A no-regret generalization of hierarchical softmax to extreme multi-label classification

Marek Wydmuch
Institute of Computing Science
Poznan University of Technology, Poland
mwydmuch@cs.put.poznan.pl

Kalina Jasinska
Institute of Computing Science
Poznan University of Technology, Poland
kjasinska@cs.put.poznan.pl

Mikhail Kuznetsov
Yahoo! Research
New York, USA
kuznetsov@oath.com

Róbert Busa-Fekete
Yahoo! Research
New York, USA
busafekete@oath.com

Krzysztof Dembczyński
Institute of Computing Science
Poznan University of Technology, Poland
kdembczynski@cs.put.poznan.pl

Abstract

Extreme multi-label classification (XMLC) is a problem of tagging an instance with a small subset of relevant labels chosen from an extremely large pool of possible labels. Large label spaces can be efficiently handled by organizing labels as a tree, like in the hierarchical softmax (HSM) approach commonly used for multi-class problems. In this paper, we investigate probabilistic label trees (PLTs) that have been recently devised for tackling XMLC problems. We show that PLTs are a *no-regret* multi-label generalization of HSM when precision@k is used as a model evaluation metric. Critically, we prove that *pick-one-label* heuristic—a reduction technique from multi-label to multi-class that is routinely used along with HSM—is not consistent in general. We also show that our implementation of PLTs, referred to as EXTREMETEXT (XT), obtains significantly better results than HSM with the pick-one-label heuristic and XML-CNN, a deep network specifically designed for XMLC problems. Moreover, XT is competitive to many state-of-the-art approaches in terms of statistical performance, model size and prediction time which makes it amenable to deploy in an online system.

1 Introduction

In several machine learning applications, the label space can be enormous, containing even millions of different classes. Learning problems of this scale are often referred to as *extreme classification*. To name a few examples of such problems, consider image and video annotation for multimedia search (Deng et al., 2011), tagging of text documents for categorization of Wikipedia articles (Dekel & Shamir, 2010), recommendation of bid words for online ads (Prabhu & Varma, 2014), or prediction of the next word in a sentence (Mikolov et al., 2013).

To tackle extreme classification problems in an efficient way, one can organize the labels into a tree. A prominent example of such label tree model is *hierarchical softmax* (HSM) (Morin & Bengio, 2005), often used with neural networks to speed up computations in multi-class classification with large output spaces. For example, it is commonly applied in natural language processing problems such as language modeling (Mikolov et al., 2013). To adapt HSM to *extreme multi-label classification* (XMLC), several very popular tools, such as FASTTEXT (Joulin et al., 2016) and LEARNED TREE (Jernite et al., 2017), apply the *pick-one-label* heuristic. As the name suggests, this heuristic randomly picks one of the labels from a multi-label training example and treats the example as a multi-class one.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

In this work, we exhaustively investigate the multi-label extensions of HSM. First, we show that the pick-one-label strategy does not lead to a proper generalization of HSM for multi-label setting. More precisely, we prove that using the pick-one-label reduction one cannot expect any multi-class learner to achieve zero regret in terms of marginal probability estimation and maximization of precision@k. As a remedy to this issue, we are going to revisit *probabilistic label trees* (PLTs) (Jasinska et al., 2016) that have been recently introduced for solving XMLC problems. We show that PLTs are a theoretically motivated generalization of HSM to multi-label classification, that is, 1) PLTs and HSM are identical in multi-class case, and 2) a PLT model can get *zero regret* (i.e., it is *consistent*) in terms of marginal probability estimation and precision@k in the multi-label setting.

Beside our theoretical findings, we provide an efficient implementation of PLTs, referred to as XT, that we build upon FASTTEXT. The comprehensive empirical evaluation shows that it gets significantly better results than the original FASTTEXT, LEARNED TREE, and XML-CNN, a specifically designed deep network for XMLC problems. XT also achieves competitive results to other state-of-the-art approaches, being very efficient in model size and prediction time, particularly in the online setting.

This paper is organized as follows. First we discuss the related work and situate our approach in the context. In Section 3 we formally state the XMLC problem and present some useful theoretical insights. Next, we briefly introduce the HSM approach, and in Section 5 we show theoretical results concerning the pick-one-label heuristic. Section 6 formally introduces PLTs and presents the main theoretical results concerning them and their relation to HSM. Section 7 provides implementation details of PLTs. The experimental results are presented in Section 8. Finally we make concluding remarks.

2 Related work

Historically, problems with a large number of labels were usually solved by nearest neighbor or decision tree methods. Some of today's algorithms are still based on these classical approaches, significantly extending them by a number of new tricks. If the label space is of moderate size (like a few thousands of labels) then an independent model can be trained for each class. This is the so-called 1-VS-ALL approach. Unfortunately, it scales linearly with the number of labels, which is too costly for many applications. The extreme classification algorithms try to improve over this approach by following different paradigms such as sparsity of labels (Yen et al., 2017; Babbar & Schölkopf, 2017), low-rank approximation (Mineiro & Karampatziakis, 2015; Yu et al., 2014; Bhatia et al., 2015), tree-based search (Prabhu & Varma, 2014; Choromanska & Langford, 2015), or label filtering (Vijayanarasimhan et al., 2014; Shrivastava & Li, 2015; Niculescu-Mizil & Abbasnejad, 2017).

In this paper we focus on tree-based algorithms, therefore we discuss them here in more detail. There are two distinct types of these algorithms: decision trees and label trees. The former type follows the idea of classical decision trees. However, the direct use of the classic algorithms can be very costly (Agrawal et al., 2013). Therefore, the FASTXML algorithm (Prabhu & Varma, 2014) tackles the problem in a slightly different way. It uses sparse linear classifiers in internal tree nodes to split the feature space. Each linear classifier is trained on two classes that are formed in a random way first and then reshaped by optimizing the normalized discounted cumulative gain. To improve the overall accuracy FASTXML uses an ensemble of trees. This algorithm, like many other decision tree methods, works in a batch mode. Choromanska & Langford (2015) have succeeded to introduce a fully online decision tree algorithm that also uses linear classifiers in internal nodes of the tree.

In label trees each label corresponds to one and only one path from the root to a leaf. Besides PLTs and HSM, there exist several other instances of this approach, for example, filter trees (Beygelzimer et al., 2009b; Li & Lin, 2014) or label embedding trees (Bengio et al., 2010). It is also worth to underline that algorithms similar to HSM have been introduced independently in many different research fields, such as nested dichotomies (Fox, 1997) in statistics, conditional probability estimation trees (Beygelzimer et al., 2009a) in multi-class classification, multi-stage classifiers (Kurzynski, 1988) in pattern recognition, and probabilistic classifier chains (Dembczyński et al., 2010) in multi-label classification under the subset 0/1 loss. All these methods have been jointly analyzed in (Dembczyński et al., 2016).

A still open problem in label tree approaches is the tree structure learning. FASTTEXT (Joulin et al., 2016) uses HSM with a Huffman tree built on the label frequencies. Jernite et al. (2017)

have introduced a new algorithm, called LEARNED TREE, which combines HSM with a specific hierarchical clustering that reassigns labels to paths in the tree in a semi-online manner. Prabhu et al. (2018) follows another approach in which a model similar to PLTs is trained in a batch mode and a tree is built by using recursively balanced k-means over the label profiles. In Section 7 we discuss this approach in more detail.

The HSM model is often used as an output layer in neural networks. The FASTTEXT implementation can also be viewed as a shallow architecture with one hidden layer that represents instances as averaged feature (i.e., word) vectors. Another neural network-based model designed for XMLC has been introduced in (Liu et al., 2017). This model, referred to as XML-CNN, uses a complex convolutional deep network with a narrow last layer to make it work with large output spaces. As we show in the experimental part, this quite expensive architecture gets inferior results in comparison to our PLTs built upon FASTTEXT.

3 Problem statement

Let $\mathcal{X}$ denote an instance space, and let $\mathcal{L} = \{1, \ldots, m\}$ 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 $\mathcal{L}_{\boldsymbol{x}} \in 2^{\mathcal{L}}$ (the subset can be empty); this subset is often called a 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 much smaller than m, i.e., $|\mathcal{L}_{\boldsymbol{x}}| \ll m$. We identify a set $\mathcal{L}_{\boldsymbol{x}}$ of relevant labels with a binary (sparse) 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 a set of all possible label vectors. We assume that observations $(\boldsymbol{x}, \boldsymbol{y})$ are generated independently and identically according to the probability distribution $\mathbf{P}(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y} = \boldsymbol{y})$ (denoted later by $\mathbf{P}(\boldsymbol{x}, \boldsymbol{y})$) defined on $\mathcal{X} \times \mathcal{Y}$.

The problem of XMLC can be defined as finding a *classifier* $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x}))$, which in general can be defined as a mapping $\mathcal{X} \to \mathcal{R}^m$, that minimizes the *expected loss* (or *risk*):

$$L_\ell(\boldsymbol{h}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbf{P}(\boldsymbol{x},\boldsymbol{y})}(\ell(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}))\,,$$

where $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$ is the (*task*) *loss*. The optimal classifier, the so-called *Bayes classifier*, for a given loss function ℓ is:

$$\boldsymbol{h}^*_\ell = \arg\min_{\boldsymbol{h}} L_\ell(\boldsymbol{h})\,.$$

The *regret* of a classifier $\boldsymbol{h}$ with respect to ℓ is defined as:

$$\mathrm{reg}_\ell(\boldsymbol{h}) = L_\ell(\boldsymbol{h}) - L_\ell(\boldsymbol{h}^*_\ell) = L_\ell(\boldsymbol{h}) - L^*_\ell\,.$$

The regret quantifies the suboptimality of $\boldsymbol{h}$ compared to the optimal classifier $\boldsymbol{h}^*$. The goal could be then defined as finding $\boldsymbol{h}$ with a small regret, ideally equal to zero.

In the following, we aim at estimating the marginal probabilities $\eta_j(\boldsymbol{x}) = \mathbf{P}(y_j = 1 \,|\, \boldsymbol{x})$. As we will show below, marginal probabilities are a key element to optimally solve extreme classification for many performance measures, like Hamming loss, macro-F measure, and precision@k. To obtain the marginal probability estimates one can use the label-wise log loss as a surrogate:

$$\ell_{\log}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \sum_{j=1}^{m} \ell_{\log}(y_j, h_j(\boldsymbol{x})) = \sum_{j=1}^{m} \left(y_j \log(h_j(\boldsymbol{x})) + (1 - y_j) \log(1 - h_j(\boldsymbol{x}))\right)\,.$$

Then the expected label-wise log loss for a single $\boldsymbol{x}$ (i.e., the so-called *conditional risk*) is:

$$\mathbb{E}_{\boldsymbol{y}} \ell_{\log}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \sum_{j=1}^{m} \mathbb{E}_{\boldsymbol{y}} \ell_{\log}(y_j, h_j(\boldsymbol{x})) = \sum_{j=1}^{m} L_{\log}(h_j(\boldsymbol{x}) \,|\, \boldsymbol{x})\,.$$

Therefore, it is easy to see that the pointwise optimal prediction for the j-th label is given by:

$$h^*_j(\boldsymbol{x}) = \arg\min_{h} L_{\log}(h_j(\boldsymbol{x}) \,|\, \boldsymbol{x}) = \eta_j(\boldsymbol{x})\,.$$

As shown in (Dembczyński et al., 2010), the Hamming loss is minimized by $h^*_j(\boldsymbol{x}) = [\![\eta_j(\boldsymbol{x}) > 0.5]\!]$. For the macro F-measure it suffices in turn to find an optimal threshold on marginal probabilities for

each label separately as proven in (Ye et al., 2012; Narasimhan et al., 2014; Jasinska et al., 2016; Dembczyński et al., 2017). In the following, we will show a similar result for precision@k which has become a standard measure in extreme classification (although it is also often criticized, as it favors the most frequent labels).

