Tree Learning: Optimal Sample Complexity and Algorithms

Dmitrii Avdiukhin[‡1], Grigory Yaroslavtsev[‡2], Danny Vainstein[‡3], Orr Fischer[‡4], Sauman Das[5], Faraz Mirza[5]

[1] Indiana University, Department of Computer Science
[2] George Mason University, Department of Computer Science [*]
[3] Tel-Aviv University, Blavatnik School of Computer Science
[4] Weizmann Institute of Science, Department of Computer Science and Applied Mathematics [†]
[5] Thomas Jefferson High School for Science and Technology
davdyukh@iu.edu, orr.fischer@weizmann.ac.il, dannyvainstein@gmail.com, grigory@grigory.us, 2023sdas@tjhsst.edu, 2023fmirza@tjhsst.edu

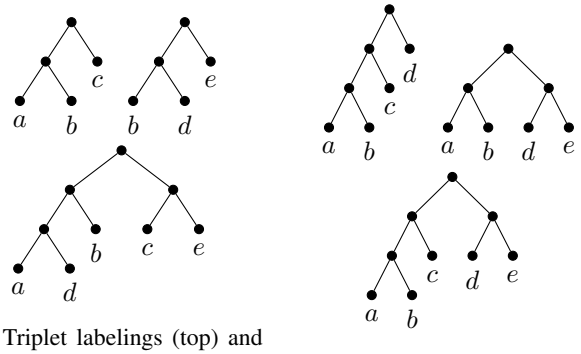
Abstract

We study the problem of learning a hierarchical tree representation of data from labeled samples, taken from an arbitrary (and possibly adversarial) distribution. Consider a collection of data tuples labeled according to their hierarchical structure. The smallest number of such tuples required in order to be able to accurately label subsequent tuples is of interest for data collection in machine learning. We present optimal sample complexity bounds for this problem in several learning settings, including (agnostic) PAC learning and online learning. Our results are based on tight bounds of the Natarajan and Littlestone dimensions of the associated problem. The corresponding tree classifiers can be constructed efficiently in near-linear time.

1 Introduction

The algorithmic problem of constructing hierarchical data representations has been of major importance for many decades, due to its applications in statistics (Ward Jr 1963; Gower and Ross 1969), entomology (Michener and Sokal 1957), plant biology (Sorensen 1948), genomics (Eisen et al. 1998) and other domains. Efficient collection of labeled data is a problem of key importance for construction of hierarchical data representations. In this paper we consider the problem of constructing tree representations of data from labeled samples, focusing on understanding the optimal number of samples required for this task.

The most basic type of a label that allows one to construct a tree representation consists of a triplet of points (x, y, z) labeled according to the induced hierarchical structure within the triplet. For example, given images of a cat, a dog, and a plane, the label would describe the cat and the dog as being more similar to each other than to the plane. This "odd one out" type of label is the simplest one to collect in a crowdsourcing setting. In this paper, we focus on understanding the number of such labeled samples required in order to construct a tree representation of the underlying data, which enables one to accurately predict subsequent labels in the future. This is then further generalized to include larger labeled subsets of data. Figure 1 shows examples of possible trees consistent with certain tuple labelings.



(a) Triplet labelings (top) and a tree consistent with the labelings (bottom)

(b) 4-tuple labelings and a tree consistent with the labelings

Figure 1: An example for the triplet and k-tuple setting, where two labeled queries and a tree consistent with the queries are shown. The tree satisfies all hierarchical relations of both labeled queries. For example, in Figure 1a, the lowest common ancestor of a and b (denoted $\mathrm{LCA}(a, b)$) is below $\mathrm{LCA}(a, c)$ and $\mathrm{LCA}(b, c)$, as in its first labeled query.

1.1 Our Results

Let n be the number of points in the dataset. We present results in two settings: PAC learning and online learning. In both cases, our objective is to build a classifier that given access to labeled tuples can predict labels for the subsequent tuples with high probability.

PAC learning In the *Probably Approximately Correct (PAC) learning* setting (Valiant 1984), the tuples are gen-

[*]Supported by NSF CCF-1657477 and Facebook Faculty Award.

[†]This project is partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 949083).

[‡]These authors contributed equally to this work

erated from a fixed unknown distribution $\mathcal{D}$. We distinguish between realizable and agnostic settings. In the *realizable* case, it is assumed that the labels are consistent (there exists a tree that respects all observed labels), and the goal is to predict their labels with probability at least $1-\epsilon$, i.e. achieve the error rate at most ϵ, while providing this guarantee with overall probability at least $1-\delta$. In the *agnostic* PAC-learning case, such a tree does not necessarily exist, and the goal is to predict the labels with probability at least $1-\epsilon-\epsilon_{T^*}$, where ϵ_{T^*} is the smallest prediction error among all trees.

In the PAC-learning setting, our main result, which also holds for non-binary trees, is as follows:

Theorem 1.1 (Informal version of Theorem 4.4)**.** *The sample complexity of (ϵ,δ)-PAC-learning hierarchically labeled tuples is $\Theta(\frac{n}{\epsilon}\cdot\text{polylog}(\frac{1}{\epsilon},\frac{1}{\delta}))$ in the realizable settings and $\Theta(\frac{n}{\epsilon^2}\cdot\text{polylog}(\frac{1}{\epsilon},\frac{1}{\delta}))$ in the agnostic setting.*

Online learning In the *online learning* setting, we have a stream of labeled tuples that can arrive in an adversarial order. The accuracy is evaluated sequentially, counting the overall number of mistakes made by the algorithm throughout the sequence. In the realizable setting, it is assumed that the tree generating the labels is fixed in advance and all labels seen by the algorithm are consistent with this tree. In the agnostic setting, this is no longer assumed, similarly to the PAC-learning scenario. Our main result in this setting is:

Theorem 1.2 (Informal)**.** *On a sequence of triplets of length T, the number of mistakes made by an online learning algorithm for predicting their hierarchical structure is $\Theta(n\log n)$ in the realizable case. In the agnostic case, the number of mistakes is at most $O(\sqrt{Tn\log n\log(T)})$ and at least $\Omega(\sqrt{Tn\log n})$ larger than the optimum number of mistakes achieved by any tree.*

A summary of our results is shown in Table 1 (which hold for triplet queries and for k-tuples for constant k).

