Data Mining and Knowledge Discovery (2023) 37:2192–2215
https://doi.org/10.1007/s10618-023-00945-5

Fast block-wise partitioning for extreme multi-label classification

Yuefeng Liang[1] · **Cho-Jui Hsieh**[2] · **Thomas C. M. Lee**[1]

Received: 22 June 2022 / Accepted: 6 April 2023 / Published online: 26 July 2023

Abstract

Extreme multi-label classification aims to learn a classifier that annotates an instance with a relevant subset of labels from an extremely large label set. Many existing solutions embed the label matrix to a low-dimensional linear subspace, or examine the relevance of a test instance to every label via a linear scan. In practice, however, those approaches can be computationally exorbitant. To alleviate this drawback, we propose a Block-wise Partitioning (BP) pretreatment that divides all instances into disjoint clusters, to each of which the most frequently tagged label subset is attached. One multi-label classifier is trained on one pair of instance and label clusters, and the label set of a test instance is predicted by first delivering it to the most appropriate instance cluster. Experiments on benchmark multi-label data sets reveal that BP pretreatment significantly reduces prediction time, and retains almost the same level of prediction accuracy.

Keywords Extreme multi-label problems · Non-convex optimization · Scalable classification · Supervised learning

Responsible editor: Dragi Kocev.

✉ Thomas C. M. Lee
tcmlee@ucdavis.edu

Yuefeng Liang
frnliang@ucdavis.edu

Cho-Jui Hsieh
chohsieh@cs.ucla.edu

[1] Department of Statistics, University of California at Davis, Davis, CA 95616, USA

[2] Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90095, USA

1 Introduction

Advances in computing technology enable the collection, maintenance, and analysis of extremely large data sets. Domains such as video annotation (Snoek et al. 2006), text classification (Deng et al. 2009; Partalas et al. 2015), automated tag suggestion (Wetzker et al. 2008) and keyword suggestion (Chang et al. 2020) generate data sets whose numbers of labels are growing magnificently. Given the abundance of features and labels, supervised learning with hundreds of thousands of labels has attracted the attention of machine learning researchers and practitioners in recent years. Automatically assigning a relevant subset of labels from an extremely large label set to an unseen instance defines the goal in extreme multi-label classification (Prabhu and Varma 2014; Bhatia et al. 2015).

The rapid augmentation of labels leads to numerous computational challenges. In this paper, we focus on one critical issue that limits the performance of existing methods in real applications: prediction time. Assume there are m labels, most of the existing methods require $O(m)$ prediction time for each test instance (Yu et al. 2014; Babbar and Schölkopf 2017; Yen et al. 2016, 2017). For example, the classical one-vs-all approach transforms multi-label problems into multiple binary classification problems. While this approach delivers competitive prediction accuracy on many data sets (Babbar and Schölkopf 2017), it becomes prohibited when m is extremely large (Niculescu-Mizil and Abbasnejad 2017).

To overcome this limitation, we exploit label popularity among all instances in large-scale data sets. It turns out that in many cases, some labels are popular among all instances, while some are frequently tagged only with certain subgroups. Given an association between a label subset and an instance subgroup, prediction performed only from the associated label subset will almost be as accurate as prediction performed from all m labels (Nasierding et al. 2009; Niculescu-Mizil and Abbasnejad 2017). This observation leads to our assumption in this paper that the feature and label spaces are so discernible that one can partition instances and labels into different clusters. This partitioning scheme allows us to construct a one-to-one correspondence between each pair of instances and label clusters so that we do not have to visit all m labels for every test instance.

Under our assumption, we show that from feature and label matrices, instance and label clusters can be determined by an alternating update procedure imposed on a discrete optimization problem. If we rearrange labels by label cluster and instances by instance cluster, the permuted label matrix approximates a diagonal block structure, with a majority of ones inside the blocks and zeros elsewhere. Moreover, when an L_2 penalty on the lengths of label clusters is imposed, the number of labels assigned to each cluster is effectively monitored by the regularization parameter.

Rather than propose a classifier that competes with the existing algorithms, we introduce a Block-wise Partitioning (BP) pretreatment on feature and label matrices to help existing multi-label classifiers achieve faster prediction while retaining prediction accuracy. As for training, one classifier is trained on one pair of instance and label clusters. As for prediction, a test instance is first classified

into a proper instance cluster, and then the corresponding classifier is applied on the paired label cluster. We discuss how Precision (P), propensity scored Precision (PSP) (Bhatia et al. 2016) and prediction time are impacted by BP pretreatment on large-scale data sets from the Extreme Classification Repository (Bhatia et al. 2016). For example, on the Wiki10-31K (Zubiaga 2012) data set, while PD-Sparse (Yen et al. 2016) achieves 81.89% at P@1, BP pretreatment accelerates its prediction by 209 times at the expense of 0.33% loss on P@1.

The rest of the paper is organized as follows. We review related work in Sect. 2 and introduce our partitioning algorithm in Sect. 3. Experimental results are presented and discussed in Sect. 4, followed by concluding remarks in Sect. 5.

2 Related work

Most state-of-the-art methods for extreme multi-label classification are based on trees (Jasinska et al. 2016; Khandagale et al. 2019; Prabhu and Varma 2014; Prabhu et al. 2018; Siblini et al. 2018; Wydmuch et al. 2018), embeddings (Bhatia et al. 2015; Evron et al. 2018; Gupta et al. 2019; Jalan and Kar 2019; Tagami 2017; Yu et al. 2014), one-versus-all (Babbar and Schölkopf 2017, 2019; Jain et al. 2019; Khandagale et al. 2020; Prabhu et al. 2018; Yen et al. 2016, 2017) and deep learning (Chang et al. 2020; Dahiya et al. 2021b, a; Jiang et al. 2021; Liu et al. 2017; Mittal et al. 2021, 2022; Qaraei et al. 2021; You et al. 2018). The first branch learns a hierarchical structure of the full label set by recursively dividing feature or label spaces. For instance, FastXML (Prabhu and Varma 2014) searches for a sparse linear separator to split each node by optimizing an nDCG (Bhatia et al. 2016) based loss function. The second branch reduces the effective number of labels under the low-rank assumption. For example, LEML (Yu et al. 2014) directly optimizes for the decompression matrices using a regularized least square objective function in a generic empirical risk minimization framework. PD-Sparse (Yen et al. 2016), DiSMEC (Babbar and Schölkopf 2017) and ProXML (Babbar and Schölkopf 2019) are three methods in the third trend that attract considerable attention in the literature. PD-Sparse makes use of both primal and dual sparsity by margin-maximizing loss with L_1 and L_2 penalties. DiSMEC revisits one-versus-all paradigm and provides prominent boosts in prediction accuracy and prediction time by explicitly inducing sparsity and doubly parallel training. While many approaches treat labels with equal importance, ProXML (Babbar and Schölkopf 2019) improves performance for tail labels that are infrequently occurring (Wei et al. 2021). XML-CNN (Liu et al. 2017), AttentionXML (You et al. 2018) and LightXML Jiang et al. (2021) are the very three representatives in the last trend.

Tree-based approaches are well known for their prediction accuracy (Prabhu and Varma 2014), and embedding-based approaches are also popular as they are straightforward and can handle label correlations (Yu et al. 2014; Bhatia et al. 2015). However, implementation on large data sets has been a subject of extensive debate, and one of the most common remarks is on prediction time (Si et al. 2017; Niculescu-Mizil and Abbasnejad 2017). Over the past few years, various methods have been proposed to ameliorate the burden. For instance, PPD-Sparse (Yen et al.

2017) adapts the parallelizability and small memory footprint of the one-versus-all technique from DiSMEC (Babbar and Schölkopf 2017), and the sub-linear complexity of primal-dual sparse method from PD-Sparse (Yen et al. 2016).

