On the computational complexity of the probabilistic label tree algorithms

Róbert Busa-Fekete* Krzysztof Dembczyński† Alexander Golovnev‡
Kalina Jasinska§ Mikhail Kuznetsov¶ Maxim Sviridenko‖ Chao Xu**

Abstract

Label tree-based algorithms are widely used to tackle multi-class and multi-label problems with a large number of labels. We focus on a particular subclass of these algorithms that use probabilistic classifiers in the tree nodes. Examples of such algorithms are hierarchical softmax (HSM), designed for multi-class classification, and probabilistic label trees (PLTs) that generalize HSM to multi-label problems. If the tree structure is given, learning of PLT can be solved with provable regret guaranties (Wydmuch et al., 2018). However, to find a tree structure that results in a PLT with a low training and prediction computational costs as well as low statistical error seems to be a very challenging problem, not well-understood yet.

In this paper, we address the problem of finding a tree structure that has low computational cost. First, we show that finding a tree with optimal training cost is NP-complete, nevertheless there are some tractable special cases with either perfect approximation or exact solution that can be obtained in linear time in terms of the number of labels m. For the general case, we obtain $O(\log m)$ approximation in linear time too. Moreover, we prove an upper bound on the expected prediction cost expressed in terms of the expected training cost. We also show that under additional assumptions the prediction cost of a PLT is $O(\log m)$.

*Yahoo! Research, email: busafekete@verizonmedia.com
†Institute of Computing Science, Poznan University of Technology, email: kdembczynski@cs.put.poznan.pl
‡Harvard University, email: alexgolovnev@gmail.com. Supported by a Rabin Postdoctoral Fellowship.
§Institute of Computing Science, Poznan University of Technology, email: kjasinska@cs.put.poznan.pl
¶Yahoo! Research, email: kuznetsov@verizonmedia.com
‖Yahoo! Research, email: sviri@verizonmedia.com
**Yahoo! Research, email: chao.xu@verizonmedia.com

1 Introduction

We consider a class of machine learning algorithms that use hierarchical structures of classifiers to reduce the computational complexity of training and prediction in large-scale problems characterized by a large number of labels. Problems of this type are often referred to as extreme classification (Prabhu and Varma, 2014). The hierarchical structure usually takes a form of a label tree in which a leaf corresponds to one and only one label. The nodes of the tree contain classifiers that direct the test examples from the root down to the leaf nodes. We study the subclass of these algorithms with probabilistic classifiers, i.e., classifiers with responses in the range $[0, 1]$. Examples of such algorithms for multi-class classification include hierarchical softmax (HSM) (Morin and Bengio, 2005), as implemented for example in fastText (Joulin et al., 2017), and conditional probability estimation trees (Beygelzimer et al., 2009a). For multi-label classification this idea is known under the name of probabilistic label trees (PLTs) (Jasinska et al., 2016), and has been implemented in Parabel (Prabhu et al., 2018) and extremeText (Wydmuch et al., 2018). Note that the PLT model can be treated as a generalization of algorithms for both multi-class and multi-label classification (Wydmuch et al., 2018).

We present a wide spectrum of theoretical results concerning training and prediction costs of PLTs. We first define the multi-label problem (Section 2). Then, we define the PLT model and state some of its important properties (Section 3). As a starting point of our analysis, we define the training cost for a single instance as the number of nodes where it is involved in training classifiers (Section 4). The rationale behind this cost is that the learning methods, often used to train the node classifiers, scale linearly with the sample size. We note that the popular 1-vs-All approach has the cost equal m, the number of labels, according to our definition. This cost can be significantly reduced by using PLTs. We then address the problem of finding a tree structure that minimizes the training cost (Section 5). We first show that the decision version of this problem is NP-complete (Section 5.1). Nevertheless, there exists a $O(\log m)$ approximation that can be computed in linear time (Section 5.2). We also consider two special cases: multi-class (Section 5.3) and multi-label with nested labels (Section 5.4), for which we obtain constant approximation and exact solution, respectively, both computed in linear time in m. We also consider the prediction cost defined as the number of nodes visited during classification of a test example (Section 6). We first show that under additional assumptions prediction can be made in $O(\log m)$ time. Finally, we prove an upper bound on the expected prediction cost expressed in terms of the expected training cost and statistical error of the node classifiers.

The problem of optimizing the training cost is closely related to the binary merging problem in databases (Ghosh et al., 2015). The hardness result in (Ghosh et al., 2015), however, does not generalize to our setting as it is limited to binary trees only. Nevertheless, our approximation result is partly based on the results from (Ghosh et al., 2015). The training cost we use is similar to the one considered in (Grave et al., 2017), but the authors there consider a specific class of shallow trees. The Huffman tree is a popular choice for HSM (many word2vec implementations (Mikolov et al., 2013) and fastText (Joulin et al., 2017) use binary Huffman trees). This strategy is justified as for multi-class with binary trees the Huffman code is optimal (Wydmuch et al., 2018). Surprisingly, the solution for the general multi-class case has been unknown prior to this work. The problem of learning the tree structure to improve the predictive performance is studied in (Jernite et al., 2017; Prabhu et al., 2018). Ideally, however, one would like to have a procedure that minimizes two objectives: the computational cost and statistical error.

2 Multi-label classification

Let $\mathcal{X}$ denote an instance space, and let $\mathcal{L} = [m]$ be a finite set of m class labels. We assume that an instance $\boldsymbol{x} \in \mathcal{X}$ is associated with a subset of labels $\mathcal{L}_{\boldsymbol{x}} \subseteq \mathcal{L}$ (the subset can be empty); this subset is often called the set of relevant labels, while the complement $\mathcal{L} \setminus \mathcal{L}_{\boldsymbol{x}}$ is considered as irrelevant for $\boldsymbol{x}$. We assume m to be a large number (e.g., $\geq 10^5$), but the size of the set of relevant labels $\mathcal{L}_{\boldsymbol{x}}$ is usually much smaller than m, i.e., $|\mathcal{L}_{\boldsymbol{x}}| \ll m$. We identify the set $\mathcal{L}_{\boldsymbol{x}}$ of relevant labels with the binary vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$, in which $y_j = 1 \Leftrightarrow j \in \mathcal{L}_{\boldsymbol{x}}$. By $\mathcal{Y} = \{0,1\}^m$ we denote the set of all possible label vectors. We assume that observations $(\boldsymbol{x}, \boldsymbol{y})$ are generated independently and identically according to a probability distribution $\mathbf{P}(\boldsymbol{x}, \boldsymbol{y})$ defined on $\mathcal{X} \times \mathcal{Y}$. Observe that the above definitions include as special cases multi-class classification (where $\|\boldsymbol{y}\|_1 = 1$) and k-sparse multi-label classification (where $\|\boldsymbol{y}\|_1 \leq k$).[1]

We are interested in multi-label classifiers that estimate conditional probabilities of labels, $\eta_j = \mathbf{P}(y_j = 1|\boldsymbol{x})$, $j \in \mathcal{L}$, as accurately as possible, i.e., with possibly small L_1-estimation error, i.e., $|\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})|$, where $\hat{\eta}_j(\boldsymbol{x})$ is an estimate of $\eta_j(\boldsymbol{x})$. This statement of the problem is justified by the fact that optimal predictions in terms of the statistical decision theory for many performance measures used in multi-label classification, such as the Hamming loss, precision@k, and the micro- and macro F-measure, are determined through the conditional probabilities of labels (Dembczyński et al., 2010; Kotlowski and Dembczyński, 2016; Koyejo et al., 2015).

3 Probabilistic label trees (PLTs)

We will work with the set $\mathcal{T}$ of rooted, leaf-labeled trees with m leaves. We denote a single tree by T and its set of leaves by L_T. The leaf $\ell_j \in L_T$ corresponds to the label $j \in \mathcal{L}$. The set of leaves of a (sub)tree rooted in an inner node v is denoted by $L(v)$. The parent node of v is denoted by $\mathrm{pa}(v)$, and the set of child nodes by $\mathrm{Ch}(v)$. The path from node v to the root is denoted by $\mathrm{Path}(v)$. The length of the path, i.e., the number of nodes on the path, is denoted by len_v. The set of all nodes is denoted by V_T. The degree of a node $v \in V_T$, i.e., the number of its children, is denoted by $\deg_v = |\mathrm{Ch}(v)|$.

PLT uses tree T to factorize the conditional probabilities of labels, $\eta_j(\boldsymbol{x}) = \mathbf{P}(y_j = 1|\boldsymbol{x})$, for $j \in \mathcal{L}$. To this end let us define for every $\boldsymbol{y}$ a corresponding vector $\boldsymbol{z}$ of length $|V_T|$,[2] whose coordinates, indexed by $v \in V_T$,[3] are given by:

$$z_v = \mathbb{I}\left\{\sum_{\ell_j \in L(v)} y_j \geq 1\right\}, \quad \text{or equivalently by } z_v = \bigvee_{\ell_j \in L(v)} y_j \,.$$

With the above definition, it holds based on the chain rule that for any $v \in V_T$:

$$\eta_v(\boldsymbol{x}) = \mathbf{P}(z_v = 1 \,|\, \boldsymbol{x}) = \prod_{v' \in \mathrm{Path}(v)} \eta(\boldsymbol{x}, v') \,, \tag{1}$$

where $\eta(\boldsymbol{x}, v) = \mathbf{P}(z_v = 1 | z_{\mathrm{pa}(v)} = 1, \boldsymbol{x})$ for non-root nodes, and $\eta(\boldsymbol{x}, v) = \mathbf{P}(z_v = 1 \,|\, \boldsymbol{x})$ for the root (see, e.g., Jasinska et al. 2016). Notice that for the leaf nodes we get the conditional probabilities of labels, i.e.,

$$\eta_{\ell_j}(\boldsymbol{x}) = \eta_j(\boldsymbol{x}) \,, \quad \text{for } \ell \in L_T \,. \tag{2}$$

[1] We use $[n]$ to denote the set of integers from 1 to n, and $\|\boldsymbol{x}\|_1$ to denote the L_1 norm of x.

[2] Note that $\boldsymbol{z}$ depends on T, but T will always be obvious from the context.

[3] We will also use leaves $v \in L_T$ to index the elements of vector $\boldsymbol{y}$.

The following result states the relation between probabilities of the parent node and its children.

Proposition 1. *For any T and $\mathbf{P}(\boldsymbol{y}|\boldsymbol{x})$, the probability $\eta_v(\boldsymbol{x})$ of any internal node $v \in V_T \setminus L_T$ satisfies:*

$$\max\left\{\eta_{v'}(\boldsymbol{x}) : v' \in \mathrm{Ch}(v)\right\} \leq \eta_v(\boldsymbol{x}) \leq \min\left\{1, \textstyle\sum_{v' \in \mathrm{Ch}(v)} \eta_{v'}(\boldsymbol{x})\right\} . \tag{3}$$

Proof. We first prove the first inequality. From the definition of tree T and z_v, we have that $z_v = 1 \Rightarrow z_{\mathrm{pa}(v)} = 1$ since $L(v) \subset L(z_{\mathrm{pa}(v)})$. Taking the expectation with respect to $\mathbf{P}(\boldsymbol{y}|\boldsymbol{x})$, we obtain that $\eta_{v'}(\boldsymbol{x}) \leq \eta_v(\boldsymbol{x})$ for every $v' \in \mathrm{Ch}(v)$.

