifier as the base classifier but irrelevant for a more powerful one. For any given dataset, a powerful enough classifier will not be affected by the hierarchy's quality.

All of this means that in many common settings, the hierarchy's quality becomes irrelevant to the overall predictive performance. At the same time, the divide-and-conquer aspect inherent in the HMC approach may provide benefits over alternatives, like either single classifiers or OVA. Ignoring this might lead practitioners to make claims about the quality of a hierarchy when, in fact, they are validating other aspects. To avoid these cases, we suggest using HMC on a random hierarchy as an appropriate benchmark to understand the extracted hierarchy's quality. By comparing with a random hierarchy, we can isolate the added value of the hierarchy's quality. If there

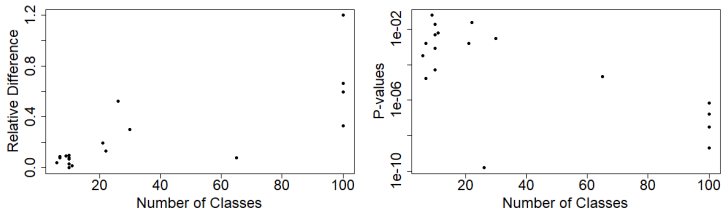


Figure 7: On the left, relative difference of accuracies as the number of classes increases, measured with respect to the random hierarchy. On the right, significance of the difference between the random hierarchy and the best informed hierarchy.

is no existing relationship between classes relevant to the chosen learning algorithm, extracted hierarchies will not provide results significantly better than those from random hierarchies.

Throughout our experiments, we have observed that the hierarchy's quality becomes more and more relevant as the number of classes increases (See Figure 7. With more classes, we expect more complex structures and good hierarchies to become crucial. The comparison with respect to random hierarchies shows that the structures found by our informed hierarchies do exist and that the chances of obtaining them randomly can be quite low.

We have noticed that some base classifiers are more sensitive to the shape of the hierarchy. Hierarchical clustering with average link tends to provide deeper dendrograms than hierarchical K-means. In our case, using logistic regression as the base classifier, we have found significant differences in favor of the chain-shape structures. However, using CART, we have not found apparent differences. Comparing different clustering algorithms just based on the classification performance gives only a partial view of the problem. Without considering how the learning algorithm used is affected by the shape of the dendrogram might lead practitioners to misleading conclusions.

We have found that the hierarchy's quality depends decisively on how we measure the dissimilarity between classes. We have reviewed two existing approaches: representative-based dissimilarities (RBD) and classifier-based dissimilarity (CBD). Using RBD is fast and simple and often produces significant improvements when compared to random hierarchies. However, we have found that hierarchies obtained

with CBD dissimilarities are generally as good or better.

We have also evaluated the hierarchies obtained using the Best of 50 heuristic presented in [25]. This method samples 50 random hierarchies and picks the best one using a cross-validation scheme. When the number of classes is not too high, this method presents very competitive results, outperforming RBD and CBD hierarchies in some cases. There are some relationships between the classes that these dissimilarities or the clustering algorithm are failing to measure and encode in a hierarchy. However, for datasets with many classes, Best of 50 underperforms compared to CBD or RBD hierarchies. The possible number of hierarchies increases very fast with the number of classes. The assumption that the sampled hierarchies are representative of the general distribution of all possible hierarchies can break as the number of classes increases.

CBD is a family of dissimilarities that depend on the classifier used. It is usually accepted in the literature that a simple classifier used to measure CBD is good enough to obtain a good hierarchy. We have proven that this is not necessarily the case. While we can get better hierarchies with the appropriate CBD dissimilarity, RBD can outperform too simple CBD hierarchies.

We have analyzed the trade-off between the complexity of the CBD classifier and the base classifiers trained on the hierarchy's nodes. Our results show that the hierarchy's quality is more relevant for the weaker base classifiers than for the more powerful ones. An interesting result is how hierarchies using simpler CBD classifiers outperform the ones using complex CBD classifiers when the base classifier is weak. This opens up interesting research lines into understanding the synergies between the quality of the extracted hierarchies and the power of the base classifiers used for HMC.