Precision@k can be formally defined as:

$$\text{precision@}k(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{h}) = \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} [\![y_j = 1]\!], \tag{1}$$

where $\hat{\mathcal{Y}}_k$ is a set of k labels predicted by $\boldsymbol{h}$ for $\boldsymbol{x}$. To be consistent with the former discussion, let us define a loss function for precision@k as $\ell_{p@k} = 1 - \text{precision@}k$. The conditional risk is then:[1]

$$L_{p@k}(\boldsymbol{h} \,|\, \boldsymbol{x}) = \mathbb{E}_{\boldsymbol{y}} \ell_{p@k}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{h}) = 1 - \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \eta_j(\boldsymbol{x}) \,.$$

The above result shows that the optimal strategy for precision@k is to predict k labels with the highest marginal probabilities $\eta_j(\boldsymbol{x})$. As the main theoretical result given in this paper is a regret bound for precision@k, let us define here the conditional regret for this metric:

$$\text{reg}_{p@k}(\boldsymbol{h} \,|\, \boldsymbol{x}) = \frac{1}{k} \sum_{i \in \mathcal{Y}_k} \eta_i(\boldsymbol{x}) - \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \eta_j(\boldsymbol{x}) \,,$$

where $\mathcal{Y}_k$ is a set containing the top k labels with respect to the true marginal probabilities.

From the above results, we see that estimation of marginal probabilities is crucial for XMLC problems. To obtain these probabilities we can use the vanilla 1-VS-ALL approach trained with the label-wise log loss. Unfortunately, 1-VS-ALL is too costly in the extreme setting. In the following sections, we discuss an alternative approach based on the label trees that estimates the marginal probabilities with the competitive accuracy, but in a much more efficient way.

4 Hierarchical softmax approaches

Hierarchical softmax (HSM) is designed for multi-class classification. Using our notation, for multi-class problems we have $\sum_{i=1}^{m} y_i = 1$, i.e., there is one and only one label assigned to an instance $(\boldsymbol{x}, \boldsymbol{y})$. The marginal probabilities $\eta_j(\boldsymbol{x})$ in this case sum up to 1.

The HSM classifier $\boldsymbol{h}(\boldsymbol{x})$ takes a form of a label tree. We encode all labels from $\mathcal{L}$ using a prefix code. Any such code can be given in a form of a tree in which a path from the root to a leaf node corresponds to a code word. Under the coding, each label $y_j = 1$ is uniquely represented by a code word $\boldsymbol{z} = (z_1, \ldots, z_l) \in \mathcal{C}$, where l is the length of the code word and $\mathcal{C}$ is a set of all code words. For $z_i \in \{0, 1\}$, the code and the label tree are binary. In general, the code alphabet can contain more than two symbols. Furthermore, z_is can take values from different sets of symbols depending on the previous values in the code word. In other words, the code can result with nodes of a different arity even in the same tree, like in (Grave et al., 2017) and (Prabhu et al., 2018). We will briefly discuss different tree structures in Section 7.

A tree node can be uniquely identified by the partial code word $\boldsymbol{z}^i = (z_1, \ldots, z_i)$. We denote the root node by $\boldsymbol{z}^0$, which is an empty vector (without any elements). The probability of a given label is determined by a sequence of decisions made by node classifiers that predict subsequent values of the code word. By using the chain rule of probability, we obtain:

$$\eta_j(\boldsymbol{x}) = \mathbf{P}(y_j = 1 \,|\, \boldsymbol{x}) = \mathbf{P}(\boldsymbol{z} \,|\, \boldsymbol{x}) = \prod_{i=1}^{l} \mathbf{P}(z_i \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x}) \,.$$

By using logistic loss and a linear model $f_{\boldsymbol{z}^i}(\boldsymbol{x})$ in each node $\boldsymbol{z}^i$ for estimating $\mathbf{P}(z_i \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})$, we obtain the popular formulation of HSM. Let us notice that since we deal here with a multi-class distribution, we have that:

$$\sum_{c} \mathbf{P}(z_i = c \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x}) = 1 \,. \tag{2}$$

[1] The derivation is given in Appendix A.

Because of this normalization, we can assume that a multi-class (or binary in the case of binary trees) classifier is situated in all internal nodes and there are no classifiers in the leaves of the tree. Alternatively, we can assume that each node, except the root, is associated with a binary classifier that estimates $\mathbf{P}(z_i = c \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})$, but then the additional normalization (2) has to be performed. This alternative formulation is important for the multi-label extension of HSM discussed in Section 6. In either way, learning of the node classifiers can be performed simultaneously as independent tasks.

Note that estimate $\hat{\eta}_j(\boldsymbol{x})$ of the probability of label j can be easily obtained by traversing the tree along the path indicated by the code of the label. Unfortunately, the task of predicting top k labels is more involved as it requires searching over the tree. Popular solutions are beam search (Kumar et al., 2013; Prabhu et al., 2018), uniform-cost search (Joulin et al., 2016), and its approximate variant (Dembczyński et al., 2012; Dembczyński et al., 2016).

5 Suboptimality of HSM for multi-label classification

To deal with multi-label problems, some popular tools, such as FASTTEXT (Joulin et al., 2016) and its extension LEARNED TREE (Jernite et al., 2017), apply HSM with the pick-one-label heuristic which randomly picks one of the positive labels from a given training instance. The resulting instance is then treated as a multi-class instance. During prediction, the heuristic returns a multi-class distribution and the k most probable labels. We show below that this specific reduction of the multi-label problem to multi-class classification is not consistent in general.

Since the probability of picking a label j from $\boldsymbol{y}$ is equal to $y_j / \sum_{j'=1}^{m} y_{j'}$, the pick-one-label heuristic maps the multi-label distribution to a multi-class distribution in the following way:

$$\eta'_j(\boldsymbol{x}) = \mathbf{P}'(y_j = 1 \,|\, \boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \frac{y_j}{\sum_{j'=1}^{m} y_{j'}} \mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}) \tag{3}$$

It can be easily checked that the resulting $\eta'_j(\boldsymbol{x})$ form a multi-class distribution as the probabilities sum up to 1. It is obvious that that the heuristic changes the marginal probabilities of labels, unless the initial distribution is multi-class. Therefore this method cannot lead to consistent classifiers in terms of estimating $\eta_j(\boldsymbol{x})$. As we show below, it is also not consistent for precision@k in general.

Proposition 1. *A classifier $\boldsymbol{h}$ such that $h_j(\boldsymbol{x}) = \eta'_j(\boldsymbol{x})$ for all $j \in \{1, \ldots, m\}$ has in general a non-zero regret in terms of precision@k.*

Proof. We prove the proposition by giving a simple counterexample. Consider the following conditional distribution for some $\boldsymbol{x}$:

$$\mathbf{P}(\boldsymbol{y} = (1,0,0) \,|\, \boldsymbol{x}) = 0.1\,, \quad \mathbf{P}(\boldsymbol{y} = (1,1,0) \,|\, \boldsymbol{x}) = 0.5\,, \quad \mathbf{P}(\boldsymbol{y} = (0,0,1) \,|\, \boldsymbol{x}) = 0.4\,.$$

The optimal top 1 prediction for this example is obviously label 1, since the marginal probabilities are $\eta_1(\boldsymbol{x}) = 0.6, \eta_2(\boldsymbol{x}) = 0.5, \eta_3(\boldsymbol{x}) = 0.4$. However, the pick-one-label heuristic will transform the original distribution to the following one: $\eta'_1(\boldsymbol{x}) = 0.35, \eta'_2(\boldsymbol{x}) = 0.25, \eta'_3(\boldsymbol{x}) = 0.4$. The predicted top label will be then label 3, giving the regret of 0.2 for precision@1. □

The proposition shows that the heuristic is in general inconsistent for precision@k. Interestingly, the situation changes when the labels are conditionally independent, i.e., $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}) = \prod_{j=1}^{m} \mathbf{P}(y_i \,|\, \boldsymbol{x})$.

Proposition 2. *Given conditionally independent labels, a classifier $\boldsymbol{h}$ such that $h_j(\boldsymbol{x}) = \eta'_j(\boldsymbol{x})$ for all $j \in \{1, \ldots, m\}$ has zero regret in terms of the precision@k loss.*

Proof. We show here only a sketch of the proof. The full proof is given in Appendix B. To prove the theorem, it is enough to show that in the case of conditionally independent labels the pick-one-label heuristic does not change the order of marginal probabilities. Let y_i and y_j be so that $\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x}) \geq \mathbf{P}(y_j = 1 \,|\, \boldsymbol{x})$. Then in the summation over all $\boldsymbol{y}$s in (3), we are interested in four different subsets of $\mathcal{Y}$, $S_{i,j}^{u,w} = \{\boldsymbol{y} \in \mathcal{Y} : y_i = u \wedge y_j = w\}$, where $u, w \in \{0,1\}$. Remark that during mapping none of $\boldsymbol{y} \in S_{i,j}^{0,0}$ plays any role, and for each $\boldsymbol{y} \in S_{i,j}^{1,1}$, the value of $y_t / (\sum_{t'=1}^{m} y_{t'}) \times \mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$, for $t \in \{i, j\}$, is the same for both y_i and y_j. Now, let $\boldsymbol{y}' \in S_{i,j}^{1,0}$ and $\boldsymbol{y}'' \in S_{i,j}^{0,1}$ be the same on all elements except the i-th and the j-th one. Then, because

of the label independence and the assumption that $\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x}) \geq \mathbf{P}(y_j = 1 \,|\, \boldsymbol{x})$, we have $\mathbf{P}(\boldsymbol{y}' \,|\, \boldsymbol{x}) \geq \mathbf{P}(\boldsymbol{y}'' \,|\, \boldsymbol{x})$. Therefore, after mapping we obtain $\eta'_i(\boldsymbol{x}) \geq \eta'_j(\boldsymbol{x})$. Thus, for independent labels, the pick-one-label heuristic is consistent for precision@k. □

6 Probabilistic label trees

The section above has revealed that HSM cannot be properly adapted to multi-label problems by the pick-one-label heuristic. There is, however, a different way to generalize HSM to obtain no-regret estimates of marginal probabilities $\eta_j(\boldsymbol{x})$. The probabilistic label trees (PLTs) (Jasinska et al., 2016) can be derived in the following way. Let us encode $y_j = 1$ by a slightly extended code $\boldsymbol{z} = (1, z_1, \ldots, z_l)$ in comparison to HSM. The new code gets 1 at the zero position what corresponds to a question whether there exists at least one label assigned to the example. As before, each node is uniquely identified by a partial code $\boldsymbol{z}^i$ which says that there is at least one positive label in a subtree rooted in that node. It can be easily shown by the chain rule of probability that the marginal probabilities can be expressed in the following way:

$$\eta_j(\boldsymbol{x}) = \mathbf{P}(\boldsymbol{z} \,|\, \boldsymbol{x}) = \prod_{i=0}^{l} \mathbf{P}(z_i \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x}) \,. \tag{4}$$

The difference to HSM is the probability $\mathbf{P}(z_0 = 1 \,|\, \boldsymbol{x})$ in the chain and a different normalization, i.e.:

$$\sum_{c} \mathbf{P}(z_i = c \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x}) \geq 1 \,. \tag{5}$$

Only for z_0 we have $\mathbf{P}(z_0 = 1 \,|\, \boldsymbol{x}) + \mathbf{P}(z_0 = 0 \,|\, \boldsymbol{x}) = 1$. Because of (5), the binary models that estimate $\mathbf{P}(z_i = c \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})$ (against $\mathbf{P}(z_i \neq c \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})$) are situated in all nodes of the tree (i.e., also in the leaves). The models can be trained independently as before for HSM. Only during prediction, one can re-calibrate the estimates when (5) is not satisfied, for example, by normalizing them to sum up to 1. It can be easily noticed that for a multi-class distribution, the resulting model of PLTs boils down to HSM, since $\mathbf{P}(z_0 = 1 \,|\, \boldsymbol{x})$ is always equal 1, and in addition, normalization (5) will take the form of (2). In Appendix D we additionally present the pseudocode of training and predicting with PLTs.