	Labeled k-tuples (for constant k)
PAC realizable*	$\Theta(\frac{n}{\epsilon}\cdot\text{polylog}(\frac{1}{\epsilon},\frac{1}{\delta}))$
PAC agnostic*	$\Theta(\frac{n}{\epsilon^2}\cdot\text{polylog}(\frac{1}{\epsilon},\frac{1}{\delta}))$
Online realizable**	$\Theta(n\log n)$
Online agnostic**	$O(\sqrt{Tn\log n\log(T)})+\text{OPT}$, $\Omega(\sqrt{Tn\log n})+\text{OPT}$

Table 1: Summary of sample complexities in each setting

1.2 Our Techniques

It is known (e.g. Daniely et al. (2015)) that PAC-learning and online learning complexities are nearly tightly described in terms of the Natarajan (Natarajan 1989) and Littlestone (Littlestone 1987) dimensions respectively. Hence, in this paper, we focus on tight bounds on the dimensions. Since the exact definitions are technical, we refer the reader to Definition 2.8 for the precise definition of Natarajan dimension and the full version for the Littlestone dimension.

Natarajan dimension In the PAC-learning setting, VC-dimension (Vapnik and Chervonenkis 1971) can be used to capture query complexity for binary classification problems. The problem of labeling tuples considered in this paper corresponds to multiclass classification since even for triplets there are three "odd one out" labels possible. A generalization of VC-dimension which captures this scenario is the Natarajan dimension (Natarajan 1989). While it can be shown via a simple probabilistic argument that this dimension is $O(n\log n)$, we argue that the exact bound is in fact linear in the number of points, which implies Theorem 1.1.

In this section, we focus on triplet constraints; in the full version, we show how to reduce k-tuples to triplets. In order to bound the sample complexity, we use a version of the definition of Natarajan dimension adapted to our setting. Given a triple $\Delta=(a,b,c)$, we denote an "odd one out" constraint separating c from a and b as $[a,b\,|\,c]$. For each triplet, there are 3 possible constraints, i.e. 3 possible labels. We are now ready to define the notion of Natarajan Shattering.

Definition 1.3 (Natarajan Shattering)**.** *Let $S=\{(a_1,b_1,c_1),\dots(a_k,b_k,c_k)\}$ be a set of triples of points. We say S is* Natarajan shattered *if for every triple $(a_i,b_i,c_i)\in S$ there exist two distinct constraints $f_1(a_i,b_i,c_i)$, $f_2(a_i,b_i,c_i)$ (e.g. $f_1(a_i,b_i,c_i)=[a_i,c_i\,|\,b_i], f_2(a_i,b_i,c_i)=[a_i,b_i\,|\,c_i]$) on this triple, such that for every subset $R\subseteq S$ there is a hierarchical tree T for which:*

- *For every $\Delta\in R$ it holds that T satisfies $f_1(\Delta)$,*
- *For every $\Delta\in S\setminus R$ it holds that T satisfies $f_2(\Delta)$.*

The Natarajan dimension of the hierarchically labeled tuples is defined as the size of the largest cardinality of a set which can be Natarajan shattered.

Intuitively, for every triplet in S, we fix 2 labels out of 3 possible. The set is shattered if all possible $2^{|S|}$ combinations of these labels are realizable by the hypothesis space.

Example. Consider a point set, consisting of 4 points: a,b,c,d. Consider two sets of triples:

- $S_1=\{(a,b,c),(b,c,d)\}$
- $S_2=\{(a,b,c),(b,c,d),(c,d,a)\}$.

For S_1 we can select $f_1(a,b,c)=[a,b\,|\,c]$, $f_2(a,b,c)=[b,c\,|\,a]$ and $f_1(b,c,d)=[b,c\,|\,d], f_2(b,c,d)=[c,d\,|\,b]$. Then for any of the four subsets of S_1 we can create a tree consistent with the f_1 choices on the subset and consistent with the f_2 choices otherwise. We show all four resulting trees on Figure 2. It is less straightforward, however, to check that for the set S_2, no possible selection of f_1,f_2 can be used to satisfy the definition of Natarajan shattering. This follows from our key technical result.

Theorem 1.4. *The Natarajan dimension of hierarchically labeled triplets on n elements is $n-2$*

The proof is based on identifying subsets with a certain property, which we refer to as *closed sets* (Definition 3.3) which prevent construction of a consistent tree. We then further characterize these sets in terms of easier to define *critical sets* (Definition 3.5). We prove that for $n-1$ elements a critical set must exist, which, combined with the fact that

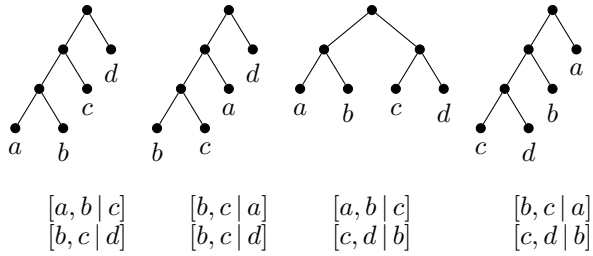


Figure 2: Shattering a set of two triplets $\{(a, b, c), (b, c, d)\}$.

a set of consistent $n - 2$ pairs of constraints exists, gives a tight bound for the Natarajan dimension.

