Machine Learning (2020) 109:2099–2119
https://doi.org/10.1007/s10994-020-05888-2

Bonsai: diverse and shallow trees for extreme multi-label classification

Sujay Khandagale[1] · **Han Xiao**[1] · **Rohit Babbar**[1]

Received: 13 July 2019 / Revised: 1 November 2019 / Accepted: 4 June 2020 / Published online: 23 August 2020

Abstract
Extreme multi-label classification (*XMC*) refers to supervised multi-label learning involving hundreds of thousands or even millions of labels. In this paper, we develop a suite of algorithms, called `Bonsai`, which generalizes the notion of label representation in XMC, and partitions the labels in the representation space to learn shallow trees. We show three concrete realizations of this label representation space including: (i) the input space which is spanned by the input features, (ii) the output space spanned by label vectors based on their co-occurrence with other labels, and (iii) the joint space by combining the input and output representations. Furthermore, the constraint-free multi-way partitions learnt iteratively in these spaces lead to shallow trees. By combining the effect of shallow trees and generalized label representation, `Bonsai` achieves the best of both worlds—fast training which is comparable to state-of-the-art tree-based methods in XMC, and much better prediction accuracy, particularly on tail-labels. On a benchmark Amazon-3M dataset with 3 million labels, `Bonsai` outperforms a state-of-the-art one-vs-rest method in terms of prediction accuracy, while being approximately 200 times faster to train. The code for `Bonsai` is available at https://github.com/xmc-aalto/bonsai.

Keywords Large-scale multi-label classification · Extreme multi-label classification · Large label space

1 Introduction

Extreme Multi-label Classification (*XMC*) refers to supervised learning of a classifier which can automatically label an instance with a small subset of relevant labels from an extremely large set of all possible target labels. Machine learning problems consisting of hundreds of thousand labels are common in various domains such as product categorization for e-commerce (McAuley and Leskovec 2013; Shen et al. 2011; Bengio et al. 2010; Agrawal et al. 2013), hash-tag suggestion in social media (Denton et al.

Editors: Larisa Soldatova, Joaquin Vanschoren.

✉ Rohit Babbar
rohit.babbar@aalto.fi

[1] Aalto University, Helsinki, Finland

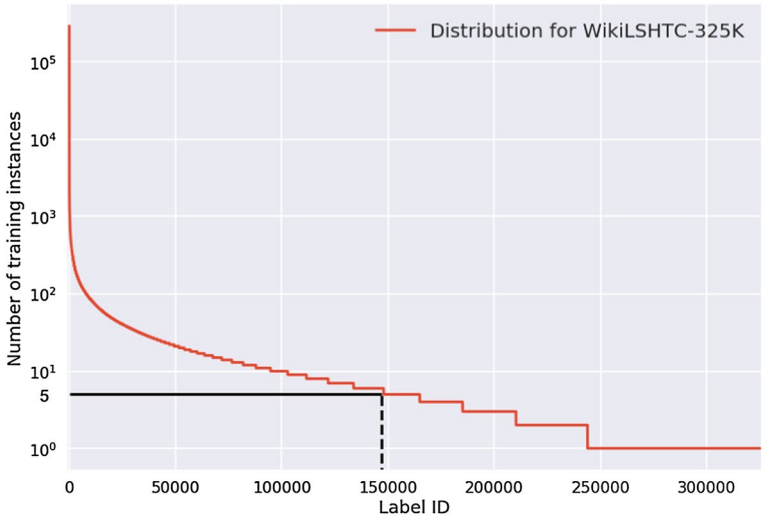

Fig. 1 Label frequency in dataset WikiLSHTC-325K shows power-law distribution. X-axis shows the label IDs sorted by their frequency in training instances and Y-axis gives the actual frequency (on log-scale). Note that more than half of the labels have fewer than 5 training instances

2015), annotating web-scale encyclopedia (Partalas et al. 2015), and image-classification (Krizhevsky et al. 2012; Deng et al. 2010). It has been demonstrated that, the framework of XMC can also be leveraged to effectively address ranking problems arising in bid-phrase suggestion in web-advertising and suggestion of relevant items for recommendation systems (Prabhu and Varma 2014).

From the machine learning perspective, building effective extreme classifiers is faced with the *computational* challenge arising due to large number of (i) output labels, (ii) input training instances, and (iii) input features. Another important statistical characteristic of the datasets in XMC is that a large fraction of labels are *tail labels*, i.e., those which have very few training instances that belong to them (also referred to as power-law, fat-tailed distribution and Zipf's law). Formally, let N_r denote the size of the r-th ranked label, when ranked in decreasing order of number of training instances that belong to that label, then:

$$N_r = N_1 r^{-\beta} \tag{1}$$

where N_1 represents the size of the 1-st ranked label and $\beta > 0$ denotes the exponent of the power law distribution. This distribution is shown in Fig. 1 for a benchmark dataset, WikiLSHTC-325K from the XMC repository (Bhatia et al. 2016). In this dataset, only $\sim$ 150,000 out of 325,000 labels have more than 5 training instances in them. Tail labels exhibit diversity of the label space, and contain informative content not captured by the head or torso labels. Indeed, by predicting well the head labels, yet omitting most of the tail labels, an algorithm can achieve high accuracy (Wei and Li 2018). However, such behavior is not desirable in many real world applications, where fit to power-law distribution has been observed (Babbar et al. 2014).

1.1 Related work

Multi-label learning has long been a topic of interest with early focus on relatively smaller scale problems (Tsoumakas and Katakis 2007; Tsoumakas et al. 208; Read et al. 2008; Vens et al. 2008; Madjarov et al. 2012). However, most works in the context of large-scale scenarios which fall under the realm of XMC, can be broadly categorized into one of the four strands:

1. *Tree-based* Tree-based methods implement a divide-and-conquer paradigm and scale to large label sets in XMC by partitioning the labels space. As a result, these scheme of methods have the computational advantage of enabling faster training and prediction (Prabhu and Varma 2014; Jain et al. 2016; Jasinska et al. 2016; Majzoubi and Choromanska 2019; Wydmuch et al. 2018). Approaches based on decision trees have also been proposed for multi-label classification and those tailored to XMC regime (Joly et al. 2019; Si et al. 2017). However, tree-based methods suffer from error propagation in the tree cascade as also observed in hierarchical classification (Babbar et al. 2013, 2016). As a result, these methods tend to perform particularly worse on metrics which are sensitive for tail-labels Prabhu et al. (2018).
2. *Label embedding* Label-embedding approaches assume that, despite large number of labels, the label matrix is effectively low rank and therefore project it to a low-dimensional sub-space. These approaches have been at the fore-front in multi-label classification for small scale problems with few tens or hundred labels (Hsu et al. 2009; Tai and Lin 2012; Weston et al. 2011; Lin et al. 2014). For power-law distributed labels in XMC settings, the crucial assumption made by the embedding-based approaches of a low rank label space breaks down (Xu et al. 2016a; Bhatia et al. 2015; Tagami 2017). Under this condition, embedding based approaches leads to high prediction error.
3. *One-vs-rest* Sometimes also referred to as binary relavance (Zhang et al. 2018), these methods learn a classifier per label which distinguishes it from rest of the labels. In terms of prediction accuracy and label diversity, these methods have been shown to be among the best performing ones for XMC (Babbar and Schölkopf 2017; Yen et al. 2017; Babbar and Schölkopf 2019). However, due to their reliance on a distributed training framework, it remains challenging to employ them in resource constrained environments.
4. *Deep learning* Deeper architectures on top of word-embeddings have also been explored in recent works (Liu et al. 2017; Joulin et al. 2017; Mikolov et al. 2013). However, their performance still remains sub-optimal compared to the methods discussed above which are based on bag-of-words feature representations. This is mainly due to the data scarcity in tail-labels which is substantially below the sample complexity required for deep learning methods to reach their peak performance.