Next, we show that the PLT model obeys strong theoretical guarantees. Let us first revise the result from (Jasinska et al., 2016) that relates the absolute difference between the true and the estimated marginal probability of label j, $|\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})|$, to the surrogate loss ℓ used to train node classifiers $f_{\boldsymbol{z}^i}$. It is assumed here that ℓ is a strongly proper composite loss (e.g, logistic, exponential, or squared loss) characterized by a constant λ (e.g. $\lambda = 4$ for logistic loss).[2]

Theorem 1. *For any distribution* $\mathbf{P}$ *and internal node classifiers* $f_{\boldsymbol{z}^i}$*, the following holds:*

$$|\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})| \leq \sum_{i=0}^{l} \mathbf{P}(\boldsymbol{z}^{i-1} \,|\, \boldsymbol{x}) \sqrt{\frac{2}{\lambda}} \sqrt{\mathrm{reg}_\ell(f_{\boldsymbol{z}^i} \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})} \,,$$

where $\mathrm{reg}_\ell(f_{\boldsymbol{z}^i} \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})$ *is a binary classification regret for a strongly proper composite loss* ℓ *and* λ *is a constant specific for loss* ℓ*.*

Due to filtering of the distribution imposed by the PLT, the regret $\mathrm{reg}_\ell(f_{\boldsymbol{z}^i} \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})$ of a classifier $f_{\boldsymbol{z}^i}$ exists only for $\boldsymbol{x}$ such that $\mathbf{P}(\boldsymbol{z}^{i-1} \,|\, \boldsymbol{x}) > 0$, therefore we condition the regret not only on $\boldsymbol{x}$, but also on $\boldsymbol{z}^{i-1}$. The above result shows that the absolute error of estimating the marginal probability of label j can be upper bounded by the regret of the node classifiers on the corresponding path from the root to a leaf. The proof of Theorem 1 is given in Appendix A. Moreover, for zero-regret (i.e., optimal) node classifiers we obtain an optimal multi-label classifier in terms of estimation of marginal probabilities $\eta_j(\boldsymbol{x})$. This result can be further extended for precision@k.

Theorem 2. *For any distribution* $\mathbf{P}$ *and classifier* $\boldsymbol{h}$ *delivering estimates* $\hat{\eta}_j(\boldsymbol{x})$ *of the marginal probabilities of labels, the following holds:*

$$\mathrm{reg}_{p@k}(\boldsymbol{h} \,|\, \boldsymbol{x}) = \frac{1}{k} \sum_{i \in \mathcal{Y}_k} \eta_i(\boldsymbol{x}) - \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \eta_j(\boldsymbol{x}) \leq 2 \max_l |\eta_l(\boldsymbol{x}) - \hat{\eta}_l(\boldsymbol{x})|$$

[2]For more detailed introduction to strongly proper composite losses, we refer the reader to (Agarwal, 2014).

The proof is based on adding and subtracting the following terms $\frac{1}{k}\sum_{i\in\mathcal{Y}_k}\hat{\eta}_i(\boldsymbol{x})$ and $\frac{1}{k}\sum_{j\in\hat{\mathcal{Y}}_k}\hat{\eta}_j(\boldsymbol{x})$ to the regret (a detailed proof is given in Appendix A). By getting together both theorems we get an upper bound of the precision@k regret expressed in terms of the regret of the node classifiers. Again, for the zero-regret node classifiers, we get optimal solution in terms of precision@k.

7 Implementation details of PLTs

Given the tree structure, the node classifiers of PLTs can be trained as logistic regression either in online (Jasinska et al., 2016) or batch mode (Prabhu et al., 2018). Both training modes have their pros and cons, but the online implementation gives a possibility of learning more complex representation of input instances. The above cited implementations are both based on sparse representation, given either in a form of a bag-of-words or its TF-IDF variant. We opt here for training a PLT in the online mode along with the dense representation. We build our implementation upon FASTTEXT and refer to it as XT which stands for EXTREMETEXT.[3] In this way, we succeeded to obtain a very powerful and compressed model. The small dense models are important for fast online prediction as they do not need too much resources. The sparse models, in turn, can be slow and expensive in terms of memory usage as they need to decompress the node models to work fast. Remark also that, in general, PLTs can be used as an output layer of any neural network architecture (also that one used in XML-CNN (Yen et al., 2017)) to speed up training and prediction time.

In contrast to the original implementation of FASTTEXT, we use L2 regularization for all parameters of the model. To obtain representation of input instances we do not compute simple averages of the feature vectors, but use weights proportional to the TF-IDF scores of features. The competitive results can be obtained with feature and instance vectors of size 500. If a node classification task contains only positive instances, we use a constant classifier predicting 1 without any training. The training of PLT in either mode, online or batch, can be easily parallelized as each node classifier can be trained in isolation from the other classifiers. In our current implementation, however, we follow the parallelization on the level of training and test instances as in original FASTTEXT.

Our implementation, because of the additional use of the L2 regularization, has more parameters than original FASTTEXT. We have found, however, that our model is remarkably robust for the hyperparameter selection, since it achieves close to optimal performance for a large set of hyperparameters that is in the vicinity of the optimal one. Moreover, the optimal hyperparameters are close to each other across all datasets. We report more information about the hyperparameter selection in Appendix E.4.

The tree structure of a PLT is a crucial modeling decision. The vanishing regret for probability estimates and precision@k holds regardless of the tree structure (see Theorem 1 and 2), however, this theory requires the regret of the node classifiers also to vanish. In practice, we can only estimate the conditional probabilities in the nodes, therefore the tree structure does indeed matter as it affects the difficulty of the node learning problems. The original PLT paper (Jasinska et al., 2016) uses simple complete trees with labels assigned to leaves according to their frequencies. Another option, routinely used in HSM (Joulin et al., 2016), is the Huffman tree built over the label frequencies. Such tree takes into account the computational complexity by putting the most frequent labels close to the root. This approach has been further extended to optimize GPU operations in (Grave et al., 2017). Unfortunately, it ignores the statistical properties of the tree structure. Furthermore, for multi-label case the Huffman tree is no longer optimal even in terms of computational cost as we show it in Appendix C. There exist, however, other methods that focus on building a tree with high overall accuracy (Tagami, 2017; Prabhu et al., 2018). In our work, we follow the later approach, which performs a simple top-down hierarchical clustering. Each label in this approach is represented by a profile vector being an average of the training vectors tagged by this label. Then the profile vectors are clustered using balanced k-means which divides the labels into two or more clusters with approximately the same size. This procedure is then repeated recursively until the clusters are smaller than a given value (for example, 100). The nodes of the resulting tree are then of different arities. The internal nodes up to the leaves' parent nodes have k children, but the leaves' parent nodes are usually of higher arity. Thanks to this clustering, similar labels are close to each other in the tree. Moreover, the tree is balanced, so its depth is logarithmic in terms of the number of labels.

[3] Implementation of XT is available at `https://github.com/mwydmuch/extremeText`.

8 Empirical results

We carried out three sets of experiments. In the first, we compare exhaustively the performance of PLTs and HSM on synthetic and benchmark data. Due to lack of space, the results are deferred to Appendix E.1 and E.2. The results on synthetic data confirm our theoretical findings: the models are the same in the case of multi-class data, the performance of HSM and PLTs is on par using multi-label data with independent labels, and PLTs significantly outperform HSM on multi-label data with conditionally dependent labels. The results on the benchmark data clearly indicate the better performance of PLTs over HSM.

In the second experiment, we compare XT, the variant of PLTs discussed in the previous section, to the state-of-the-art algorithms on five benchmark datasets taken from XMLC repository,[4] and their text equivalents, by courtesy of Liu et al. (2017). We compare the models in terms of precision@$\{1, 3, 5\}$, model size, training and test time. The competitors for our XT are original FASTTEXT, its variant LEARNED TREE, a PLT-like batch learning algorithm PARABEL (we use the variant that uses a single tree instead of an ensemble), a XMLC-designed convolutional deep network XML-CNN, a decision tree ensemble FASTXML, and two 1-vs-All approaches tailored to XMLC problems, PPD-SPARSE and DISMEC. The hyperparameters of the models have been tuned using grid search. The range of the hyperparameters is reported in E.4.

The results presented in Table 1 demonstrate that XT outperforms the HSM approaches with the pick-one-label heuristic, namely FASTTEXT and LEARNED TREE, with a large margin. This proves the superiority of PLTs as the proper generalization of HSM to multi-label setting. In all the above methods we use vectors of length 500 and we tune the other hyperparameters appropriately for a fair comparison.

Moreover, XT scales well to extreme datasets achieving performance close to the state-of-the-art, being at the same time 10000x and 100x faster compared to DISMEC and PPDSPARSE during prediction. XT always responds below 2ms, what makes it a competitive alternative for an online setting. XT is also close to PARABEL in terms of performance. However, the reported times and model sizes of PARABEL are given for the batch prediction. The prediction times seem to be faster, but PARABEL needs to decompress the model during prediction, what makes it less suitable for online prediction. It is only efficient when the batches are sufficiently large. Finally, we would like to underline that XT outperforms XML-CNN, the more complex neural network, in terms of predictive performance with computational costs that are an order of magnitude smaller. Moreover, XML-CNN requires pretrained embedding vectors, whereas XT can be used with random initialization.

In the third experiment we perform an ablation analysis in which we compare different components of the XT algorithm. We analyze the influence of the Huffman tree vs. top-down clustering, the simple averaging of features vectors vs. the TF-IDF-based weighting, and no regularization vs. L2 regularization. Figure 1 clearly shows that the components need to be combined together to obtain the best results. The best combination uses top-down clustering, TF-IDF-based weighting, and L2 regularization, while top-down clustering alone gets worse results than Huffman trees with TF-IDF-based weighting and L2 regularization. In Appendix E.3 we give more detailed results of the ablation analysis performed on a larger spectrum of benchmark datasets.

9 Conclusions

In this paper we have proven that probabilistic label trees (PLTs) are no-regret generalization of HSM to the multi-label setting. Our main theoretical contribution is the precision@k regret bound for PLTs. Moreover, we have shown that the pick-one-label heuristic commonly-used with HSM in multi-label problems leads to inconsistent results in terms of marginal probability estimation and precision@k. Our implementation of PLTs referred to as XT, built upon FASTTEXT, gets state-of-the-art results, being significantly better than the original FASTTEXT, LEARNED TREE, and XML-CNN. The XT results are also close to the best known ones that are obtained by expensive 1-vs-All approaches, such as PPDSPARSE and DISMEC, and outperforms the other tree-based methods on many benchmarks. Our online variant has the advantage of producing very often much smaller models that can be efficiently used in fast online prediction.