Deep learning approaches have been successful in delivering state-of-the-art results in extreme classification benchmarks, as they borrow strength from different machine learning domains and learn informative text representation (Chang et al. 2021; Jiang et al. 2021; You et al. 2018). XML-CNN (Liu et al. 2017) is one of the first methods that bring deep learning into extreme classification. It learns word embeddings directly from raw text and feeds it to one-dimensional convolutional neural networks (CNN). To embrace the power of neural embedding-based models, AttentionXML (You et al. 2018) uses Bidirectional Long Short-Term Memory Networks (BiLSTMs) and probabilistic label trees (PTL) to handle raw text, but its promising performance comes at the cost of long training time and big model size (Chang et al. 2021). Various methods have been proposed to address these commonly shared challenges in deep extreme classification. Notably, LightXML Jiang et al. (2021) aims to lower computational complexity by adopting transformer models as text encoders, and performing label shortlist and label re-ranking (Chang et al. 2021).

In contrast to the above methods, we develop a pretreatment for those algorithms. Many classical sequential, agglomerative, hierarchical and non-overlapping (SAHN) (Day and Edelsbrunner 1984) clustering methods can be used as baseline pretreatments, and the following three approaches are more relevant to ours. Label Partitioning for Sub-linear Ranking (LPSR) follows a two-step approach. At the training stage, it clusters the training instances, and then assigns a fixed number of potential labels to each cluster. At the testing stage, a test instance is first put in one of the clusters, and its labels are predicted only from the labels attached to that cluster (Weston et al. 2013). While LPSR and our proposal stand on the same testing stage, three differences can be observed at the training stage. First, the optimized number of clusters in our method is selected by the algorithm, whereas the number in LPSR is predefined. Second, the cardinality of our label clusters may vary across clusters, while the number of potential labels in each cluster in LPSR is fixed. Moreover, our instance and label clusters are updated via optimization, whereas the assignment of training instances in LPSR solely depends on the feature matrix. Consequently, our method permits more flexibility in cluster structure.

The second method, Clustering Based Multi-Label Classification (CBMLC), groups the training instances into a user-specified number of clusters, and trains a multi-label classification model for each cluster (Nasierding et al. 2009). The testing stage is identical to that of LPSR. There are several differences between CBMLC and our approach. First, our goal is to minimize the prediction time while CBMLC focuses on reducing the training time. Second, we formally form a clustering objective to achieve better prediction time using both features and labels, and expedite prediction by reducing label size, whereas CBMLC takes features into consideration only in the clustering step. Third, our number of clusters is not user-specified.

Another related approach is to pre-select a small fraction of candidate labels via label filters (LF) (Niculescu-Mizil and Abbasnejad 2017) before the base classifier is applied. As the number of filters increases, a larger proportion of labels will be

filtered out, and thus prediction time decreases without a significant impact on prediction performance. Although LF shares the same goal as we do – to speed up existing classifiers at prediction time by reducing the number of candidate labels, infrequent, unnoticeable but valuable tail labels will be filtered out by LF; those labels can be captured and stay in one of our label clusters, given that they are closely related to a group of training instances. What is more, LF tends to select the same candidate labels for all instances, while our method assigns customized label subsets to different instance clusters. As a result, our method reflects more natural characteristics of features and labels without discarding rare but rewarding signals.

3 Proposed algorithm

This section defines the learning problem we consider. We first introduce definitions and notations, and then explore the optimization algorithm in detail.

3.1 Problem formulation

To formulate our partitioning problem, we adopt the following notations: lower-case letters such as x and y denote elements of sets; upper-case letters such as C and L denote maps; bold lower-case letters such as $\mathbf{x}$ and $\mathbf{y}$ denote vectors; bold upper-case letters such as $\mathbf{X}$ and $\mathbf{Y}$ denote matrices; calligraphic letters such as $\mathcal{X}$ and $\mathcal{L}$ denote sets.

Let $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \in \mathbb{R}^d$, $i = 1, \cdots, n$, be a d-dimensional input feature vector that forms an instance, and $\mathbf{y}_i = (y_{i1}, y_{i2}, \ldots, y_{im}) \in \{0, 1\}^m$, $i = 1, \cdots, n$, be the corresponding m-dimensional label vector. $y_{ij} = 1$ if the ith data point is tagged with the jth label. Let $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ be the training data set. $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ is the $n \times d$ feature matrix, and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top$ is the $n \times m$ label matrix. We are interested in simultaneously partitioning instances and labels into q ($1 \leq q \leq \min(m, n)$) instance and label clusters. Write the q instance and label clusters as $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_q\}$ and $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_q\}$ respectively, where $\mathcal{X}_l \subseteq \{1, \ldots, n\}$ and $\mathcal{L}_l \subseteq \{1, \ldots, m\}$ for $l = 1, \ldots, q$. We define a one-to-one deterministic mapping L from instance clusters to label clusters such that

$$L(\mathcal{X}_l) = \mathcal{L}_l \quad \text{for all } l.$$

Therefore, our goal is to learn a map

$$C : \mathbb{R}^d \longrightarrow \{\mathcal{X}_1^*, \mathcal{X}_2^*, \ldots, \mathcal{X}_q^*\}$$

that classifies instances, and an optimal set of label clusters $\{\mathcal{L}_1^*, \mathcal{L}_2^*, \ldots, \mathcal{L}_q^*\}$.

The partitioned label matrix approximates a block diagonal matrix upon row and column permutations. The diagonal blocks are matrices of any size with a majority of ones, and the off-diagonal elements are mostly zeros. Instance clusters are deemed to be disjoint, and overlapping is allowed between any two label clusters, which means there can be overlap between $\mathcal{L}_l^*$ and $\mathcal{L}_{l'}^*$ for any (l, l') pair.

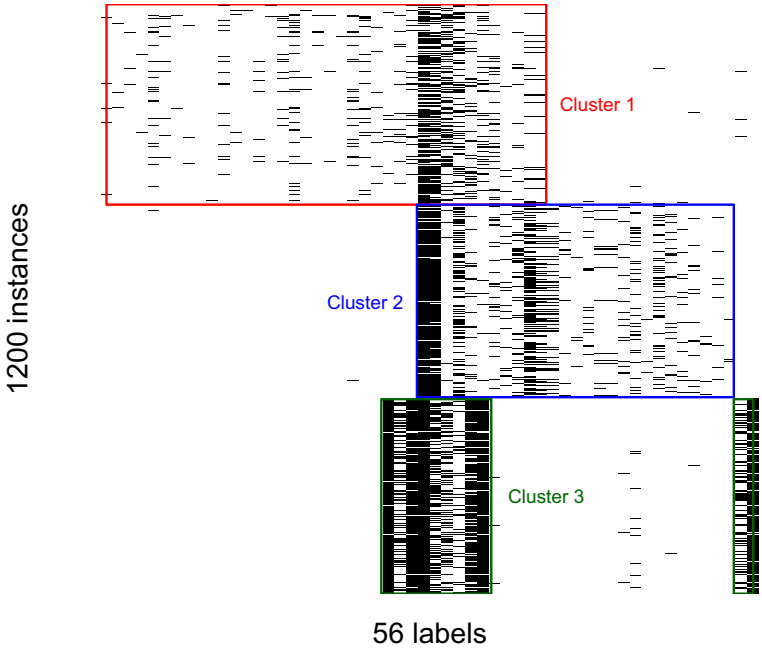

Fig. 1 Truncated and permuted training label matrix of Mediamill. Black pixels are tagged labels. Clusters are marked by bounding rectangles in different colors

Allowing overlapping label clusters is crucial for achieving good performance on extreme classification problems. As we stated in Sect. 1, some popular labels are often assigned to many samples, and our clustering approach is able to capture this information by assigning those popular labels to many clusters. Figure 1 illustrates the approximate block diagonal structure on small-scale data set Mediamill (Snoek et al. 2006). For illustration purposes, the number of clusters q is set to 3, and only 56 out of $m = 101$ labels and the first 400 train instances from each cluster are displayed. The vertical axis represents instances, and the horizontal axis represents labels. It can be visualized that a small set of labels located at the horizontal center are included in all label clusters. If these popular labels were assigned to only one label cluster, the other two would fail to capture the strong signals and thus suffer from an undesired loss in predictive power.