Littlestone dimension We next show a tight bound on the Littlestone dimension. For the purpose of the introduction, it is convenient to view the Littlestone dimension in our setting as follows. Assume that we have two players Alice and Bob who play the following game for t iterations. Initially, there is an empty set of constraints C. In every iteration, Alice passes Bob a triplet and two distinct elements from this triplet. E.g., suppose that the triplet is (a, b, c) and the chosen elements are a, b. Bob then chooses one of the constraints where one of the chosen elements is the "odd one out", i.e. $[b, c \mid a]$ or $[a, c \mid b]$, and adds it to C. The Littlestone dimension is the maximum number of rounds t for which Alice has a strategy that results in C admitting a tree that satisfies all constraints in C, regardless of Bob's strategy.

Theorem 1.5. *The Littlestone dimension of hierarchically labeled triples is* $\Theta(n \log n)$.

To prove the lower bound on the Littlestone dimension, we need to describe Alice's strategy of producing an adaptive sequence of queries of size $\Omega(n \log n)$ such that for any choice of Bob's query answers there is a hierarchical tree consistent with the query answers at the end of the sequence. The sequence we construct is intuitively a sort tournament on V, for which there is always some ordering $v_1, \ldots, v_n$ on the point set V such that if we place the points in order in a hierarchical tree whose internal nodes are shaped like a path, this tree satisfies all constraints.

Non-binary trees When all constraints are of type "odd one out", it suffices to consider binary trees: if a non-binary tree is consistent with the constraints, then its binarization (i.e. we replace a node with ℓ children with an arbitrary binary tree on these children) is also consistent. However, one may generalize the problem to non-binary trees by considering constraints of the form *"points i, j and k must be split simultaneously"*. For the PAC-learning setting, in the full version, we resolve this setting by extending Theorem 1.1 appropriately. In the online learning setting, Theorem 1.2 generalizes as well – the same lower bound holds for the non-binary setting and the upper bound holds trivially.

1.3 Related Work

An early work by Aho et al. (1981) shows that given access to m consistently labeled triples on n vertices, the tree satisfying them can be constructed in $O(mn)$ time. This was improved to $O(m \log^2 n)$ time using the techniques in Henzinger, King, and Warnow (1999); Holm, de Lichtenberg, and Thorup (2001), and to $O(\min(n^2, m \log n) + \sqrt{mn} \log^{2.5}(n))$ by Thorup (1999).

Compared with our settings where we are given queries from some distribution and aim for a bounded error rate on unseen triplets, a related line of work considers the problem of exactly reconstructing the entire tree using adaptively or non-adaptively chosen triplet queries (see e.g. Kannan, Lawler, and Warnow (1996); Emamjomeh-Zadeh and Kempe (2018)). For non-adaptive queries, a lower bound of $\Omega(n^3)$ queries precludes any non-trivial results (Emamjomeh-Zadeh and Kempe 2018). For adaptive queries, $\Theta(n \log n)$ consistent queries and running time are necessary and sufficient for the construction of the tree (Kannan, Lawler, and Warnow 1996). This can be extended to handle a mix of independently correct labels and adversarial noise, resulting in similar bounds for constant noise levels (Emamjomeh-Zadeh and Kempe 2018).

In the case when labeled data is allowed to be inconsistent, minimizing the number of disagreements with labeled triplets is known to be hard to approximate (Chester, Dondi, and Wirth 2013). Recent work Chatziafratis, Mahdian, and Ahmadian (2021) gives algorithms for maximizing the number of agreements between the tree and the labeled data. In a related line of work, results are known for optimizing certain objectives while respecting triplet constraints (Chatziafratis, Niazadeh, and Charikar 2018).

2 Preliminaries

We begin with a formal definition of a hierarchical tree.

Definition 2.1. *Given a set of points V, we define a* hierarchical tree *T as a binary tree such that V is bijectively mapped on the leaves of T.*

With a slight abuse of notation, for a fixed tree T, we treat elements of V as the leaves of T. For two leaves i, j and a hierarchical tree T, we denote the least common ancestor, i.e. the internal node corresponding to the smallest subtree containing both i and j, as $\mathrm{LCA}_T(i, j)$.

Triplet Constraints We are interested in satisfying structural constraints on the elements of V. We consider different types of constraints: We first consider constraints on three elements (triplet constraints), then generalize our result to an arbitrary number of elements, and then for the case when the trees are not necessarily binary.

Definition 2.2. *For a hierarchical tree T and a triplet of distinct points $(a, b, c) \subseteq V$[‡] we say that T* satisfies the constraint *denoted $[a, b|c]$ if T cuts c from a and b, i.e. $\mathrm{LCA}_T(a, c) = \mathrm{LCA}_T(b, c)$. We call such a constraint an orientation of triplet (a, b, c).*

Definition 2.3. *For a triplet $t = (a, b, c) \subseteq V$ we say that an* orientation *of the triplet is a constraint over the nodes a, b, c, i.e. $[a, b \mid c]$, $[a, c \mid b]$, or $[b, c \mid a]$. We further denote it as $\overrightarrow{t}$. Given a set of triplets $\Delta = \{(a_i, b_i, c_i)\}_i$ over V, we*

[‡]Since the original order inside the tuples doesn't matter, with a slight abuse of notation we treat tuples (e.g. triplets) as sets.

say that an orientation of Δ is a set of orientations over each triplet in Δ. We similarly denote it as $\overrightarrow{\Delta}$.