[4] Additional statistics of these datasets are also included in Appendix F. Address of the XMLC repository: `http://manikvarma.org/downloads/XC/XMLRepository.html`

Table 1: Precision@k scores with $k = \{1, 3, 5\}$ and statistics of FASTXML, PPDSPARSE, DISMEC, PARABEL (with 1 tree), FASTTEXT (FT), LEARNED TREE (LT), EXTREMETEXT (XT) and XML-CNN methods. Notation: N – number of samples, T – CPU time, m – number of labels, d – number of features, $*$ – result of offline prediction, $\star$ – calculated on GPU, † – not reported by authors, ‡ – cannot be calculated due to lack of a text version of a dataset.

Dataset	Metrics	FASTXML	PPDSPARSE	DISMEC	FT	LT	**XT**	PARABEL	XML-CNN
Wiki-30K	P@1	82.03	73.80	85.20	80.78	80.85	**85.23**	83.77	82.78
$N_{train} = 14146$	P@3	67.47	60.90	**74.60**	50.46	50.59	73.18	71.96	66.34
$N_{test} = 6616$	P@5	57.76	50.40	**65.90**	36.79	37.68	63.39	62.44	56.23
$d = 101938$	T_{train}	16m	†	†	10m	12m	18m	**5m**	88m$\star$
$m = 30938$	T_{test}/N_{test}	3.00ms	†	†	1.88ms	10.09ms	**0.83ms**	1.63ms$*$	1.39ms$\star$
	model size	354M	†	†	513M	513M	**259M**	**109M**$*$	$\star$
Delicious-200K	P@1	42.81	45.05	44.71	42.22	42.71	**47.85**	43.32	‡
$N_{train} = 196606$	P@3	38.76	38.34	38.08	37.90	36.27	**42.08**	38.49	‡
$N_{test} = 100095$	P@5	36.34	34.90	34.7	35.05	33.43	**39.13**	35.83	‡
$d = 782585$	T_{train}	458m	4781m	1080h	271m	563m	502m	105m	‡
$m = 205443$	T_{test}/N_{test}	4.86ms	275ms	5m	1.97ms	1.98ms	**1.41ms**	**1.31ms**$*$	‡
	model size	15.4G	9.4G	18.0G	9.0G	9.0G	**1.9G**	**1.8G**$*$	‡
WikiLSHTC	P@1	49.35	64.08	**64.94**	41.13	50.15	58.73	61.53	‡
$N_{train} = 1778351$	P@3	32.69	41.26	**42.71**	24.09	31.95	39.24	40.07	‡
$N_{test} = 587084$	P@5	24.03	30.12	**31.5**	17.44	23.59	29.26	29.25	‡
$d = 617899$	T_{train}	724m	236m	750h	207	212m	550m	**34m**	‡
$m = 325056$	T_{test}/N_{test}	2.17ms	37.76ms	43m	1.25ms	4.76ms	**0.81ms**	0.92ms$*$	‡
	model size	9.3G	5.2G	3.8G	6.5G	6.5G	**3.3G**	**1.1G**$*$	‡
Wiki-500K	P@1	54.10	70.16	**70.20**	32.73	37.18	64.48	66.12	59.85
$N_{train} = 1813391$	P@3	29.45	50.57	**50.60**	19.02	21.62	45.84	47.02	39.28
$N_{test} = 783743$	P@5	21.21	39.66	**39.70**	14.46	16.01	35.46	36.45	29.81
$d = 2381304$	T_{train}	3214m	1771m	7495h	496m	531m	1253m	**168m**	7032m$\star$
$m = 501070$	T_{test}/N_{test}	8.03ms	113.70ms	155m	2.05ms	6.43ms	**1.07ms**	4.68ms$*$	21.06ms$\star$
	model size	63G	3.4G	14.7G	11G	11G	**5.5G**	**2.0G**$*$	**3.7G**$\star$
Amazon-670K	P@1	34.24	45.32	**45.37**	25.47	27.67	39.90	41.59	35.39
$N_{train} = 490449$	P@3	29.30	40.37	**40.40**	21.47	20.96	35.36	37.18	33.74
$N_{test} = 153025$	P@5	26.12	36.92	**36.96**	18.61	17.72	32.04	33.85	32.64
$d = 135909$	T_{train}	422m	102m	373h	162m	182m	241m	**8m**	3134m$\star$
$m = 670091$	T_{test}/N_{test}	3.39ms	66.09ms	23m	7.84ms	5.13ms	**1.72ms**	**0.68ms**$*$	16.18ms$\star$
	model size	10G	6.0G	3.8G	3.2G	3.2G	**1.5G**	**0.7G**$*$	**1.5G**$\star$

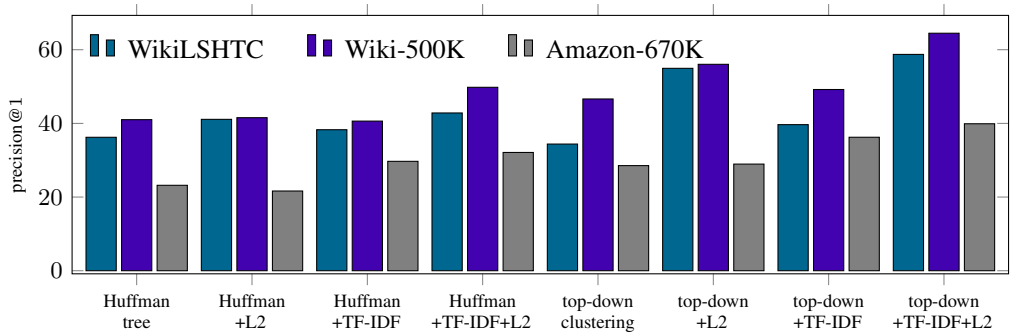

Figure 1: The ablation analysis of different variants of XT on WIKILSHTC, WIKI-500K, and AMAZON-670K.

Acknowledgements

The work of Kalina Jasinska was supported by the Polish National Science Center under grant no. 2017/25/N/ST6/00747. The work of Krzysztof Dembczyński was supported by the Polish Ministry of Science and Higher Education under grant no. 09/91/DSPB/0651. Computational experiments have been performed in Poznan Supercomputing and Networking Center.

References

Agarwal, S. Surrogate regret bounds for bipartite ranking via strongly proper losses. *JMLR*, 15: 1653–1674, 2014.

Agrawal, R., Gupta, A., Prabhu, Y., and Varma, M. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, pp. 13–24. ACM, 2013.

Babbar, Rohit and Schölkopf, Bernhard. Dismec: Distributed sparse machines for extreme multi-label classification. In *WSDM 2017*, pp. 721–729. ACM, 2017.

Bengio, S., Weston, J., and Grangier, D. Label embedding trees for large multi-class tasks. In *NIPS 24*, pp. 163–171, 2010.

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

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

Bhatia, K., Jain, H., Kar, P., Varma, M., and Jain, P. Sparse local embeddings for extreme multi-label classification. In *NIPS 29*, pp. 730–738, 2015.

Choromanska, A. and Langford, J. Logarithmic time online multiclass prediction. In *NIPS 29*, pp. 55–63, 2015.

Dekel, O. and Shamir, O. Multiclass-multilabel classification with more classes than examples. In *AISTATS*, pp. 137–144, 2010.

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

Dembczyński, K., Waegeman, W., Cheng, W., and Hüllermeier, E. An analysis of chaining in multi-label classification. In *ECAI*, 2012.

Dembczyński, Krzysztof, Kotłowski, Wojciech, Waegeman, Willem, Busa-Fekete, Róbert, and Hüllermeier, Eyke. Consistency of probabilistic classifier trees. In *ECMLPKDD*. Springer, 2016.

Dembczyński, Krzysztof, Kotłowski, Wojciech, Koyejo, Oluwasanmi, and Natarajan, Nagarajan. Consistency analysis for binary classification revisited. In Precup, Doina and Teh, Yee Whye (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 961–969. PMLR, 2017.

Deng, J., Satheesh, S., Berg, A. C., and Fei-Fei, L. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS 24*, pp. 567–575, 2011.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

Fox, J. *Applied regression analysis, linear models, and related methods*. Sage, 1997.

Grave, Edouard, Joulin, Armand, Cissé, Moustapha, Grangier, David, and Jégou, Hervé. Efficient softmax approximation for GPUs. In Precup, Doina and Teh, Yee Whye (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1302–1310, International Convention Centre, Sydney, Australia, 2017. PMLR.

Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., and Hüllermeier, E. Extreme F-measure maximization using sparse probability estimates. In *ICML*, 2016.

Jernite, Yacine, Choromanska, Anna, and Sontag, David. Simultaneous learning of trees and representations for extreme classification and density estimation. In Precup, Doina and Teh, Yee Whye (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1665–1674, International Convention Centre, Sydney, Australia, 2017. PMLR.

Joulin, Armand, Grave, Edouard, Bojanowski, Piotr, and Mikolov, Tomas. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016. URL `http://arxiv.org/abs/1607.01759`.

Kumar, A., Vembu, S., Menon, A.K., and Elkan, C. Beam search algorithms for multilabel learning. In *Machine Learning*, 2013.

Kurzynski, Marek. On the multistage bayes classifier. *Pattern Recognition*, 21(4):355–365, 1988.

Langford, J., Strehl, A., and Li, L. Vowpal wabbit, 2007. `http://mloss.org/software/view/53/`.

Li, Ch.-L. and Lin, H.-Ti. Condensed filter tree for cost-sensitive multi-label classification. In *ICML*, pp. 423–431, 2014.

Liu, Jingzhou, Chang, Wei-Cheng, Wu, Yuexin, and Yang, Yiming. Deep learning for extreme multi-label text classification. In *SIGIR 2017*, pp. 115–124. ACM, 2017.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *NIPS 26*, pp. 3111–3119. Curran Associates, Inc., 2013.

Mineiro, P. and Karampatziakis, N. Fast label embeddings via randomized linear algebra. In *ECML/PKDD 2015*, pp. 37–51, 2015.

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

Narasimhan, H., Vaish, R., and S., Agarwal. On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *NIPS 27*, pp. 1493–1501, 2014.

Niculescu-Mizil, Alexandru and Abbasnejad, Ehsan. Label Filters for Large Scale Multilabel Classification. In Singh, Aarti and Zhu, Jerry (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1448–1457. PMLR, 2017.

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

Prabhu, Yashoteja, Kag, Anil, Harsola, Shrutendra, Agrawal, Rahul, and Varma, Manik. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In Champin, Pierre-Antoine, Gandon, Fabien L., Lalmas, Mounia, and Ipeirotis, Panagiotis G. (eds.), *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018*, pp. 993–1002. ACM, 2018.

Shrivastava, A. and Li, P. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (mips). In *UAI*, 2015.

Tagami, Yukihiro. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 455–464, New York, NY, USA, 2017. ACM.

Vijayanarasimhan, Sudheendra, Shlens, Jonathon, Monga, Rajat, and Yagnik, Jay. Deep networks with large output spaces. *CoRR*, abs/1412.7479, 2014. URL `http://arxiv.org/abs/1412.7479`.