After we find the optimal partitions, we learn a classifier from each pair of $\mathcal{X}_l$ and $\mathcal{Y}_l$. Finally, given a test instance $\hat{\mathbf{x}}$, we first classify it into one of $\{\mathcal{X}_1^*, \mathcal{X}_2^*, \ldots, \mathcal{X}_q^*\}$ by conducting a multinomial logistic regression on the feature space, and then predict its most relevant labels $\hat{\mathbf{y}}$ from its label cluster.

3.2 Block-wise partitioning (BP)

As we mentioned in Sect. 3.1, our goal is to learn a function C and a set of label clusters $\{\mathcal{L}_1^*, \mathcal{L}_2^*, \ldots, \mathcal{L}_q^*\}$ for any given q. Let $\Omega_q = (C, \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_q\})$. We wish to find the optimal Ω_q^* which minimizes the objective function

$$f(\Omega_q) = -\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{1}[j \in (L \circ C)(\mathbf{x}_i), y_{ij} = 1] + \lambda \sum_{l=1}^{q} |\mathcal{L}_l|^2, \tag{1}$$

where λ is a regularization parameter. The intuition behind $f(\Omega_q)$ is to search for an equilibrium between the number of ones captured by the diagonal blocks and the total size of label clusters. We first observe that this optimization problem is a significant challenge, as the objective function is neither convex nor differentiable. Furthermore, our goal is to perform optimization for data sets where d, m, and n are substantially large. Nevertheless, when either one of the two sets of clusters is fixed, the objective function can be minimized. To this end, we divide the optimization problem into two phases, and implement alternating minimization.

To start with, we specify q, the initial number of paired clusters, and set $t = 0$. The selection criteria of q will be discussed in Sect. 3.3. Given a chosen q, we initialize C by applying the sparse k-means clustering on the entire training feature matrix. In other words, we initialize $\{\mathcal{X}_1^{(0)}, \mathcal{X}_2^{(0)}, \ldots, \mathcal{X}_q^{(0)}\}$ by features.

3.2.1 Label clusters selection

From $t = 1$, we fix $\{\mathcal{X}_1^{(t-1)}, \mathcal{X}_2^{(t-1)}, \ldots, \mathcal{X}_q^{(t-1)}\}$ and update $\{\mathcal{L}_1^{(t)}, \mathcal{L}_2^{(t)}, \ldots, \mathcal{L}_q^{(t)}\}$. This is a discrete optimization problem, and a naive implementation will need to check the objective function (1) for every possible set of labels. Instead, we find the optimal set can be efficiently computed using the following procedure.

Given an instance cluster $\mathcal{X}_l$, we first calculate column sums in the label matrix for all m labels, and then sort the m column sums in descending order. By the definition of the objective function, we know if one label is included in $\mathcal{L}_l$, then all the labels with larger column sum should also be included in $\mathcal{L}_l$ (otherwise exchanging two labels will decrease objective function). Denote J as the number of labels in $\mathcal{L}_l$. Thus we add the sorted m labels one by one to the label set $\mathcal{L}_l$, and each time check the objective function

$$-\sum_{i \in \mathcal{X}_l}\sum_{j=1}^{J} \mathbf{1}[j \in (L \circ C)(\mathbf{x}_i)] \cdot \mathbf{1}[y_{ij} = 1] + \lambda J^2 \tag{2}$$

until adding one additional label will increase (2). It is obvious that further increasing J will lead to a larger objective function value, thus assigning the first J labels to $\mathcal{L}_l$ will be optimal.

In the following paragraph, we show that our algorithm can find the optimal label clusters when fixing instance clusters. For any instance cluster, in sequel, we denote C_i as the column sum of label matrix block for the ith label, $i = 1, \ldots, m$.

Proposition 1 *Let $\{C_i\}_{i=1}^m$ be a non-increasing sequence of non-negative integers. Denote $a_n = -\sum_{i=1}^n C_i + \lambda n^2$ where $\lambda > 0$. The sequence of real numbers $\{a_n\}_{n=1}^m$ is strictly convex.*

Proof To prove the strict convexity of $\{a_n\}_{n=1}^m$, it suffices to show that for $2 \le n \le m - 1$,

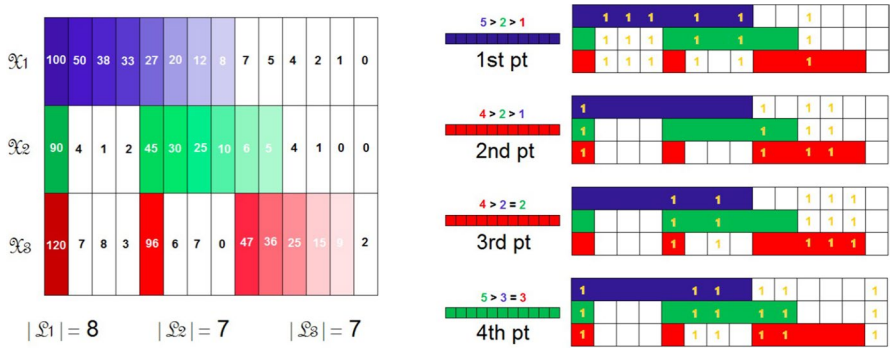

Fig. 2 An example of label clusters selection (left) and instance clusters selection (right). For label clusters selection, lighter colors represent lower column sums. Best viewed in color

$$a_{n-1} + a_{n+1} > 2a_n.$$

Since $a_n = -\sum_{i=1}^{n} C_i + \lambda n^2$, we have

$$\begin{aligned} a_{n-1} + a_{n+1} - 2a_n &= -\sum_{i=1}^{n-1} C_i + \lambda(n-1)^2 - \sum_{i=1}^{n+1} C_i + \lambda(n+1)^2 + \sum_{i=1}^{n} 2C_i - 2\lambda n^2 \\ &= -C_n - C_{n+1} + 2\lambda n^2 + 2\lambda + 2C_n - 2\lambda n^2 \\ &= C_n - C_{n+1} + 2\lambda > 0 \end{aligned}$$

as $C_n \geq C_{n+1}$ and $\lambda > 0$. This completes the proof.

□

Since convexity guarantees that (2) can be minimized, it implies Algorithm 1 finds the optimal label sets. Figure 2 (left) visualizes Algorithm 1 via an example with $q = 3$ and $m = 14$. Different colors distinguish different clusters. The top-left number 100, for instance, is the column sum of the first column, i.e., label 1, in the training label matrix for all samples in $\mathcal{X}_1$. As the color turns lighter from left to right, the column sums decrease, and the algorithm selects the first eight labels in the corresponding label cluster $\mathcal{L}_1$.

In this procedure, computing the count of each label in each instance cluster costs $O(\text{nnz}(\mathbf{Y}))$ time in total, where $\text{nnz}(\mathbf{Y})$ denotes the number of nonzero elements in $\mathbf{Y}$. Sorting for all the clusters costs $O(qm \log(m))$ time. Once the label counts are computed and sorted, every step of checking (2) only costs $O(1)$ time, so the overall time complexity for this step is $O(\text{nnz}(\mathbf{Y}) + qm \log(m))$.

3.2.2 Instance clusters selection

Now we fix $\{\mathcal{L}_l^{(t)}\}_{l=1}^{q}$ and update $\{\mathcal{X}_1^{(t)}, \mathcal{X}_2^{(t)}, \ldots, \mathcal{X}_q^{(t)}\}$. We take the following steps for every instance. We first assign an instance to every one of the q instance clusters, and then count the number of attached labels that belong to the corresponding label cluster.

If the Ith assignment yields the greatest count, then we put the instance in $\mathcal{X}_I$. Algorithm 2 optimally solves the subproblem resulting from (1) because the summation of maxima in each independent component is indeed the maximum. Figure 2 (right) envisions Algorithm 2 for the first four sample points under the same framework as Fig. 2 (left). The 1's in yellow are the ground truth labels in the training label matrix. For the first sample, there are five 1's in blue, two in green, and one in red. At this step, we fix label clusters and update instance clusters. If we place the first sample into $\mathcal{X}_1$, then five ground truth labels will be captured in $\mathcal{L}_1$. The same idea follows for green and red. Since placing it into $\mathcal{X}_1$ yields the highest number of ground truth labels to be captured, we put the first sample into $\mathcal{X}_1$, and mark it in blue.