That is, orientation is a particular choice of constraint(s) generated based on a given triplet(s). In order to apply our results in the PAC-learning setting, we are interested in the sets of constraints that can not be satisfied:

Definition 2.4. *We define a set of constraints as* contradictory *if there is no hierarchical tree that satisfies all constraints in the set.*

k-tuple constraints Any orientation of a triplet uniquely defines a tree on this triplet. We can use this intuition to define constraints on k elements.

Definition 2.5. *Let $A = (a_1, \dots, a_k)$ be a k-tuple, and let T_A be a binary tree with leaves $a_1, \dots, a_k$. Then we say that a binary tree T* satisfies constraint *T_A if any triplet constraint $[a_i, a_j \mid a_t]$ satisfied by T_A is also satisfied by T.*

Definition 2.6. *For a given k-tuple $(a_1, \dots, a_k) \subseteq V$, an orientation of the tuple is any constraint on $a_1, \dots, a_k$.*

Non-binary trees When non-binary trees are allowed, some nodes i, j, k can be separated at the same time, i.e. $\mathrm{LCA}_T(a, b) = \mathrm{LCA}_T(a, c) = \mathrm{LCA}_T(b, c)$. We denote such case as $[a|b|c]$. Similarly, we allow k-tuple constraints where multiple elements can be separated at the same time.

Sample complexity Let $\mathcal{D}$ be the distribution on $X \times Y$, where X is the set of inputs and Y is the set of labels. Let $H \subseteq Y^X$ be a hypothesis space[‡]. For a given $h \in H$, we define the error rate as $\mathrm{err}_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$. Let $h^*_{\mathcal{D}} = \mathrm{argmin}_{h\in H} \mathrm{err}_{\mathcal{D}}(h)$. We say that the settings are realizable if $\mathrm{err}_{\mathcal{D}}(h^*_{\mathcal{D}}) = 0$; otherwise, we say that the settings are agnostic.

Definition 2.7 (Sample complexity)**.** *We define* sample complexity *$m_H(\varepsilon, \delta)$ as the minimum number of samples, such that there exists a predictor that, for any distribution $\mathcal{D}$, given $m_H(\varepsilon, \delta)$ samples from $\mathcal{D}$, achieves error rate at most $\mathrm{err}_{\mathcal{D}}(h^*_{\mathcal{D}}) + \epsilon$ with probability at least δ. We denote the sample complexity as $m^r_H(\varepsilon, \delta)$ for the realizable case and $m^a_H(\varepsilon, \delta)$ for the agnostic case.*

For the binary classification task ($|Y| = 2$), the sample complexity can be estimated using VC-dimension (Vapnik and Chervonenkis 1971). Its analog for the multi-class settings is the Natarajan dimension.

Definition 2.8 (Natarajan dimension (Natarajan 1989))**.** *Let X be the set of inputs, Y be the set of labels, and let $H \subseteq Y^X$ be a hypothesis class. Then $S \subseteq X$ is N-shattered by H if there exist $f_1, f_2 \colon X \to Y$ such that $f_1(x) \neq f_2(x)$ for all $x \in S$ and for every $T \subseteq S$ there exists $g \in H$ such that:*

$$g(x) = f_1(x) \text{ for } x \in T \text{ and } g(x) = f_2(x) \text{ for } x \notin T$$

The Natarajan dimension $\mathrm{NDim}(H)$ of H is the maximum size of an N-shattered set.

In our case, the hypothesis space is defined by the set of constraints induced by all possible hierarchical trees:

[‡] Y^X is the set of all functions $X \to Y$

Definition 2.9. *Given a set V be a set and an integer $k \geq 3$, let X be the set of k-tuples on V and Y be the set of orientations of the k-tuples. Then we use $H_k(V)$ to denote a set of mappings $X \to Y$ such that for each mapping there exists a hierarchical tree where each k-tuple is oriented according to the mapping.*

The following result gives a tight estimate of the sample complexity based on the Natarajan dimension of the problem

Lemma 2.10 (Ben David et al. (1995))**.** *If $|Y| < \infty$, then for the sample complexity $m^r_H(\varepsilon, \delta)$, the following holds for some universal constants C_1 and C_2 for the realizable case:*

$$m^r_H(\varepsilon, \delta) \geq C_1 \frac{\mathrm{NDim}(H) + \log \frac{1}{\delta}}{\varepsilon}$$

$$m^r_H(\varepsilon, \delta) \leq C_2 \frac{\mathrm{NDim}(H) \log \frac{1}{\varepsilon} \log |Y| + \log \frac{1}{\delta}}{\varepsilon}$$

For the agnostic case:

$$m^a_H(\varepsilon, \delta) \geq C_1 \frac{\mathrm{NDim}(H) + \log \frac{1}{\delta}}{\varepsilon^2}$$

$$m^a_H(\varepsilon, \delta) \leq C_2 \frac{\mathrm{NDim}(H) \log |Y| + \log \frac{1}{\delta}}{\varepsilon^2}$$

3 Contradictory Orientations

As described in Section 1.2, the key component in our analysis is a tight bound on the Natarajan dimension. To find it, we first consider a simpler question: "*for a given n, what is the minimum m, such that for any set of m triplets on $[n]$ there exists a contradictory orientation of these triplets?*" Recall that the definition of the Natarajan dimension restricts the candidate orientations so that every triplet has only two allowed orientations, giving 2^m possible label combinations. In this section, we answer this question without such a restriction, i.e. we check whether all 3^m label combinations are possible[‡], and handle the restriction in the next section. In what follows we prove that $m = n - 1$.