Ye, N., Chai, A., Lee, W., and Chieu, H. Optimizing F-measures: A tale of two approaches. In *ICML*, 2012.

Yen, Ian E.H., Huang, Xiangru, Dai, Wei, Ravikumar, Pradeep, Dhillon, Inderjit, and Xing, Eric. PPDSparse: A parallel primal-dual sparse method for extreme classification. In *KDD 2017*, pp. 545–553. ACM, 2017.

Yu, Hsiang-Fu, Jain, Prateek, Kar, Purushottam, and Dhillon, Inderjit. Large-scale multi-label learning with missing labels. In *ICML 2014*, volume 32, pp. 593–601. PMLR, 2014.

A Regret for precision@k

Precision@k is formally defined as:

$$\text{precision@}k(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{h}) = \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} [\![y_j = 1]\!],$$

where $\hat{\mathcal{Y}}_k$ is a set of k labels predicted by $\boldsymbol{h}$ for $\boldsymbol{x}$. The loss function for precision@k can be defined as $\ell_{p@k} = 1 - \text{precision@}k$. The conditional risk is then:

$$\begin{aligned} L_{p@k}(\boldsymbol{h} \,|\, \boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{y}} \ell_{p@k}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{h}) \\ &= 1 - \sum_{\boldsymbol{y} \in \mathcal{Y}} \mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}) \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} [\![y_j = 1]\!] \\ &= 1 - \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \sum_{\boldsymbol{y} \in \mathcal{Y}} \mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}) [\![y_j = 1]\!] \\ &= 1 - \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \eta_j(\boldsymbol{x}) \,. \end{aligned}$$

The above result shows that the optimal strategy for precision@k is to predict k labels with the highest marginal probabilities $\eta_j(\boldsymbol{x})$.

We show now that PLTs obey strong theoretical guarantees. We first recall the result from (Jasinska et al., 2016) that relates the absolute difference between the true and the estimated marginal probability of label j, $|\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})|$, to the surrogate loss ℓ used to train node classifiers $f_{\boldsymbol{z}^i}$. It is assumed here that ℓ is a strongly proper composite loss (e.g, logistic, exponential, or squared-error loss) characterized by a constant λ (e.g. $\lambda = 4$ for logistic loss).[5]

Theorem 1. *For any distribution* $\mathbf{P}$ *and internal node classifiers* $f_{\boldsymbol{z}^i}$*, the following holds:*

$$|\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})| \leq \sum_{i=0}^{l} \mathbf{P}(\boldsymbol{z}^{i-1} \,|\, \boldsymbol{x}) \sqrt{\frac{2}{\lambda}} \sqrt{\text{reg}_\ell(f_{\boldsymbol{z}^i} \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})} \,,$$

where $\text{reg}_\ell(f_{\boldsymbol{z}^i} \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})$ *is a binary classification regret for a strongly proper composite loss* ℓ *and* λ *is a constant specific for loss* ℓ*.*

Proof. From Equation (4) we have:

$$\eta_j(\boldsymbol{x}) = \mathbf{P}(\boldsymbol{z} \,|\, \boldsymbol{x}) = \prod_{i=0}^{l} \mathbf{P}(z_i \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x}) = \mathbf{P}(\boldsymbol{z}^{n-1} \,|\, \boldsymbol{x}) \prod_{i=n}^{l} \mathbf{P}(z_i \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x}) \,,$$

for any $1 \leq n \leq l$. A similar equation holds for the estimates $\hat{\eta}_j(\boldsymbol{x})$, $\widehat{\mathbf{P}}(z_i \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x})$, and $\widehat{\mathbf{P}}(\boldsymbol{z}^{n-1} \,|\, \boldsymbol{x})$.

By expressing $\eta_j(\boldsymbol{x})$ and $\hat{\eta}_j(\boldsymbol{x})$ in the aforementioned way we get:

$$\begin{aligned} |\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})| &= \left| \prod_{i=0}^{l} \mathbf{P}(z_i \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x}) - \prod_{i=0}^{l} \widehat{\mathbf{P}}(z_i \,|\, \boldsymbol{z}^{i-1}, \boldsymbol{x}) \right| \\ &= \left| \mathbf{P}(\boldsymbol{z}^{l-1} \,|\, \boldsymbol{x}) \mathbf{P}(z_l \,|\, \boldsymbol{z}^{l-1}, \boldsymbol{x}) - \widehat{\mathbf{P}}(\boldsymbol{z}^{l-1} \,|\, \boldsymbol{x}) \widehat{\mathbf{P}}(z_l \,|\, \boldsymbol{z}^{l-1}, \boldsymbol{x}) \right| \end{aligned}$$

[5] For more detailed introduction to strongly proper composite losses, we refer the reader to (Agarwal, 2014)

By adding and subtracting $\widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x})\mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})$ and using the fact that $|a-b| \leq |a-c|+|b-c|$, and that probability values are in $[0,1]$, we can write:

$$\begin{aligned}
|\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})| &= \Big|\mathbf{P}(\boldsymbol{z}^l \mid \boldsymbol{x}) - \widehat{\mathbf{P}}(\boldsymbol{z}^l \mid \boldsymbol{x})\Big| \\
&= \Big|\mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\mathbf{P}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x}) - \widehat{\mathbf{P}}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x})\Big| \\
&= \Big|\mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\mathbf{P}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x}) - \widehat{\mathbf{P}}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x}) \\
&\quad + \widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x})\mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x}) - \widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x})\mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\Big| \\
&\leq \Big|\mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\mathbf{P}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x}) - \widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x})\mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\Big| \\
&\quad + \Big|\widehat{\mathbf{P}}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x}) - \widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x})\mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\Big| \\
&= \mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\Big|\mathbf{P}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x}) - \widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x})\Big| \\
&\quad + \widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x})\Big|\widehat{\mathbf{P}}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x}) - \mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\Big| \\
&\leq \mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\Big|\mathbf{P}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x}) - \widehat{\mathbf{P}}(z_l \mid \boldsymbol{z}^{l-1}, \boldsymbol{x})\Big| \\
&\quad + \Big|\mathbf{P}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x}) - \widehat{\mathbf{P}}(\boldsymbol{z}^{l-1} \mid \boldsymbol{x})\Big|
\end{aligned}$$

We notice that rightmost term corresponds to the absolute value of the difference of probabilities corresponding to one-element shorter code $\boldsymbol{z}^{l-1}$. Therefore we can use recursion and write:

$$|\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})| \leq \sum_{i=0}^{l} \mathbf{P}(\boldsymbol{z}^{i-1} \mid \boldsymbol{x})\Big|\mathbf{P}(z_i|\boldsymbol{z}^{i-1}, \boldsymbol{x}) - \widehat{\mathbf{P}}(z_i|\boldsymbol{z}^{i-1}, \boldsymbol{x})\Big|. \tag{6}$$

Next, we express the above bound in terms of the regret of the strongly proper composite losses. The $(\boldsymbol{x}, z_i)$ pairs are generated i.i.d. according to $\mathbf{P}(\boldsymbol{x}, z_i \mid \boldsymbol{z}^{i-1})$. Assume that a node classifier has a form of a real-valued function $f_{\boldsymbol{z}^i}$. Moreover, there exists a strictly increasing (and therefore invertible) link function $\psi : [0,1] \to \mathbb{R}$ such that $f_{\boldsymbol{z}^i}(\boldsymbol{x}) = \psi(\mathbf{P}(z_i \mid \boldsymbol{z}^{i-1}, \boldsymbol{x}))$. Recall that the regret of $f_{\boldsymbol{z}^i}$ in terms of a loss function ℓ at point $\boldsymbol{x}$ is defined as:

$$\mathrm{reg}_\ell(f_{\boldsymbol{z}^i} \mid \boldsymbol{z}^{i-1}, \boldsymbol{x}) = L_\ell(f_{\boldsymbol{z}^i} \mid \boldsymbol{z}^{i-1}, \boldsymbol{x}) - L^*_\ell(\boldsymbol{z}^{i-1}, \boldsymbol{x})\,,$$

where $L_\ell(f_{\boldsymbol{z}^i} \mid \boldsymbol{z}^{i-1}, \boldsymbol{x})$ is the expected loss at point $\boldsymbol{x}$:

$$L_\ell(f_{\boldsymbol{z}^i} \mid \boldsymbol{z}^{i-1}, \boldsymbol{x}) = \mathbf{P}(z_i \mid \boldsymbol{z}^{i-1}, \boldsymbol{x})\ell(1, f_{\boldsymbol{z}^i}(\boldsymbol{x})) + (1 - \mathbf{P}(z_i \mid \boldsymbol{z}^{i-1}, \boldsymbol{x}))\ell(-1, f_{\boldsymbol{z}^i}(\boldsymbol{x}))\,,$$

and $L^*_\ell(\boldsymbol{x})$ is the minimum expected loss at point $\boldsymbol{x}$.

If a node classifier is trained by a learning algorithm that minimizes a strongly proper composite loss, then the bound (6) can be expressed in terms of the regret of this loss function (Agarwal, 2014):

$$\Big|\mathbf{P}(z_i \mid \boldsymbol{z}^{i-1}, \boldsymbol{x}) - \psi^{-1}(f_{\boldsymbol{z}^i})\Big| \leq \sqrt{\frac{2}{\lambda}}\sqrt{\mathrm{reg}_\ell(f_{\boldsymbol{z}^i} \mid \boldsymbol{z}^{i-1}, \boldsymbol{x})}\,.$$

By putting the above inequality into (6), we get

$$\begin{aligned}
|\eta(\boldsymbol{x}, j) - \hat{\eta}(\boldsymbol{x}, j)| &\leq \sum_{i=0}^{l} \mathbf{P}(\boldsymbol{z}^{i-1} \mid \boldsymbol{x})\,\Big|\mathbf{P}(z_i \mid \boldsymbol{z}^{i-1}, \boldsymbol{x}) - \widehat{\mathbf{P}}(z_i \mid \boldsymbol{z}^{i-1}, \boldsymbol{x})\Big| \\
&= \sum_{i=0}^{l} \mathbf{P}(\boldsymbol{z}^{i-1} \mid \boldsymbol{x})\,\Big|\mathbf{P}(z_i \mid \boldsymbol{z}^{i-1}, \boldsymbol{x}) - \psi^{-1}(f_{\boldsymbol{z}^i})\Big| \\
&\leq \sum_{i=0}^{l} \mathbf{P}(\boldsymbol{z}^{i-1} \mid \boldsymbol{x})\sqrt{\frac{2}{\lambda}}\sqrt{\mathrm{reg}_\ell(f_{\boldsymbol{z}^i} \mid \boldsymbol{z}^{i-1}, \boldsymbol{x})}
\end{aligned}$$

□

The above result shows that the absolute error of estimating the marginal probability of label j can be upperbounded by the regret of the node classifiers on the corresponding path from the root to a leaf. Moreover, for zero-regret (i.e., optimal) node classifiers we obtain an optimal multi-label classifier in terms of estimation of marginal probabilities $\eta_j(\boldsymbol{x})$. This result can be further extended for precision@k.