In this procedure, computing the count of each instance in each label cluster costs $O(\text{nnz}(\mathbf{Y}))$ time in total, and comparing the counts of q assignments costs $O(q)$ time for each instance, so the overall time complexity for this step is $O(\text{nnz}(\mathbf{Y}) + nq)$.

We iterate the above two selection procedures until the objective reaches a local minimum. Once the difference between $f(\Omega_q^{(t-1)})$ and $f(\Omega_q^{(t)})$ is less than 10^{-5}, we stop at time t, and report $\{\mathcal{X}_1^{(t)}, \mathcal{X}_2^{(t)}, \ldots, \mathcal{X}_q^{(t)}\}$ and $\{\mathcal{L}_1^{(t)}, \mathcal{L}_2^{(t)}, \ldots, \mathcal{L}_q^{(t)}\}$ as the optimal set of instance and label clusters, respectively.

As the time complexities of label clusters selection and instance clusters selection are $O(\text{nnz}(\mathbf{Y}) + qm\log(m))$ and $O(\text{nnz}(\mathbf{Y}) + nq)$ respectively, the total time complexity of this alternating minimization procedure is $O(\text{nnz}(\mathbf{Y}) + qm\log(m) + nq)$.

Algorithm 1 Label Clusters Selection

Input: $\{\mathcal{X}_l^{(t-1)}\}_{l=1}^q$
Output: $\{\mathcal{L}_l^{(t)}\}_{l=1}^q$
for $l = 1$ **to** q **do**
  $P_{lj}^{(t-1)} = \sum_{i \in \mathcal{X}_l^{(t-1)}} \mathbf{1}[y_{ij} = 1]$ for all j
  Sort $\{P_{lj'}^{(t-1)}\}_{j'=1}^m$ in descending order
  Find $J^* = \arg\min_J - \sum_{j'=1}^J P_{lj'}^{(t-1)} + \lambda J^2$ using a linear scan
  Assign $\{1, \ldots, J^*\}$ to $\mathcal{L}_l^{(t)}$
end for

Algorithm 2 Instance Clusters Selection

Input: $\{\mathcal{L}_l^{(t)}\}_{l=1}^q$
Output: $\{\mathcal{X}_l^{(t)}\}_{l=1}^q$
for $i = 1$ **to** n **do**
  $Q_{il}^{(t)} = \sum_{j \in \mathcal{L}_l^{(t)}} \mathbf{1}[y_{ij} = 1]$ for all l
  Find $I = \arg\max_l Q_{il}^{(t)}$
  Assign i to $\mathcal{X}_I^{(t)}$
end for

3.3 Optimal parameters

The regularization parameter λ monitors the tradeoff between prediction speed and prediction accuracy. The optimal λ depends on one's preference for prediction efficiency and tolerance of accuracy loss. Conditional on the tolerance level, we are able to apply cross-validation to select the optimal λ. For each validation set, we record an interval of λ's that keeps the loss of prediction accuracy within the tolerance level. The optimal range of λ's is taken as the intersection of all intervals. In real data experiments, we conduct 5-fold cross-validation, and set the tolerance level to 2%.

Note that our algorithm assumes that the numbers of label clusters and instance clusters are equal to q, but in practice, the "true" number of paired clusters is unknown. Nevertheless, our algorithm automatically searches for the optimal q within a search space. On the one hand, if the diagonal blocks in the training label matrix capture as many ones as possible, then important labels are likely to stay in at least one of the label clusters, and thus predictive power is unaffected. On the other hand, if q is too large, some of the paired clusters may be empty. Hence, there exists a unique q such that the maximal proportion of ones in the full training label matrix is attained, and all q paired clusters are non-empty. For instance, in the large-scale data set AmazonCat-13K (McAuley and Leskovec 2013), when $\lambda = 1$, as q increases from 2 to 5, the proportion of 1's being captured in clusters increases from 80 to 84%. If we take $q = 6$, one pair of instance and label clusters shrinks to null. Therefore, our algorithm selects $q = 5$ for AmazonCat-13K.

3.4 Prediction

To predict labels of test instances, we allocate all instances into proper instance clusters based on their features. This task is a supervised learning problem, which is handled by the linear classification package LIBLINEAR (Fan et al. 2008). We use L2-regularized linear logistic regression in LIBLINEAR, with the default one-versus-all approach for multi-class classification. The training is efficient since LIBLINEAR only takes $O(\text{nnz}(\mathbf{X})q)$ time per epoch. The final step of the whole process is to apply the trained classifiers on the assigned testing instance cluster and corresponding label cluster. The prediction only requires q inner products, and q is much smaller than the original label size. Hence, instead of performing a linear scan on all m labels, we can just scan at most $q \cdot \max_l |\mathcal{L}_l^*|$ labels in all q clusters. To demonstrate the efficiency of LIBLINEAR, we report the train and prediction time, together with the number of instance clusters in all data sets on which we experiment in Sect. 4.8.

4 Experiments

We evaluate the performance of BP pretreatment on combinations of data sets and classifiers with respect to prediction accuracy and prediction time. All experiments are conducted on a machine with an Intel Xeon E5-2690 2.90GHz CPU and 256GB RAM.

Table 1 Data set statistics for extreme multi-label classification problems. ASpL and ALpS represent the average sample per label and average labels per sample respectively

Data	# Training	# Testing	# Features	# Labels	ASpL	ALpS
AmazonCat-13K	1,186,239	306,782	203,882	13,330	448.57	5.04
Wiki10-31K	14,146	6,616	101,938	30,938	8.52	18.64
Delicious-200K	196,606	100,095	782,585	205,443	72.29	75.54
WikiLSHTC-325K	1,778,351	587,084	1,617,899	325,056	17.46	3.19
Amazon-670K	490,449	153,025	135,909	670,091	3.99	5.45

4.1 Data

We conduct experiments on five large-scale data sets from the Extreme Classification Repository. Table 1 shows the associated details. We use the same train-test splits as the Repository.

subsectionClassifiers

We apply BP on one embedding and three one-versus-all based classifiers, and then compare BP with four existing pretreatment methods. We also compare BP-enabled classifiers with deep learning classifiers in prediction time.

- LEML (Yu et al. 2014) is a low-rank Empirical-Risk-Minimization solver. We implement BP with this method because its prediction time complexity is linear to m.
- PD-Sparse (Yen et al. 2016) uses L_1 regularization along with multi-class loss. Since the intermediary weight vectors need to be stored in size linear to m, we implement BP with this algorithm.
- DiSMEC (Babbar and Schölkopf 2017) is a distributed framework based on one-versus-all linear classifiers with explicit model size controlled by pruning small weights.
- ProXML (Babbar and Schölkopf 2019) exploits the label-wise independence of Hamming loss among labels and manages to detect tail labels.
- LPSR-NB (Weston et al. 2013) learns a hierarchy of labels as well as a classifier for the entire label set. Since the computational cost for each task is high, training Naive Bayes (NB) classifier as the base classifier is the only possible strategy for large-scale data sets. Results of this approach are taken from Table 1(a) in the SLEEC paper (Bhatia et al. 2015), Table 2 in the DiSMEC paper (Babbar and Schölkopf 2017) and the Repository.
- CBMLC (Nasierding et al. 2009) comprises a clustering algorithm (either k-means (Jain and Dubes 1988) or EM (Dempster et al. 1977)) and a multi-label classification algorithm. Since none of the four originally proposed multi-label classifier candidates can fully scale to extreme datasets without modification, we instantiate the clustering component using the k-means algorithm and implement the classification component using LEML, PD-Sparse and DiSMEC.
- LF (Niculescu-Mizil and Abbasnejad 2017) chooses candidate labels by projecting every test instance on a filtering line, and keeps only the labels that

Table 2 P@k, PSP@k and theoretical prediction speedup for algorithms with BP pretreatment. Values with superscript † are obtained from a subset of test samples as the built-in testing procedure of PD-Sparse does not scale to that data set