Theorem 3.1. *For any $n > 2$, any set of triplets of size at least $n - 1$ on these points has a contradictory orientation.*

3.1 Closed Set

We can think about every constraint $[a, b \mid c]$ as an edge (a, b) corresponding to a separate vertex c, and we say that $[a, b \mid c]$ "generates" edge (a, b). Clearly, if a hierarchical tree satisfies constraint $[a, b \mid c]$, then there must exist a tree node such that one child's subtree contains a and b, and another child's subtree contains c. Hence, the tree violates the constraint if it cuts edge (a, b) before first separating c from a and b.

When building a tree in a top-down manner, a node corresponds to a set of elements $S \subseteq V$, which we want to partition. When partitioning S, only triplets lying entirely in S (which we call *induced* by S) can lead to a contradiction.

Definition 3.2 (Induced triplets/constraints)**.** *Let Δ be a set of triplets over V and let $S \subseteq V$. We define the set*

[‡] One may think of this as a naïve generalization of VC shattering, although not directly applicable to PAC learning

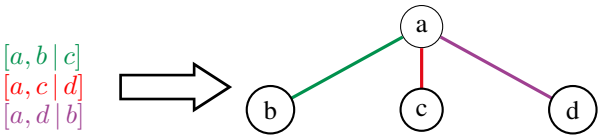


Figure 3: Let $S = \{a, b, c, d\}$ and $\overrightarrow{\Delta} = \{[a, b \mid c], [a, c \mid d], [a, d \mid b]\}$. Since S is connected by edges generated by $\overrightarrow{\Delta}|_S$, the constraints are contradictory. When first splitting S, we must cut an edge, which leads to a contradiction: cutting edge (a, b) violates $[a, b \mid c]$, cutting (a, c) violates $[a, c \mid d]$, and cutting (a, d) violates $[a, d \mid b]$.

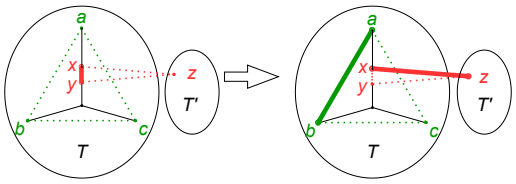


Figure 4: A simple case from Theorem 8: selecting orientation $[a, b \mid c]$ allows one to reorient any edge on the path from a to b. In this case, on this path, there is an edge (x, y) generated by constraint $[x, y \mid z]$, where z belongs to another spanning tree. Reorienting $[x, y \mid z]$ connects the trees

of triplets in Δ which are induced *by S as $\Delta|_S = \{t \in \Delta \mid t \subseteq S\}$ (i.e., the set of all triplets that lie entirely within S). Similarly, given a set of constraints C, we define $C|_S = \{[a, b \mid c] \in C \mid a, b, c \in S\}$.*

The above reasoning implies that it's impossible to split a tree node with set S if it's connected by the edges generated by the constraints induced by S, since splitting S would cut at least one edge (see Figure 3 for example). We will show that the existence of the hierarchical tree is determined by the existence of such set S. For fixed Δ, if it's possible to orient Δ so that $\overrightarrow{\Delta}|_S$ connects S, we call S a closed set.

Definition 3.3 (Closed set)**.** *Let Δ be a set of triplets over V. We say that a set $S \subseteq V$ is* closed *w.r.t. Δ if there exists an orientation of Δ denoted as $\overrightarrow{\Delta}$, such that $E|_S = \{(a, b) \mid [a, b \mid c] \in \overrightarrow{\Delta}|_S\}$ connects S.*

The next Lemma shows that a contradictory orientation exists iff a closed subset exists. The proof is in the full version.

Lemma 3.4. *A set of triplets Δ over V allows for a contradictory orientation if and only if there exists a set $S \subseteq V$ that is closed w.r.t. Δ.*

The proof idea is the following. If the set is not connected by the edges generated by $\overrightarrow{\Delta}|_S$, then by separating connected components we don't cut any edge, and hence don't violate any constraints. On the other hand, if the set is connected, when we first split S, we cut at least one edge (a, b) such that $[a, b \mid c] \in \overrightarrow{\Delta}|_S$. But this violates the constraint, as it requires c to be first separated from a and b.

3.2 Critical Set

We next show that, when Δ is sufficiently large, there exists a set S such that we can always construct a contradictory orientation of $\Delta|_S$. We call such a set a critical set.

Definition 3.5 (Critical set)**.** *Let Δ be a set of triplets over V of size $|\Delta| \geq |V| - 1$. We say that set $S \subseteq V$ is* critical *w.r.t. Δ if it satisfies the following conditions:*

- *S induces at least $|S| - 1$ triplets, i.e. $\left|\Delta|_S\right| \geq |S| - 1$,*
- *Among all such sets, S has the minimal cardinality.*

Note that this is well defined since V satisfies the condition: $|\Delta| = \left|\Delta|_V\right| \geq |V| - 1$. Intuitively, $|S| - 1$ is the minimum possible number of edges which can connect S. The surprising fact is that this condition suffices, which leads to the main result of this section (Theorem 3.1).