Therefore, a central challenge in XMC is to build classifiers which retain the accuracy of one-vs-rest paradigm while being as efficiently trainable as the tree-based methods. Recently, there have been efforts for speeding up the training of existing classifiers by better initialization and exploiting the problem structure (Fang et al. 2019; Liang et al. 2018; Jalan et al. 2019). In a similar vein, a recently proposed tree-based method, `Parabel` (Prabhu et al. 2018), partitions the label space recursively into two child nodes using 2-means clustering. It also maintains a balance between these two label partitions in terms of number of labels. Each intermediate node in the resulting binary label-tree is like a meta-label which captures the generic properties of its constituent labels. The leaves of the tree consist of the actual labels from the training data. During training and prediction each of these labels

is distinguished from other labels under the same parent node through the application of a binary classifier at internal nodes and one-vs-all classifier for the leaf nodes. By combination of tree-based partitioning and one-vs-rest classifier, it has been shown to give better performance than previous tree-based methods (Prabhu and Varma 2014; Jain et al. 2016; Jasinska et al. 2016) while simultaneously allowing efficient training.

However, in terms of prediction performance, `Parabel` remains sub-optimal compared to one-vs-rest approaches. In addition to error propagation *due to cascading effect of the deep trees*, its performance is particularly worse on tail labels. This is the result of two strong constraints in its label partitioning process, (i) each parent node in the tree has only *two* child nodes, and (ii) at each node, the labels are partitioned into *equal sized* parts, such that the number of labels under the two child nodes differ by at most one. As a result of the coarseness imposed by the binary partitioning of labels, the tail labels get subsumed by the head labels.

1.2 Bonsai overview

In this paper, we develop a family of algorithms, called `Bonsai`. At a high level, `Bonsai` follows a similar paradigm which is common in most tree-based approaches, i.e., label partitioning followed by learning classifiers at the internal nodes. However, it has two main features, which distinguish it from state-of-the-art tree based approaches. These are summarized below :

- *Generalized label representation* In this work, we argue that the notion of representing the labels is quite general, and there are various meaningful manifestations of the label representation space. As three concrete examples, we show the applicability of the following representations of labels: (i) input space representation as a function of feature vectors (ii) output space representation based on their co-occurrence with other labels, and (iii) a combination of the output and input representations. In this regard, our work generalizes the approach taken in many earlier works, which have represented labels only in the input space (Prabhu et al. 2018; Wydmuch et al. 2018), or only in the output space (Tsoumakas et al. 208). We show that these representations, when combined with shallow trees (described in the next section), surpass existing methods demonstrating the efficacy of the proposed generalized representation.
- *Shallow trees* To avoid error propagation in the tree cascade, we propose to construct a shallow tree architecture. This is achieved by enabling (i) a flexible clustering via K-means for $K > 2$, and (ii) relaxing balancedness constraints in the clustering step. Multi-way partitioning initializes diverse sub-groups of labels, and the unconstrained nature maintains the diversity during the entire process. These are in contrast to tree-based methods which impose such constraints for a balanced tree construction. As we demonstrate in our empirical findings, by relaxing the constraints, `Bonsai` leads to prediction diversity and significantly better tail-label coverage.

By synergizing the effect of a richer label representation and shallow trees, `Bonsai` achieves the best of both worlds—prediction diversity better than state-of-the-art tree-based methods with comparable training speed, and prediction accuracy at par with one-vs-rest methods. The code for `Bonsai` is available at https://github.com/xmc-aalto/bonsai.

2 Formal description of Bonsai

We assume to be given a set of N training points $\left\{(\mathbf{x}_i, \mathbf{y}_i)\right\}_{i=1}^{N}$ with D dimensional feature vectors $\mathbf{x}_i \in \mathbb{R}^D$ and L dimensional label vectors $\mathbf{y}_i \in \{0,1\}^L$. Without loss of generality, let the set of labels be represented by $\{1, \ldots, \ell, \ldots, L\}$. Our goal is to learn a multi-label classifier in the form of a vector-valued output function $f : \mathbb{R}^D \mapsto \{0,1\}^L$. This is typically achieved by minimizing an empirical estimate of $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[\mathcal{L}(\mathbf{W};(\mathbf{x},\mathbf{y}))]$ where $\mathcal{L}$ is a loss function, and samples $(\mathbf{x}, \mathbf{y})$ are drawn from some underlying distribution $\mathcal{D}$. The desired parameters $\mathbf{W}$ can take one of the myriad of choices. In the simplest (and yet effective) of setups for XMC such as linear classification, $\mathbf{W}$ can be in the form of matrix. In other cases, it can be representative of a deeper architecture or a cascade of classifiers in a tree structured topology. Due to their scalability to extremely large datasets, `Bonsai` follows a tree-structured partitioning of labels.

In this section, we next present in detail the two main components of `Bonsai`: (i) generalized label representation and (ii) shallow trees.

2.1 Label representation

In the extreme classification setting, labels can be represented in various ways. To motivate this, as an analogy in terms of publications and their authors, one can think of labels as authors, the papers they write as their training instances, and multiple co-authors of a paper as the multiple labels. Now, one can represent authors (labels) either solely based on the content of the papers they authored (input space representation), or based only on their co-authors (output space representation) or as a combination of the two.

Formally, let each label ℓ be represented by η-dimensional vector $\mathbf{v}_\ell \in \mathbb{R}^\eta$. Now, $\mathbf{v}_\ell$ can be represented as a function (i) *only* of input features via the input vectors in training instances $\left\{\mathbf{x}_i\right\}_{i=1}^{N}$, (ii) *only* of output features via the label vectors in the training instances $\left\{\mathbf{y}_i\right\}_{i=1}^{N}$ or (iii) as a combination of *both* $\left\{(\mathbf{x}_i, \mathbf{y}_i)\right\}_{i=1}^{N}$. We now present three concrete realizations of the label representation $\mathbf{v}_\ell$. We later show that these representations can be seamlessly combined with shallow tree cascade of classifiers, and yield state-of-the-art performance on XMC tasks.

(a) *Input space label representation* The label representation for label ℓ can be arrived at by summing all the training examples for which it is active. Let $\mathbf{V}_i$ be the label representation matrix given by

$$\mathbf{V}_i = \mathbf{Y}^T\mathbf{X} = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_L^T \end{bmatrix}_{L\times D} \quad \text{where} \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}_{N\times D}, \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^T \\ \mathbf{y}_2^T \\ \vdots \\ \mathbf{y}_N^T \end{bmatrix}_{N\times L}. \tag{2}$$

We follow the notation that each bold letter such as $\mathbf{x}$ is a vector in column format and $\mathbf{x}^T$ represents the corresponding row vector. Hence, each row $\mathbf{v}_\ell$ of matrix $\mathcal{V}_i$ which represents the label ℓ, is given by the sum of all the training instances for which label ℓ is active. This can also be represented as, $\mathbf{v}_\ell = \sum_{i=1}^{N} \mathbf{y}_{i\ell}\mathbf{x}_i$. Note that even though $\mathbf{v}_\ell$ also depends on the label vectors, it is still in the same space as the input instance and has dimensionality D. Furthermore, each $\mathbf{v}_\ell$ can be normalized to unit length in euclidean norm as follows: $\mathbf{v}_\ell := \mathbf{v}_\ell / \|\mathbf{v}_\ell\|_2$.

(b) *Output space representation* In the multi-label setting, another way to represent the labels is to represent them solely as a function of the degree of their co-occurrence with other labels. That is, if two labels co-occur with similar set of labels, then these are bound to be related to each other, and hence should have similar representation. In this case, the label representation matrix $\mathbf{V}_o$ is given by

$$\mathbf{V}_o = \mathbf{Y}^T\mathbf{Y} = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_L^T \end{bmatrix}_{L\times L} \quad \text{where} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^T \\ \mathbf{y}_2^T \\ \vdots \\ \mathbf{y}_N^T \end{bmatrix}_{N\times L} . \tag{3}$$

Here $\mathbf{V}_o$ is an $L \times L$ symmetric matrix, where each row $\mathbf{v}_\ell^T$, corresponds to the number of times the label ℓ co-occurs with all other labels. Hence these label co-occurrence vectors $\mathbf{v}_\ell$ give us another way of representing the label ℓ. It may be noted that in contrast to the previous case, being an output space representation, the dimensionality of the label vector is same as that of the output space having the same dimensionality, i.e. $\eta = L$.