Let us denote a set of the top k labels with respect to the true marginals by $\mathcal{Y}_k$ and a set of the top k labels with respect to predicted marginals by $\hat{\mathcal{Y}}_k$. The conditional regret for precision@k is given then by:

$$\operatorname{reg}_{p@k}(\boldsymbol{h} \,|\, \boldsymbol{x}) = \frac{1}{k} \sum_{i \in \mathcal{Y}_k} \eta_i(\boldsymbol{x}) - \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \eta_j(\boldsymbol{x})$$

Theorem 2. *For any distribution* $\mathbf{P}$ *and classifier* $\boldsymbol{h}$ *delivering estimates* $\hat{\eta}_j(\boldsymbol{x})$ *of the marginal probabilities of labels, the following holds:*

$$\operatorname{reg}_{p@k}(\boldsymbol{h} \,|\, \boldsymbol{x}) = \frac{1}{k} \sum_{i \in \mathcal{Y}_k} \eta_i(\boldsymbol{x}) - \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \eta_j(\boldsymbol{x}) \leq 2 \max_l |\eta_l(\boldsymbol{x}) - \hat{\eta}_l(\boldsymbol{x})|$$

Proof. Let us add and subtract the following two terms, $\frac{1}{k} \sum_{i \in \mathcal{Y}_k} \hat{\eta}_i(\boldsymbol{x})$ and $\frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \hat{\eta}_j(\boldsymbol{x})$, to the regret and reorganize the expression in the following way:

$$\begin{aligned}
\operatorname{reg}_{p@k}(\boldsymbol{h} \,|\, \boldsymbol{x}) = & \underbrace{\frac{1}{k} \sum_{i \in \mathcal{Y}_k} \eta_i(\boldsymbol{x}) - \frac{1}{k} \sum_{i \in \mathcal{Y}_k} \hat{\eta}_i(\boldsymbol{x})}_{\leq \frac{1}{k} \sum_{i \in \mathcal{Y}_k} |\eta_i(\boldsymbol{x}) - \hat{\eta}_i(\boldsymbol{x})|} \\
& + \underbrace{\frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \hat{\eta}_j(\boldsymbol{x}) - \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \eta_j(\boldsymbol{x})}_{\leq \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} |\hat{\eta}_j(\boldsymbol{x}) - \eta_j(\boldsymbol{x})|} \\
& + \underbrace{\frac{1}{k} \sum_{i \in \mathcal{Y}_k} \hat{\eta}_i(\boldsymbol{x}) - \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} \hat{\eta}_j(\boldsymbol{x})}_{\leq 0}
\end{aligned}$$

Because of the relations given under the braces, we finally get:

$$\operatorname{reg}_{p@k}(\boldsymbol{h} \,|\, \boldsymbol{x}) \leq 2 \max_l |\eta_l(\boldsymbol{x}) - \hat{\eta}_l(\boldsymbol{x})| \ .$$

□

By getting together both theorems we get an upper bound of the precision@k regret expressed in terms of the regret of the node classifiers. Again, for the zero-regret node classifiers, we get optimal solution in terms of precision@k.

B Hierarchical softmax with the pick-one-label heuristic

Proposition 2. *Given conditionally independent labels, a classifier* $\boldsymbol{h}$ *such that* $h_j(\boldsymbol{x}) = \eta'_j(\boldsymbol{x})$ *for all* $j \in \{1, \ldots, m\}$ *has zero regret in terms of the precision@*k *loss.*

Proof. To proof the proposition it suffices to show that for conditionally independent labels the order of labels induced by the marginal probabilities $\eta_j(\boldsymbol{x})$ is the same as the order induced by the values of $\eta'_j(\boldsymbol{x})$ obtained by the pick-one-label heuristic (3):

$$\eta'_j(\boldsymbol{x}) = \mathbf{P}'(y_j = 1 \,|\, \boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \frac{y_j}{\sum_{j'=1}^{m} y_{j'}} \mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}).$$

In other words, for any two labels $i, j \in \{1, \ldots, m\}$, $i \neq j$, $\eta_i(\boldsymbol{x}) \geq \eta_j(\boldsymbol{x}) \Leftrightarrow \eta'_i(\boldsymbol{x}) \geq \eta'_j(\boldsymbol{x})$.

Let $\eta_i(\boldsymbol{x}) \geq \eta_j(\boldsymbol{x})$. The summation over all $\boldsymbol{y}$ in (3) can be written in the following way:

$$\eta'_j(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} y_j N(\boldsymbol{y}) \mathbf{P}(\boldsymbol{y}|\boldsymbol{x}) \,,$$

where $N(\boldsymbol{y}) = (\sum_{i=1}^{m} y_i)^{-1}$ is a value that depends only on the number of positive labels in $\boldsymbol{y}$. In this summation we consider four subsets of $\mathcal{Y}$, creating a partition of this set:

$$\mathcal{S}_{i,j}^{u,w} = \{\boldsymbol{y} \in \mathcal{Y} : y_i = u \wedge y_j = w\}, \quad u, w \in \{0, 1\}.$$

The subset $\mathcal{S}_{i,j}^{0,0}$ does not play any role because $y_i = y_j = 0$ and therefore do not contribute to the final sum. Then (3) can be written in the following way for the i-th and j-th label:

$$\eta'_i(\boldsymbol{x}) = \sum_{\boldsymbol{y}:\mathcal{S}_{i,j}^{1,0}} N(\boldsymbol{y}) \mathbf{P}(\boldsymbol{y}|\boldsymbol{x}) + \sum_{\boldsymbol{y} \in \mathcal{S}_{i,j}^{1,1}} N(\boldsymbol{y}) \mathbf{P}(\boldsymbol{y}|\boldsymbol{x}) \tag{7}$$

$$\eta'_j(\boldsymbol{x}) = \sum_{\boldsymbol{y}:\mathcal{S}_{i,j}^{0,1}} N(\boldsymbol{y}) \mathbf{P}(\boldsymbol{y}|\boldsymbol{x}) + \sum_{\boldsymbol{y} \in \mathcal{S}_{i,j}^{1,1}} N(\boldsymbol{y}) \mathbf{P}(\boldsymbol{y}|\boldsymbol{x}) \tag{8}$$

The contribution of elements from $\mathcal{S}_{i,j}^{1,1}$ is equal for both $\eta'_i(\boldsymbol{x})$ and $\eta'_j(\boldsymbol{x})$. It is so because the value of $N(\boldsymbol{y})\mathbf{P}(\boldsymbol{y}|\boldsymbol{x})$ is the same for all $\boldsymbol{y} \in \mathcal{S}_{i,j}^{1,1}$: the conditional joint probabilities $\mathbf{P}(\boldsymbol{y}|\boldsymbol{x})$ are fixed and they are multiplied by the same factors $N(\boldsymbol{y})$.

Consider now the contributions of $\mathcal{S}_{i,j}^{1,0}$ and $\mathcal{S}_{i,j}^{0,1}$ to the relevant sums. By the definition of $\mathcal{Y}$, $\mathcal{S}_{i,j}^{1,0}$, and $\mathcal{S}_{i,j}^{0,1}$, there exists bijection $b_{i,j} : \mathcal{S}_{i,j}^{1,0} \to \mathcal{S}_{i,j}^{0,1}$, such that for each $\boldsymbol{y}' \in \mathcal{S}_{i,j}^{1,0}$ there exists $\boldsymbol{y}'' \in \mathcal{S}_{i,j}^{0,1}$ equal to $\boldsymbol{y}'$ except on the i-th and the j-th position.

Notice that because of the conditional independence assumption the joint probabilities of elements in $\mathcal{S}_{i,j}^{1,0}$ and $\mathcal{S}_{i,j}^{0,1}$ are related to each other. Let $\boldsymbol{y}'' = b_{i,j}(\boldsymbol{y}')$, where $\boldsymbol{y}' \in \mathcal{S}_{i,j}^{1,0}$ and $\boldsymbol{y}'' \in \mathcal{S}_{i,j}^{0,1}$. The joint probabilities are:

$$\mathbf{P}(\boldsymbol{y}'|\boldsymbol{x}) = \eta_i(\boldsymbol{x})(1 - \eta_j(\boldsymbol{x})) \prod_{l \in \mathcal{L} \setminus \{i,j\}} \eta_l(\boldsymbol{x})^{y_l} (1 - \eta_l(\boldsymbol{x}))^{1 - y_l}$$

and

$$\mathbf{P}(\boldsymbol{y}''|\boldsymbol{x}) = (1 - \eta_i(\boldsymbol{x}))\eta_j(\boldsymbol{x}) \prod_{l \in \mathcal{L} \setminus \{i,j\}} \eta_l(\boldsymbol{x})^{y_l} (1 - \eta_l(\boldsymbol{x}))^{1 - y_l}.$$

One can easily notice the relation between these probabilities:

$$\mathbf{P}(\boldsymbol{y}'|\boldsymbol{x}) = \eta_i(\boldsymbol{x})(1 - \eta_j(\boldsymbol{x}))q_{i,j} \quad \text{and} \quad \mathbf{P}(\boldsymbol{y}''|\boldsymbol{x}) = (1 - \eta_i(\boldsymbol{x}))\eta_j(\boldsymbol{x})q_{i,j},$$

where $q_{i,j} = \prod_{l \in \mathcal{L} \setminus \{i,j\}} \eta_l(\boldsymbol{x})^{y_l} (1 - \eta_l(\boldsymbol{x}))^{1 - y_l} \geq 0$. Consider now the difference of these two probabilities:

$$\begin{aligned} \mathbf{P}(\boldsymbol{y}'|\boldsymbol{x}) - \mathbf{P}(\boldsymbol{y}''|\boldsymbol{x}) &= \eta_i(\boldsymbol{x})(1 - \eta_j(\boldsymbol{x}))q_{i,j} - (1 - \eta_i(\boldsymbol{x}))\eta_j(\boldsymbol{x})q_{i,j} \\ &= q_{i,j}(\eta_i(\boldsymbol{x})(1 - \eta_j(\boldsymbol{x})) - (1 - \eta_i(\boldsymbol{x}))\eta_j(\boldsymbol{x})) \\ &= q_{i,j}(\eta_i(\boldsymbol{x}) - \eta_j(\boldsymbol{x})). \end{aligned}$$

From the above we see that $\eta_i(\boldsymbol{x}) \geq \eta_j(\boldsymbol{x}) \Rightarrow \mathbf{P}(\boldsymbol{y}'|\boldsymbol{x}) \geq \mathbf{P}(\boldsymbol{y}''|\boldsymbol{x})$. Due to the properties of the bijection $b_{i,j}$, the number of positive labels in $\boldsymbol{y}'$ and $\boldsymbol{y}''$ is the same and $N(\boldsymbol{y}') = N(\boldsymbol{y}'')$, therefore we also get $\eta_i(\boldsymbol{x}) \geq \eta_j(\boldsymbol{x}) \Rightarrow \sum_{\boldsymbol{y}:\mathcal{S}_{i,j}^{1,0}} N(\boldsymbol{y})\mathbf{P}(\boldsymbol{y}|\boldsymbol{x}) \geq \sum_{\boldsymbol{y}:\mathcal{S}_{i,j}^{0,1}} N(\boldsymbol{y})\mathbf{P}(\boldsymbol{y}|\boldsymbol{x})$, which by (7) and (8) gives us finally $\eta_i(\boldsymbol{x}) \geq \eta_j(\boldsymbol{x}) \Rightarrow \eta'_i(\boldsymbol{x}) \geq \eta'_j(\boldsymbol{x})$.