Theorem 3.6. *Let Δ be a set of triplets over V of size $|\Delta| \geq |V| - 1$. Then any critical set w.r.t. Δ is closed w.r.t. Δ.*

We present the full proof of Theorem 3.6 in the full version. The proof relies on the operation which we call *reorientation*: given constraint $[a, b \mid c]$, we change it to either $[a, c \mid b]$ or $[b, c \mid a]$, which respectively changes the generated edge from (a, b) to either (a, c) or (b, c).

The proof is by contradiction: for a critical set S, we assume that no orientation of $\Delta|_S$ generates edges connecting S. We consider the orientation whose generated edges result in the smallest number of connected components. We show that performing certain reorientations can reduce the number of connected components, leading to a contradiction.

Since the edges generated by the orientation don't connect S, there must exist an "unused" triplet (a, b, c), such that adding or removing its edges (a, b), (a, c) and (b, c) doesn't change the number of connected components. All of a, b, c belong to the same tree T in the spanning forest of S (otherwise, we could orient (a, b, c) so that it connects two trees, reducing the number of connected components). By orienting this triplet as, for example, $[a, b \mid c]$, we can reorient any edge on the path from a to b in T without disconnecting vertices in T. If there exists a reorientation that connects T to another tree in the spanning forest, using this reorientation we decrease the number of connected components, as shown in Figure 4. Such a reorientation might not always be immediately available, but we show that it's always possible to build a chain of reorientations so that the number of connected components decreases.

k-tuples In the full version, we show that any set of k-tuples of size at least $\left\lceil \frac{|V|-1}{k-2} \right\rceil$ has a contradictory specification and that the bound is tight for k-tuples[‡]. The main idea is that, for the sake of analysis, we can replace a k-tuple with $k - 2$ "independent" triplets, meaning that all 3^{k-2} orientations of these triplets are not contradictory. Namely, all the triplets share two elements and differ in the third one.

Non-binary trees When constraints of form $[a, b \mid c]$ are allowed, after adjusting the definitions, the overall idea is

[‡]As we show in the next section, the $1/(k-2)$ factor doesn't propagate to the sample complexity

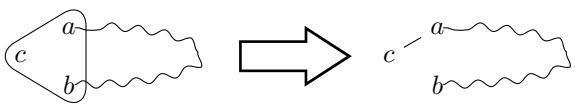


Figure 5: Given constraint $[a|b|c]$, when a and b are connected, we can also connect them to c

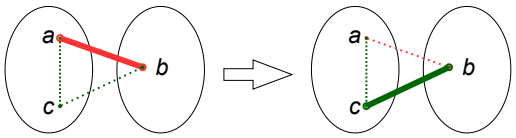


Figure 6: Theorem 4.2: when orientation $[a, b \mid c]$ is not allowed, we can reorient it while preserving connectivity

the same: a contradictory orientation exists iff a closed set exists, and any critical set is closed. While the definition of a critical set doesn't change – it's a minimum-size set S with at least $|S| - 1$ induced edges – the definition of a closed set changes substantially, as described next.

As before, we have a set $\mathcal{E} = E|_S$ of edges induced by S, but now constraints of form $[a|b|c]$ may produce additional edges. The main idea is as follows: if e.g. a and b are already connected by $\mathcal{E}$, they can't be separated by the first cut. But by definition of $[a|b|c]$, c also can't be separated from a and b by the first cut. Since we perform the first cut based on the connectivity of the set, the fact that c can't be separated from a and b can be expressed by adding edge (a, c), i.e. $\mathcal{E} \leftarrow \mathcal{E} \cup \{(a, c)\}$, hence connecting a, b and c (Figure 5).

Now, we can say that S is *closed* w.r.t. Δ if there exists an orientation of Δ such that S is connected after performing all such possible operations. Similarly to the binary-tree case, the existence of such a set implies a contradictory orientation. This intuition is formalized in the full version.

Obviously, critical sets are closed: by only using constraints of type $[a, b \mid c]$, we can use Theorem 3.6 directly. However, constraints of type $[a|b|c]$ result in an additional option in the definition of N-shattering (we have to choose 2 labels out of 4 instead of 3), and hence the Natarajan dimension can potentially increase. In the next section, we show that this is not the case for our problem.

4 PAC-Learning and Natarajan Dimension

In this section, we present tight sample complexity bounds for PAC learning for k-tuple constraints. Recall that the set $\Delta = \{t_i\}_{i=1}^n$ of k-tuples is N-shattered if for every k-tuple t_i we can select two different orientations $\overrightarrow{t}_i^{(1)}$ and $\overrightarrow{t}_i^{(2)}$ such that every combination of orientations of different k-tuples is not contradictory, i.e. for any $f : [n] \to [2]$ the orientation $\{\overrightarrow{t}_i^{(f(i))}\}_{i=1}^n$ is not contradictory. Given the Natarajan dimension, Lemma 2.10 gives the tight bound on sample complexity up to the factor $O(\frac{1}{\varepsilon} \log |Y|)$, and $|Y|$ is constant for constant k. We first lower-bound the Natarajan dimension:

Lemma 4.1. *For any V and $k \geq 3$, we have* $\mathrm{NDim}(H_k(V)) \geq |V| - k + 1$.

Proof. Let A be an arbitrary subset of V of size $k - 1$, and let $B = V \setminus A$. We construct the set of k-tuples as $\Delta = \{A \cup \{b\} \mid b \in B\}$. Let T_A be an arbitrary hierarchical tree on A. For each $A \cup \{b\} \in \Delta$, we construct the orientations as follows: we replace a leaf a of T_A with a new node with two children: a and b. By choosing two different leaves, we obtain two different orientations of $A \cup \{b\}$, as required by the definition of N-shattering.