Data	Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	Speedup
AmazonCat-13K (ALpS=5.04)	LEML	**88.80**	70.65	54.22	45.62	52.69	53.41	1×
	BP LEML (accuracy)	87.03	**71.50**	**54.43**	**48.74**	**55.23**	**54.85**	37×
	BP LEML (speed)	86.87	69.94	52.26	47.72	53.60	52.32	58×
	PD-Sparse	**89.04**	**73.46**	**59.37**	**49.58**	**61.63**	**68.23**	1×
	BP PD-Sparse (accuracy)	87.38	71.74	57.40	48.16	60.44	67.19	8×
	BP PD-Sparse (speed)	87.38	71.74	57.40	48.16	60.44	67.19	8×
	DiSMEC	**93.38**	**79.07**	**64.09**	**59.08**	**67.10**	**71.19**	1×
	BP DiSMEC (accuracy)	92.13	77.57	62.49	53.58	63.30	67.66	5×
	BP DiSMEC (speed)	92.11	77.56	62.27	53.50	62.91	66.89	6×
	ProXML	89.27	**74.52**	**60.06**	65.95	70.05	70.68	1×
	BP ProXML (accuracy)	**89.52**	74.51	59.01	**73.37**	**78.27**	**80.93**	17×
	BP ProXML (speed)	89.21	74.08	58.21	73.01	77.14	78.74	31×
Wiki10-31K (ALpS = 18.64)	LEML	73.10	62.13	**54.06**	**9.43**	**10.07**	**10.53**	1×
	BP LEML (accuracy)	**75.57**	**62.51**	53.73	9.14	9.48	9.66	212×
	BP LEML (speed)	74.85	61.96	52.52	9.02	9.35	9.35	274×
	PD-Sparse	**81.89**	**65.34**	**53.74**	**12.48**	**11.89**	**12.67**	1×
	BP PD-Sparse (accuracy)	81.56	65.02	53.04	11.83	11.28	12.10	209×
	BP PD-Sparse (speed)	80.89	64.30	52.62	11.19	10.56	11.42	289×
	DiSMEC	**85.20**	**74.90**	**65.90**	**13.55**	**13.08**	**13.81**	1×
	BP DiSMEC (accuracy)	84.08	74.22	64.95	10.42	12.07	12.99	40×
	BP DiSMEC (speed)	84.08	73.95	64.30	10.41	11.90	12.54	90×
	ProXML	85.25	**76.53**	**67.33**	17.17	16.07	**16.38**	1×
	BP ProXML (accuracy)	**85.29**	76.50	66.79	**17.31**	**16.10**	16.28	71×
	BP ProXML (speed)	85.02	75.95	65.89	16.54	15.46	15.50	125×

Table 2 (continued)

Data	Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	Speedup
Delicious-200K (ALpS = 75.54)	LEML	40.01	36.99	35.07	**6.06**	**7.24**	**8.10**	1×
	BP LEML (accuracy)	**40.52**	**37.40**	**35.47**	5.97	7.10	7.93	390×
	BP LEML (speed)	39.67	37.35	35.44	5.95	6.89	7.64	1467×
	PD-Sparse	**41.97**	**35.28**	**32.87**	**5.29**	**5.80**	**6.24**	1×
	BP PD-Sparse (accuracy)	41.47	34.79	31.94	5.14	5.60	6.05	1756×
	BP PD-Sparse (speed)	40.94	34.33	31.22	5.09	5.95	6.15	2104×
	DiSMEC	**45.53**	**38.67**	**35.52**	6.52	**7.61**	**8.39**	1×
	BP DiSMEC (accuracy)	44.27	38.27	35.23	**6.66**	7.42	8.03	436×
	BP DiSMEC (speed)	43.63	37.92	35.09	6.53	7.32	7.96	2075×
WikiLSHTC-325K (ALpS = 3.19)	LEML	19.82	11.43	8.39	3.48	3.79	4.27	1×
	BP LEML (accuracy)	**29.68**	**17.86**	**13.20**	**7.30**	**8.10**	**8.97**	84×
	BP LEML (speed)	21.99	11.29	7.52	4.36	3.97	3.82	2559×
	PD-Sparse	**60.98**	**38.12**	**26.53**	**28.34**	**33.50**	**36.62**	1×
	BP PD-Sparse (accuracy)	59.26	37.48	26.14	27.72	32.44	35.48	3×
	BP PD-Sparse (speed)	59.01	37.33	26.03	27.63	32.11	35.39	4×
	DiSMEC	**64.14**	**42.45**	**31.52**	**29.10**	**35.60**	**39.50**	1×
	BP DiSMEC (accuracy)	63.54	42.38	30.97	27.45	33.68	37.70	10×
	BP DiSMEC (speed)	62.88	40.87	29.53	26.65	33.67	37.16	18×
	ProXML	**63.60**	**41.50**	**30.80**	**34.80**	**37.70**	**41.00**	1×
	BP ProXML (accuracy)	63.03	41.41	30.28	33.82	35.79	39.24	10×
	BP ProXML (speed)	62.43	39.88	29.76	33.03	35.75	38.71	18×

Table 2 (continued)

Data	Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	Speedup
Amazon-670K (ALpS = 5.45)	LEML	8.13	6.83	6.02	2.07	2.26	2.47	1×
	BP LEML (accuracy)	**14.21**	**10.66**	**8.60**	**5.87**	**5.47**	**5.29**	387×
	BP LEML (speed)	7.87	5.29	4.12	3.00	2.51	2.34	2780×
	PD-Sparse	24.41†	21.54†	**19.71**†	18.83†	15.77†	**14.54**†	1×
	BP PD-Sparse (accuracy)	**26.80**	**22.28**	18.82	**20.09**	**16.80**	12.29	28×
	BP PD-Sparse (speed)	23.90	19.27	17.83	17.87	14.34	11.78	42×
	DiSMEC	**44.71**	**39.65**	**36.09**	**27.80**	**30.60**	**34.20**	1×
	BP DiSMEC (accuracy)	43.78	38.88	35.44	24.87	28.36	32.25	11×
	BP DiSMEC (speed)	42.91	37.89	34.16	24.11	28.14	31.99	22×
	ProXML	**43.50**	**38.70**	**35.30**	**30.80**	**32.80**	**35.10**	1×
	BP ProXML (accuracy)	42.55	37.90	34.74	27.90	30.57	33.14	11×
	BP ProXML (speed)	41.72	36.94	33.42	27.15	30.36	32.88	22×

Bold values indicate the best result among different pretreatment methods under the same classifier

have training instances in the neighborhood of this projection. The results of this approach are taken from the LF paper.

- Agglomerative Hierarchical Cluster (AHC) tree (Day and Edelsbrunner 1984) initiates an individual cluster for each instance, and pairs of clusters are merged as one moves up the hierarchy. We employ AHC in feature clustering and apply LEML, PD-Sparse and DiSMEC in label scanning.
- AttentionXML (You et al. 2018) is a label tree-based deep learning model that utilizes label correlations to generate label partitions and shortlist candidate labels. Since BP is mostly applicable to sparse tf-idf data representation other than dense raw text data, we do not implement BP on AttentionXML, and the results of this deep learning approach are adopted from the AttentionXML paper.

4.2 Parameter setting

For LEML, we set the rank to 500 and the number of iterations to 5 for an early stop.[1] All other parameters are set as default. For PD-Sparse, DiSMEC and ProXML, we use the default parameters provided by the authors.

4.3 Evaluation metrics

Precision@k (P@k) counts the fraction of correct predictions in the top k scoring labels in $\hat{\mathbf{y}}$ (this is a standard evaluation on Extreme Repository) (Hsu et al. 2009; Agrawal et al. 2013). To examine the performance on tail labels, we also present propensity scored Precision@k (PSP@k), as propensity score helps in making metrics unbiased (Jain et al. 2016). We report two versions of precision scores based on two variants of BP. Accuracy BP aims at the highest P@k in the search space of λ. Speed BP aims at the highest speedup at the expense of at most 2% P@k loss. To evaluate improvement in prediction efficiency, we consider theoretical speedup, a measure with regard to number of vector multiplications (Niculescu-Mizil and Abbasnejad 2017). We also use elapsed time to measure actual speedup.