(c) *Joint input–output representation* Given the previous input and output space representations of labels, a natural way to extend it is by combining these representations via concatenation. This is achieved as follows, for a training instance i with feature vector $\mathbf{x}_i$ and corresponding label vector $\mathbf{y}_i$, let $\mathbf{z}_i$ be the concatenated vector given by, $\mathbf{z}_i = [\mathbf{x}_i \odot \mathbf{y}_i]$. Then, the joint representation can be computed in the matrix $\mathbf{V}_j$ as follows

$$\mathbf{V}_j = \mathbf{Y}^T\mathbf{Z} = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_L^T \end{bmatrix}_{L\times(D+L)} \quad \text{where} \quad \mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_N^T \end{bmatrix}_{N\times(D+L)} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^T \\ \mathbf{y}_2^T \\ \vdots \\ \mathbf{y}_N^T \end{bmatrix}_{N\times L} \tag{4}$$

Here each row $\mathbf{v}_\ell$ of the label representation matrix $\mathcal{V}_j$ which is the label representation in the joint space, is therefore a concatenation of representations obtained from $\mathcal{V}_i$ and $\mathcal{V}_o$, hence being of length $(D + L)$. Since both the input vectors $\mathbf{x}_i$ and output vectors $\mathbf{y}_i$ are highly sparse, this does not lead to any major computational burden in training.

It may be noted that our notion of label representation generalizes similar approaches in recent works (i) which are either based solely on the input space representation (Prabhu et al. 2018; Wydmuch et al. 2018), or (ii) those which are based on output space representation only (Tsoumakas et al. 208). As also shown later in our empirical findings, in combination with shallow tree cascade of classifiers, partitioning of:

- output space representation ($\mathbf{V}_o$) yields competitive results compared to state-of-the-art classifiers in XMC such as `Parabel`.
- joint representation ($\mathbf{V}_j$) further surpasses the state-of-the-art methods in terms of prediction performance and label diversity.

2.2 Label partitioning via *K*-means clustering

Once we have obtained the representation $\mathbf{v}_\ell$ for each label ℓ in the set $S = \{1, \ldots, L\}$, the next step is to iteratively partition S into disjoint subsets. This is achieved by K-means clustering, which also presents many choices such as number of clusters and degree of balanced-ness among the clusters. Our goal, in this work, is to avoid propagation error in a deep tree cascade. We, therefore, choose a relatively large value of K (e.g. ≥ 100) which leads to shallow trees.

The clustering step in `Bonsai` first partitions S into K disjoint sets $\{S_1, \ldots, S_K\}$. Each of the elements, S_k, of the above set can be thought of as a meta-label which semantically groups actual labels together in one cluster. Then, K child nodes of the root are created, each contains one of the partitions, $\{S_k\}_{k=1}^K$. The same process is repeated on each of the newly-created K child nodes in an iterative manner. In each sub-tree, the process terminates either when the node's depth exceeds pre-defined threshold $d_{\max}$ or the number of associated labels is no larger than K, e.g, $|S_k| \leq K$.

Formally, without loss of generality, we assume a non-leaf node has labels $\{1, \ldots, L\}$. We aim at finding K cluster centers $\mathbf{c}_1, \ldots, \mathbf{c}_K \in \mathbb{R}^\eta$, i.e., in an appropriate space (input, output, or joint) by optimizing the following:

$$\min_{\mathbf{c_1},\ldots,\mathbf{c_K}\in\mathbb{R}^\eta} \left[\sum_{k=1}^{K} \sum_{\ell \in \mathbf{c}_i} d(\mathbf{v}_\ell, \mathbf{c}_k) \right] \tag{5}$$

where $d(.,.)$ represents a distance function and $\mathbf{v}_\ell$ represents the vector representation of the label ℓ. The distance function is defined in terms of the dot product as follows: $d(\mathbf{v}_\ell, \mathbf{c}_k) = 1 - \mathbf{v}_\ell^T \cdot \mathbf{c}_k$. The above problem is **NP**-hard and we use the standard K-means algorithm (also known as Lloyd's algorithm) (Lloyd 1982)[1] for finding an approximate solution to Eq. (5).

The K-way unconstrained clustering in `Bonsai` has the following advantages over `Parabel` which enforces binary and balanced partitioning:

1. *Initializing label diversity in partitioning* By setting $K > 2$, `Bonsai` allows a varied partitioning of the labels space, rather than grouping all labels in two clusters. This facet of `Bonsai` is especially favorable for tail labels by allowing them to be part of separate clusters if they are indeed very different from the rest of the labels. Depending on the similarity to other labels, each label can choose to be part of one of the K clusters.
2. *Sustaining label diversity* `Bonsai` sustains the diversity in the label space by *not enforcing* the balanced-ness constraint of the form, $||S_k| - |S_{k'}|| \leq 1, \forall 1 \leq k, k' \leq K$ (where $|.|$ operator is overloaded to mean set cardinality for the inner one and absolute value for the outer ones) among the partitions. This makes the `Bonsai` partitions more data-dependent since smaller partitions with diverse tail-labels are very moderately penalized under this framework.
3. *Shallow tree cascade* Furthermore, K-way unconstrained partitioning leads to shallower trees which are less prone propagation error in deeper trees constructed by `Parabel`. As we will show in Sect. 3, the diverse partitioning reinforced by shallower architecture

[1] We also tried K-means++ and observed that faster convergence did not out-weigh extra computation time for seed initialization.

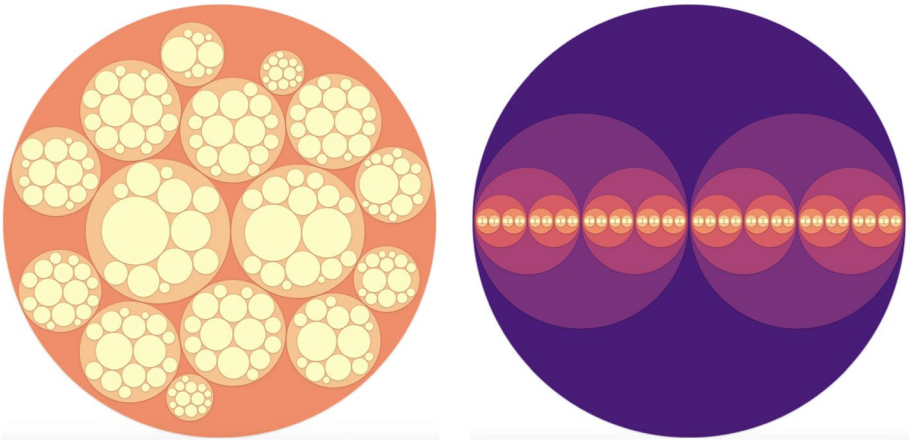

Fig. 2 Comparison of partitioned label space by `Bonsai` and `Parabel` on **EURLex-4K** dataset. Each circle corresponds to one label partition (also a tree node), the size of circle indicates the number of labels in that partition and lighter color indicates larger node level. The largest circle is the whole label space. Note that `Bonsai` produces label partitions of varying sizes, while `Parabel` gives perfectly balanced partitioning

leads to better prediction performance, and significant improvement is achieved on tail labels.

A pictorial description of the partitioning scheme of `Bonsai` and its difference compared to `Parabel` is also illustrated in Fig. 2.

2.3 Learning node classifiers

Once the label space is partitioned into a diverse and shallow tree structure, we learn a K -way One-vs-All linear classifier at each node. These classifiers are trained independently using only the training examples that have at least one of the node labels. We distinguish the leaf nodes and non-leaf nodes in the following way: (i) for non-leaf nodes, the classifier learns K linear classifiers separately, each maps to one of the K children. During prediction, the output of each classifier determines whether the test point should traverse down the corresponding child. (ii) for leaf nodes, the classifier learns to predict the actual labels on the node.

Without loss of generality, given a node in the tree, denote by $\{c_k\}_{k=1}^{K}$ as the set of its children. For the special case of leaf nodes, the set of children represent the final labels. We learn K linear classifiers parameterized by $\{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$, where $\mathbf{w}_k \in \mathbb{R}^D$ for $\forall k = 1, \ldots, K$. Each output label determines if the corresponding K children should be traversed or not.