It's easy to check that Δ is N-shattered using these orientations: all elements from A are in agreement across all constraints since they all are oriented according to T_A, while every element from B participates in only one constraint, and hence can't lead to a contradiction. Therefore, $\mathrm{NDim}(H_k(V)) \geq |B| = |V| - k + 1$. □

Next, we upper-bound the Natarajan dimension for triplets using results from Section 3.

Theorem 4.2. $\mathrm{NDim}(H_3(V)) = |V| - 2$ *for any* V.

We provide the full proof in the full version. First, note that the result doesn't immediately follow from Theorem 3.6 since Theorem 3.6 finds a contradictory orientation among all possible 3^m orientations of m constraints. However, Natarajan shattering allows us to choose one of only two orientations of each triplet, i.e. it allows 2^m possible orientations. Hence, we need to handle the case when the contradicting orientation from Theorem 3.6 orients some triplet in a non-allowed way.

Let S be a critical set, $\overrightarrow{\Delta}|_S$ be its induced orientation connecting S, and $[a, b|c] \in \overrightarrow{\Delta}|_S$ be a non-allowed orientation of a triplet. Note that it means that both remaining orientations $[a, c|b]$ and $[b, c|a]$ are allowed. If removing edge (a, b) doesn't disconnect S, we reorient it arbitrarily. Otherwise, removing (a, b) separates S into two connected components. We reorient (a, b) in the following way: if c belongs to the same part as a, then we use edge (b, c); otherwise, we use edge (a, c). As shown in Figure 6, such reorientation preserves connectivity of S. Combining Lemma 4.1 and Theorem 4.2 yields the following corollary.

Corollary 4.3. *For any V and $k \geq 3$, we have* $|V| - k + 1 \leq \mathrm{NDim}(H_k(V)) \leq |V| - 2$.

Combined with Lemma 2.10, it gives our main result.

Theorem 4.4. *For constant k, the sample complexity of learning hierarchically labeled k-tuples, denoted by $m^r_{H_k}(\varepsilon, \delta)$ in the realizable setting and $m^a_{H_k}(\varepsilon, \delta)$ in the agnostic setting, is bounded by:*

$$C_1 \frac{n + \log \frac{1}{\delta}}{\varepsilon} \leq m^r_{H_k}(\varepsilon, \delta) \leq C_2 \frac{n \log \frac{1}{\varepsilon} + \log \frac{1}{\delta}}{\varepsilon}$$

$$C_1 \frac{n + \log \frac{1}{\delta}}{\varepsilon^2} \leq m^a_{H_k}(\varepsilon, \delta) \leq C_2 \frac{n + \log \frac{1}{\delta}}{\varepsilon^2}$$

In the full version, we prove that the identical result holds for the non-binary case. The proof idea is similar to that of Theorem 4.2, with the only change that we must handle the case when $[a|b|c]$ is one of the allowed constraints.

5 Experiments

In this section, we empirically verify our theoretical findings by evaluating the prediction accuracy of binary trees constructed from labeled triplets on the unlabeled triplets from the same distribution. We consider the binary realizable case, when the triplets are labeled according to a ground-truth binary tree, the binary agnostic case, and the non-binary realizable case.

We build a tree using a top-down algorithm. Every tree node corresponds to a set, and we build a graph described in Section 3.1. When possible, we separate the connected components, but in the agnostic case the graph might be connected, and we partition the graph using the minimum cut, which is known to achieve $O(n)$ approximation.

Datasets For the realizable case, we perform experiments on randomly generated trees and on ImageNet (Deng et al. 2009) hierarchy[‡]. For non-binary tree experiments (see the full version), we use the full hierarchy (48,860 points), while for binary tree experiments, we sample 256 leaves that induce a binary subtree.

For the agnostic case, we consider 2 datasets: 1) randomly sampled vectors from the uniform distribution on $[0,1]^{100}$ and 2) Spambase[‡] dataset containing feature vectors for 4601 different emails for the purpose of spam detection.

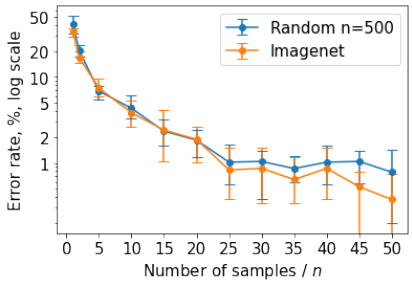
Binary Realizable case Given a ground-truth tree, we sample triplets from this tree uniformly at random. Since the orientations for these triplets are not contradictory, we can construct a tree and make predictions according to the tree. We first build a non-binary tree using the algorithm described in Aho et al. (1981), and then binarize it by replacing non-binary nodes with random binary trees on their children.

For this approach, Figure 7a shows the dependence of the error rate on k, where k is the ratio of the number of samples to the number of labels. The results imply that the error rate depends on k and is independent of the number of leaves or other properties of the ground-truth tree. From Theorem 4.4 we know that the sample complexity is roughly proportional to $\frac{n}{\varepsilon}$, and hence we expect $\varepsilon \cdot k = \frac{\varepsilon}{n} \cdot n_{\text{samples}}$ to be approximately constant. Figure 7b confirms this hypothesis since $\varepsilon \cdot k$ is approximately 0.4 for various values of n and k.