For the second inequality, obviously we have $\eta_v(\boldsymbol{x}) \leq 1$. Furthermore, if $z_v = 1$, then there exists at least one $v' \in \mathrm{Ch}(v)$ for which $z_{v'} = 1$. In other words, $z_v \leq \sum_{v' \in \mathrm{Ch}(v)} z_{v'}$. Therefore, by taking expectation with respect to $\mathbf{P}(\boldsymbol{y}|\boldsymbol{x})$ we obtain $\eta_v(\boldsymbol{x}) \leq \sum_{v' \in \mathrm{Ch}(v)} \eta_{v'}(\boldsymbol{x})$. □

To estimate $\eta(\boldsymbol{x}, v)$, for $v \in V_T$, we use a function class $\mathcal{H} : \mathbb{R}^d \mapsto [0, 1]$ which contains probabilistic classifiers of choice, for example, logistic regressors. We assign a classifier from $\mathcal{H}$ to each node of the tree T. We shall index this set of classifiers by the elements of V_T as $H = \{\hat{\eta}(v) \in \mathcal{H} : v \in V_T\}$. We also denote by $\hat{\eta}(\boldsymbol{x}, v)$ the estimate of $\eta(\boldsymbol{x}, v)$ obtained for a given $\boldsymbol{x}$ in node $v \in V_T$. The estimates obey the analogous equations to (1) and (2). However, as the probabilistic classifiers $\hat{\eta}(v) \in H$ can be trained independently from each other, Proposition 1 may not apply to the estimated probabilities. This can be fixed by a proper normalization during prediction.

The quality of the estimates of conditional probabilities, $\hat{\eta}_j(\boldsymbol{x})$, $j \in \mathcal{L}$, can be expressed in terms of the L_1-estimation error in each node classifier, i.e., by $|\eta(\boldsymbol{x}, v) - \hat{\eta}(\boldsymbol{x}, v)|$. Based on similar results from (Beygelzimer et al., 2009b) and (Wydmuch et al., 2018) we get the following bound, which for $\ell_j \in L_T$ gives the guarantees for $\hat{\eta}_j(\boldsymbol{x})$, $j \in \mathcal{L}$.

Theorem 1. *For any tree T and $\mathbf{P}(\boldsymbol{y}|\boldsymbol{x})$ the following holds for $v \in V_T$:*

$$|\eta_v(\boldsymbol{x}) - \hat{\eta}_v(\boldsymbol{x})| \leq \sum_{v' \in \mathrm{Path}(v)} \eta_{\mathrm{pa}(v')}(\boldsymbol{x}) \left|\eta(\boldsymbol{x}, v') - \hat{\eta}(\boldsymbol{x}, v')\right| , \tag{4}$$

where for the root node $\eta_{\mathrm{pa}(r_T)}(\boldsymbol{x}) = 1$.

Proof. This result can be found as a part of the proof of Theorem 1 in Appendix A in (Wydmuch et al., 2018). It is presented in Eq. (6) therein. However, this result is stated only for conditional probabilities of labels $\eta_j(\boldsymbol{x})$ and their estimates $\hat{\eta}_j(\boldsymbol{x})$. The generalization to any node $v \in V_T$ is straightforward as the chain rule (1) applies to any node v and the necessary transformations to get the result can be applied. □

4 Training complexity

Training data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ consist of tuples of feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$ and label vector $\boldsymbol{y}_i \in \{0, 1\}^m$. The labels for the entire training set can be written in a matrix form $\boldsymbol{Y} = [y_{i,j}]$ whose j-th column is denoted by $\dot{\boldsymbol{y}}_j$. We also use a corresponding matrix $\boldsymbol{Z} = [z_{i,v}]$, with columns indexed by $v \in V_T$ and denoted by $\dot{\boldsymbol{z}}_v$.

We define the training complexity of PLTs in terms of the number of nodes in which a training example $(\boldsymbol{x}, \boldsymbol{y})$ is used. This number follows from the definition of the tree and the PLT model (1). We use each training example in the root (to estimate $\mathbf{P}(z_{r_T} = 1|\boldsymbol{x})$) and in each node v for which

Algorithm 1 PLT.AssignToNodes$(T, \boldsymbol{x}, \boldsymbol{y})$

$P = \emptyset$, $N = \{r_T\}$ ▷ Initialize positive and negative nodes (r_T added to deal with $\boldsymbol{y}$ of all zeros)
for $j \in \mathcal{L}_{\boldsymbol{x}}$ **do** ▷ For all labels of the training example
  $v = \ell_j$ ▷ Set v to a leaf corresponding to label j
  while v not null **and** $v \notin P$ **do** ▷ On a path to the first positive node (excluded) or the root (included)
    $P = P \cup \{v\}$ ▷ Assign a node to positive nodes
    $N = N \setminus \{v\}$ ▷ Remove the node from negative nodes if added there before
    for $v' \in \mathrm{Ch}(v)$ **do** ▷ For all its children
      if $v' \notin P$ **then** ▷ If a child is not a positive node
        $N = N \cup \{v'\}$ ▷ Assign it to negative nodes
    $v = \mathrm{pa}(v)$ ▷ Move up along the path
return (P, N) ▷ Return a set of positive and negative nodes for the training example

$z_{\mathrm{pa}(v)} = 1$ (to estimate $\mathbf{P}(z_v = 1 | z_{\mathrm{pa}(v)} = 1, \boldsymbol{x})$). Therefore, we define the training cost for a single training example $(\boldsymbol{x}, \boldsymbol{y})$ by:

$$c(T, \boldsymbol{y}) = 1 + \sum_{v \in V_T \setminus r_T} z_{\mathrm{pa}(v)} \,. \tag{5}$$

Algorithm 1 shows the AssignToNodes method which identifies for a training example the set of *positive* and *negative nodes*, i.e., the nodes for which the training example is treated respectively as positive (i.e, $(\boldsymbol{x}, z_v = 1)$) or negative (i.e., $(\boldsymbol{x}, z_v = 0)$) (see the pseudocode and the comments there for details of the method).[4] Based on this assignment a learning algorithm of choice, either batch or online, trains the node classifiers $\hat{\eta}(v, \boldsymbol{x})$. The *training cost* for set $\mathcal{D}$ is then expressed by:

$$c(T, \boldsymbol{Y}) = \sum_{i=1}^{n} c(T, \boldsymbol{y}_i) \,.$$

The above quantities are justified from the learning point of view by the following reasons. On the one hand, in an online setting, the complexity of an update of PLT based on a single sample $(\boldsymbol{x}, \boldsymbol{y})$ is indeed $O(c(T, \boldsymbol{y}))$, using a linear classifier in the inner node trained by optimizing some smooth loss with stochastic gradient descent (SGD) which is often the method of choice along with PLTs. Moreover, even if SGD is used in an offline setting, the SOTA packages, like fastText, run several epochs over the training data. Therefore, their training time is $O(c(T, \boldsymbol{Y}) \cdot \#\text{epochs})$, not taking into account the complexity of other layers. On the other hand, if we update the inner node models in a batch setting, the training time is again linear in $c(T, \boldsymbol{Y})$ for several large-scale learning methods whose training process is based on optimizing some smooth loss, such as logistic regression (Allen-Zhu, 2017).

The next proposition gives an upper bound for the cost $c(T, \boldsymbol{y})$.

Proposition 2. *For any tree T and vector $\boldsymbol{y}$ it holds that:*

$$c(T, \boldsymbol{y}) \leq 1 + \|\boldsymbol{y}\|_1 \cdot \mathrm{depth}_T \cdot \deg_T \,,$$

where $\mathrm{depth}_T = \max_{v \in L_T} \mathrm{len}_v - 1$ *is the depth of the tree, and* $\deg_T = \max_{v \in V_T} \deg_v$ *is the highest degree of a node in T.*

[4] Notice that the AssignToNodes method has time complexity $O(c(T, \boldsymbol{y}))$ assuming that the set operations are performed in time $O(1)$ (e.g., the set is implemented by hash table).

Proof. First notice that a training example is always used in the root node, either as a positive example $(\boldsymbol{x}, 1)$, if $\|\boldsymbol{y}\|_1 > 0$, or as a negative example $(\boldsymbol{x}, 0)$, if $\|\boldsymbol{y}\|_1 = 0$. Therefore the cost is bounded by 1. If $\|\boldsymbol{y}\|_1 > 0$, the training example is also used as a positive example in all the nodes on paths from the root to leaves corresponding to labels j for which $y_j = 1$ in $\boldsymbol{y}$. As the root has been already counted, we have at most $\mathrm{depth}_T = \max_v \mathrm{len}_v - 1$ such nodes for each positive label in $\boldsymbol{y}$. Moreover, the training example is used as a negative example in all siblings of the nodes on the paths determined above, unless it is already a positive example in the sibling node. The highest degree of a node in the tree is $\deg_T$. Taking the above into account, the cost $c(T, \boldsymbol{y})$ is upperbounded by $1 + \|\boldsymbol{y}\|_1 \cdot \mathrm{depth}_T \cdot \deg_T$. The bound is tight, for example, if $\|\boldsymbol{y}\|_1 = 1$ and T is a perfect $\deg_T$-ary tree (all non-leaf nodes have equal degree and the paths to the root from all leaves are of the same length).

□

Remark 1. *Consider k-sparse multi-label classification (i.e., $\|\boldsymbol{y}\|_1 \leq k$). For a balanced tree of constant $\deg_T = \lambda(\geq 2)$ and $\mathrm{depth}_T = \log_\lambda m$, the training cost is $c(T, \boldsymbol{y}) = O(k \log m)$.*

In the proposition below we express the cost in terms of vectors $\dot{\boldsymbol{z}}_v$. Each such vector indicates the positive examples for node v. We refer to $\|\dot{\boldsymbol{z}}_v\|_1$ as the *Hamming weight* of the node $v \in V_T$. Moreover, we use $c(v) = c(T, \boldsymbol{Y}, v) = \|\dot{\boldsymbol{z}}_v\|_1 \cdot \deg_v$ for the cost of the node $v \in V_T$.

Proposition 3. *For any tree T and label matrix $\boldsymbol{Y}$ it holds that:*

$$c(T, \boldsymbol{Y}) = n + \sum_{v \in V_T \setminus r_T} \|\dot{\boldsymbol{z}}_{\mathrm{pa}(v)}\|_1 = n + \sum_{v \in V_T} \|\dot{\boldsymbol{z}}_v\|_1 \cdot \deg_v = n + \sum_{v \in V_T} c(v)\,.$$

Proof. Obviously, we have that:

$$\sum_{i=1}^{n} c(T, \boldsymbol{y}) = \sum_{i=1}^{n} \left(1 + \sum_{v \in V_T \setminus r_T} z_{i,\mathrm{pa}(v)}\right) = n + \sum_{v \in V \setminus r_T} \|\dot{\boldsymbol{z}}_{\mathrm{pa}(v)}\|_1\,,$$

as elements $z_{i,\mathrm{pa}(v)}$ constitute matrix $\boldsymbol{Z} = [\dot{\boldsymbol{z}}_1, \ldots, \dot{\boldsymbol{z}}_{|V|}]$ with columns $\dot{\boldsymbol{z}}_v$ corresponding to the nodes of T. Next, notice that for each $v \in V_T \setminus L_T$, we have:

$$\sum_{v' \in \mathrm{Ch}(v)} z_v = z_v \sum_{v' \in \mathrm{Ch}(v)} 1 = z_v \cdot \deg_v\,. \tag{6}$$

Therefore,

$$\sum_{i=1}^{n} c(T, \boldsymbol{y}) = n + \sum_{v \in V_T \setminus r_T} \|\dot{\boldsymbol{z}}_{\mathrm{pa}(v)}\|_1 = n + \sum_{v \in V} \|\dot{\boldsymbol{z}}_v\|_1 \cdot \deg_v\,.$$

The last sum is over all nodes as for $v \in L_T$ we have $\deg_v = 0$. The final equation is obtained by definition of the cost of the node $v \in V_t$, i.e., $c(v) = c(T, \boldsymbol{Y}, v) = \|\dot{\boldsymbol{z}}_v\|_1 \cdot \deg_v$. □

Next we show a counterpart of Proposition 1 for training data.

Proposition 4. *For any T and label matrix $\boldsymbol{Y}$, the Hamming weight $\|\dot{\boldsymbol{z}}_v\|_1$ of any internal node $v \in V_T \setminus L_T$ satisfies:*

$$\max\left\{\|\dot{\boldsymbol{z}}_{v'}\|_1 : v' \in \mathrm{Ch}(v)\right\} \leq \|\dot{\boldsymbol{z}}_v\|_1 \leq \min\left\{n, \textstyle\sum_{v' \in \mathrm{Ch}(v)} \|\dot{\boldsymbol{z}}_{v'}\|_1\right\}\,, \tag{7}$$

with equality on the left holding for label covering distributions, i.e., $\forall \boldsymbol{y}_i \exists \ell_j \in L(v) : \forall_{\ell_k \in L(v) \setminus \ell_j} (y_{i,k} = 1 \Rightarrow y_{i,j} = 1)$, and equality on the right holding for multi-class distributions, i.e., $\forall \boldsymbol{y}_i \sum_{\ell_j \in L(v)} y_{i,j} = 1$.