4.4 Prediction accuracy on classifiers

We implement BP with LEML, DP-Sparse, DiSMEC and ProXML. Table 2 demonstrates that BP speeds up prediction under all scenarios with less than 2% loss in P@k and less than 6% loss in PSP@k. Results of BP ProXML on Delicious-200K are not reported, as both (Panos et al. 2021) and our experiments show that ProXML does not scale to Delicious-200K. For the embedding method, accuracy BP LEML produces better P@k than the original LEML for all data sets, and even better PSP@k in some cases. By partitioning the original label space into different subspaces, BP effectively reinforces low-rankness, a fundamental assumption that LEML must satisfy to achieve good performance. For one-versus-all methods, although the original PD-Sparse does not scale to challenging data sets, BP frees it

[1] These choices are adopted from the Extreme Classification Repository.

from the size constraint. P@k of DiSMEC and ProXML experiences a small drop when in most cases. For classifiers that require a linear scan over the whole label space, skipping a subset of labels slightly impacts predictive performance.

4.5 Role of λ

Figure 3 reveals that λ manages to control the trade-off between P@k and speedup. Even though a smaller λ allows more labels in label clusters, minimizing λ does not necessarily yield the best prediction performance.

4.6 Theoretical and actual speedup

The theoretical speedup column in Table 2 is measured by the number of vector multiplications. BP speeds up prediction up to 2780 times under all scenarios. A negative correlation is observed between speedup and average labels per sample (ALpS) on PD-Sparse, DiSMEC and ProXML. Note that ALpS in AmazonCat-13K, WikiLSHTC-325K and Amazon-670K is below 6, and every sample in Delicious-200K has 76 positive labels on average. Due to the nature of the polished one-versus-all approach, choosing the top 5 labels from 76 is much easier than choosing the top 5 from 6. As the label size in each cluster should have been substantially large to avoid unwanted elimination, speedup may be affected. Nevertheless, even though the speedup is less in low ALpS data sets, we still speed up PD-Sparse, DiSMEC and ProXML for more than 10 times on the largest Amazon-670K. We also report actual prediction time per test instance (in milliseconds) on AmazonCat-13K, Wiki10-31K and Delicious-200K for speed BP in Table 3. Overall, the actual speedup is similar to and bounded by the theoretical speedup.

4.7 BP processing time

Table 4 shows that BP does not have much overhead in the pre-processing step. Although our goal is to speed up prediction, the training time (data partition time + BP training time) is also shortened. Note that all clusters can be trained and tested in a parallel manner. Parallelization can make extreme classifiers' training and testing faster than the single-core implementation.

4.8 Efficiency of LIBLINEAR

To visualize the efficiency of the LIBLINEAR package, we report the training and prediction time, together with the number of instance clusters q in all data sets for LEML and DiSMEC in Table 5. These numbers are compatible with the training time complexity $O(\text{nnz}(\mathbf{X})q)$ time per epoch and the prediction time complexity $O(q \cdot \max_l |\mathcal{L}_l^*|)$ time per epoch mentioned in Sect. 3.4.

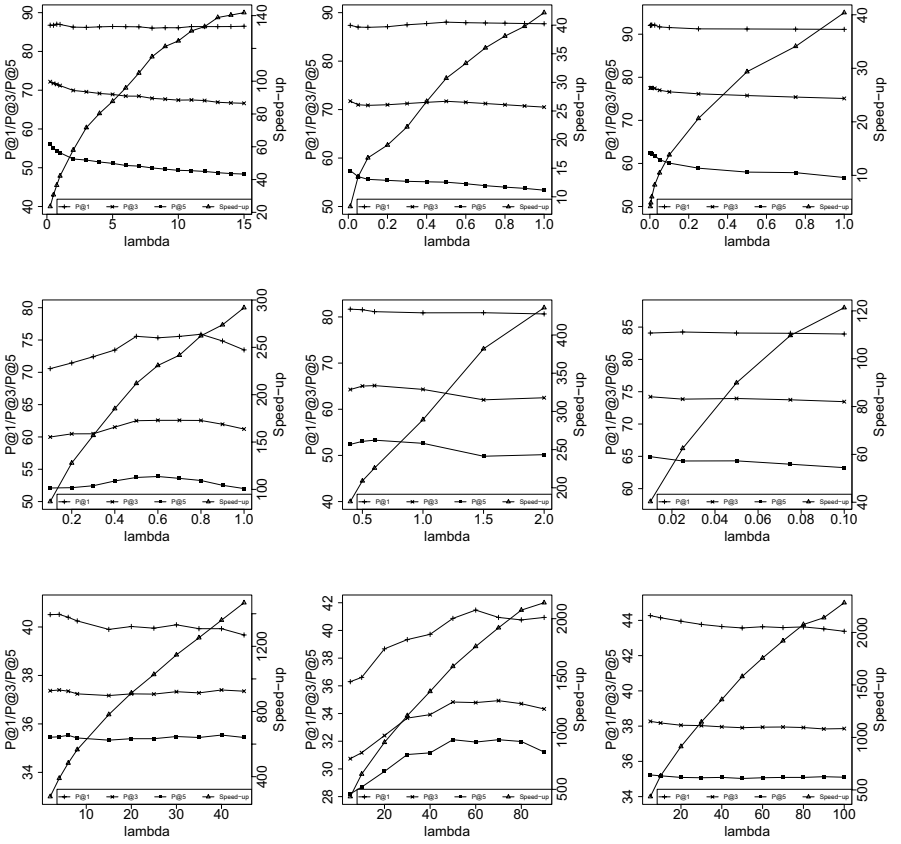

Fig. 3 Precision (y1 axis) and speedup (y2 axis) as a function of λ. From top to bottom: AmazonCat-13K, Wiki10-31K, and Delicious-200K. From left to right: LEML, PD-Sparse, and DiSMEC. Presented λ may be out of the optimal range

Table 3 Actual prediction time (in milliseconds) per test instance for speed BP

Method	Data	Original Time	BP Time	Actual Speedup	Theoretical Speedup
LEML	AmazonCat-13K	5.76	0.11	52×	58×
	Wiki10-31K	34.1	0.15	227×	274×
	Delicious-200K	0.128	0.0001	1280×	1467×
PD-Sparse	AmazonCat-13K	0.62	0.078	8×	8×
	Wiki10-31K	3.38	0.03	113×	289×
	Delicious-200K	5.27	0.01	527×	2104×
DiSMEC	AmazonCat-13K	0.2	0.038	5×	6×
	Wiki10-31K	0.12	0.0015	80×	90×
	Delicious-200K	307.3	0.154	1995×	2075×

Table 4 Elapsed time (in seconds) for LEML and DiSMEC with speed BP pretreatment

Method	Procedure	A.-13K	W.-31K	D.-200K	W.-325K	A.-670K
BP LEML	Data partitioning	2102	107	1675	7841	3049
BP LEML	Training	17626	1239	4581	8800	5753
LEML	Original training	28200	4448	18599	43473	9802
BP DiSMEC	Data partitioning	220	6	118	89613	35735
BP DiSMEC	Training	6587	541	4134	95243	49348
DiSMEC	Original training	12459	2339	39439	300655	193842

Table 5 LIBLINEAR train and prediction time (in seconds) for LEML and DiSMEC with BP pretreatment

Data	Method	Train	Prediction
AmazonCat-13K	BP LEML	573	3
($q = 5$)	BP DiSMEC	405	12
Wiki10-31K	BP LEML	21	1
($q = 3$)	BP DiSMEC	8	1
Delicious-200K	BP LEML	1233	9
($q = 10$)	BP DiSMEC	257	10
WikiLSHTC-325K	BP LEML	8447	8
($q = 1575$)	BP DiSMEC	2487	27
Amazon-670K	BP LEML	3375	5
($q = 2000$)	BP DiSMEC	937	14