The implication in the other side, i.e., $\eta_i(\boldsymbol{x}) \geq \eta_j(\boldsymbol{x}) \Leftarrow \mathbf{P}(\boldsymbol{y}'|\boldsymbol{x}) \geq \mathbf{P}(\boldsymbol{y}''|\boldsymbol{x})$ holds obviously for $q_{i,j} > 0$. For $q_{i,j} = 0$, we can notice, however, that $\mathbf{P}(\boldsymbol{y}'|\boldsymbol{x})$ and $\mathbf{P}(\boldsymbol{y}''|\boldsymbol{x})$ do not contribute to the appropriate sums as they are zero, and therefore we can follow a similar reasoning as above, concluding that $\eta_i(\boldsymbol{x}) \geq \eta_j(\boldsymbol{x}) \Leftarrow \eta'_i(\boldsymbol{x}) \geq \eta'_j(\boldsymbol{x})$.

Thus for conditionally independent labels, the order of labels induced by marginal probabilities $\eta_j(\boldsymbol{x})$ is equal to the order induced by $\eta'_j(\boldsymbol{x})$. As the precision@k is optimized by k labels with the highest marginal probabilities, we have that prediction consisted of k labels with highest $\eta'_j(\boldsymbol{x})$ has zero regret for precision@k. □

C Huffman codes for PLTs

In this section we analyze computational cost of PLTs in binary case, i.e. every inner node has two children. We define the cost of a tree as the total expected fraction of instances which are used in the inner nodes and show that for the multi-class case minimization of this cost coincides with minimization of Huffman criteria. However, in the multi-label case, this does no longer hold.

We shall use prefix codes, as it is introduced in Section 4, to identify a path from the root to the leaf. Accordingly, a prefix code $\boldsymbol{z} = (z_1, \ldots, z_\ell) \in \mathcal{C}$ determines a path with length $|\boldsymbol{z}| = \ell$. The probability to observe a (possibly partial) prefix code is $p_{\boldsymbol{z}^i} = \mathbf{P}(\boldsymbol{z}^i)$ with respect to the data distribution. If we are given a data that consists of n instances, then the expected number of positive instances that is used to train the node classifier in node $\boldsymbol{z}^i$ is $np_{\boldsymbol{z}^i}$. Note that in the training process of PLT, only the positive instances shows up in the training data of a child node of $\boldsymbol{z}^i$, thus one can define the expected computational cost of a PLT with fixed structure, thus with fixed set of prefix code $\mathcal{C}$, as

$$\sum_{\boldsymbol{z} \in \mathcal{C}} \sum_{i=1}^{|\boldsymbol{z}|} p_{\boldsymbol{z}^{i-1}}. \tag{9}$$

The next proposition defines the relation of (9) and Huffman coding. Huffman coding can be naturally defined in the multi-class case using $\mathbf{P}(y_i = 1)$ as weights.

Proposition 3. *Huffman code minimizes the expected computational cost that is given in (9) among binary codes if the data is multi-class.*

Proof. Recall that the Huffman code minimizes the following criterion:

$$\sum_{\boldsymbol{z} \in \mathcal{C}} p_{\boldsymbol{z}} |\boldsymbol{z}|.$$

In multi-class setting score $p_{\boldsymbol{z}^i}$ of each inner node $\boldsymbol{z}^i$ equals to the sum of scores of its children, and the objective function given in (9) can be rewritten as

$$\sum_{\boldsymbol{z} \in \mathcal{C}} \sum_{i=1}^{|\boldsymbol{z}|} p_{\boldsymbol{z}^{i-1}} = 1 + \sum_{\boldsymbol{z} \in \mathcal{C}} p_{\boldsymbol{z}} \sum_{i=1}^{|\boldsymbol{z}|} \deg(\boldsymbol{z}^i),$$

where $\deg(\boldsymbol{z}^i) = \#\{c : \hat{z}_{i+1} = c, \boldsymbol{z}^i = \hat{\boldsymbol{z}}^i, \hat{\boldsymbol{z}} \in \mathcal{C}\}$ is the number of children of the node $\boldsymbol{z}^i$. In the case of binary codes $\deg(\boldsymbol{z}^i) = 2$ and

$$1 + \sum_{\boldsymbol{z} \in \mathcal{C}} p_{\boldsymbol{z}} \sum_{i=1}^{|\boldsymbol{z}|} \deg(\boldsymbol{z}^i) = 1 + 2 \sum_{\boldsymbol{z} \in \mathcal{C}} p_{\boldsymbol{z}} |\boldsymbol{z}|$$

which completes the proof. □

Note that Grave et al. (2017) considered similar notion of computational cost and optimized in a restricted setup, assuming that the tree depth is at most two. Of course, in practice, the $p_{\boldsymbol{z}^i}$ values are not known, but are estimated based on observations from the data distribution. Let us denote an estimate $\hat{p}_{\boldsymbol{z}^i}$ of $p_{\boldsymbol{z}^i}$. An interesting question to address is that to what extent the empirical computational cost $\sum_{\boldsymbol{z} \in \mathcal{C}} \sum_{i=1}^{|\boldsymbol{z}|} \hat{p}_{\boldsymbol{z}^{i-1}}$ concentrates around its expected values, and on what parameter of the code/tree structure does depend on.

For multi-label case score of an inner node cannot be represented by a sum of children scores, and this result does not hold.

D Pseudocode of PLTs

The pseudocode below presents the training and prediction procedures of PLTs in detail. PLTs can be trained either in the online or batch mode. Algorithm 1 follows the former mode used, for example, in XT, the state-of-the-art variant of PLTs described in Section 7. Note that learning of feature embeddings used in XT is not included in the pseudocode.

In Algorithm 2 we present an inference algorithm, which uses the uniform-cost search for finding the top k labels. The algorithm searches the tree by starting from the root node $\boldsymbol{z}^0$. It uses a priority queue Q to store pairs of the node $\boldsymbol{z}^i$ and its probability estimate $\hat{\eta}_{\boldsymbol{z}^i}$. The algorithm pops the element with the highest probability estimate from the queue. If the element is a leaf we add the corresponding label to the final prediction. Otherwise, the algorithm estimates probabilities of the children nodes and adds them into the queue. Once we found the k-th label or we run out of the pairs in the queue, we stop the search procedure.

Algorithm 1 Incremental learning of a PLT:

Input:
T: a label tree with t nodes
A_{online}: an incremental learning algorithm
$\mathcal{D}_N$: a set of training examples $(\boldsymbol{x}, \boldsymbol{y})$

Output:
$\mathcal{F}_t = \{f_{\boldsymbol{z}^i}\}^t$: a set of t node (binary) classifiers

$\mathcal{F}_t = \emptyset$
for each node $\boldsymbol{z}^i \in T$ **do** ▷ Initialization of binary classifiers
    $\mathcal{F}_t \leftarrow \mathcal{F}_t \cup$ new classifier $f_{\boldsymbol{z}^i}$
for each training example $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_N$ **do**
    if $\sum_{j=1}^{m} y_j = 0$ **then** ▷ Select nodes for the positive and negative update
        $\mathcal{Z}_{\text{positive}} \leftarrow \emptyset$
        $\mathcal{Z}_{\neg\text{positive}} \leftarrow \boldsymbol{z}^0$
    else
        for each $y_j = 1, j \in \{1, \ldots, m\}$ **do**
            $\boldsymbol{z} \leftarrow$ encode j
            $\mathcal{Z}_{\text{positive}} \leftarrow \mathcal{Z}_{\text{positive}} \cup \left\{ \bigcup_{i=0}^{l} \boldsymbol{z}^i \right\}$
        for each $\boldsymbol{z}^i \in \mathcal{Z}_{\text{positive}}$ **do**
            $\mathcal{Z}_{\neg\text{positive}} \leftarrow \mathcal{Z}_{\neg\text{positive}} \cup \left\{ \bigcup_{z_i} (\boldsymbol{z}^{i-1}, z_i) \right\} \setminus \mathcal{Z}_{\text{positive}}$
    for each node $\boldsymbol{z}^i \in \mathcal{Z}_{\text{positive}}$ **do** ▷ Update node classifiers
        $f_{\boldsymbol{z}^i} \leftarrow A_{\text{online}}(f_{\boldsymbol{z}^i}, \boldsymbol{x}, 1)$
    for each node $\boldsymbol{z}^i \in \mathcal{Z}_{\neg\text{positive}}$ **do**
        $f_{\boldsymbol{z}^i} \leftarrow A_{\text{online}}(f_{\boldsymbol{z}^i}, \boldsymbol{x}, 0)$
return $\mathcal{F}_t$.

E Additional experimental results

E.1 Comparison of PLTs and HSM on synthetic data

In this section we validate our theoretical results presented in Section 5, which show that HSM is not amenable to model the marginal probabilities in general for multi-label problems. In this case, HSM with pick-one-label heuristic should be outperformed in terms of precision@k by PLTs which are consistent for this performance measure. To validate this claim empirically, we compare the performance of PLTs and HSM on synthetic datasets of three types: multi-label data with independent labels, multi-label data with conditionally dependent labels and multi-class data.

All synthetic models are based on linear models parametrized by a weight vector $\boldsymbol{w}$ of size d. The values of the vector are sampled uniformly from a d-dimensional sphere of radius 1. Each instance $\boldsymbol{x}$, in turn, is represented as a vector sampled from a d-dimensional disc of the same radius.

Algorithm 2 Top-k prediction with a PLT:

Input:
T: a label tree with t nodes
$\mathcal{F}_t = \{f_{\boldsymbol{z}^i}\}^t$: a set of t node (binary) classifiers
$\boldsymbol{x}$: a test example
k: a size of prediction

Output:
$\hat{\boldsymbol{y}}$: a vector with top k labels for the test example

$\hat{\boldsymbol{y}} \leftarrow \{0\}^m$
$Q \leftarrow$ PRIORITYQUEUE() ▷ Initialization of priority queue
$\mathrm{add}\big(Q, (1, \boldsymbol{z}^0)\big)$
while $Q \neq \emptyset$ and $\sum_{j=1}^{m} \hat{y}_j < k$ **do** ▷ Check if k labels not found
  $\big(\hat{\eta}_{\boldsymbol{z}^i}, \boldsymbol{z}^i\big) \leftarrow \mathrm{pop}(Q)$
  if $i \in \mathcal{C}$ **then** ▷ Check if leaf node reached
    $j \leftarrow$ decode $\boldsymbol{z}^i$
    $\hat{y}_j \leftarrow 1$
  else
    for each z_{i+1} **do** ▷ For each node's children
      $\boldsymbol{z}^{i+1} \leftarrow (\boldsymbol{z}^i, z_i)$
      $\mathrm{add}(Q, (\hat{\eta}_{\boldsymbol{z}^i} \cdot \widehat{\mathbf{P}}_{f_{\boldsymbol{z}^i}}(z_{i+1} \mid \boldsymbol{z}^i, \boldsymbol{x}), \boldsymbol{z}^{i+1}))$
return $\hat{\boldsymbol{y}}$.

Multi-class distribution. We associate a weigh vector $\boldsymbol{w}_j$ with each label $j \in \{1, \ldots, m\}$. The model assigns probabilities to labels at point $\boldsymbol{x}$ based on the softmax schema

$$\eta_j(\boldsymbol{x}) = \frac{\exp(c\boldsymbol{w}_j^\top \boldsymbol{x})}{\sum_{j'=1}^{m} \exp(c\boldsymbol{w}_{j'}^\top \boldsymbol{x})} \tag{10}$$

and draws the positive label according to this probability distribution over labels. Scaling factor c is added to control noise in the model. Higher values of c give less noisy model.