Proof. The proof follows the same steps as the proof of Proposition 1 with the difference that instead of expectation with respect to $\mathbf{P}(\boldsymbol{y} \mid \boldsymbol{x})$, we take the sum over the training examples.

The left inequality becomes equality, for example, for the label covering distribution, since $z_v = z_{v'}$ for the child node v' under which there is label j, i.e., $j \in L(v')$, or v' is the leaf node corresponding to label ℓ_j.

The right inequality becomes equality, for example, for the multi-class distribution, since there is always only one child v' for which $z_{v'} = 1$. □

Another important quantity we use is the expected training cost:

$$C_{\mathbf{P}}(T) = \mathbb{E}_{\boldsymbol{y}}\left[c(T, \boldsymbol{y})\right] = \sum_{\boldsymbol{y} \in \mathcal{Y}} c(T, \boldsymbol{y}) \mathbf{P}(\boldsymbol{y})\,.$$

Propositions 2–4 can be easily generalized to the expected training cost.

Proposition 5. *For any tree T and distribution $\mathbf{P}(\boldsymbol{y})$ it holds that:*

$$C_{\mathbf{P}}(T) = 1 + \sum_{v \in V_T \setminus r_T} \mathbf{P}(z_{\mathrm{pa}(v)} = 1) = 1 + \sum_{v \in V_T} \mathbf{P}(z_v = 1) \cdot \deg_v\,.$$

Proof. The result follows immediately by taking the expectation of $c(T, \boldsymbol{y})$ and applying (6) similarly as in Proposition 3. Namely, we have

$$\begin{aligned}
C_{\mathbf{P}}(T) = \mathbb{E}[c(T, \boldsymbol{y})] &= \sum_{\boldsymbol{y}} c(T, \boldsymbol{y}) \mathbf{P}(\boldsymbol{y}) \\
&= \sum_{\boldsymbol{y}} \left(1 + \sum_{v \in V_T \setminus r_T} z_{\mathrm{pa}(v)}\right) \mathbf{P}(\boldsymbol{y}) \\
&= 1 + \sum_{v \in V_T \setminus r_T} \sum_{\boldsymbol{y}} z_{\mathrm{pa}(v)} \mathbf{P}(\boldsymbol{y}) \\
&= 1 + \sum_{v \in V_T \setminus r_T} \mathbf{P}(z_{\mathrm{pa}(v)} = 1) \\
&= 1 + \sum_{v \in V_T} \mathbf{P}(z_v = 1) \cdot \deg_v\,.
\end{aligned}$$

The last sum is over all nodes as for $v \in L_T$ we have $\deg_v = 0$. □

Proposition 6. *For any tree T and distribution $\mathbf{P}(\boldsymbol{y})$ it holds that:*

$$C_{\mathbf{P}}(T) \leq 1 + \mathrm{depth}_T \cdot \deg_T \cdot \sum_{j=1}^{m} \mathbf{P}(y_j = 1)\,,$$

where $\mathrm{depth}_T = \max_{v \in L_T} \mathrm{len}_v - 1$ is the depth of the tree, and $\deg_T = \max_{v \in V_T} \deg_v$ is the highest degree of a node in T.

Proof. The proof follows immediately from Proposition 2 by taking the expectation over $\mathbf{P}(\boldsymbol{y})$. □

Proposition 7. *For any T and distribution $\mathbf{P}$, the probability $\mathbf{P}(z_v = 1)$ of any internal node $v \in V(T) \setminus L(T)$ satisfies:*

$$\max\left\{\mathbf{P}(z_{v'} = 1) : v' \in \mathrm{Ch}(v)\right\} \leq \mathbf{P}(z_v = 1) \leq \min\left\{1, \sum_{v' \in \mathrm{Ch}(v)} \mathbf{P}(z_{v'} = 1)\right\}. \tag{8}$$

Proof. The proposition follows immediately from Proposition 1 by taking the expectation over $\mathbf{P}(\boldsymbol{x})$. □

Next, we state the relation between the finite sample and expected training costs. Using the fact that $c(T, \boldsymbol{y})$ has bounded difference property, we can compute its deviation from its mean as follows.

Proposition 8. *For any PLT with label tree T, it holds that*

$$\mathbf{P}\big(\,|c(T, \boldsymbol{y}) - C_{\mathbf{P}}(T)| > \epsilon\big) \leq 2e^{-2\epsilon^2 / \sum_{i=1}^{m} d_i^2},$$

where $d_i = \sum_{j \in \mathrm{Path}(\ell_i)} \deg_{\mathrm{pa}(j)}$.

Proof. We can directly apply the concentration result for functions with bounded difference (see Section 3.2 of Boucheron et al. 2013). It only remains to upper bound $\sup_{\boldsymbol{y}, \boldsymbol{y}^{(i)}} |c(T, \boldsymbol{y}) - c(T, \boldsymbol{y}^{(i)})|$ for any i, where $\boldsymbol{y}^{(i)}$ is the same as $\boldsymbol{y} \in \{0, 1\}^m$ except that the i component is flipped. First, consider the case when $y_i = 0$ and let us flip its value. Based on Proposition 5, the training algorithm of PLT updates each children of an inner node v if there is at least one leaf ℓ in the subtree below it for which $y_\ell = 1$, otherwise it does not update the children classifier with the given example. Thus $|c(T, \boldsymbol{y}) - c(T, \boldsymbol{y}^{(i)})|$ cannot be bigger than d_i. The same argument applies to the case when $y_i = 1$ which concludes the proof. □

Note that $d_i \leq 2 \log m$ for balanced binary trees, thus $\frac{1}{n} \sum_{i=1}^{n} c(T, \boldsymbol{y}_i)$ is close to its expected value with $\Omega(\sqrt{m} \log m)$ samples with high probability. This lower bound suggests that one should not consider optimizing the training complexity based on fewer examples, since the empirical value $\frac{1}{n} \sum_{i=1}^{n} c(T, \boldsymbol{y}_i)$ which one would like to optimize over the space of labeled trees, might significantly deviate from its expected value.

5 Optimizing the training complexity ($\min_{T \in \mathcal{T}} c(T, \boldsymbol{Y})$)

In this section, we focus on the algorithmic and hardness results for minimizing the cost $c(T, \boldsymbol{Y})$. In the analysis, we mainly refer to matrices $\boldsymbol{Y}$ and $\boldsymbol{Z}$ via their columns $\dot{\boldsymbol{y}}_j \in \{0, 1\}^n$, $j \in [m]$, and $\dot{\boldsymbol{z}}_v \in \{0, 1\}^n$, $v \in V_T$, respectively. We assume $\boldsymbol{Y}$ to be stored efficiently, for example, as a sparse matrix whenever it is possible. We also use $p_j = \|\dot{\boldsymbol{y}}_j\|_1 / n$ and $p_v = \|\dot{\boldsymbol{z}}_v\|_1 / n$, which are the fractions of positive examples in the corresponding nodes.

5.1 Hardness of training cost minimization

First we formally define the decision version of the cost minimization problem.

Definition 1 (PLT training cost problem)**.** *For a label matrix $\boldsymbol{Y}$ and a parameter w decide whether there exists a tree $T \in \mathcal{T}$ such that $c(T, \boldsymbol{Y}) \leq w$.*

We prove NP-hardness of PLT training cost by a reduction from the Clique problem (which is one of the classical NP-complete problems, Garey and Johnson 1979) defined as follows.

Definition 2 (Clique)**.** *For an undirected graph $G = (V, E)$ and a parameter $1 \leq k \leq |V|$, decide whether G contains a clique on k nodes.*

Theorem 2. *The PLT training cost problem is NP-complete.*

The proof of this theorem is given in the Appendix. We remark that a problem similar to PLT training cost has been studied in the database literature. In particular, the problem of finding an optimal *binary* tree is proven to be NP-hard in (Ghosh et al., 2015). Note that the result of (Ghosh et al., 2015) does not imply hardness of the PLT training cost problem.

5.2 Logarithmic approximation for multi-label case

Despite the hardness of the problem, we are able to give a simple algorithm which achieves an $O(\log m)$ approximation. As remarked above, the problem of finding an optimal *binary* PLT tree is equivalent to the binary merging problem considered in (Ghosh et al., 2015).

Definition 3 (Binary merging)**.** *For a ground set U of size n, and a collection of m sets, $A_1, ..., A_m$ where each $A_i \subseteq U$, a merge schedule is a pair of a full binary tree[5] $T^*_{binmerge}$ with m labeled leaves, and a permutation $\pi : [m] \to [m]$ which assigns every set A_i to the leaf number $\pi(i)$. The binary merging problem is to find a merge schedule of the minimum cost:*

$$cost(T, \pi, A_1, ..., A_m) = \sum_{v \in T} |A_v| \,,$$

where A_v is the union of sets A_i assigned to the leaves of the subtree rooted at the node v.

While binary merging is NP-complete, it admits an $O(\log m)$ approximation (Ghosh et al., 2015). The lemma below, showing that any PLT training cost problem can be 2-approximated by a binary PLT tree, gives a simple $O(\log m)$-approximation for the PLT training cost problem: it suffices to find an optimal binary tree using the algorithms from (Ghosh et al., 2015) (e.g., one of the algorithms presented there is a simple modification of the Huffman tree building algorithm).

Lemma 1. *For any PLT training cost instance $\boldsymbol{Y}$, it holds that*

$$\min_{T \in \mathcal{T}} c(T, \boldsymbol{Y}) \leq 2 \min_{T \in \mathcal{T}_{bin}} c(T, \boldsymbol{Y}) \,,$$

where $\mathcal{T}_{bin}$ denotes the set of trees in which each internal node (including the root) has degree 2.

Proof. Consider an optimal tree $T^*_{\boldsymbol{Y}} \in \operatorname{argmin}_{T \in \mathcal{T}} c(T, \boldsymbol{Y})$. Starting from the root, replace every node with more than 2 children by an arbitrary binary tree whose set of leaves is the set of children of this node. Consider a node v of $T^*_{\boldsymbol{Y}}$, let $v_1, \ldots, v_d$ be the children of v. The cost of the node v is $c(v) = \deg_v \cdot \|\dot{\boldsymbol{z}}_{v_1} \vee \ldots \vee \dot{\boldsymbol{z}}_{v_d}\|_1$. Any binary tree with the leaves $v_1, \ldots, v_d$ has $(\deg_v - 1)$ internal nodes, each of them has degree two and the Hamming weight of its label $\dot{\boldsymbol{z}}$ is at most $\|\dot{\boldsymbol{z}}_{v_1} \vee \ldots \vee \dot{\boldsymbol{z}}_{v_d}\|_1$. Thus, the sum of the costs of the internal nodes of this binary tree is at most $2(\deg_v - 1) \cdot \|\dot{\boldsymbol{z}}_{v_1} \vee \ldots \vee \dot{\boldsymbol{z}}_{v_d}\|_1 < 2\deg_v \cdot \|\dot{\boldsymbol{z}}_{v_1} \vee \ldots \vee \dot{\boldsymbol{z}}_{v_d}\|_1 = 2c(v)$. When we repeat this procedure for all internal nodes of $T^*_{\boldsymbol{Y}}$, we increase the cost of each node by at most a factor of 2. Thus, the resulting binary tree is a 2-approximation of $T^*_{\boldsymbol{Y}}$. □