For each of the child node c_k, we define the training data as $T_k = (\mathbf{X}_k, \mathbf{s}_k)$, where $\mathbf{X}_k = \{\mathbf{x}_i \mid \mathbf{y}_{ik} = 1, i = 1, \ldots, N\}$. Let $\mathbf{s}_k \in \{+1, -1\}^N$ represent the vector of signs depending on whether $\mathbf{y}_{ik} = 1$ corresponds to $+1$ and $\mathbf{y}_{ik} = 0$ for -1. We consider the following optimization problem for learning linear SVM with squared hinge loss and ℓ_2 -regularization

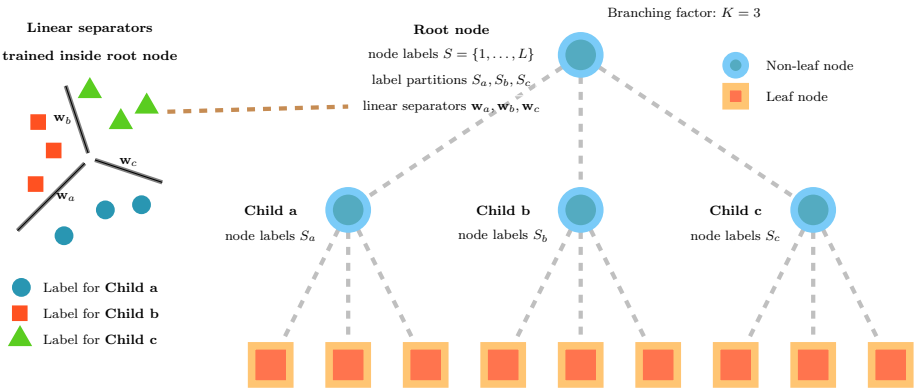

Fig. 3 Illustration of `Bonsai` architecture. During training, label are partitioned hierarchically, resulting in a tree structure of label partitions. In order to obtain diverse and shallow trees, the branching factor, K is set to large values (e.g, $K \geq 100$) in `Bonsai` (shown as 3 for better pictorial illustration). Inside non-leaf nodes, linear classifiers are trained to predict which child nodes to traverse down during prediction. Inside leaf nodes, linear classifiers are trained to predict the actual labels

$$\min_{\mathbf{w_k}} \left[||\mathbf{w}_k||_2^2 + C \sum_{i=1}^{|\mathbf{X}_k|} \mathcal{L}(s_{k_i} \mathbf{w}_k^T \mathbf{x}_i) \right] \tag{6}$$

where $\mathcal{L}(z) = (\max(0, 1 - z))^2$. This is solved using the Newton method based primal implementation in LIBLINEAR (Fan et al. 2008). To restrict the model size, and remove spurious parameters, thresholding of small weights is performed as in Babbar and Schölkopf (2017). Similar to Parabel, one-vs-all classifiers are also learnt at the leaf nodes, which consist of the actual labels.

The tree-structured architecture of `Bonsai` is illustrated in Fig. 3. The details of `Bonsai` 's training procedure are shown in Algorithm 1. The partitioning process in Sect. 2.1 is described as the procedure `GROW` in the algorithm. The One-vs-All procedure is shown as ONE-VS-ALL in Algorithm 1.

2.4 Prediction error propagation in shallow versus deep trees

During prediction, a test point $\mathbf{x}$ traverses down the tree. At each non-leaf node, the classifier narrows down the search space by deciding which subset of child nodes $\mathbf{x}$ should further traverse. If the classifier decides not to traverse down some child node c, all descendants of c will not be traversed. Later, as $\mathbf{x}$ reaches to one or more leaf nodes, One-vs-All classifiers are evaluated to assign probabilities to each label. `Bonsai` uses beam search to avoid the possibility of evaluating all nodes. At each level, B most probable nodes, whose scores are calculated from previous level, are further traversed down.

Algorithm 1: Training algorithm: GROW$(n, K, d_{\max})$ partitions label space recursively and returns K children nodes of n. K-means(L, K) partitions label set L into K disjoint sets using standard K-means algorithm. Label features are derived from training data T. ONE-VS-ALL$(T, \{\ell_1, \ldots, \ell_K\})$ learns K one-vs-rest linear classifiers $\{\mathbf{w}_k\}_{k=1}^{K}$.

Input : Training data $T = \{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{N}\}$, where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \{0,1\}^L$
branching factor $K \geq 2$, maximum depth $d_{\max}$
Output: a tree rooted at r

$r \leftarrow$ new node;
$r.d \leftarrow 0$; // d: node depth
$r.L \leftarrow \{1, \ldots, L\}$; // $\mathcal{L}$: associated label set
$r.T \leftarrow \{1, \ldots, N\}$; // $\mathcal{I}$: associated training instance ids
$\{n_1, \ldots, n_K\} \leftarrow$ GROW$(r, d_{\max}, K)$; // grow the root recursively
$r.\mathcal{C} \leftarrow \{n_1, \ldots, n_K\}$; // $\mathcal{C}$: set of child nodes
return r;
procedure GROW(n, d_{max}, K):
  $S_1, \ldots, S_K \leftarrow K$-means$(n.L, K)$; // K-way split of labels
  for $k = 1, \ldots, K$ **do**
    $n_k \leftarrow$ new node;
    $n_k.L \leftarrow S_k$;
    $n_k.d \leftarrow n.d + 1$;
    $n_k.T \leftarrow \{i \in n.T \mid \exists \ell \in S_k \text{ s.t. } \mathbf{y}_{i\ell} = 1\}$;
    if $K \geq |n_k.L|$ *or* $n_k.d \geq d_{max}$ **then**
      $n_k.\mathbf{w} \leftarrow$ ONE-VS-ALL$(n_k.T, n_k.L)$; // n_k is a leaf
    else
      $\{c_1, \ldots, c_K\} \leftarrow$ GROW$(n_k, d_{\max}, K)$; // n_k is non-leaf
      $n_k.\mathcal{C} \leftarrow \{c_1, \ldots, c_K\}$;
    end
    $n.\mathbf{w} \leftarrow$ ONE-VS-ALL$(n.T, \{\ell_{n_1}, \ldots, \ell_{n_K}\})$; // each n_k maps to a meta label
  end
  return $\{n_k\}_{k=1}^{K}$;

However, the above search space pruning strategy implies errors made at non-leaf nodes could propagate to their descendants. `Bonsai` sets relatively large values to the branching factor K (typically 100), resulting in much shallower trees compared to `Parabel`, and hence significantly reducing error propagation, particularly for tail-labels.

More formally, given a data point $\mathbf{x}$ and a label ℓ that is relevant to $\mathbf{x}$, we denote e as the leaf node ℓ belongs to and $\mathcal{A}(e)$ as the set of ancestor nodes of e and e itself. Note that $|\mathcal{A}(e)|$ is path length from root to e. Denote the parent of n as $p(n)$. We define the binary indicator variable z_n to take value 1 if node n is visited during prediction and 0 otherwise. From the chain rule, the probability that ℓ is predicted as relevant for $\mathbf{x}$ is as follows:

$$\Pr(\mathbf{y}_\ell = 1 \mid \mathbf{x}) = \Pr(\mathbf{y}_\ell = 1 \mid z_e = 1, \mathbf{x}) \times \prod_{n \in \mathcal{A}(e)} \Pr(z_n = 1 \mid z_{p(n)} = 1, \mathbf{x})$$

Consider the **Amazon-3M** dataset with $L \approx 3 \times 10^6$, setting $K = 2$ produces a tree of depth 16. Assuming $\Pr(z_n = 1 \mid z_{p(n)} = 1, \mathbf{x}) = 0.95$, for $\forall n \in p(n)$ and $\Pr(\mathbf{y}_\ell = 1 \mid z_e = 1, \mathbf{x}) = 1$, it gives $\Pr(\mathbf{y}_\ell = 1 \mid \mathbf{x}) = (0.95)^{16} \approx 0.46$. This is to say, even if $\Pr(z_n = 1 \mid z_{p(n)} = 1, \mathbf{x})$ is high (e.g, 0.95) at each $n \in \mathcal{A}(e)$, multiplying them together can result in small probability (e.g, 0.46) if the depth of the tree, i.e., $|\mathcal{A}(e)|$ is *large*. We choose to mitigate this issue by increasing K, and hence limiting the propagation error.

2.5 Computational complexity

Training time analysis The training process can be decomposed into three steps: (i) learning the label representation, (ii) building the k-ary trees and (iii) learning a one-vs-rest classifier at each node.

First, we assume only $\log(L)$ labels are relevant for every data point on average. Also, let $\tilde{D}$ be the average feature density i.e. for dense features, $\tilde{D} = D$. For the three variants of the label representation $\mathbf{v}_\ell$ discussed in Sect. 2.1, learning the label representations requires a cost of $\mathcal{O}(N\tilde{D}\log L)$, $\mathcal{O}(N\log^2 L)$ and $\mathcal{O}(N(\tilde{D}+\log L)\log L)$ for the input, output and the joint input–output space respectively.