4.9 Comparison with other pretreatment methods

Among the four pretreatment methods to be compared with BP, AHC takes the simplest approach: it only partitions the instance space based on features and leaves label space unchanged. Its clustering step requires $O(n^2 \log n)$ time in the worst scenario (Day and Edelsbrunner 1984), but no extra cost is involved before any label classification step. CBMLC performs both feature clustering and label assignment (Nasierding et al. 2009). Its standard k-means clustering on feature space requires $O(ndq)$ time per epoch, but the computational cost for its default label assignment solutions is extremely high in the extreme classification setting. Therefore, the first round of comparison is among AHC, CBMLC and accuracy BP, where we focus on the effectiveness of feature clustering and aim to achieve the highest prediction accuracy for all three methods. Table 6 indicates that CBMLC and AHC perform better than accuracy BP in terms of P@k with LEML. Although BP helps with low-rankness as we observe in Sect. 4.4, keeping the original set of labels is still the best way to preserve the quality of label embedding in LEML's encoder-decoder architecture. For one-versus-all methods such as PD-Sparse and DiSMEC, nevertheless, CBMLC and AHC perform worse than accuracy BP. AHC and CBMLC in the extreme classification scenarios tend to overlook the relationship between labels and features. Doing a linear scan

Table 6 P@k and PSP@k for algorithms with accuracy BP, CBMLC and AHC pretreatments

Data	Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
AmazonCat-13K (ALpS=5.04)	BP LEML (accuracy)	87.03	71.50	54.43	**48.74**	55.23	54.85
	CBMLC LEML	89.03	**72.38**	**56.18**	46.91	**55.34**	**54.92**
	AHC LEML	**89.12**	72.33	55.87	47.10	55.21	54.84
	BP PD-Sparse (accuracy)	87.38	**71.74**	**57.40**	**48.16**	**60.44**	**67.19**
	CBMLC PD-Sparse	**87.70**	71.34	57.09	47.33	59.51	66.40
	AHC PD-Sparse	87.54	71.12	56.96	47.30	58.96	66.01
	BP DiSMEC (accuracy)	**92.13**	**77.57**	**62.49**	**53.58**	**63.30**	**67.66**
	CBMLC DiSMEC	92.06	77.53	62.19	53.31	62.40	66.55
	AHC DiSMEC	92.01	77.25	61.88	53.26	62.37	66.00
Wiki10-31K (ALpS = 18.64)	BP LEML (accuracy)	**75.57**	62.51	53.73	**9.14**	9.48	9.66
	CBMLC LEML	74.43	63.32	**55.00**	8.97	10.65	**10.93**
	AHC LEML	74.83	**63.34**	54.93	9.02	**10.87**	10.64
	BP PD-Sparse (accuracy)	**81.56**	**65.02**	**53.04**	**11.83**	**11.28**	**12.10**
	CBMLC PD-Sparse	71.16	56.51	46.28	9.85	9.20	9.98
	AHC PD-Sparse	78.52	59.99	48.38	10.92	10.04	10.71
	BP DiSMEC (accuracy)	**84.08**	**74.22**	**64.95**	**10.42**	**12.07**	**12.99**
	CBMLC DiSMEC	84.01	73.83	64.22	10.39	11.79	12.38
	AHC DiSMEC	84.04	74.01	64.45	10.40	11.98	12.61
Delicious-200K (ALpS = 75.54)	BP LEML (accuracy)	40.52	37.40	35.47	5.97	7.10	7.93
	CBMLC LEML	42.12	38.05	**35.98**	6.84	7.93	**8.88**
	AHC LEML	**42.15**	**38.07**	35.58	**6.87**	**7.97**	8.54
	BP PD-Sparse (accuracy)	**41.47**	**34.79**	**31.94**	**5.14**	**5.60**	**6.05**
	CBMLC PD-Sparse	30.07	25.27	23.06	4.02	4.16	4.89
	AHC PD-Sparse	38.83	27.44	24.52	4.79	4.75	5.20
	BP DiSMEC (accuracy)	**44.27**	**38.27**	**35.23**	**6.66**	**7.42**	**8.03**
	CBMLC DiSMEC	43.99	37.83	35.00	6.59	7.17	7.72
	AHC DiSMEC	44.09	37.97	35.14	6.61	7.19	7.81

Bold values indicate the best result among different pretreatment methods under the same classifier

over the whole label space without emphasis on the most relevant set of labels introduces distraction from infrequent and unrepresentative signals.

The second round of comparisons is among LPSR, LF and BP. All three methods follow the two-step procedure to update instance and label clusters, but LPSR's computational cost is so exorbitant that NB remains the only possible base classifier in extreme classification (Babbar and Schölkopf 2017). LF is similar to BP as they both conduct alternating minimization to find the optimal sets of unknown parameters with a similar level of computation complexity. LF requires $O(nd + nq + n\log n + m\log m)$ time per update (Niculescu-Mizil and Abbasnejad 2017), and BP requires $O(\text{nnz}(\mathbf{Y}) + nq + qm\log(m))$ time as

Table 7 P@k and theoretical prediction speedup for accuracy BP NB, conservative LF NB and LPSR-NB

Data	Method	P@1	P@5	Speedup
Wiki10-31K (ALpS = 18.64)	NB	80.00	57.55	1x
	Accuracy BP NB	**80.89**	56.98	**38x**
	Conservative LF NB	80.62	**57.90**	34x
	LPSR-NB	72.71	49.50	–
Delicious-200K (ALpS = 75.54)	NB	**45.33**	**38.08**	1x
	Accuracy BP NB	45.16	37.29	**558x**
	Conservative LF NB	45.25	37.98	428x
	LPSR-NB	18.59	14.07	–

Bold values indicate the best result among different pretreatment methods under the same classifier

mentioned in Sect. 3.2. As LF is shown to work well mostly with NB and SVM (Crammer and Singer 2001) in (Niculescu-Mizil and Abbasnejad 2017), we apply all three methods to NB. To aim for the highest prediction accuracy, we adopt the idea of conservative LF that attempts to preserve performance other than speedup from (Niculescu-Mizil and Abbasnejad 2017). Table 7 shows that BP and LF significantly outperform LPSR, and BP has slightly worse accuracy but better theoretical speedup than LF.

4.10 Performance on tail labels

The PSP@k columns in Table 2 show that BP boosts embedding-based LEML's performance on many tail labels, but it introduces 1% to 6% drop to one-versus-all methods in many cases. The PSP@k columns in Table 6 reveal that BP outperforms CBMLC and AHC in one-versus-all scenarios, but gives mixed outcomes to LEML. Allowing more labels to be captured in label clusters prevents more tail labels from elimination, but this comes at the cost of longer prediction time.

4.11 Computation time comparison with deep learning methods

Despite long training time and big model size (Chang et al. 2021), deep learning methods in general deliver the most accurate prediction outcomes (Bhatia et al. 2016). For example, on AmazonCat-13K, while the best representative of one-versus-all methods, DiSMEC, gives 93.38 for P@1, deep learning methods AttentionXML and LightXML bring 95.92 and 96.77 respectively (Bhatia et al. 2016). Conditional on the availability of computation resources, deep learning methods outperform the other three trends when it comes to time-insensitive tasks. However, it is still worth investigating the tradeoff between accuracy and efficiency when timely iterative scoring is at risk and computation power is a constraint.

While it takes AttentionXML 1.63 milliseconds to test one sample on AmazonCat-13K with 8 Nvidia GTX 1080Ti GPUs (You et al. 2018), it takes

BP DiSMEC 0.038 millisecond to do the same job without GPU according to Table 3. We thus face a tradeoff between 43 times faster in prediction and 3.79% (95.92 − 92.13) drop in accuracy based on Table 2. On Wiki10-31K, it takes AttentionXML 4.53 milliseconds (You et al. 2018) and BP DiSMEC 0.0015 millisecond to finish the same task. It now comes to the debate of 3020 times faster in prediction versus 3.39% (87.47 − 84.08) drop in accuracy. Besides prediction time, according to Table 3, it takes BP DiSMEC 1.89 hours and AttentionXML 13.11 hours (You et al. 2018) to train their corresponding model on AmazonCat-13K. LightXML is proven to be faster than AttentionXML in training and testing in some use cases Jiang et al. (2021), but BP-empowered methods can still achieve faster computation by taking some cost in accuracy.

5 Concluding remarks

We apply BP pretreatment on extreme multi-label classifiers whose test time complexity is linear to the size of the label set. The non-convex and discrete optimization problem launched with BP is solved by an intuitive and straightforward alternating minimization procedure. Partitioning on the instance and label matrices embraces the strength of existing algorithms by a significant order of magnitude speedup together with a similar level of prediction accuracy.