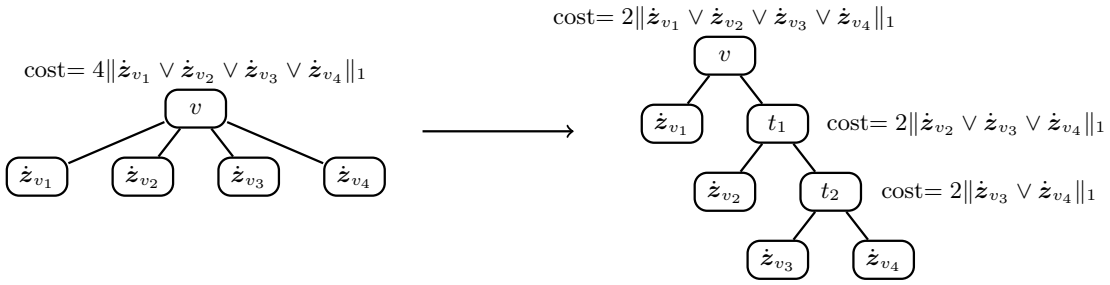

Figure 1: The transformation from Lemma 1: the node v of high degree is transformed into a binary tree of total cost less than twice the original cost: $2\|\dot{\boldsymbol{z}}_{v_1} \vee \dot{\boldsymbol{z}}_{v_2} \vee \dot{\boldsymbol{z}}_{v_3} \vee \dot{\boldsymbol{z}}_{v_4}\|_1 + 2\|\dot{\boldsymbol{z}}_{v_2} \vee \dot{\boldsymbol{z}}_{v_3} \vee \dot{\boldsymbol{z}}_{v_4}\|_1 + 2\|\dot{\boldsymbol{z}}_{v_3} \vee \dot{\boldsymbol{z}}_{v_4}\|_1 \leq 6\|\dot{\boldsymbol{z}}_{v_1} \vee \dot{\boldsymbol{z}}_{v_2} \vee \dot{\boldsymbol{z}}_{v_3} \vee \dot{\boldsymbol{z}}_{v_4}\|_1 \leq 2 \cdot 4\|\dot{\boldsymbol{z}}_{v_1} \vee \dot{\boldsymbol{z}}_{v_2} \vee \dot{\boldsymbol{z}}_{v_3} \vee \dot{\boldsymbol{z}}_{v_4}\|_1$.

We are able, however, to give another algorithm, based on ternary complete trees, with a slightly better constant in the approximation ratio.[6]

Theorem 3. *There exists an algorithm which runs in time $O(m+n)$ and achieves an approximation guarantee of $\frac{3\log m}{\log 3}$ for the PLT training cost problem, i.e., the output T of the algorithm satisfies*

$$c(T, \boldsymbol{Y}) \leq \frac{3\log m}{\log 3} \cdot \min_{T \in \mathcal{T}} c(T, \boldsymbol{Y})\,.$$

Proof. The algorithm constructs in linear time a complete ternary tree T of depth $\lceil \log_3 m \rceil$, and assigns the m vectors $\dot{\boldsymbol{y}}_i$ to the leaves arbitrarily. From the definition of the cost function we have that for every tree T^*: $c(T^*, \boldsymbol{Y}) \geq n + \sum_{i=1}^{m} \|\dot{\boldsymbol{y}}_i\|_1$. On the other hand, from Proposition 3 we have that $c(T, \boldsymbol{Y}) \leq n + 3\lceil \log_3 m \rceil \sum_{i=1}^{m} \|\dot{\boldsymbol{y}}_i\|_1$, which completes the proof. □

We remark that any improvement of the approximation ratio of Theorem 3 would solve an open problem. Indeed, since the proof of Lemma 1 is constructive and efficient, any $o(\log m)$-approximation algorithm for the PLT training cost problem would imply an $o(\log m)$-approximation of an optimal binary tree, and this would improve the best known approximation ratio for the binary merging problem.

5.3 Multi-class case

In the multi-class case, we have $\sum_{j=1}^{m} y_{i,j} = 1$ for each $\boldsymbol{y}_i$ in $\boldsymbol{Y}$. For ease of exposition, we assume that the columns $\dot{\boldsymbol{y}}_1, \ldots, \dot{\boldsymbol{y}}_m$ are sorted such that $0 < p_1 \leq \ldots \leq p_m$.

Remark that for trees of a fixed degree λ for all internal nodes, the optimal solution is the λ-ary Huffman tree. Here, we do not have this restriction and have different costs for nodes of different degrees, which makes the problem more difficult. Nevertheless, we give two efficient algorithms which find almost optimal solutions for every instance of the multi-class PLT training cost problem. Namely, these algorithms find a solution within a small additive error. Moreover, these algorithms run in linear time $O(n+m)$.

[5] A full binary tree is a tree where every non-leaf node has exactly 2 children.

[6] We use log to denote the logarithm base 2.

We will use the entropy function defined as $H(p_1, \ldots, p_k) = \sum_{i=1}^{k} p_j \log(1/p_j)$, for $k \geq 1$, and $p_1, \ldots, p_k > 0$.[7] We will use the fact that for $p_1 + \ldots + p_k \leq 1, H(p_1, \ldots, p_k) \leq \log k$ (this follows from Jensen's inequality). We will also make use of the following corollary of Jensen's inequality.

Proposition 9. *Let $k \geq 1$, and $p_1, \ldots, p_k > 0$. Let $p = \sum_{i=1}^{k} p_i$. Then*

$$H(p) \geq H(p_1, \ldots, p_k) - p \log k \ .$$

Proof. Since $x \log\left(\frac{1}{x}\right)$ is concave for $x > 0$, by Jensen's inequality we have that:

$$\begin{aligned} H(p_1, \ldots, p_k) &= \sum_{i=1}^{k} p_i \log\left(\frac{1}{p_i}\right) \\ &\leq k \cdot \left(\frac{p}{k}\right) \log\left(\frac{k}{p}\right) \\ &= p\left(\log\left(\frac{1}{p}\right) + \log k\right) \\ &= H\left(p\right) + p \log k \ . \end{aligned}$$

□

We start by showing a lower bound for the multi-class case.

Lemma 2. *Let $\boldsymbol{Y}$ be an instance of the multi-class case. The cost of any tree T for $\boldsymbol{Y}$ is at least*

$$c(T, \boldsymbol{Y}) \geq n + \frac{3n}{\log 3} \cdot H(p_1, \ldots, p_m) \ .$$

Proof. We prove this Lemma by induction on the number of inner nodes of T. If T has only one inner node (the root), then

$$c(T, \boldsymbol{Y}) = n + nm \geq n + n \cdot \frac{3 \log m}{\log 3} \geq n + \frac{3n}{\log 3} \cdot H(p_1, \ldots, p_m) \ ,$$

because $\log m \geq H(p_1, \ldots, p_m)$ for every integer $m \geq 1$.

Now assume T has more than one inner nodes. Consider an inner node v of T on the longest distance from the root. All children of v are leaves. W.l.o.g. assume that the children of v are $\dot{\boldsymbol{y}}_1, \ldots, \dot{\boldsymbol{y}}_k$ for $k \geq 2$. In the multi-class case we have that $\|\dot{\boldsymbol{z}}_v\|_1 = n \cdot \sum_{i=1}^{k} p_i$, and the cost $c(v) = \deg_v \cdot n \cdot \sum_{i=1}^{k} p_i = kn \cdot \sum_{i=1}^{k} p_i$. Now let T' be the tree T with the children of v removed (while keeping the label $\dot{\boldsymbol{z}}_v$ of the new leaf v). Then $c(T, \boldsymbol{Y}) = kn \cdot \sum_{i=1}^{k} p_i + c(T', \boldsymbol{Y}')$ where $\boldsymbol{Y}'$ derived from $\boldsymbol{Y}$ by replacing the columns $\dot{\boldsymbol{y}}_1, \ldots, \dot{\boldsymbol{y}}_k$ by the column $\dot{\boldsymbol{y}}_1 \vee \ldots \vee \dot{\boldsymbol{y}}_k$. By the induction

[7] For ease of exposition, we do not require the arguments of the entropy function to sum up to 1.

hypothesis, $c(T', \boldsymbol{Y}') \geq n + \frac{3n}{\log 3} \cdot H(\sum_{i=1}^{k} p_i, p_{k+1}, \ldots, p_m)$. Let $p = \sum_{i=1}^{k} p_i$. Then we have that

$$\begin{aligned}
c(T, \boldsymbol{Y}) &= knp + c(T', \boldsymbol{Y}') \\
&\geq n + knp + \frac{3n}{\log 3} \cdot H\left(p, p_{k+1}, \ldots, p_m\right) \\
&= n + knp + \frac{3n}{\log 3} \cdot p \log\left(\frac{1}{p}\right) + \frac{3n}{\log 3} \cdot H(p_{k+1}, \ldots, p_m) \\
&\geq n + knp + \frac{3n}{\log 3} \cdot (H(p_1, \ldots, p_k) - p \log k) + \frac{3n}{\log 3} \cdot H(p_{k+1}, \ldots, p_m) \\
&= n + knp - \frac{3np \log k}{\log 3} + \frac{3n}{\log 3} \cdot H(p_1, \ldots, p_m) \\
&= n + np\left(k - \frac{3 \log k}{\log 3}\right) + \frac{3n}{\log 3} \cdot H(p_1, \ldots, p_m) \\
&\geq n + \frac{3n}{\log 3} \cdot H(p_1, \ldots, p_m)\,,
\end{aligned}$$

where the second inequality is due to Proposition 9, and the last ineqaulity $k - \frac{3 \log k}{\log 3} \geq 0$ holds for every integer $k \geq 1$. □

As an upper bound, we prove that both a ternary Shannon code and a ternary Huffman tree give an almost optimal solution in the multi-class case. Both algorithms will construct a tree T where each node (possibly except for one) has exactly three children. Remark that in the multi-class case the Hamming weight of each internal node is the sum of the Hamming weights of all leaves in its subtree (which follows from Proposition 4).

Theorem 4. *A ternary Shannon code and a ternary Huffman tree for $p_1 \leq \ldots \leq p_m$, which both can be constructed in time $O(n + m)$, solve the multi-class PLT training cost problem with an additive error of at most $3n$, i.e., the output T of the algorithm satisfies*

$$c(T, \boldsymbol{Y}) \leq \min_{T \in \mathcal{T}} c(T, \boldsymbol{Y}) + 3n\,.$$

Proof. Recall that for a leaf i corresponding to the vector $\dot{\boldsymbol{y}}_i$, len_i denotes the number of nodes on the path from i to the root of the tree. Since in ternary Shannon and Huffman trees, the degree of each node is at most 3, the total cost of these trees is at most $c(T, \boldsymbol{Y}) \leq n + 3n \cdot \sum_{i=1}^{m} (\mathrm{len}_i - 1) p_i$.

It is known that the value of the Shannon code is upper bounded by $v = \sum_{i=1}^{m} (\mathrm{len}_i - 1) p_i < \sum_{i=1}^{m} p_i \log_3\left(\frac{1}{p_i}\right) + 1$ (see, e.g., Section 5.4 in Cover and Thomas 2012). This implies that the cost of the corresponding ternary Shannon tree T is

$$\begin{aligned}
c(T, \boldsymbol{Y}) &\leq n + 3n \cdot \sum_{i=1}^{m} (\mathrm{len}_i - 1) p_i \\
&< n + 3n \left(\sum_{i=1}^{m} p_i \log_3\left(\frac{1}{p_i}\right) + 1\right) \\
&= n + \frac{3n}{\log 3} \cdot H(p_1, \ldots, p_m) + 3n\,.
\end{aligned}$$

It is also know that the weight of the ternary Huffman code is upper bounded by the same quantity $\sum_{i=1}^{m} p_i \log_3 \left(\frac{1}{p_i}\right) + 1$ (see, e.g., Section 5.8 in Cover and Thomas 2012). Thus, the same upper bound holds for a ternary Huffman tree for the PLT training cost problem. This, together with Lemma 2, implies approximation with an additive error of at most $3n$.