When building the label tree, it takes $\mathcal{O}(cKL\tilde{D})$ to cluster the labels at each level, where c is the number of iterations needed for K-means clustering to converge. Since `Bonsai` produces trees with small depth values, which can be considered small constant, the total time cost at this step is $\mathcal{O}(cKL\tilde{D})$.

With the learnt label tree, K independent linear classifiers are learnt at each internal tree node which decide which child nodes a training point should traverse. Learning internal node classifiers at each level takes $\mathcal{O}(KN\tilde{D}\log L)$. Therefore, the total time cost on learning internal node classifiers is $\mathcal{O}(KN\tilde{D}\log L)$ since we omit the tree depth, which is a small constant.

Lastly, the one-vs-rest leaf node classifiers are trained at a cost of $\mathcal{O}(MN\tilde{D}\log L)$ assuming that a leaf node can contain at most M labels. As we do not have any balanced-ness constraints on the K-means clustering, M can be equal to L in the worst case. However, in practice M is found to be much smaller. So the overall complexity of `Bonsai` for training T trees is $\mathcal{O}\left(\left(\frac{cKL}{N\log L}+K+M\right)TN\tilde{D}\log L\right)$.

Prediction time analysis The prediction process can be decomposed into two steps: (i) traversal down from the root through the intermediate nodes, (ii) label prediction at the leaf nodes.

For part (i), we use the fact that: at each level, at most B nodes are further traversed down. The time cost at each level is $\mathcal{O}(B\tilde{D}K)$. Therefore, traversing down all levels takes $\mathcal{O}(B\tilde{D}K)$ since tree depth is a small constant. For part (ii), at most B leaf nodes are evaluated. This step has complexity $\mathcal{O}(B\tilde{D}M)$, assuming that a leaf node contains at most M labels. Therefore, if we predict using T trees, the total complexity is $\mathcal{O}(TB\tilde{D}K + TB\tilde{D}M)$.

Comparison with `Parabel` We highlight the difference of complexity between `Bonsai` and `Parabel`.

For training, `Parabel` takes $\mathcal{O}((\frac{cKL}{N}+K\log L+M)TN\tilde{D}\log L)$[2] while `Bonsai` takes $\mathcal{O}\left(\left(\frac{cKL}{N\log L}+K+M\right)TN\tilde{D}\log L\right)$. `Bonsai` differs from `Parabel` in three ways: (i) a factor of $\log L$ (equals tree depths in `Parabel`) is absent in the first two terms in `Bonsai` as tree depths are small constants in `Bonsai`. (ii) M is generally larger in `Bonsai` since balanced-ness is not enforced in label partitioning. (iii) c is also larger in `Bonsai` because `Bonsai` sets a larger K value during K-means clustering, which takes more iterations to converge.

For prediction, `Parabel` takes $\mathcal{O}(TB\tilde{D}K\log L + TB\tilde{D}M)$ while `Bonsai` takes $\mathcal{O}(TB\tilde{D}K + TB\tilde{D}M)$. The main difference is similar as in the case of training: (i) `Bonsai`

[2] c is omitted in the original version, since $K = 2$ and it takes only a few iterations for K-means to converge.

Table 1 Multi-label datasets used in the experiment

Dataset	# Training	# Test	# Labels	# Features	APpL	ALpP
EURLex-4K	15,539	3809	**3993**	5000	25.7	5.3
Wikipedia-31K	14,146	6616	**30,938**	101,938	8.5	18.6
WikiLSHTC-325K	1,778,351	587,084	**325,056**	1,617,899	17.4	3.2
Wikipedia-500K	1,813,391	783,743	**501,070**	2,381,304	24.7	4.7
Amazon-670K	490,499	153,025	**670,091**	135,909	3.9	5.4
Amazon-3M	1,717,899	742,507	**2,812,281**	337,067	31.6	36.1

APpL and ALpP represent average points per label and average labels per point respectively

gets rid of the $\log L$ factor in the first term because `Bonsai` trees are shallow; (ii) meanwhile, M is generally larger in the case of `Bonsai`.

Though their training/prediction complexity are not directly comparable, we find that `Parabel` is faster in both training and prediction in practice. Therefore, we conclude that: in practice the role of larger M and c values rule over the absence of $\log L$ factor, therefore making `Bonsai` slower than `Parabel`.

3 Experimental evaluation

In this section, we detail the dataset description, and the set up for comparison of the proposed approach against state-of-the-art methods in XMC.

3.1 Dataset and evaluation metrics

We perform empirical evaluation on publicly available datasets from the XMC repository[3] curated from sources such as Amazon for item-to-item recommendation tasks and Wikipedia for tagging tasks. The datasets of various scales in terms of number of labels are used, **EURLex-4K** consisting of approximately 4000 labels to **Amazon-3M** consisting of 3 million labels. The datasets also exhibit a wide range of properties in terms of number of training instances, features, and labels. Though the overall feature dimensionality is high, each training instance is a tf-idf weighted sparse representation of features. The document length corresponding to each training sample can be further reduced by keeping the highest scores based on truncating at some pre-defined threshold (Khandagale and Babbar 2019). The detailed statistics of the datasets are shown in Table 1.

With applications in recommendation systems, ranking and web-advertising, the objective of the machine learning system in XMC is to correctly recommend/rank/advertise among the top-k slots. We therefore use evaluation metrics which are standard and commonly used to compare various methods under the XMC setting—Precision@k ($prec@k$) and normalised Discounted Cumulative Gain ($nDCG@k$). Given a label space of dimensionality L, a predicted label vector $\hat{\mathbf{y}} \in \mathbb{R}^L$ and a ground truth label vector $\mathbf{y} \in \{0,1\}^L$:

[3] http://manikvarma.org/downloads/XC/XMLRepository.html.

$$prec@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{k} \sum_{\ell \in rank_k(\hat{\mathbf{y}})} \mathbf{y}_\ell \tag{7}$$

$$nDCG@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{DCG@k}{\sum_{\ell=1}^{\min(k, ||\mathbf{y}||_0)} \frac{1}{\log(\ell+1)}} \tag{8}$$

where $DCG@k = \frac{\mathbf{y}_\ell}{\sum_{l=1} \frac{1}{\log(\ell+1)}}$, and $rank_k(\hat{\mathbf{y}})$ returns the k largest indices of $\hat{\mathbf{y}}$.

For better readability, we report the percentage version of above metrics (multiplying the original scores by 100). In addition, we consider $k \in \{1, 3, 5\}$.

3.2 Methods for comparison

We consider three different variants of the proposed family of algorithms, `Bonsai`, which is based on the generalized label representations (discussed in Sect. 2.1) combined with the shallow tree cascades. We refer the algorithms learnt by partitioning the input space, output space and the joint space as `Bonsai`-i, `Bonsai`-o, and `Bonsai`-io respectively. These are compared against six state-of-the-art algorithms from each of the three main strands for XMC namely, label-embedding, tree-based and one-vs-all methods:

- *Label-embedding methods* Due to the fat-tailed distribution of instances among labels, `SLEEC` (Bhatia et al. 2015) makes a locally low-rank assumption on the label space, `RobustXML` (Xu et al. 2016b) decomposes the label matrix into tail labels and non tail labels so as to enforce an embedding on the latter without the tail labels damaging the embedding. `LEML` (Yu et al. 2014) makes a global low-rank assumption on the label space and performs a linear embedding on the label space. As a result, it gives much worse results, and is not compared explicitly in the interest of space.
- *Tree-based methods* `FastXML` (Prabhu and Varma 2014) learns an ensemble of trees which partition the label space by directly optimizing an nDCG based ranking loss function, `PFastXML` (Jain et al. 2016) replaces the nDCG loss in `FastXML` by its propensity scored variant which is unbiased and assigns higher rewards for accurate tail label predictions, `Parabel` (Prabhu et al. 2018) which has been described earlier in the paper.
- *One-vs-All methods* `PD-Sparse` (Yen et al. 2016) enforces sparsity by exploiting the structure of a margin-maximizing loss with L1-penalty, `DiSMEC` (Babbar and Schölkopf 2017) learns one-vs-rest classifiers for every label with weight pruning to control model size. It may be noted that even though the actual number of true labels is unknown for the test set, evaluation metrics based on top-k predictions can still be computed for predictions for One-vs-All methods.

