PParabel: Parallel Partitioned Label Trees for Extreme Classification

Jiaqi Lu[1], Jun Zheng[2], and Wenxin Hu[2](✉)

[1] School of Computer Science and Software Engineering, East China Normal University, 3663 Zhong Shan Rd. N., Shanghai, China
[2] The Computer Center, East China Normal University, 3663 Zhong Shan Rd. N., Shanghai, China
wxhu@cc.ecnu.edu.cn

Abstract. Extreme classification consists of extreme multi-class or multi-label predictions, whose objective is to learn classifiers that can label each data point with the most relevant labels. Recently, some approaches such as 1-vs-all method have been proposed to accomplish the task. However, their training time is linear with the number of classes, which makes them unrealistic in real-world applications such as text and image tagging. In this work, we are motivated to present a two-stage thread-level parallelism which is based on Partitioned Label Trees for Extreme Classification (Parabel). Our method is able to train the tree nodes in different parallel ways according to their number of labels. We compare our algorithm with recent state-of-the-art approach on some publicly available real-world datasets which have up to 670,000 labels. The experimental results demonstrate that our algorithm achieves the shortest training time.

Keywords: Extreme multi-label classification · Thread-level parallelism · OpenMP

1 Introduction

Extreme classification was coined by John Langford[1] and Manik Varma[2] in 2013. It is the emerging research field in machine learning which solves classification problems in presence of a large number of categories (which are also called classes or labels) [8]. And the number of these categories is often more than 10^5. To be specific, extreme classification consists of extreme multi-class (only one label is correct) or multi-label predictions (more than one label is relevant to the given item).

In this work, we focus on extreme multi-label classification where the label set has dimensionality of the order of hundreds of thousands or even millions,

[1] http://hunch.net/~jl/.
[2] http://manikvarma.org/.

X. Tang et al. (Eds.): NPC 2019, LNCS 11783, pp. 82–92, 2019.
https://doi.org/10.1007/978-3-030-30709-7_7

because this task has been of more and more significance in real-world applications such as text tagging. The goal in extreme multi-label classification is to learn a classifier which can annotate a new instance with relevant labels from the extremely large label set. Take web tagging as an example, the pages in Wikipedia are all tagged with several relevant labels. Extreme multi-label classification can be used to learn a classifier to automatically label new pages by training on the existing pages. Furthermore, extreme multi-label classification can effectively address machine learning problems in web-scale data mining, such as recommendation systems and ad landing pages' queries [1,10,11]. Due to its capability for dealing with web-scale data, extreme multi-label classification has attracted more and more attention in recent years.

The popular approaches to extreme multi-label classification can be divided into two categories, namely 1-vs-all approaches [2,9,12,13] and tree-based approaches [5,6,10,11]. 1-vs-all approaches train a classifier for each label and they usually take months to train on large datasets on a standard desktop [11]. It is intolerable since extreme multi-label classification has been applied in real-world applications such as recommendation systems and ad landing pages' queries which are required to quickly predict the labels of items and give users an immediate answer. To overcome this, DiSMEC [2] and PPDSparse [12] take advantage of distributed systems and partition the training jobs on several computing nodes. Although it is effective, the cost of hardware is heavy. Taking dataset, WikiLSHTC-325K[3], as an example, it has 1,778,351 training instances and 325,056 categories. On this dataset, DiSMEC needs 3 h train on 1000 cores. While for PPDSparse, it takes much shorter training time (i.e. 353 s on 100 cores). If we reduce the hardware cost and train on a single core, tree-based approaches can train much faster. For example, PfastreXML [5] only needs 7.42 h on a single core relative to 749 h for DiSMEC. However, tree-based approaches have not been parallel to accelerate the training process. So is Parabel which is the fastest 1-vs-all approach built with tree structure [11]. To overcome this, we analyze the data independence between nodes and propose PParabel method to accelerate the training process which is the fastest method on one core [11].

Our contributions are shown as follows:

- We analyze the hierarchy of Parabel and find that each label only exists in one node on the same level which means nodes on the same level have data independence. With data independence, we can make the training process parallel at each level.
- We parallelize the training process in two stages. In the first stage, we parallelize the training process of nodes on the same level. In the second stage, we parallelize the k-means in nodes with OpenMP according to the number of labels in nodes.
- We conduct our training process in a thread-level parallelism way and apply OpenMP to accelerate our training. We can enable PParabel work on standard desktops to minimize hardware costs.