Multi-label distribution with conditionally independent labels. The model is similar to the previous one used for the multi-class distribution. The difference lays is normalization as the marginal probabilities do not have to sum up to 1. To get a probability of the j-th label, we use the logistic transformation:

$$\eta_j(\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_j^\top \boldsymbol{x})}{1 + \exp(\boldsymbol{w}_j^\top \boldsymbol{x})}.$$

Then, we assign a label to an instance based on:

$$y_j = [\![r < \eta_j(\boldsymbol{x})]\!],$$

where the random value r is sampled uniformly and independently from range $[0, 1]$ for each instance $\boldsymbol{x}$ and label $j \in \{1, \ldots, m\}$.

Multi-label distribution with conditionally dependent labels. To model conditionally dependent labels we use the mixing matrix model based on latent scoring functions generated by $\boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m)$. The $m \times m$ mixing matrix $\boldsymbol{M}$ introduces dependencies between noise $\boldsymbol{\epsilon}$, which stands for the source of randomness in the model. The models $\boldsymbol{w}_j$ are sampled from a sphere of radius 1, as in previous cases. The values in the mixing matrix $\boldsymbol{M}$ are sampled uniformly and independently from $[-1, 1]$. The random noise vector $\boldsymbol{\epsilon}$ is sampled from $N(0, 0.25)$. The label vector $\boldsymbol{y}$ is then obtained by element-wise evaluation of the following expression:

$$\boldsymbol{y} = [\![\boldsymbol{M}(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{\epsilon}) > 0]\!]$$

Notice that if $\boldsymbol{M}$ was an identity matrix the model would generate independent labels.

Table 2: Means and standard deviations of 50 runs of each experiment. The p-values on the right indicate the significance of the observed differences. The results of HSM and PLTs on multi-class data were always equal.

	HSM		PLT		p-values		
	mean	stdev	mean	stdev	t-test	sign	Wilcoxon
multi-class	21.90	2.74	21.90	2.74			
multi-label independent	32.57	0.34	32.58	0.33	0.4367	0.3222	0.5980
multi-label dependent	70.68	5.73	71.68	5.65	9.80E-14	3.71E-11	3.90E-09

Experimental setting. PLTs and HSM are usually implemented as online learning algorithms, i.e., the node classifiers are updated in an incremental way, example by example. To minimize the impact of the hyperparameter tuning of online algorithms, we have decided to implement batch versions of both algorithms using the LIBLINEAR-based (Fan et al., 2008) logistic regression. The sets of training instances for each node classifier are appropriately constructed by taking the corresponding conditioning of the probabilistic models into account. In the case of HSM with the pick-one-label heuristic, we first transform each multi-label example to s multi-class copies of it, one copy for each its label. Each such copy gets then a weight of $1/s$. Such transformation should be concordant with the theoretical model (3).

In the experiments we used the following parameters of the synthetic models: $d = 3$ (i.e., the number of features), $n = 100000$ instances (split $1 : 1$ for training and test subsets), and $m = 32$ labels or classes. In the case of the multi-class model we report results with the scaling factor $c = 10$. The choice of c does not change the interpretation of the results. To train logistic regression we use a fixed value of the regularization parameter, standing for a very weak regularization. For all experiments we report the results in terms of precision@1.

Observations. Table 2 presents the average results of all experiments besides with the standard deviation of obtained values. As expected the performance of PLTs and HSM on the multi-class data are exactly the same. For the other models, we additionally include the p-values of statistical tests run to verify, whether there is a significant difference in performance between PLTs and HSM with the pick-one-label heuristic. In the case of the multi-label data with conditionally independent labels the test shows that there is no evidence to reject the hypothesis that the performance of PLTs and HSM is the same. In the case of multi-label data with conditionally dependent labels, the statistical tests show that PLTs are significantly better than HSM with the pick-one-label heuristic.

E.2 Comparison of PLTs and HSM on benchmark data

In Table 3 we present similar results, but obtained on the benchmark datasets. We use two models. The first one follows the sparse representation and uses an implementation of PLTs and HSM in VOWPAL WABBIT (Langford et al., 2007). The second one is based on FASTTEXT and produces the dense representation. In all models we use Huffman trees. The results clearly indicate the better performance of PLTs over HSM.

Table 3: Precision@k with $k = \{1, 3, 5\}$ of a simple HSM and a simple PLT implementaions.

	VOWPAL WABBIT		FASTTEXT			VOWPAL WABBIT		FASTTEXT	
Dataset	HSM	PLT	HSM	PLT	Dataset	HSM	PLT	HSM	PLT
	56.98	**74.55**	66.39	**73.19**		86.69	**91.46**	90.18	**92.98**
EUR-Lex	46.99	**60.60**	54.05	**57.79**	**AmazonCat-13K**	72.00	**76.00**	72.53	**75.75**
	39.09	**50.05**	44.73	**46.98**		57.97	**61.40**	56.20	**59.53**
	70.20	**84.34**	83.02	**85.11**		41.58	**45.27**	42.17	**46.98**
Wiki-30K	60.11	**72.34**	69.66	**73.12**	**Delicious-200K**	33.24	**38.95**	37.94	**40.99**
	53.17	**62.72**	59.50	**62.67**		28.04	**35.59**	35.77	**38.04**
	36.90	**41.63**	41.28	**41.78**		33.64	**36.85**	25.04	**26.18**
WikiLSHTC-325K	22.30	**26.78**	24.68	**24.96**	**Amazon-670K**	28.58	**32.48**	21.06	**22.76**
	16.60	**20.39**	18.08	**18.53**		25.01	**29.15**	18.28	**20.29**

E.3 The ablation analysis on benchmark datasets

Table 4 contains results of the ablation analysis in which we compare different components of the XT algorithm. We analyze the influence of the Huffman tree vs. top-down clustering, the simple averaging of features vectors vs. the TF-IDF-based weighting, and no regularization vs. L2 regularization. For every configuration, we conducted a grid search of hyperparameters from ranges reported in Appendix E.4. The results clearly show that the components need to combined together to obtain the best results. The best combination is usually the one that uses top-down clustering, TF-IDF-based weighting, and L2 regularization. It is worth to notice that top-down clustering alone gets worse results than Huffman trees with TF-IDF-based weighting and L2 regularization.

Table 4: Precision@k scores with $k = \{1, 3, 5\}$ of different variants of PLT in FASTTEXT (EXTREMETEXT)

Dataset	Metrics	Huff.	Huff. + TF-IDF	Huff. + L2	Huff. + L2 + TF-IDF	Clus.	Clus. + TF-IDF	Clus. + L2	Clus. + L2 + TF-IDF
Eurlex	P@1	63.39	71.20	62.79	74.60	68.31	75.05	65.05	**77.68**
	P@3	50.48	57.26	48.99	60.36	54.62	61.65	50.87	**63.37**
	P@5	41.19	46.93	40.16	49.72	45.23	50.99	41.71	**52.85**
AmazonCat-13K	P@1	90.10	89.19	90.84	91.08	72.95	77.13	91.73	**92.43**
	P@3	72.67	72.81	73.10	75.27	61.66	65.76	75.07	**77.65**
	P@5	57.69	58.30	57.69	60.34	48.38	52.72	59.63	**62.74**
Wiki-30K	P@1	78.04	78.14	82.28	**85.23**	78.40	78.69	82.13	85.21
	P@3	63.31	67.47	68.75	**73.52**	65.62	66.40	69.22	73.18
	P@5	52.65	56.00	58.33	**63.71**	55.61	56.77	58.69	63.39
Delicious-200K	P@1	45.71	47.24	46.48	**47.85**	44.58	45.13	46.55	47.31
	P@3	40.69	40.88	41.38	**42.08**	40.42	40.63	41.08	41.63
	P@5	38.02	37.74	38.78	**39.13**	38.15	38.23	38.25	38.88
WikiLSHTC-325K	P@1	36.23	38.28	41.10	42.83	34.39	39.66	54.95	**58.73**
	P@3	20.60	22.43	25.62	26.33	21.17	24.79	36.42	**39.24**
	P@5	14.70	16.33	19.10	19.45	15.09	18.28	27.25	**29.26**
Wiki-500K	P@1	41.01	40.62	41.55	49.80	46.63	49.20	56.04	**64.48**
	P@3	24.79	25.68	25.94	32.39	31.59	34.52	38.87	**45.84**
	P@5	18.64	19.74	19.79	24.62	24.30	26.83	30.46	**35.46**
Amazon-670K	P@1	23.19	29.70	21.64	32.11	28.54	36.24	28.95	**39.90**
	P@3	19.80	25.84	18.11	27.77	25.52	32.15	25.74	**35.36**
	P@5	17.48	22.96	15.83	24.64	23.30	29.19	23.22	**32.04**

E.4 Tuning of hyperparameters

The PLT has only one global hyperparameter which is the degree of the tree denoted by b. The other hyperparameters are associated with the node classifiers. The HSM and PLT in Vowpal Wabbit was tuned with the stochastic gradient descent with a step size η_t calculated separately for each node according to $\eta_t = \eta \times (1/t)^p$ where t is number of node updates and η and p are hyperparameters. In FASTTEXT-based methods, HSM, LEARNED TREE and XT, η_t decreased linearly during training from η to 0.0. In the XT the optimization methods has been extended by L2 regularization, so it has one additional parameter. Balanced k-means clustering used to build a tree in XT has also a stopping parameter ϵ set by default to 0.001.

Table 5: The hyperparameters of the HSM and PLT methods and their ranges used in hyperparameter optimization. Notation: b – tree arity, η – learning rate

Hyperparameter	Range
b	$\{2, \ldots, 32\}$
η	$[0.0001 - 1.0]$
number of epochs	$\{20, 30, 40\}$

F Information about the benchmark datasets

Table 8 contains the basic statistics of the benchmark datasets used in the experiments taken from the Extreme Classification Repository: `http://manikvarma.org/downloads/XC/XMLRepository.html`.

Table 6: The hyperparameters of the FASTTEXT and LEARNED TREE methods and their ranges used in hyperparameter optimization. Notation: b – tree arity, η – learning rate

Hyperparameter	Range
b	$\{2, \ldots, 32\}$
η	$[0.0001 - 1.0]$
number of epochs	$\{20, 30, 40\}$
dim	$\{500\}$

Table 7: The hyperparameters of the XT method and their ranges used in hyperparameter optimization. Notation: b – tree arity (number of centroids used in k-means clustering), ϵ – stoping condition of k-means clustering, η – learning rate

Hyperparameter	Range
b	$\{2\}$
ϵ	$\{0.001\}$
η	$\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$
L2 regularization	$\{0.001, 0.002, 0.003\}$
number of epochs	$\{20, 30, 40\}$
dim	$\{500\}$
max leaves	$\{100\}$

Table 8: Statistics of the benchmark datasets.

Dataset	Number of features	Number of labels	Number of train points	Number of test points	Avg. points per label	Avg. labels per point
EURLex-4K	5000	3993	15539	3809	25.73	5.31
AmazonCat-13K	203882	13330	1186239	306782	448.57	5.04
Wiki10-31K	101938	30938	14146	6616	8.52	18.64
Delicious-200K	782585	205443	196606	100095	72.29	75.54
WikiLSHTC-325K	1617899	325056	1778351	587084	17.46	3.19
Wikipedia-500K	2381304	501070	1813391	783743	24.75	4.77
Amazon-670K	135909	670091	490449	153025	3.99	5.45