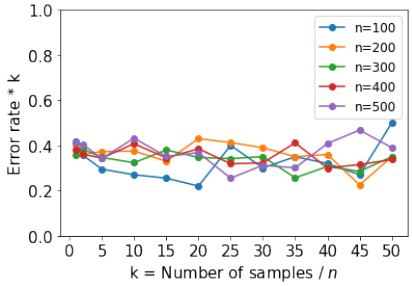
Agnostic case For this scenario, we assume that the input is vectors in the Euclidean space, and we generate triplet constraints as follows. Given a triplet of vectors (a, b, c), we create constraint $[a, b \mid c]$ if $\|a-b\| \leq \min(\|a-c\|, \|b-c\|)$. Such constraints can be contradictory if the dataset is not hierarchical, meaning that the setting is agnostic. Similarly to the realizable case, Figure 8 shows the dependence of the error rate on k, where k is the ratio of the number of samples to the number of labels. Since random vectors don't have any hierarchical structure, known samples don't provide sufficient information about the unseen samples, and hence the error rate is close to trivial $2/3$ regardless of k. On the other hand, since Spambase feature vectors have a hierarchical structure, they error rate on this dataset is significantly lower and slowly decreases with k.

[‡]https://github.com/waitwaitforget/ImageNet-Hierarchy-Visualization

[‡]UCI Machine Learning Repository: http://archive.ics.uci.edu/ml



(a) Prediction error depending on the number of samples for a random tree with $n \in \{100, 500\}$ nodes, and the ImageNet hierarchy. For each dataset, we average the results over 10 runs, and the error bars correspond to 10% and 90% quantiles.



(b) For random trees with n nodes, we show dependence of $k \cdot \text{err}$ on k, where k as the ratio of number of samples to n. As Theorem 4.4 indicates, this number is close to a constant for different n and k. Each data point shows the mean values over 10 runs.

Figure 7: Binary Realizable case

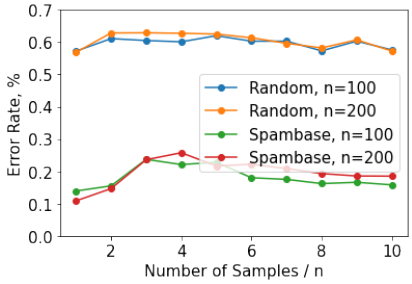


Figure 8: Agnostic case. Prediction error depending on the number of samples. We sample n random 100-dimensional vectors and a subset of the Spambase dataset of size n.

6 Conclusion

In this paper we give almost optimal bounds on the sample complexity of learning hierarchical tree representations of data from labeled tuples in both distributional (PAC-learning) and online case. Our experimental results confirm the convergence bounds predicted by the theory on trees generated from the ImageNet dataset.

References

Aho, A. V.; Sagiv, Y.; Szymanski, T. G.; and Ullman, J. D. 1981. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3): 405–421.

Ben David, S.; Cesabianchi, N.; Haussler, D.; and Long, P. M. 1995. Characterizations of learnability for classes of (0,..., n)-valued functions. *Journal of Computer and System Sciences*, 50(1): 74–86.

Chatziafratis, V.; Mahdian, M.; and Ahmadian, S. 2021. Maximizing Agreements for Ranking, Clustering and Hierarchical Clustering via MAX-CUT. In Banerjee, A.; and Fukumizu, K., eds., *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, 1657–1665. PMLR.

Chatziafratis, V.; Niazadeh, R.; and Charikar, M. 2018. Hierarchical Clustering with Structural Constraints. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 773–782. PMLR.

Chester, A.; Dondi, R.; and Wirth, A. 2013. Resolving Rooted Triplet Inconsistency by Dissolving Multigraphs. In Chan, T. H.; Lau, L. C.; and Trevisan, L., eds., *Theory and Applications of Models of Computation, 10th International Conference, TAMC 2013, Hong Kong, China, May 20-22, 2013. Proceedings*, volume 7876 of *Lecture Notes in Computer Science*, 260–271. Springer.

Daniely, A.; Sabato, S.; Ben-David, S.; and Shalev-Shwartz, S. 2015. Multiclass learnability and the ERM principle. *J. Mach. Learn. Res.*, 16: 2377–2404.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Eisen, M. B.; Spellman, P. T.; Brown, P. O.; and Botstein, D. 1998. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25): 14863–14868.

Emamjomeh-Zadeh, E.; and Kempe, D. 2018. Adaptive Hierarchical Clustering Using Ordinal Queries. In Czumaj, A., ed., *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, 415–429. SIAM.

Gower, J. C.; and Ross, G. J. 1969. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 18(1): 54–64.

Henzinger, M. R.; King, V.; and Warnow, T. J. 1999. Constructing a Tree from Homeomorphic Subtrees, with Applications to Computational Evolutionary Biology. *Algorithmica*, 24(1): 1–13.

Holm, J.; de Lichtenberg, K.; and Thorup, M. 2001. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4): 723–760.

Kannan, S.; Lawler, E. L.; and Warnow, T. J. 1996. Determining the Evolutionary Tree Using Experiments. *J. Algorithms*, 21(1): 26–50.

Littlestone, N. 1987. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Mach. Learn.*, 2(4): 285–318.

Michener, C. D.; and Sokal, R. R. 1957. A quantitative approach to a problem in classification. *Evolution*, 11(2): 130–162.

Natarajan, B. K. 1989. On Learning Sets and Functions. *Mach. Learn.*, 4: 67–97.

Sorensen, T. A. 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons. *Biol. Skar.*, 5: 1–34.

Thorup, M. 1999. Decremental Dynamic Connectivity. *J. Algorithms*, 33(2): 229–243.

Valiant, L. G. 1984. A Theory of the Learnable. *Commun. ACM*, 27(11): 1134–1142.

Vapnik, V.; and Chervonenkis, A. Y. 1971. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications*, 16(2): 264.

Ward Jr, J. H. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301): 236–244.