Since we are considering only bag-of-words representation across all datasets, we do not compare against deep learning methods explicitly. However, it may be noted that despite using raw data and corresponding word-embeddings, deep learning methods in XMC are still sub-optimal in terms of prediction performance in XMC (Liu et al. 2017; Joulin et al. 2017; Kim 2014). More details on the performance of deep methods can be found in Wydmuch et al. (2018).

Bonsai is implemented in C++ on a 64-bit Linux system. For all the datasets, we set the branching factor $K = 100$ at every tree depth. We will explore the effect of tree depth in details later. This results in depth-1 trees (excluding the leaves which represent the final labels) for smaller datasets such as **EURLex-4K**, **Wikipedia-31K** and depth-2 trees for larger datasets such as **WikiLSHTC-325K** and **Wikipedia-500K**. Bonsai learns an ensemble of three trees similar to Parabel.

For all the other state-of-the-art approaches, we used the hyperparameter values as suggested in the various papers in order to recreate the results reported in them.

4 Experimental results

In this section, we report the main findings of our empirical evaluation.

4.1 Precision@*k*

The comparison of Bonsai against other baselines is shown in Table 2. The results are averaged over five runs with different initializations of the clustering algorithm. The important findings from these results are the following:

- The competitive performance of the different variants of Bonsai shows the success and applicability of the notion of *generalized label representation*, and their concrete realization discussed in Sect. 2.1. It also highlights that it is possible to enrich these representations further, and achieve better partitioning.
- The consistent improvement of Bonsai over Parabel on all datasets validates the choice of higher fanout and advantages of using *shallow trees*.
- Another important insight from the above results is that when the average number of labels per training point are higher such as in Wikipedia-31K, Amazon-670K and Amazon-3M, the joint space label representation, used in bonsai-io, leads to better partitioning and further improves the strong performance of input only label representation in Bonsai-i. However, it degrades when the average number of labels per point is low (≤ 5) for datasets such as WikiLSHTC-325K and Wikipedia-500K, in which cases the information captured between input and output representations does not synergize well.
- Even though DiSMEC performs slightly better on Wiki-500K and Wikipedia-31K, its computational complexity of training and prediction is orders of magnitude higher than Bonsai. As a result, while Bonsai can be run in environments with limited computational resources, DiSMEC requires a distributed infrastructure for training and prediction.

4.2 Performance on tail labels

We also evaluate prediction performance on tail labels using propensity scored variants of *prec@k* and *nDCG@k*. For label ℓ, its propensity p_ℓ is related to number of its positive training instances N_ℓ by $p_\ell = \frac{1}{\left(1+Ce^{-A\log(N_\ell+B)}\right)}$ where A, B are application specific parameters and $C = (\log N - 1)(B+1)^A$. Here N is the total number of training samples, and

Table 2 (P@k) on benchmark datasets for $k = 1$, 3 and 5

Dataset	Our Approach (Bonsai)			Embedding based		Tree based		Linear one-vs-rest	
	Bonsai-i	Bonsai-o	Bonsai-io	SLEEC	Robust XML	Fast-XML	Parabel	PD-Sparse	DiSMEC
EURLex-4K									
P@1	**83.0**	82.5	82.9	79.3	78.7	71.4	82.2	76.4	82.4
P@3	**69.7**	69.4	69.4	64.3	63.5	59.9	68.7	60.4	68.5
P@5	**58.4**	58.1	58.0	52.3	51.4	50.4	57.5	49.7	57.7
Wikipedia-31K									
P@1	84.7	84.70	84.8	**85.5**	85.5	82.5	84.2	73.8	84.1
P@3	73.6	73.57	73.6	73.6	74.0	66.6	72.5	60.9	**74.6**
P@5	64.7	64.81	64.8	63.1	63.8	56.7	63.4	50.4	**65.9**
WikiLSHTC-325K									
P@1	**66.6**	63.4	65.8	55.5	53.5	49.3	65.0	58.2	64.4
P@3	**44.5**	42.8	44.1	33.8	31.8	32.7	43.2	36.3	42.5
P@5	**33.0**	32.0	32.7	24.0	29.9	24.0	32.0	28.7	31.5
Wikipedia-500K									
P@1	69.2	68.7	69.1	48.2	41.3	54.1	68.7	–	**70.2**
P@3	49.8	48.8	49.7	29.4	30.1	35.5	49.6	–	**50.6**
P@5	38.8	37.6	38.8	21.2	19.8	26.2	38.6	–	**39.7**
Amazon-670K									
P@1	45.5	44.5	**45.7**	35.0	31.0	33.3	44.9	–	44.7
P@3	40.3	39.8	**40.6**	31.2	28.0	29.3	39.8	–	39.7
P@5	36.5	36.4	**36.9**	28.5	24.0	26.1	36.0	–	36.1
Amazon-3M									
P@1	48.4	47.5	**48.5**	–	–	44.2	47.5	–	47.8
P@3	**45.6**	44.7	45.5	–	–	40.8	44.6	–	44.9
P@5	43.4	42.6	**43.5**	–	–	38.6	42.5	–	42.8

For each case of P@k and dataset, the best performed score is highlighted in bold. Entries marked "–" imply the corresponding method could not scale to the particular dataset, thus the scores are unavailable

parameters A, B vary across datasets, and were chosen as suggested in Jain et al. (2016). With this formulation, $p_\ell \approx 1$ for head labels and $p_\ell \ll 1$ for tail labels.

Let $\mathbf{y} \in \{0,1\}^L$ and $\hat{\mathbf{y}} \in \mathbb{R}^L$ denote the true and predicted label vectors respectively. As detailed in Jain et al. (2016), propensity scored variants of $P@k$ and $nDCG@k$ are given by

$$PSP@k(\hat{\mathbf{y}}, \mathbf{y}) := \frac{1}{k} \sum_{\ell \in rank_k(\hat{\mathbf{y}})} \mathbf{y}_\ell / p_\ell \tag{9}$$

$$PSnDCG@k(\hat{\mathbf{y}}, \mathbf{y}) := \frac{PSDCG@k}{\sum_{\ell=1}^{\min(k, ||\mathbf{y}||_0)} \frac{1}{\log(\ell+1)}} \tag{10}$$

where $PSDCG@k := \sum_{\ell \in rank_k(\hat{\mathbf{y}})} [\frac{\mathbf{y}_\ell}{p_\ell \log(\ell+1)}]$, and $rank_k(\mathbf{y})$ returns the k largest indices of $\mathbf{y}$. To match against the ground truth, as suggested in Jain et al. (2016), we use $100 \cdot \mathbb{G}(\{\hat{\mathbf{y}}\})/\mathbb{G}(\{\mathbf{y}\})$ as the performance metric. For M test samples, $\mathbb{G}(\{\hat{\mathbf{y}}\}) = \frac{-1}{M} \sum_{i=1}^{M} \mathbb{L}(\hat{\mathbf{y}}_i, \mathbf{y})$, where $\mathbb{G}(.)$ and $\mathbb{L}(.,.)$ signify gain and loss respectively. The loss $\mathbb{L}(.,.)$ can take two forms, *(i)* $\mathbb{L}(\hat{\mathbf{y}}_i, \mathbf{y}) = -PSnDCG@k$, and *(ii)* $\mathbb{L}(\hat{\mathbf{y}}_i, \mathbf{y}) = -PSP@k$. This

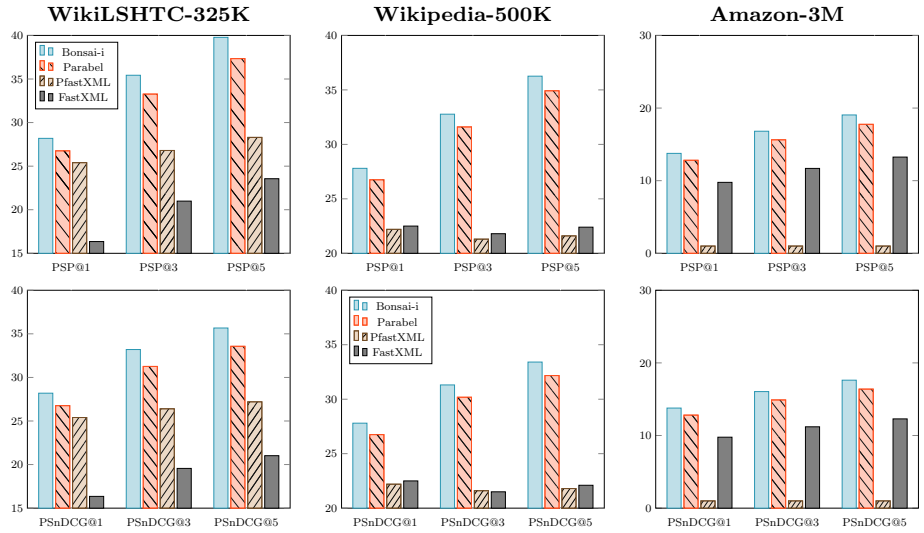

Fig. 4 Comparison of PSP@k (top row) and PSnDCG@k (bottom row) over tree-based methods. The reported metrics capture prediction performance over tail labels

leads to the two metrics which are sensitive to tail labels and are denoted by PSP@k, and PSnDCG@k.