Throughout this paper, we have evaluated the quality of the hierarchies based on the results of a classifier trained on them. This evaluation depends on the particular characteristics of the datasets, such as the number of classes and the intrinsic complexity of the data. But it also depends critically on the learning algorithm used for the base classifiers. This means that special care needs to be taken when evaluating methods to extract hierarchies across datasets. Statistical tests like Friedman's could change their result if, for example, we use datasets with few classes and a powerful base learner or if we use datasets with many classes and weak learners.

Acknowledgments

This work was supported by "Stiftelsen för kunskaps- och kompetensutveckling".

References

[1] M. Aly, Survey on multiclass classification methods, Neural Netw 19 (2005) 1–9. doi:10.1.1.175.107.

[2] S. Bengio, J. Weston, D. Grangier, Label embedding trees for large multi-class tasks, Adv Neural Inf Process Syst 23 (1) (2010) 163–171. doi:10.5555/2997189.2997208.

[3] R. P. Stanley, What is enumerative combinatorics?, in: Enumerative combinatorics, Springer, 1986, pp. 1–63. doi:10.1007/978-1-4615-9763-6_1.

[4] V. Vural, J. G. Dy, A hierarchical method for multi-class support vector machines, ICML '04, ACM, 2004, pp. 105–. doi:10.1145/1015330.1015427.

[5] D. Silva-palacios, C. Ferri, M. J. Ramírez-quintana, Probabilistic class hierarchies for multiclass classification, Journal of Computational Science (2018) 1–10doi:https://doi.org/10.1016/j.jocs.2018.01.006.

[6] T. Li, S. Zhu, M. Ogihara, Hierarchical document classification using automatically generated hierarchy, J. Intell. Inf. Syst. 29 (2) (2007) 211–230. doi:10.1007/s10844-006-0019-7.

[7] S. Godbole, Exploiting confusion matrices for automatic generation of topic hierarchies and scaling up multi-way classifiers, Progress Rep., IIT Bombay (2002) 17.

[8] J. Weston, C. Watkins, Multi-class support vector machines, Tech. rep., Citeseer (1998).

[9] Y. Lee, Y. Lin, G. Wahba, Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data, Journal of the American Statistical Association 99 (465) (2004) 67–81. doi:10.1198/016214504000000098.

[10] J. Engel, Polytomous logistic regression, Stat. Neerl. 42 (4) (1988) 233–252. doi:10.1111/j.1467-9574.1988.tb01238.x.

[11] T. Hastie, R. Tibshirani, Classification by pairwise coupling, in: Advances in neural information processing systems, 1998, pp. 507–513. doi:10.1214/aos/1028144844.

[12] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes, Pattern Recognition 44 (8) (2011) 1761–1776. doi:10.1016/j.patcog.2011.01.017.

[13] R. Rifkin, A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. 5 (2004) 101–141.

[14] W. Liu, I. W. Tsang, K.-R. Müller, An easy-to-hard learning paradigm for multiple classes and multiple labels, J. Mach. Learn. Res. 18 (1) (2017) 3300–3337. doi:10.5555/3122009.3176838.

[15] R. Babbar, I. Partalas, E. Gaussier, M. R. Amini, On flat versus hierarchical classification in large-scale taxonomies, in: Advances in Neural Information Processing Systems 26, Curran Associates, Inc., 2013, pp. 1824–1832. doi:10.5555/2999792.2999816.

[16] Y. Qu, L. Lin, F. Shen, C. Lu, Y. Wu, Y. Xie, D. Tao, Joint hierarchical category structure learning and large-scale image classification, IEEE Transactions on Image Processing 26 (9) (2017) 4331–4346. doi:10.1109/TIP.2016.2615423.