Now we show that in our case, PLT trees corresponding to Shannon and Huffman codes can be constructed even more efficiently. We assume a sparse representation of th input by the numbers $n_i = \|\dot{\boldsymbol{y}}_i\|_1$. From now on we will only store and work with n_i. Since all n_i are integers from 1 to n, we can sort them using Bucket sort in time $O(n+m)$. In Shannon code, the depth $|\mathrm{Path}(\ell_i)| = \lceil \log_3(1/p_i) \rceil$. We can construct the corresponding tree T going from the root. We add internal nodes one by one, and connect leaves of the corresponding depth to this tree in the ascending order of n_i. This algorithm takes one pass over the sorted data, and also runs in time $O(n+m)$. Thus, the running time of the algorithm is $O(n+m)$.

For the Huffman code, we will also store a Bucket sorting of the current set of n_i. Namely, we introduce an array $s[1 \dots n]$ where $s[i]$ equals the number of vectors of Hamming weight i. Initially, this array can be constructed in time $O(n+m)$. In each iteration of the Huffman algorithm, we choose three smallest elements, and add a new one with a larger value of n_i. To implement all iterations of this procedure, it suffices to make only one pass through the array s from the index 1 to the index n. The running time of this algorithm is then again $O(n+m)$. This concludes the proof. □

This approximation is quite tight for the multi-class case, since it implies that $\frac{1}{n}\sum_{i=1}^{n} c(T, \boldsymbol{y}_i) \le \frac{1}{n}\sum_{i=1}^{n} c(T^*_{\boldsymbol{Y}}, \boldsymbol{y}_i) + 3$ which means that the difference with respect to the optimal tree is at most 3 on average. Also, note that any algorithm for the multi-class case trivially gives an approximation for the k-sparse multi-label case. For example, the algorithm from Theorem 4 finds a solution for the k-sparse multi-label case of cost at most $k \cdot (\min_{T \in \mathcal{T}} c(T, \boldsymbol{Y}) + 3n)$.

Below we show that for the multi-class case there exists an optimal tree $T^* \in \mathcal{T}$ which has a form similar to the tree from Theorem 4: every internal node of T^* also has 2 or 3 children nodes, and the order of the leaves in T^* coincides with the order of the leaves in Theorem 4.

Lemma 3. *For the multi-class case, there exists an optimal tree $T^*_{\boldsymbol{Y}} \in \operatorname{argmin}_{T \in \mathcal{T}} c(T, \boldsymbol{Y})$ in which each internal node has 2 or 3 children. Moreover, in this tree the order of the leaves in descending order of their depths is the ascending order of their Hamming weights.*

Proof. Consider an optimal PLT tree T^*. Consider a node $v \in T^*$ having children $v_1, ..., v_k$, $k \ge 4$ (if no such node exist, T^* already has the desired structure). Assume that $\|\dot{\boldsymbol{z}}_{v_1}\|_1 \le \ldots \le \|\dot{\boldsymbol{z}}_{v_k}\|_1$. Consider the subtree $T(v)$ rooted at the node v: its root node has the cost $c(v) = k \sum_{i=1}^{k} \|\dot{\boldsymbol{z}}_{v_i}\|_1$, and the whole subtree has the cost $c(T(v), \boldsymbol{Y}) = k \sum_{i=1}^{k} \|\dot{\boldsymbol{z}}_{v_i}\|_1 + \sum_{i=1}^{k} c(T(v_i), \boldsymbol{Y})$ where $c(T(v_i), \boldsymbol{Y})$ is the cost of the subtree rooted at v_i.

We make the following transformation of the subtree rooted at v: move the children nodes $v_1, ..., v_{k-2}$ under a new node u, and make u a new child of v such that v has three child nodes (v_{k-1}, v_{k-2}, u), and u has $k-2$ child nodes $(v_1, ..., v_{k-2})$. The cost of the new subtree rooted at v is $c(T'(v), \boldsymbol{Y}) = 3 \sum_{i=1}^{k} \|\dot{\boldsymbol{z}}_{v_i}\|_1 + (k-2) \sum_{i=1}^{k-2} \|\dot{\boldsymbol{z}}_{v_i}\|_1 + \sum_{i=1}^{k} c(T(v_i), \boldsymbol{Y})$.

Let $S = \sum_{i=1}^{k} \|\dot{\boldsymbol{z}}_{v_i}\|_1$. Then we have that $\sum_{i=1}^{k-2} \|\dot{\boldsymbol{z}}_{v_i}\|_1 \leq \frac{k-2}{k} \cdot S$. Now consider the difference of the costs of the transformed and original subtrees:

$$\begin{aligned} c(T'(v), \boldsymbol{Y}) - c(T(v), \boldsymbol{Y}) &= (3-k) \sum_{i=1}^{k} \|\dot{\boldsymbol{z}}_{v_i}\|_1 + (k-2) \sum_{i=1}^{k-2} \|\dot{\boldsymbol{z}}_{v_i}\|_1 \\ &\leq (3-k)S + \frac{(k-2)^2}{k} S \\ &= S\left(\frac{4}{k} - 1\right). \end{aligned}$$

Thus, the cost of the trasformed tree never exceeds the cost of the original tree for $k \geq 4$. We will apply this transformation to every node of degree ≥ 4. Since each such transformation decreases the total number of children of nodes with more that 3 children, this process will eventually terminate. Thus, after a finite number of steps, we will get a tree where each node has degree 2 or 3.

To prove the second part of the lemma, we observe that if the ascending Hamming weight order of the leaves does not match the descending order of their depth, then we can swap two leaves without increasing the cost of the tree. Repeating this transformation a finite number of times we get the desired ordering of the leaves. □

Although Lemma 3 gives a characterization of the shape and order of an optimal tree, the number of trees of this form is still exponential in m.

5.4 Nested multi-label case (Matryoshka label structure)

In this section, we study the case where the labels have nested structure which is also known as Matryoshka structure. For $\dot{\boldsymbol{y}}, \dot{\boldsymbol{y}}' \in \{0,1\}^n$, we say $\dot{\boldsymbol{y}} \leq \dot{\boldsymbol{y}}'$ if $\forall i \in [n]\colon y_i \leq y'_i$. In this section we will assume that the m vectors of $\boldsymbol{Y}$ satisfy $\dot{\boldsymbol{y}}_1 \leq \ldots \leq \dot{\boldsymbol{y}}_m$. We also assume that $\dot{\boldsymbol{y}}_1$ contains at least one positive element: $\|\dot{\boldsymbol{y}}_1\|_1 \geq 1$. We start with two structural results for an optimal tree in this case.

Lemma 4. *Let $\boldsymbol{Y}$ be an instance of the nested multi-label case. There exists an optimal tree $T^*_{\boldsymbol{Y}} \in \operatorname{argmin}_{T \in \mathcal{T}} c(T, \boldsymbol{Y})$ where each node has at most one internal node among its children.*

Proof. Consider an optimal tree $T \in \operatorname{argmin}_{T' \in \mathcal{T}} c(T', \boldsymbol{Y})$. Let v be a node with two internal nodes v_1 and v_2 among its children. Let $v_3, \ldots, v_k$ be the remaining children of v (leaves and internal nodes). W.l.o.g. assume that $\|\dot{\boldsymbol{z}}_{v_1}\|_1 \leq \|\dot{\boldsymbol{z}}_{v_2}\|_1$. We construct a tree T' such that $c(T', \boldsymbol{Y}) \leq c(T, \boldsymbol{Y})$, and T' has fewer nodes with two inner nodes among their children. Repeatedly applying this procedure we will get an optimal tree without nodes with more than one inner node among its children.

We define the tree T' as T where the node v_1 is no longer a child of v but rather a child of v_2. It is easy to see that the only two nodes which change their costs after this transformation are v and v_2. The cost of v is decreased by $\|\dot{\boldsymbol{z}}_v\|_1$ (because the $\deg_v$ is decreased by 1) while the cost of v_2 is increased by $\|\dot{\boldsymbol{z}}_{v_2}\|_1$ (because the $\deg_{v_2}$ is increased by 1). Note that $\|\dot{\boldsymbol{z}}_{v_2}\|_1 \leq \|\dot{\boldsymbol{z}}_v\|_1$ (since v is the parent of v_2 in T). This completes the proof.

□

Lemma 5. *Let $\boldsymbol{Y}$ be an instance of the nested multi-label case. There exists a $k \geq 1$, indices $1 = i_1 < \ldots < i_k = m$, and an optimal PLT tree T^* which has the following form:*

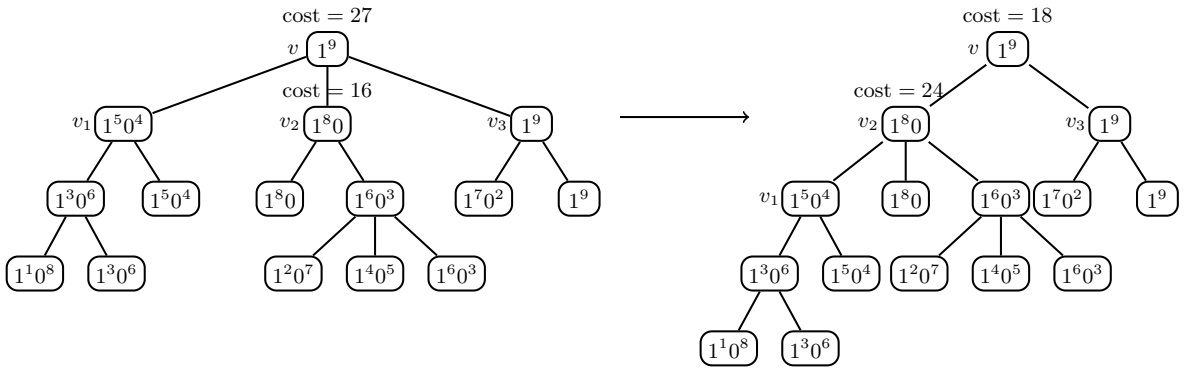

Figure 2: An example of the transformation from Lemma 4. The total cost of the two nodes v and v_2 which change their costs under this transformation is reduced from 43 to 42.

- *the internal nodes are denoted by* $v_1, \ldots, v_{k-1}$*, the root is* v_{k-1}*, and the leaves are* $\dot{\boldsymbol{y}}_1, \ldots, \dot{\boldsymbol{y}}_m$;
- *for* $k > j \geq 2$*, the internal node* v_j *has children* v_{j-1} *and* $\dot{\boldsymbol{y}}_{i_j+1}, \ldots, \dot{\boldsymbol{y}}_{i_{j+1}}$;
- *the node* v_1 *has children* $\dot{\boldsymbol{y}}_{i_1}, \ldots, \dot{\boldsymbol{y}}_{i_2}$.

Proof. For $\boldsymbol{Y} = [\dot{\boldsymbol{y}}_1, \ldots, \dot{\boldsymbol{y}}_m]$ such that $\dot{\boldsymbol{y}}_1 \leq \ldots \leq \dot{\boldsymbol{y}}_m$, by Lemma 4, there exists an optimal tree T^* where each internal node has at most one internal node among its children. Therefore, there exists some k, indices $1 = i_1 < \ldots < i_k = m$, and a permutation π such that T^* is as follows:

- the internal nodes are denoted by $v_1, \ldots, v_{k-1}$, the root is v_{k-1}, and the leaves are $\dot{\boldsymbol{y}}_1, \ldots, \dot{\boldsymbol{y}}_m$;
- for $k > j \geq 2$, the internal node v_j has children v_{j-1} and $\dot{\boldsymbol{y}}_{\pi(i_j+1)}, \ldots, \dot{\boldsymbol{y}}_{\pi(i_{j+1})}$;
- the node v_1 has children $\dot{\boldsymbol{y}}_{i_1}, \ldots, \dot{\boldsymbol{y}}_{i_2}$.