Figure 4 shows the result w.r.t PSP@k, and PSnDCG@k among the tree-based approaches. Again, `Bonsai-i` shows consistent improvement over `Parabel`. For instance, on WikiLSHTC-325K, the relative improvement over `Parabel` is approximately 6.7% on PSP@5. This further validates the applicability of the shallow tree architecture resulting from the design choices of K-way partitioning along with flexibility to allow unbalanced partitioning in `Bonsai`, which allows tail labels to be assigned into different partitions w.r.t the head ones.

4.3 Unique label coverage

We also evaluate *coverage@k*, denoted $C@k$, which is the percentage of normalized unique labels present in an algorithm's top-k labels. Let $\mathbf{P} = P_1 \cup P_2 \cup \ldots \cup P_M$ where $P_i = \{l_{i1}, l_{i2}, \ldots, l_{ik}\}$ i.e the set of top-k labels predicted by the algorithm for test point i and M is the number of test points. Also, let $\mathbf{L} = L_1 \cup L_2 \cup \ldots \cup L_M$ where $L_i = \{g_{i1}, g_{i2}, \ldots, g_{ik}\}$ i.e the top-k propensity scored ground truth labels for test point i, then, coverage@k is given by

$$C@k = |\mathbf{P}|/|\mathbf{L}|$$

The comparison between `Bonsai-i` and `Parabel` of this metric on five different datasets is shown in Table 3. It shows that the proposed method is more effective in discovering

Table 3 Coverage@k ($C@k$) statistics comparing `Parabel` and `Bonsai-i`

Dataset	Methods	$C@1$	$C@3$	$C@5$
EUR-Lex	`Parabel`	**31.46**	43.11	54.38
	`Bonsai-i`	31.38	**44.09**	**55.61**
Wiki10	`Parabel`	7.00	5.77	6.76
	`Bonsai-i`	**7.52**	**6.82**	**8.01**
WikiLSHTC	`Parabel`	22.73	35.94	43.18
	`Bonsai-i`	**24.14**	**38.49**	**46.37**
Amazon-670k	`Parabel`	32.73	33.77	38.82
	`Bonsai-i`	**33.28**	**34.76**	**40.11**
Amazon-3M	`Parabel`	21.16	20.49	21.81
	`Bonsai-i`	**22.27**	**21.89**	**23.36**

Along each $C@k$ and dataset configuration, the best performing score is highlighted in bold

correct unique labels. These results further reinforce the results in the previous section on the diversity preserving feature of `Bonsai`.

4.4 Impact of tree depth

We next evaluate prediction performance produced by `Bonsai` trees with different depth values. We set the fan-out parameter K appropriately to achieve the desired tree depth. For example, to partition 4000 labels into a hierarchy of depth two, we set $K = 64$.

In Fig. 5, we report the result on three datasets, averaged over ten runs under each setting. The trend is consistent—as the tree depth increases, prediction accuracy tends to drop, though it is not very stark for Wikipedia-31K.

Furthermore, in Fig. 6, we show that the shallow architecture is an integral part of the success of the `Bonsai` family of algorithms. To demonstrate this, we plugged in the label representation used in `Bonsai-o` into `Parabel`, called `Parabel-o` in the figure. As can be seen, `Bonsai-o` outperforms `Parabel-o` by a large margin showing that shallow trees substantially alleviate the prediction error.

4.5 Training and prediction time

Growing shallower trees in `Bonsai` comes at a slight price in terms of training time. It was observed that `Bonsai` leads to approximately 2–3x increase in training time compared to `Parabel`. For instance, on a single core, `Parabel` takes 1 h for training on the WikiLSHTC-325K dataset, while `Bonsai` takes approximately 3 h for the same task. However, it may also be noted that the training process can be performed in an offline manner. Though, unlike `Parabel`, `Bonsai` does not come with logarithmic dependence on the number of labels for the computational complexity of prediction. However, its prediction time is typically in milli-seconds, and hence it remains quite practical in XMC applications with real-time constraints such as recommendation systems and advertising.

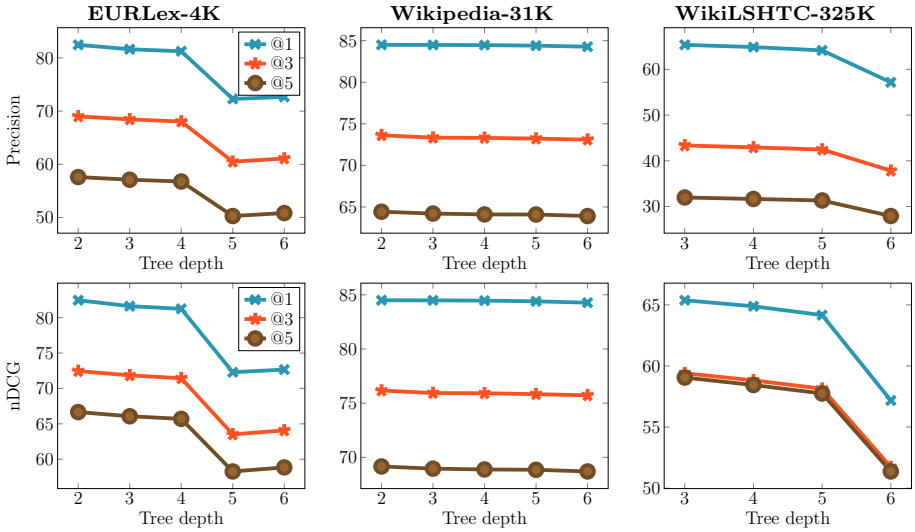

Fig. 5 Effect of tree depth: `Bonsai` trees with different depths are evaluated w.r.t *prec@k* (top row) and *nDCG@k* (bottom row). As tree depth increases, performance tends to drop

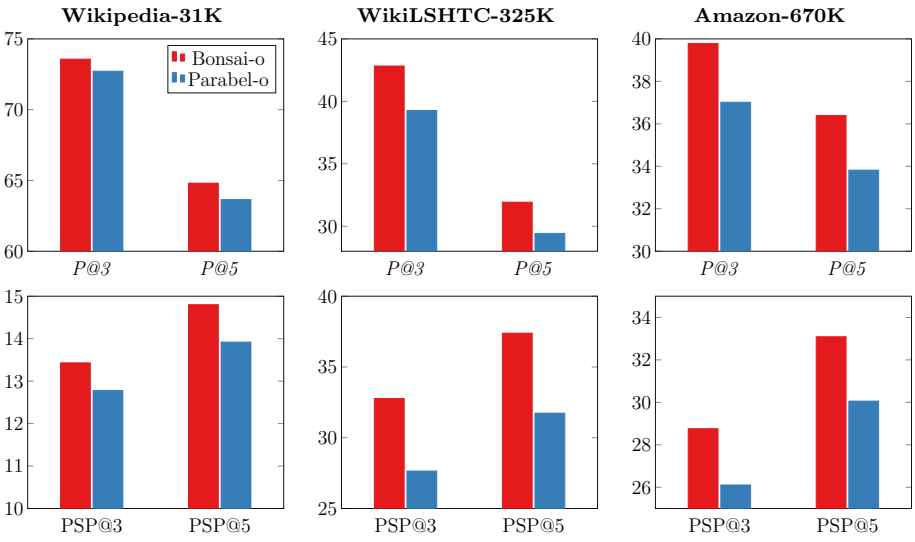

Fig. 6 Comparison of *prec@k* and PSP@k scores of Bonsai-o and Parabel-o over three benchmark datasets

5 Conclusion

In this paper, we present `Bonsai`, which is a class of algorithms for learning shallow trees for label partitioning in extreme multi-label classification. Compared to the existing tree-based methods, it improves this process in two fundamental ways. *Firstly*, it generalizes the notion of label representation beyond the input space representation, and shows the efficacy

of output space representation based on its co-occurrence with other labels, and by further combining these in a joint representation. *Secondly*, by learning shallow trees which prevent error propagation in the tree cascade and hence improving the prediction accuracy and tail-label coverage. The synergizing effects of these two ingredients enables `Bonsai` to retain the training speed comparable to tree-based methods, while achieving better prediction accuracy as well as significantly better tail-label coverage. As a future work, the generalized label representation can be further enriched by combining with embeddings from raw text. This can lead to the amalgamation of methods studied in this paper with those that are based on deep learning.