Author contributions All authors contributed to the development of the proposed method and the writing of the manuscript. YL carried out most of the numerical experiments. All authors reviewed the manuscript.

Funding Liang and Lee were supported by the National Science Foundation under Grants CCF-1934568, DMS-1916125 and DMS-2113605. Hsieh was supported by the National Science Foundation under Grants CCF-1934568, IIS-1901527 and IIS-2008173.

Data availability All data are publicly available with references provided in the paper.

Code availability The code can be obtained from the authors. It will also be made publicly available in github once the paper is accepted for publication.

Declarations

Conflict of interest The authors do not have any conflicts of interest/competing interests to declare.

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: Proceedings of the 22nd international conference on World Wide Web, ACM, pp 13–24

Babbar R, Schölkopf B (2017) Dismec: distributed sparse machines for extreme multi-label classification. In: Proceedings of the tenth ACM international conference on web search and data mining, ACM, pp 721–729

Babbar R, Schölkopf B (2019) Data scarcity, robustness and extreme multi-label classification. Mach Learn, 1–23

Bhatia K, Dahiya K, Jain H, Kar P, Mittal A, Prabhu Y, Varma M (2016) The extreme classification repository: multi-label datasets and code. URL http://manikvarma.org/downloads/XC/XMLRepository.html

Bhatia K, Jain H, Kar P, Varma M, Jain P (2015) Sparse local embeddings for extreme multi-label classification. Adv Neural Inf Process Syst, 730–738

Chang W-C, Jiang D, Yu H-F, Teo CH, Zhang J, Zhong K, Kolluri K, Hu Q, Shandilya N, Ievgrafov V et al (2021) Extreme multi-label learning for semantic matching in product search. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2643–2651

Chang W-C, Yu H-F, Zhong K, Yang Y, Dhillon IS (2020) Taming pretrained transformers for extreme multi-label text classification. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp 3163–3171

Crammer K, Singer Y (2001) On the algorithmic implementation of multiclass kernel-based vector machines. J Mach Learn Res 2:265–292

Dahiya K, Agarwal A, Saini D, Gururaj K, Jiao J, Singh A, Agarwal S, Kar P, Varma M (2021a) Siamesexml: siamese networks meet extreme classifiers with 100m labels. In: International conference on machine learning, PMLR, pp 2330–2340

Dahiya K, Saini D, Mittal A, Shaw A, Dave K, Soni A, Jain H, Agarwal S, Varma M (2021b) Deepxml: A deep extreme multi-label learning framework applied to short text documents. In: Proceedings of the 14th ACM international conference on web search and data mining, pp 31–39

Day WH, Edelsbrunner H (1984) Efficient algorithms for agglomerative hierarchical clustering methods. J Classif 1:7–24

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the em algorithm. J R Stat Soc Ser B (Methodological) 39:1–22

Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: Computer vision and pattern recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, pp 248–255

Evron I, Moroshko E, Crammer K (2018) Efficient loss-based decoding on graphs for extreme classification. Adv Neural Inf Process Syst, 31

Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J (2008) Liblinear: a library for large linear classification. J Mach Learn Resarch 9:1871–1874

Gupta V, Wadbude R, Natarajan N, Karnick H, Jain P, Rai P (2019) Distributional semantics meets multi-label learning. Proc AAAI Conf Artif Intell 33:3747–3754

Hsu DJ, Kakade SM, Langford J, Zhang T (2009) Multi-label prediction via compressed sensing. In: Advances in neural information processing systems, pp 772–780

Jain AK, Dubes RC (1988) Algorithms for clustering data. Prentice-Hall Inc

Jain H, Balasubramanian V, Chunduri B, Varma M (2019) Slice: scalable linear extreme classifiers trained on 100 million labels for related searches. In: Proceedings of the twelfth ACM international conference on web search and data mining, pp 528–536

Jain H, Prabhu Y, Varma M (2016) Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 935–944

Jalan A, Kar P (2019) Accelerating extreme classification via adaptive feature agglomeration. In: Proceedings of the 28th international joint conference on artificial intelligence, pp 2600–2606

Jasinska K, Dembczynski K, Busa-Fekete R, Pfannschmidt K, Klerx T, Hullermeier E (2016) Extreme f-measure maximization using sparse probability estimates. In: International conference on machine learning, pp 1435–1444

Jiang T, Wang D, Sun L, Yang H, Zhao Z, Zhuang F (2021) Lightxml: transformer with dynamic negative sampling for high-performance extreme multi-label text classification. In: Proceedings of the AAAI conference on artificial intelligence, vol 35, pp 7987–7994

Khandagale S, Xiao H, Babbar R (2019) Bonsai-diverse and shallow trees for extreme multi-label classification. arXiv preprint arXiv:1904.08249

Khandagale S, Xiao H, Babbar R (2020) Bonsai: diverse and shallow trees for extreme multi-label classification. Mach Learn 109:2099–2119

Liu J, Chang W-C, Wu Y, Yang Y (2017) Deep learning for extreme multi-label text classification. In Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 115–124

McAuley J, Leskovec J (2013) Hidden factors and hidden topics: understanding rating dimensions with review text. In Proceedings of the 7th ACM conference on recommender systems, ACM, pp 165–172

Mittal A, Dahiya K, Agrawal S, Saini D, Agarwal S, Kar P, Varma M (2021) Decaf: deep extreme classification with label features. In Proceedings of the 14th ACM international conference on web search and data mining, pp 49–57

Mittal A, Dahiya K, Malani S, Ramaswamy J, Kuruvilla S, Ajmera J, Chang K-h, Agarwal S, Kar P, Varma M (2022) Multi-modal extreme classification. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 12393–12402

Nasierding G, Tsoumakas G, Kouzani AZ (2009) Clustering based multi-label classification for image annotation and retrieval. In: 2009 IEEE international conference on systems, man and cybernetics SMC , IEEE, pp 4514–4519

Niculescu-Mizil A, Abbasnejad E (2017) Label filters for large scale multilabel classification. In: Artificial intelligence and statistics, pp 1448–1457

Panos A, Dellaportas P, Titsias MK (2021) Large scale multi-label learning using gaussian processes. Mach Learn 110:965–987

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:1503.08581

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, International world wide web conferences steering committee, pp 993–1002

Prabhu Y, Varma M (2014) Fastxml: a fast, accurate and stable tree-classifier for extreme multi-label learning. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 263–272

Qaraei M, Schultheis E, Gupta P, Babbar R (2021) Convex surrogates for unbiased loss functions in extreme classification with missing labels. In: Proceedings of the web conference, vol 2021, pp 3711–3720

Si S, Zhang H, Keerthi SS, Mahajan D, Dhillon IS, Hsieh C-J (2017) Gradient boosted decision trees for high dimensional sparse output. In: International conference on machine learning, pp 3182–3190

Siblini W, Kuntz P, Meyer F (2018) Craftml, an efficient clustering-based random forest for extreme multi-label learning

Snoek CG, Worring M, Van Gemert JC, Geusebroek J-M, Smeulders AW (2006) The challenge problem for automated detection of 101 semantic concepts in multimedia. In: Proceedings of the 14th ACM international conference on multimedia, ACM, pp 421–430

Tagami Y (2017) 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, ACM, pp 455–464

Wei T, Tu W-W, Li Y-F, Yang G-P (2021) Towards robust prediction on tail labels. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, pp 1812–1820

Weston J, Makadia A, Yee H (2013) Label partitioning for sublinear ranking. In: International conference on machine learning, pp 181–189

Wetzker R, Zimmermann C, Bauckhage C (2008) Analyzing social bookmarking systems: a del. icio. us cookbook. In: Proceedings of the ECAI 2008 mining social data workshop, pp 26–30

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

Yen IE, 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

You R, Dai S, Zhang Z, Mamitsuka H, Zhu S (2018) Attentionxml: extreme multi-label text classification with multi-label attention based recurrent neural networks. arXiv preprint arXiv:1811.01727

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

Zubiaga A (2012) Enhancing navigation on wikipedia with social tags. arXiv preprint arXiv:1202.5469