Thus, it suffices to show that the identity permutation $\pi = \mathrm{id}$ minimizes the cost of the tree $c(T^*, \boldsymbol{Y})$. If $\pi = \mathrm{id}$, then the cost of the internal node v_j is $c(v_j)_{\mathrm{id}} = \deg_{v_j} \|\dot{\boldsymbol{z}}_{v_{i_j+1}}\|_1 = (i_{j+1} - i_j + 1)\|\dot{\boldsymbol{y}}_{i_j+1}\|_1$. Note that this holds for v_1 as well, since its children are $\dot{\boldsymbol{y}}_1, \ldots, \dot{\boldsymbol{y}}_{i_2}$. Now, assume that there exists a permutation π' for which we get a smaller cost of the tree. Then, the cost $c(v_j)_{\pi'}$ of at least one internal node v_j is smaller under the permutation π': $c(v_j)_{\pi'} < c(v_j)_{\mathrm{id}}$. Note that

$$c(v_j)_{\pi'} = \deg_{v_j} \|\dot{\boldsymbol{z}}_{v_{i_j+1}}\|_1 = (i_{j+1} - i_j + 1) \max_{1 \leq k \leq i_{j+1}} \|\dot{\boldsymbol{y}}_{\pi'(k)}\|_1 \geq (i_{j+1} - i_j + 1)\|\dot{\boldsymbol{y}}_{i_j+1}\|_1 = c(v_j)_{\mathrm{id}}$$

which finishes the proof. □

Note that by the nested property, an inner node v_j for $j \geq 2$ has cost $c(v_j) = \deg_{v_j} \|\dot{\boldsymbol{z}}_{v_j}\|_1 = (i_{j+1} - i_j + 1)\|\dot{\boldsymbol{y}}_{i_j+1}\|_1$. Thus, the problem of finding an optimal solution of the form guaranteed by Lemma 5 can be stated as follows: Given a sequence of m non-negative integers $A = (\|\dot{\boldsymbol{y}}_1\|_1, \ldots, \|\dot{\boldsymbol{y}}_m\|_1)$ such that $\|\dot{\boldsymbol{y}}_1\|_1 \leq \ldots \leq \|\dot{\boldsymbol{y}}_m\|_1$, we need to partition A into k contiguous subsequences $S_1, \ldots, S_k$, in order to minimize the value of $|S_1| \cdot \max(S_1) + \sum_{j=2}^{k} (|S_j| + 1) \cdot \max(S_j)$.

Given the structural result of Lemma 5, one can find an optimal solution for the nested multi-label case by a simple dynamic programming algorithm in quadratic time $(m^2 + mn)$. Instead, we will show that this problem can actually be solved in linear time $O(m)$, assuming a sparse representation

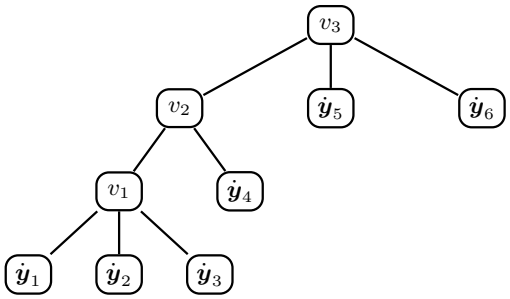

Figure 3: An example of an optimal solution from Lemma 5.

of the input, for example as a sequence of the numbers $\|\dot{\boldsymbol{y}}_1\|_1 \leq \ldots \leq \|\dot{\boldsymbol{y}}_m\|_1$. For this, we will reduce the problem to the concave least-weight sequence problem in time $O(m)$, and the latter problem can be solved in linear time $O(m)$ (Wilber, 1988).

Definition 4 (Concave least-weight sequence)**.** *Let n be an integer, and $w(i,j)$ be a real-valued function defined for integers $0 \leq i,j \leq n$ with the property that $w(i_0,j_0)+w(i_1,j_1) \leq w(i_0,j_1)+w(i_1,j_0)$ for all $0 \leq i_0 < i_1 < j_0 < j_1 \leq n$. The concave least-weight sequence problem is to find an integer $k \geq 1$ and a sequence of integers $0 = \ell_0 < \ell_1 < \ldots < \ell_{k-1} < \ell_k = n$ such that $\sum_{j=0}^{k-1} w(l_i, l_{i+1})$ is minimized.*

Now we are ready to present the main result of this section.

Theorem 5. *There exists an $O(m)$ time algorithm that solves the nested multi-label PLT training cost problem exactly.*

Proof. Recall that it suffices to reduce the problem of partitioning $A = (a_1, \ldots, a_m)$ with $a_1 \leq \ldots \leq a_m$ into k contiguous sequences $S_1, \ldots, S_k$ minimizing $|S_1| \cdot \max(S_1) + \sum_{j=2}^{k}(|S_j|+1) \cdot \max(S_j)$ to the concave least-weight sequence problem.

We will define the function $w(\cdot,\cdot)$ of the concave least-weight sequence problem such that for $i < j$, $w(i,j)$ corresponds to taking the set $S = \{a_{i+1}, \ldots, a_j\}$. Formally,

$$w(i,j) = \begin{cases} (j-i+1)a_j, & \text{if } 0 < i < j\,; \\ ja_j, & \text{if } 0 = i < j\,. \end{cases}$$

It now remains to show that the function $w(\cdot,\cdot)$ is concave: for all $0 \leq i_0 < i_1 < j_0 < j_1 \leq n$, $w(i_0,j_0)+w(i_1,j_1) \leq w(i_0,j_1)+w(i_1,j_0)$. For this, consider the following two cases.

Case 1: $i_0 > 0$.

$$\begin{aligned} &w(i_0,j_0)+w(i_1,j_1)-w(i_0,j_1)-w(i_1,j_0) \\ &\quad = (j_0-i_0+1)a_{j_0} + (j_1-i_1+1)a_{j_1} - (j_1-i_0+1)a_{j_1} - (j_0-i_1+1)a_{j_0} \\ &\quad = (i_1-i_0)a_{j_0} - (i_1-i_0)a_{j_1} \\ &\quad = (i_1-i_0)(a_{j_0}-a_{j_1}) \\ &\quad \leq 0\,. \end{aligned}$$

Case 2: $i_0 = 0$.

$$\begin{aligned} & w(i_0, j_0) + w(i_1, j_1) - w(i_0, j_1) - w(i_1, j_0) \\ & \quad = (j_0 - i_0)a_{j_0} + (j_1 - i_1 + 1)a_{j_1} - (j_1 - i_0)a_{j_1} - (j_0 - i_1 + 1)a_{j_0} \\ & \quad = (1 - i_1)a_{j_1} - (1 - i_1)a_{j_0} \\ & \quad = (1 - i_1)(a_{j_1} - a_{j_0}) \\ & \quad \leq 0\,. \end{aligned}$$

□

6 Prediction complexity

We consider a prediction for a feature vector $\boldsymbol{x}$ in which we find all labels such that:

$$\hat{\eta}_j(\boldsymbol{x}) \geq \tau\,, \quad j \in \mathcal{L}\,,$$

where $\tau \in [0, 1]$ is a threshold. A natural choice of τ is 0.5 as it leads to the optimal predictions for the Hamming loss (Dembczyński et al., 2012). The threshold-based prediction can also be used for maximizing the micro- and macro-F measures as shown in (Kotlowski and Dembczyński, 2016; Koyejo et al., 2015). Consider the tree search procedure presented in Algorithm 2. It starts with the root node and traverses the tree by visiting the nodes $v \in V_T$ for which $\hat{\eta}_{pa(v)}(\boldsymbol{x}) \geq \tau$. Obviously, the prediction consists of labels corresponding to the visited leaves for which $\hat{\eta}_{\ell_j}(\boldsymbol{x}) \geq \tau$.

Let us define formally the *prediction cost* $c_\tau(T, \boldsymbol{x})$ for a single instance $\boldsymbol{x}$ as the number of calls to node classifiers in the Algorithm 2 during prediction, i.e.:[8]

$$c_\tau(T, \boldsymbol{x}) = 1 + \sum_{v \in V_T \setminus r_T} \mathbb{I}\left\{\hat{\eta}_{pa(v)}(\boldsymbol{x}) \geq \tau\right\} = 1 + \sum_{v \in V_T} \mathbb{I}\left\{\hat{\eta}_v(\boldsymbol{x}) \geq \tau\right\} \cdot \deg_v.$$

The *expected prediction cost* is then $C_{\mathbf{P}(\boldsymbol{x}),\tau}(T) = \mathbb{E}_{\boldsymbol{x}}[c_\tau(T, \boldsymbol{x})]$.

Algorithm 2 PLT.PREDICT$(T, \boldsymbol{x}, \tau)$

```
1:  ŷ = 0, Q = ∅                          ▷ Initialize the prediction vector to all zeros and a stack
2:  Q.add((r_T, η̂(x, r_T)))               ▷ Add the tree root and the corresponding estimate of probability
3:  while Q ≠ ∅ do                        ▷ In the loop
4:      (v, η̂_v(x)) = Q.pop()             ▷ Pop an element from the stack
5:      if η̂_v(x) ≥ τ then                ▷ If the probability estimate is greater or equal τ
6:          if v is a leaf then           ▷ If the node is a leaf
7:              y_v = 1                   ▷ Set the corresponding label in the prediction vector
8:          else                          ▷ If the node is an internal node
9:              for v' ∈ ch(v) do         ▷ For all child nodes
10:                 η̂_v'(x) = η̂_v(x) × η̂(x, v')      ▷ Compute η̂_v'(x)
11:                 Q.add((v', η̂_v'(x)))              ▷ Add the node and the computed probability estimate
12: return ŷ                              ▷ Return the prediction vector
```

To express the complexity of the prediction algorithm for a given $\boldsymbol{x}$ in terms of the depth and the degree of T, we assume that $\sum_{i=1}^{m} \eta_j(\boldsymbol{x})$ is upperbounded by a constant P (e.g., for the

[8] Notice that the time complexity of Algorithm 2 is $O(c_\tau(T, \boldsymbol{x}))$.

case of k-sparse multi-label classification we have $P = k$). Let us denote the L_1-estimation error in each node $v \in V_T$ by ϵ_v, i.e., $\epsilon_v = |\eta(\boldsymbol{x}, v) - \hat{\eta}(\boldsymbol{x}, v)|$. Then, from Theorem 1, we have that $|\eta_v(\boldsymbol{x}) - \hat{\eta}_v(\boldsymbol{x})| \leq \sum_{v' \in \mathrm{Path}(v)} \eta_{\mathrm{pa}(v)}(\boldsymbol{x}) \cdot \epsilon_v$. Assuming that $\hat{\eta}_v(\boldsymbol{x})$ are properly normalized to satisfy Proposition 1, we prove the following result which is a counterpart of Proposition 2 for training cost.

Theorem 6. *For Algorithm 2 with threshold τ and any $\boldsymbol{x} \in \mathcal{X}$, we have that:*

$$c_\tau(T, \boldsymbol{x}) \leq 1 + \lfloor \hat{P}/\tau \rfloor \cdot \mathrm{depth}_T \cdot \deg_T \,, \tag{9}$$

where $\hat{P} = \sum_{j=1}^m \hat{\eta}_j(\boldsymbol{x}) \leq P + \sum_{v \in V_T} |L(v)| \eta_{\mathrm{pa}(v)}(\boldsymbol{x}) \cdot \epsilon_v$, $\mathrm{depth}_T = \max_{v \in L_T} \mathrm{len}_v - 1$, and $\deg_T = \max_{v \in V_T} \deg_v$.

Proof. Let us first show an upper bound on $\hat{P}$:

$$\begin{aligned} \hat{P} = \sum_{j=1}^m \hat{\eta}_j(\boldsymbol{x}) &\leq \sum_{j=1}^m \left(\eta_j(\boldsymbol{x}) + \sum_{v \in \mathrm{Path}(\ell_j)} \eta_{\mathrm{pa}(v)}(\boldsymbol{x}) \cdot \epsilon_v \right) \qquad (10) \\ &\leq P + \sum_{j=1}^m \sum_{v \in \mathrm{Path}(\ell_j)} \eta_{\mathrm{pa}(v)}(\boldsymbol{x}) \cdot \epsilon_v \\ &= P + \sum_{v \in V_T} |L(v)| \eta_{\mathrm{pa}(v)}(\boldsymbol{x}) \cdot \epsilon_v \,, \end{aligned}$$

where (10) follows from Theorem 1 and P is an upper bound on $\sum_{j=1}^m \eta_j(\boldsymbol{x})$.