References

Agrawal, R., Gupta, A., Prabhu, Y., & Varma, M. (2013). Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *World Wide Web conference*.

Babbar, R., & Schölkopf, B. (2017). Dismec: Distributed sparse machines for extreme multi-label classification. In *International conference on web search and data mining* (pp. 721–729).

Babbar, R., & Schölkopf, B. (2019). Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, *108*(8–9), 1329–1351.

Babbar, R., Partalas, I., Gaussier, E., & Amini, M.R. (2013). On flat versus hierarchical classification in large-scale taxonomies. In *Advances in neural information processing systems* (pp. 1824–1832).

Babbar, R., Metzig, C., Partalas, I., Gaussier, E., & Amini, M.R. (2014). On power law distributions in large-scale taxonomies. In *ACM SIGKDD explorations newsletter* (pp. 47–56).

Babbar, R., Partalas, I., Gaussier, E., Amini, M. R., & Amblard, C. (2016). Learning taxonomy adaptation in large-scale classification. *The Journal of Machine Learning Research*, *17*(1), 3350–3386.

Bengio, S., Weston, J., & Grangier, D. (2010). Label embedding trees for large multi-class tasks. In *Neural information processing systems* (pp. 163–171).

Bhatia, K., Jain, H., Kar, P., Varma, M., & Jain, P. (2015). Sparse local embeddings for extreme multi-label classification. In *Neural information processing systems*.

Bhatia, K., Dahiya, K., Jain, H., Prabhu, Y., & Varma, M. (2016). *The extreme classification repository: Multi-label datasets and code*. http://manikvarma.org/downloads/XC/XMLRepository.html.

Deng, J., Berg, A.C., Li, K., & Fei-Fei, L. (2010). What does classifying more than 10,000 image categories tell us? In *European conference on computer vision*.

Denton, E., Weston, J., Paluri, M., Bourdev, L., Fergus, R. (2015). User conditional hashtag prediction for images. In *ACM SIGKDD international conference on knowledge discovery and data mining*.

Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., & Lin, C. J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, *9*(Aug), 1871–1874.

Fang, H., Cheng, M., Hsieh, C.J., & Friedlander, M. (2019) Fast training for large-scale one-versus-all linear classifiers using tree-structured initialization. In *Proceedings of the 2019 SIAM international conference on data mining, SIAM* (pp. 280–288).

Hsu, D., Kakade, S., Langford, J., & Zhang, T. (2009). Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*.

Jain, H., Prabhu, Y., & Varma, M. (2016). Extreme multi-label loss functions for recommendation, tagging, ranking and other missing label applications. In *ACM SIGKDD international conference on knowledge discovery and data mining*.

Jalan, A., & Kar, P. (2019). *Accelerating extreme classification via adaptive feature agglomeration*. arXiv preprint arXiv:190511769.

Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., & Hüllermeier, E. (2016). Extreme f-measure maximization using sparse probability estimates. In *International conference on machine learning*.

Joly, A., Wehenkel, L., & Geurts, P. (2019). *Gradient tree boosting with random output projections for multi-label classification and multi-output regression*. arXiv preprint arXiv:190507558.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th conference of the European chapter of the association for computational linguistics* (Vol. 2, pp. 427–431), Short Papers.

Khandagale, S., & Babbar, R. (2019) A simple and effective scheme for data pre-processing in extreme classification. In *27th European symposium on artificial neural networks, ESANN 2019*, Bruges, Belgium, 24–26 April 2019.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1746–1751).

Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Neural information processing systems* (pp. 1097–1105).

Liang, Y., Hsieh, C.J., & Lee, T. (2018). *Block-wise partitioning for extreme multi-label classification*. arXiv preprint arXiv:181101305.

Lin, Z., Ding, G., Hu, M., & Wang, J. (2014). Multi-label classification via feature-aware implicit label space encoding. In *International conference on machine learning* (pp. 325–333).

Liu, J., Chang, W.C., Wu, Y., & Yang, Y. (2017). Deep learning for extreme multi-label text classification. In *SIGIR, ACM* (pp. 115–124).

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137.

Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, *45*(9), 3084–3104.

Majzoubi, M., & Choromanska, A. (2019). *Ldsm: Logarithm-depth streaming multi-label decision trees*. arXiv preprint arXiv:190510428.

McAuley, J., & Leskovec, J. (2013). Hidden factors and hidden topics: Understanding rating dimensions with review text. In *RecSys, ACM* (pp. 165–172).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Neural information processing systems* (pp. 3111–3119).

Partalas, I., Kosmopoulos, A., Baskiotis, N., Artieres, T., Paliouras, G., Gaussier, E., Androutsopoulos, I., Amini, M.R., & Galinari, P, (2015). *Lshtc: A benchmark for large-scale text classification*. arXiv preprint arXiv:150308581.

Prabhu, Y., & Varma, M. (2014). Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *ACM SIGKDD international conference on knowledge discovery and data mining, ACM* (pp. 263–272).

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., & Varma, M. (2018). Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web conference on World Wide Web* (pp. 993–1002).

Read, J., Pfahringer, B., & Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. In *8th IEEE international conference on data mining, IEEE* (pp. 995–1000).

Shen, D., Ruvini, J.D., Somaiya, M., & Sundaresan, N. (2011). Item categorization in the e-commerce domain. In *Proceedings of the 20th ACM international conference on information and knowledge management, ACM* (pp. 1921–1924).

Si, S., Zhang, H., Keerthi, S.S., Mahajan, D., Dhillon, I.S., & Hsieh, C.J. (2017). Gradient boosted decision trees for high dimensional sparse output. In *International conference on machine learning*.

Tagami, Y. (2017). Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *ACM SIGKDD international conference on knowledge discovery and data mining, ACM, IEEE*.

Tai, F., & Lin, H. T. (2012). Multilabel classification with principal label space transformation. *Neural Computation*, *24*(9), 2508–2542.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, *3*(3), 1–13.

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In: *Proceedings of the ECML/PKDD 2008 workshop on mining multidimensional data (MMD08)*.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, *73*(2), 185.

Wei, T., & Li, Y.F. (2018). Does tail label help for large-scale multi-label learning. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 2847–2853), AAAI Press.

Weston, J., Bengio, S., & Usunier, N. (2011). Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*.

Wydmuch, M., Jasinska, K., Kuznetsov, M., Busa-Fekete, R., & Dembczynski, K. (2018). A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *Advances in neural information processing systems* (pp. 6355–6366).

Xu, C., Tao, D., & Xu, C. (2016a). Robust extreme multi-label learning. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*.

Xu, C., Tao, D., & Xu, C. (2016b). Robust extreme multi-label learning. In *ACM SIGKDD international conference on knowledge discovery and data mining, ACM* (pp. 1275–1284).

Yen, I.E., Huang, X., Dai, W., Ravikumar, P., Dhillon, I., & Xing, E. (2017). Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, ACM* (pp. 545–553).

Yen, I.E.H., Huang, X., Ravikumar, P., Zhong, K., & Dhillon, I. (2016). Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International conference on machine learning* (pp. 3069–3077).

Yu, H.F., Jain, P., Kar, P., & Dhillon, I. (2014) Large-scale multi-label learning with missing labels. In *International conference on machine learning* (pp. 593–601).

Zhang, M. L., Li, Y. K., Liu, X. Y., & Geng, X. (2018). Binary relevance for multi-label learning: An overview. *Frontiers of Computer Science*, *12*(2), 191–202.