[17] C. N. Silla, A. A. Freitas, A survey of hierarchical classification across different application domains, Data Min Knowl Discov 22 (1-2) (2011) 31–72. doi:10.1007/s10618-010-0175-9.

[18] Y. Chen, M. M. Crawford, J. Ghosh, Integrating support vector machines in a hierarchical output space decomposition framework, in: IGARSS 2004, Vol. 2, 2004, pp. 949–952 vol.2. doi:10.1109/IGARSS.2004.1368565.

[19] R. Babbar, I. Partalas, E. Gaussier, M.-R. Amini, C. Amblard, Learning taxonomy adaptation in large-scale classification, J. Mach. Learn. Res. 17 (1) (2016) 3350–3386. doi:10.5555/2946645.3007051.

[20] A. Naik, H. Rangwala, Improving large-scale hierarchical classification by rewiring: a data-driven filter based approach, J. Intell. Inf. Syst. 52 (1) (2019) 141–164. doi:10.1007/s10844-018-0509-4.

[21] X. Li, Y. Zhou, Y. Zhou, W. Wang, Mmf: multi-task multi-structure fusion for hierarchical image classification, in: ICANN, Springer, 2021, pp. 61–73. doi:10.1007/978-3-030-86380-7_6.

[22] F. Morin, Y. Bengio, Hierarchical probabilistic neural network language model., in: Aistats, Vol. 5, Citeseer, 2005, pp. 246–252.

[23] A. Mnih, G. E. Hinton, A scalable hierarchical distributed language model, in: Advances in neural information processing systems, 2009, pp. 1081–1088. doi:10.5555/2981780.2981915.

[24] E. Frank, S. Kramer, Ensembles of nested dichotomies for multi-class problems, ICML '04, ACM, 2004, p. 39. doi:10.1145/1015330.1015363.

[25] V. Melnikov, E. Hüllermeier, On the effectiveness of heuristics for learning nested dichotomies: an empirical analysis, Machine Learning 107 (8-10) (2018) 1537–1560. doi:10.1007/s10994-018-5733-1.

[26] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, M. Varma, Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising, in: Proc. 2018 World Wide Web Conference, 2018, pp. 993–1002. doi:10.1145/3178876.3185998.

[27] S. Khandagale, H. Xiao, R. Babbar, Bonsai: diverse and shallow trees for extreme multi-label classification, Machine Learning 109 (11) (2020) 2099–2119. doi:10.1007/s10994-020-05888-2.

[28] B. Larsen, C. Aone, Fast and effective text mining using linear-time document clustering, in: Proc.5th ACM SIGKDD, KDD '99, ACM, 1999, pp. 16–22. doi:10.1145/312129.312186.

[29] R. K. Sevakula, N. K. Verma, Balanced binary search tree multiclass decomposition method with possible non-outliers, SN Applied Sciences 2 (6) (2020) 1–15. doi:10.1007/s42452-020-2853-6.

[30] S. Chennupati, S. Sah, S. Nooka, R. Ptucha, Hierarchical Decomposition of Large Deep Networks, Electronic Imaging 2016 (19) (2016) 1–6.

[31] G. Tsoumakas, I. Katakis, I. Vlahavas, Effective and efficient multilabel classification in domains with large number of labels, Proc. ECML/PKDD Workshop on Mining Multidimensional Data (MMD'08) (2008) 30–44doi:10.1007/978-3-642-12837-0_11.

[32] C. Nadeau, Y. Bengio, Inference for the generalization error, Machine Learning 52 (3) (2003) 239–281. doi:10.1023/A:1024068626366.

[33] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30. doi:10.5555/1248547.1248548.

[34] O. Asian, O. T. Yildiz, E. Alpaydin, Calculating the vc-dimension of decision trees, in: 24th International Symposium on Computer and Information Sciences, 2009, pp. 193–198. doi:10.1109/ISCIS.2009.5291847.