As stated before the theorem, we assume that the estimates satisfy the following:

$$\hat{\eta}_v(\boldsymbol{x}) \leq \min \left\{ 1, \sum_{v' \in \mathrm{Ch}(v)} \hat{\eta}_{v'}(\boldsymbol{x}) \right\} , \tag{11}$$

and

$$\max \left\{ \hat{\eta}_{v'}(\boldsymbol{x}), v' \in \mathrm{Ch}(v) \right\} \leq \hat{\eta}_v(\boldsymbol{x}) \,. \tag{12}$$

These are important properties of the true probabilities stated in Proposition 1. To satisfy them by the estimates we can perform the following normalization steps for the child nodes during prediction:

$$\begin{aligned} \hat{\eta}_{v'}(\boldsymbol{x}) &\leftarrow \min(\hat{\eta}_{v'}(\boldsymbol{x}), \hat{\eta}_v) \,, \quad \text{for all } v' \in \mathrm{Ch}(v) \,, \\ \hat{\eta}_{v'}(\boldsymbol{x}) &\leftarrow \frac{\hat{\eta}_{v'}(\boldsymbol{x}) \cdot \hat{\eta}_v}{\sum_{v' \in \mathrm{Ch}(v)} \hat{\eta}_{v'}(\boldsymbol{x})} \,, \quad \text{if } \hat{\eta}_v > \sum_{v' \in \mathrm{Ch}(v)} \hat{\eta}_{v'}(\boldsymbol{x}) \,, \quad \text{for all } v' \in \mathrm{Ch}(v) \,. \end{aligned}$$

The error terms ϵ_v concern then the normalized estimates.

Now, we move to the main part of the proof. Consider the subtree T' of T, which consists of all nodes $v \in T$ for which $\hat{\eta}_v(\boldsymbol{x}) \geq \tau$. If there are no such nodes, from the pseudocode of Algorithm 2, we see that only the root classifier is called. The upperbound (9) in this case obviously holds. However, it might not be tight as $\hat{\eta}_v(\boldsymbol{x}) < \tau$ does not imply $\hat{P} \geq \tau$ because of (11).

If T' has at least one node, Algorithm 2 visits each node of T' (i.e., calls a corresponding classifier and add the node to a stack), since for each parent node we have (12). Moreover, Algorithm 2 visits

all children of nodes T' (some of them are already in T'). Let the subtree T'' consist of all nodes of T' and their child nodes. Certainly $T' \subseteq T'' \subseteq T$. To prove the theorem we count first the number of nodes in T' and then the number of nodes in T'', which gives as the final result.

If the number of nodes in T' is greater than or equal to 1, then certainly r_T is in T'. Let us consider next the number of leaves of T'. Observe that $\sum_{v \in L_{T'}} \hat{\eta}_v(\boldsymbol{x}) \leq \hat{P}$. This is because $\sum_{v \in L_{T'}} \hat{\eta}_v(\boldsymbol{x}) \leq \sum_{v \in L_T} \hat{\eta}_v(\boldsymbol{x}) \leq \hat{P}$, i.e., $v \in L_{T'}$ might be an internal node in T and its $\hat{\eta}_v(\boldsymbol{x})$ is at most the sum of probability estimates of the leaves underneath v according to (11). From this we get the following upper bound on the number of leaves in T':

$$|L_{T'}| \leq \lfloor \hat{P}/\tau \rfloor \,. \tag{13}$$

Since the degree of internal nodes in T' might be 1, to upperbound the number of all nodes in T' we count the number of nodes on all paths from leaves to the root, but counting the root node only once, i.e.:

$$|V_{T'}| \leq 1 + \sum_{v \in L_{T'}} (\mathrm{len}_v - 1) \,.$$

Next, notice that for each $v \in T'$ its all siblings are in T'' unless v is the root node. This is because if non-root node v is in T' then its parent is also in T' according to (12) and T'' contains all child nodes of nodes in T'. The rest of nodes in T'' are the child nodes of leaves of T', unless a leaf of T' is also a leaf of T. Therefore, we have

$$|V_{T''}| \leq 1 + \sum_{v \in L_{T'}} \deg_T (\mathrm{len}_v - 1) + \sum_{v \in L_{T'}} \deg_T \mathbb{I}\{v \notin L_T\} \,,$$

with $\deg_T$ being the highest possible degree of a node. Since (13) and

$$\mathrm{len}_v - 1 + \mathbb{I}\{v \notin L_T\} \leq \mathrm{depth}_T \,,$$

i.e., the longest path cannot be longer than the depth of the tree plus 1, we finally get:

$$|V_{T''}| \leq 1 + \lfloor \hat{P}/\tau \rfloor \cdot \mathrm{depth}_T \cdot \deg_T \,.$$

This ends the proof as the number of nodes in T'' is equivalent to the number of calls to the node classifiers, i.e., $c_\tau(T, \boldsymbol{x})$.

□

Remark 2. *For a tree of constant* $\deg_T = \lambda (\geq 2)$ *and* $\mathrm{depth}_T = \log_\lambda m$*, the cost of Algorithm 2 is* $O(\log m)$ *if* $\hat{P}$ *is upper bounded by a constant or node classifiers predict with no error, i.e.,* $\epsilon_v = 0$*, for all* $v \in V_T$*.*

The above result does not, however, relate directly the prediction cost to the training cost. The next theorem shows this relation in terms of expected costs.

Theorem 7. *Using the notation above, it holds that*

$$C_{\mathbf{P}(\boldsymbol{x}),\tau}(T) \leq \frac{1}{\tau} \left(C_{\mathbf{P}}(T) + \sum_{v \in V_T} \mathbb{E}_{\boldsymbol{x}} \left[\eta_{\mathrm{pa}(v)}(\boldsymbol{x}) \cdot \epsilon_v \right] \cdot |T(v)| \cdot \deg_v \right) - \frac{1-\tau}{\tau},$$

where $|T(v)|$ *denotes the number of inner nodes in the subtree* $T(v)$ *rooted at* v*.*

Proof. We can upper bound the expected inference cost as follows:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}}\left[c_\tau(T,\boldsymbol{x})\right] &= \mathbb{E}_{\boldsymbol{x}}\left[1+\sum_{v\in V_T}\mathbb{I}\left\{\hat{\eta}_v(\boldsymbol{x})\geq\tau\right\}\deg_v\right] \leq \mathbb{E}_{\boldsymbol{x}}\left[1+\sum_{v\in V_T}\frac{\hat{\eta}_v(\boldsymbol{x})}{\tau}\deg_v\right] \\
&\leq 1+\sum_{v\in V_T}\frac{\mathbb{E}_{\boldsymbol{x}}\left[\eta_v(\boldsymbol{x})+|\hat{\eta}_v(\boldsymbol{x})-\eta_v(\boldsymbol{x})|\right]}{\tau}\deg_v \\
&\leq \frac{1}{\tau}\left(1+\sum_{v\in V_T}\mathbb{E}_{\boldsymbol{x}}\left[\eta_v(\boldsymbol{x})+\sum_{v'\in\mathrm{Path}(v)}\eta_{\mathrm{pa}(v')}(\boldsymbol{x})\cdot\epsilon_v\right]\cdot\deg_v\right)-\frac{1-\tau}{\tau} \qquad (14) \\
&\leq \frac{1}{\tau}\left(C_{\mathbf{P}}(T)+\sum_{v\in V_T}\mathbb{E}_{\boldsymbol{x}}\left[\eta_{\mathrm{pa}(v)}(\boldsymbol{x})\cdot\epsilon_v\right]\cdot|T(v)|\cdot\deg_v\right)-\frac{1-\tau}{\tau}, \qquad (15)
\end{aligned}$$

where (14) follows from Theorem 1. □

Proposition 10. *The bound in Theorem 7 is tight.*

Proof. We construct a tight example as follows. First, we assume that the classifier predicts with no error, $\epsilon_v = 0$ for all $\boldsymbol{x}, v \in V_T$. Second, we consider all node probabilities $\eta(\boldsymbol{x}, v) = 1$ for all $v \in V_T$ except for the root node, for which $\eta(\boldsymbol{x}, r) = \tau$. In this case all conditional probabilities $\eta_j(\boldsymbol{x}) = \tau$, $j \in \mathcal{L}$, and we can write the exact values of the expected costs:

$$C_{\mathbf{P}}(T) = 1 + \sum_{v\in V_T}\tau\deg_v = \tau\cdot|V_T|+1,$$

and

$$C_{\mathbf{P}(\boldsymbol{x}),\tau}(T) = 1 + \sum_{v\in V_T}\deg_v = |V_T|+1.$$

□

Corollary 1. *In the case of exact predictions, $\epsilon_v = 0$ for all $\boldsymbol{x}, v \in V_T$, for $\tau = 0.5$ we have*

$$C_{\mathbf{P}(\boldsymbol{x}),0.5}(T) \leq 2C_{\mathbf{P}}(T) - 1.$$

Remark 3. *A natural question is whether the lower bound on $C_{\mathbf{P}(\boldsymbol{x}),\tau}(T)$ exists of the form*

$$C_{\mathbf{P}(\boldsymbol{x}),\tau}(T) = \Omega\left(C_{\mathbf{P}}(T)\right)$$

under the assumption $\epsilon_v = 0$ for all $\boldsymbol{x}, v \in V_T$. To see that no such bound exists without additional assumptions on $\eta_v(\boldsymbol{x})$, one can consider the following example: let all $\eta_v(\boldsymbol{x}) = \tau - \varepsilon$, $v \in V_T$, for $\varepsilon > 0$. In this case we have

$$C_{\mathbf{P}}(T) = (\tau-\varepsilon)\cdot|V_T|+1, \quad C_{\mathbf{P}(\boldsymbol{x}),\tau}(T) = 1.$$

Remark 4. *The above theorems contain the term $\eta_{\mathrm{pa}(v)}(\boldsymbol{x})\cdot\epsilon_v$ either multiplied by $|L(v)|$ or $|T(v)|$. It nicely shows an interplay between the L_1-estimation error (ϵ_v) and the* importance *($\eta_{\mathrm{pa}(v)}(\boldsymbol{x})$ multiplied by $|L(v)|$ or $|T(v)|$) of node $v \in V_T$. For example, the root node has the highest importance, but the error there should be the smallest as all training examples are used there and the learning problem is relatively simple as the root classifier has to estimate the probability whether there exists at least one label in $\boldsymbol{y}$. In many cases, we can assume that this probability is 1 and the error will be 0 then. In turn, the error in the leaves nodes can be substantial as the number of training examples there is the smallest, but their importance is also the smallest.*

7 Conclusions and future work

In this paper, we addressed the problem of optimizing the training and test costs of PLTs. We showed that optimizing the training cost is an NP-complete problem, nevertheless it has several tractable special cases for which either exact or approximate solution can be efficiently found. We also show guarantees for the test cost and characterize its relation to the training cost.

Several exciting open questions have arisen from this work. One is to prove either a reasonable lower bound for the general multi-label case or the tightness of the $O(\log m)$ approximation. Second is to find a tree with the optimal statistical error and show its relation to the computational cost objective.

References

Z. Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *J. Mach. Learn. Res.*, 18(1):8194–8244, 2017.

A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl. Conditional probability tree estimation analysis and algorithms. In *UAI 2009*, pages 51–58. AUAI Press, 2009a.

A. Beygelzimer, J. Langford, and P. Ravikumar. Error-correcting tournaments. In *ALT 2009*, pages 247–262. Springer, 2009b.

S. Boucheron, G. Lugosi, and P. Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University press, 2013.

T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

K. Dembczyński, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286, 2010.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On loss minimization and label dependence in multi-label classification. *Mach. Learn.*, 88:5–45, 2012.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

M. Ghosh, I. Gupta, S. Gupta, and N. Kumar. Fast compaction algorithms for NoSQL databases. In *ICDCS 2015*, pages 452–461, 2015.

E. Grave, A. Joulin, M. Cissé, D. Grangier, and H. Jégou. Efficient softmax approximation for GPUs. In *ICML 2017*, pages 1302–1310. PMLR, 2017.

K. Jasinska, K. Dembczyński, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hüllermeier. Extreme F-measure maximization using sparse probability estimates. In *ICML 2016*, pages 1435–1444. PMLR, 2016.

Y. Jernite, A. Choromanska, and D. Sontag. Simultaneous learning of trees and representations for extreme classification and density estimation. In *ICML 2017*, pages 1665–1674. PMLR, 2017.

A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. In *EACL 2017*, pages 427–431, 2017.

W. Kotlowski and K. Dembczyński. Surrogate regret bounds for generalized classification performance metrics. In *ACML 2016*, pages 301–316, 2016.

O. Koyejo, N. Natarajan, P. Ravikumar, and I. S. Dhillon. Consistent multilabel classification. In *NIPS 2015*, pages 3321–3329, 2015.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*, pages 3111–3119, 2013.

F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *AISTATS 2005*, pages 246–252, 2005.

Y. Prabhu and M. Varma. FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD 2014*, pages 263–272. ACM, 2014.

Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW 2018*, pages 993–1002. ACM, 2018.

R. Wilber. The concave least-weight subsequence problem revisited. *J. Algorithms*, 9(3):418–425, 1988.

M. Wydmuch, K. Jasinska, M. Kuznetsov, R. Busa-Fekete, and K. Dembczyński. A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *NeurIPS 2018*, pages 6358–6368, 2018.

A Proof of Theorem 2

Theorem 2. *The PLT training cost problem is NP-complete.*

Proof. It is easy to see that PLT training cost $\in$ NP. Indeed, given an optimal PLT training cost tree, one can in polynomial time verify that its cost is at most w.[9]

Now we will show that PLT training cost is NP-hard by giving a polynomial time reduction from Clique to PLT training cost. The reduction consists of two steps. In the first step, we reduce Clique to Directed Clique with $k = \frac{2|V|}{3}$. In the second step, we reduce this version of the Directed Clique problem to PLT training cost.

Preparations. Let $G' = (V', E')$ and $1 \leq k' \leq |V'|$ be an instance of the Clique problem. W.l.o.g. let us assume that G' does not have isolated nodes. We will now construct an undirected graph $G^* = (V^*, E^*)$ and $k^* = \frac{2|V^*|}{3}$ such that G' contains a clique of size k' if and only if G^* contains a cliques of size k^*.

[9] We also need to verify that there exists an optimal tree of polynomial size. Since there always exists an optimal tree where each internal node has at least 2 children, there exists an optimal tree with at most $m - 1$ internal nodes.

First, we add all nodes and edges of G' to G^*. If $k' \leq \frac{2|V'|}{3}$, then we add $2|V'| - 3k'$ nodes to G^* which are connected to all nodes of G^*. It is now easy to see that G' contains a k'-cliques if and only if G^* contains a $k^* = \frac{2|V^*|}{3}$-clique.

If $k' > \frac{2|V'|}{3}$, then we first make k' even. Namely, if k' is odd, we increase k' by one and add one node to G' connected to all other nodes. Now we add a path on $\frac{3k-2|V'|}{2}$ nodes to G^*, and connect one node of the path to an arbitrary node of G. Again, it is easy to see that the new instance of the clique problem (with $k^* = \frac{2|V^*|}{3}$) is equivalent to the original one. We note that the constructed graph does not have isolated nodes.

We now construct a graph $G = (V, E)$ and $k = \frac{2|V|}{3}$ such that it contains a directed k-clique (with self-loops) if and only if G' has a k'-clique. To this end, we set $V = V^*$, for each edge $\{u, v\}$ we create two arcs (u, v) and (v, u) in G, and we also add all $|V|$ self-loops in G.

Reduction. Let $n = |V|$ and $m = |E|$, w.l.o.g. assume that $n \geq 30$ and n is a multiple of 30. Now we reduce the Clique problem with $k = \frac{2n}{3}$ to an instance of PLT training cost with m vectors in $d = n + \ell$ dimension where

$$\ell = 0.145n^3 \ .$$

(Since n is a multiple of 10, ℓ is an integer.) For every arc $e = (v_i, v_j)$ of the graph G, we create a Boolean vector $\dot{\boldsymbol{y}}_e \in \{0, 1\}^d$ as follows. In the first n coordinates, we set only the ith and jth coordinates to 1 (if e is a self-loop, we only set one coordinate to 1). The last ℓ coordinates are always set to 1. Now we set

$$w = \ell(m + 1) + k^3 + nm - nk^2 + n = \ell(m + 1) - \frac{4n^3}{27} + nm + n \ .$$

We claim that this instance of the PLT training cost problem has a $T \in \mathcal{T}$ tree of cost at most $w + n$ if and only if G contains a k-clique. We also remark that both reductions run in time polynomial in n and m.

Correctness. First we show that if G contains a k-clique, then there is a $T \in \mathcal{T}$ of cost at most $w + n$. We construct a tree with two internal nodes (including the root) as follows. All the k^2 arcs of the directed k-clique (with self-loops) are connected to the internal node v. This internal node and the remaining $m - k^2$ arcs are connected to the root r. Let us now compute the labels $\dot{\boldsymbol{z}}_v$ and $\dot{\boldsymbol{z}}_r$. Since all children of v correspond to the edges forming a k-clique, the vector $\dot{\boldsymbol{z}}_v$ has exactly k ones in the first n coordinates, and it has all ℓ ones in the remaining coordinates. Since there are no isolated nodes in G, the vector $\dot{\boldsymbol{z}}_r$ is 1^d. Now the cost $c(v) = (k + \ell)k^2$, $c(r) = (n + \ell)(m - k^2 + 1)$, and the total cost of the constructed tree is

$$c(T, \boldsymbol{Y}) = n + c(v) + c(r) = n + (k + \ell)k^2 + (n + \ell)(m - k^2 + 1) = n + w \ .$$

Now we will show that a tree T of cost at most $w + n$ gives us a k-clique in G. To this end, we will consider three cases. Let i be the number of internal nodes of T (recall that the root counts as an internal node, thus, $i \geq 1$). We will show that if $i = 1$ or $i > 3$, then the cost of T is larger than $w + n$, and if $i = 2$ then any tree of cost $w + n$ must look exactly as the one above, which gives us a k-clique.

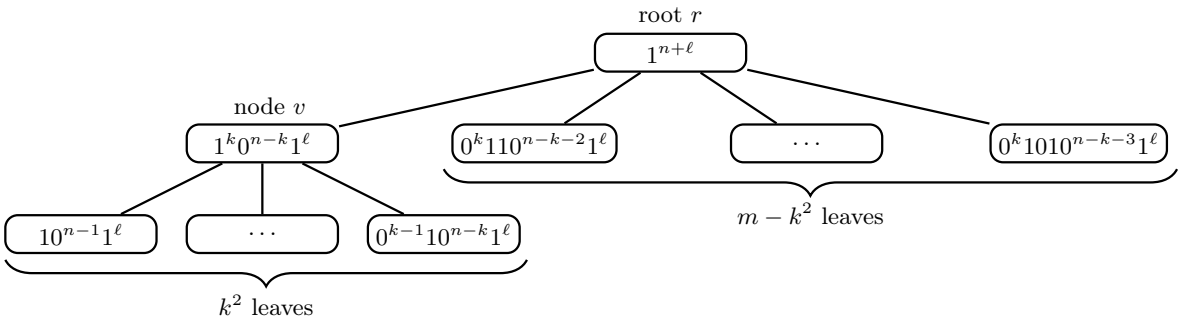

Figure 4: The tree corresponding to a k-clique. All k^2 arcs forming the clique are connected to the internal node v on the left, all the remaining arc are connected dicrectly to the root r. The label of the root is $\dot{\boldsymbol{z}}_r = 1^d = 1^{n+\ell}$, and the label $\dot{\boldsymbol{z}}_v$ of the node v has k ones in the first part (corresponding to the nodes of the clique) and ℓ ones in the second part.

- $i = 1$. If T has only one internal node (the root), then the cost of the tree is the cost of the root plus n. Since there are no isolated nodes in G, the label of the root $\dot{\boldsymbol{z}}_r$ has all ones. Thus,

$$c(r) = (n+\ell)m = nm + \ell(m+1) - 0.125n^3 > \ell(m+1) - \frac{4n^3}{27} + nm + n = w$$

  for $n \geq 7$. This implies that $c(T, \boldsymbol{Y}) = n + c(r) > n + w$.

- $i = 2$. Assume that one internal node v is connected to e leaves which span $t \leq n$ nodes of G, where $e \leq t^2$, and the root r is connected to this internal node v and the remaining $m - e$ leaves. Both $\dot{\boldsymbol{z}}_v$ and $\dot{\boldsymbol{z}}_r$ have ℓ ones in the last ℓ coordinates, and these two nodes v and r together have $(m+1)$ children. Thus, the total contribution to the cost $c(T, \boldsymbol{Y})$ of the last ℓ coordinates of $\dot{\boldsymbol{z}}_v$ and $\dot{\boldsymbol{z}}_r$ is $(m+1)\ell$. The root $\dot{\boldsymbol{z}}_r$ has n ones in the first n coordinates, and $\dot{\boldsymbol{z}}_v$ has t ones. Therefore, the contribution of the first n coordinates of $\dot{\boldsymbol{z}}_v$ and $\dot{\boldsymbol{z}}_r$ to $c(T, \boldsymbol{Y})$ is $et + (m - e + 1)n$. Now we have that $c(r) + c(v) = (m+1)\ell + et + (m - e + 1)n$. We will show that $c(r) + c(v) \leq w$ if and only if $e = k^2$ and $t = k$ which corresponds to a k-clique in the original graph.

$$c(r) + c(v) - w = et - en + \frac{4n^3}{27} \geq t^3 - t^2 n + \frac{4n^3}{27} \, .$$

  By taking the derivative with respect to t, we see that this expession is greater than 0 for $t \neq \frac{2n}{3}$. Thus, we have that $t = \frac{2n}{3} = k$, and $e = k^2$.

- $i \geq 3$. Each inner node has ℓ ones in the last ℓ coordinates, and all inner nodes together have $(m + i - 1)$ children. This means that the last ℓ coordinates of the corresponding vectors $\dot{\boldsymbol{z}}$ contribute $(m+i-1)\ell$ to the cost $c(T, \boldsymbol{Y})$. Let $m_1, \ldots, m_i$ be the numbers of leaves connected to each of the internal nodes, $\sum_{j=1}^{i} m_j = m$. Since m_j arcs span at least $\sqrt{m_j}$ nodes, the contribution of the first n coordinates of the corresponding vectors $\dot{\boldsymbol{z}}$ to the cost $c(T, \boldsymbol{Y})$ is at least $\sum_{j=1}^{i} m_j^{3/2}$. Thus, by Hölder's inequality,

$$c(T, \boldsymbol{Y}) \geq n + (m+i-1)\ell + \sum_{j=1}^{i} m_j^{3/2} \geq n + (m+i-1)\ell + \frac{m^{3/2}}{\sqrt{i}} \, .$$

In order to show that $c(T, \boldsymbol{Y}) - n > w$ for all $i \geq 3$ and $1 \leq m \leq n^2$, we show that the following function is always positive:

$$f(m, i) = (m + i - 1)\ell + \frac{m^{3/2}}{\sqrt{i}} - w = (i - 2) \cdot 0.145n^3 + \frac{m^{3/2}}{\sqrt{i}} + \frac{4n^3}{27} - nm - n \; .$$

By taking the derivative w.r.t. m, we see that for every value of $i \geq 3, f(m, i)$ takes its minimum at $m = n^2$. By plugging in $m = n^2$, we get

$$g(i) = n^3 \left(0.145(i - 2) + \frac{1}{\sqrt{i}} - \frac{23}{27} \right) - n \; .$$

By taking the derivative of $g(i)$, we get that $g(i)$ is minimized when $i = 3$, in which case we have

$$c(T, \boldsymbol{Y}) - w - n \geq n^3 \left(0.145 + \frac{1}{\sqrt{3}} - \frac{23}{27} \right) - n > 0.1n^3 - n \geq 0$$

for $n \geq 4$.

□