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

- We shorten the training time from one day to just one hour without appending more machines.

The rest of the paper is organized as follows. Section 2 introduces the existing approaches to extreme multi-label classification. Section 3 describes the detail of our proposed PParabel method. Section 4 reports our experiments and we will analyze the results. At the end of this paper, we conclude our work and indicate future directions.

2 Related Work

The existing approaches to solving the extreme multi-label classification task can be divided into four categories, namely 1-vs-all approaches [2,9,12,13], label-embedding approaches [3,4,14], tree-based approaches [5,6,10,11] and deep learning based approaches [7].

1-vs-all approaches: 1-vs-all approaches train a separate classifier per label on the whole dataset. This kind of approach leads to training time linear in the number of classes. Therefore, when it comes to big datasets, the training costs can be heavy [11]. But this kind of method ignores the relevance between labels which makes each label independent and easy to be parallelized. DiSMEC [2] and PPDSparse [12] took advantage of the irrelevance between labels and scaled the training progress in large-scale distributed settings. Despite it can make the method easy to be parallelized, the cost of hardware is heavy.

Label-embedding approaches: Label-embedding approaches make the assumption that label matrix is low-rank. Therefore, it can be projected into low dimensional space. In this way, effective number of labels can be reduced. However, since the training points follow a power-law distribution, it will lead to low accuracy [2]. Moreover, embedding approaches need long time for training and prediction even on small embedding dimensions, let alone large datasets.

To overcome these limitations, SLEEC [3] was proposed. It learned local embedding instead of global embedding. To be specific, it used kNN method to preserve nearest neighbors in the label space. However, SLEEC ignores to model the label structure. [15] proposed a deep embedding method for extreme multi-label classification to overcome this. The deep embedding method uses label graph to depict the label structure. In the label graph, an edge exists if the two labels are active at the same sample. With the label graph established, DeepWalk method is used to make word2vec representation for all nodes in the graph. Then the distance between features and labels can be computed and all the training points can be clustered.

Tree-based approaches: Tree-based approaches usually have two types: decision trees and label trees. FastXML [10] is a state-of-the-art classifier for extreme multi-label classification. It recursively partitioned a parent's feature space between its children. To learn the hierarchy, FastXML optimizes the normalized Discounted Cumulative Gain (nDCG).

Another popular tree-based approach is PfastreXML [5]. The algorithm replaces the nDCG loss with its propensity scored variant. It also assigns higher rewards for accurate tail label predictions. In this way, it can improve tail label prediction which is the most challenging factor of extreme multi-label classification.

Unlike the above two methods, Parabel [11] learned a few balanced label hierarchies. The root node of each hierarchy contains the whole set of labels. Each label tree recursively partitions the nodes into two balanced nodes until the number of labels in the leaf nodes is smaller than a threshold. When it comes to leaf nodes, a classifier will be learned for each label. It can be conducted on a single core with the shortest training time while matching its prediction accuracy with other methods. Although Parabel learned balanced label trees in a parallel way, it didn't optimize the node partition process in a parallel way to save time.

Deep learning based approaches: Deep learning based approach is a new way for extreme multi-label classification. Although it has achieved great success in other areas, it has not been applied to extreme classification until 2017. The first attempt is XML-CNN [7]. It utilizes the CNN model to learn a rich number of feature representations. Unlike the traditional CNN model, XML-CNN adopts a dynamic max pooling scheme to get more than one feature. Therefore, it can capture more fine-grained features.

Nowadays, the most popular approach is 1-vs-all approach since we can make log-time training and prediction. But how can we reduce the hardware cost and accelerate the training process is still a problem. Therefore, we are motivated to propose our solution in the next section.

3 Methodology: Parallel Partitioned Label Trees (PParabel)

Our method is designed to accelerate the node partition process parallel. There are two main components in our method, label trees and idle threads. Label trees are used for training the model. Internal nodes in label trees are processed parallel in idle threads. Figure 1 shows the main structure of PParabel. The proposed method is described in the algorithmic format in Algorithm 1. Detailed information is shown as follows. As we all know, every node in the tree which is learnt by Parabel is partitioned into two groups, and not a single label can

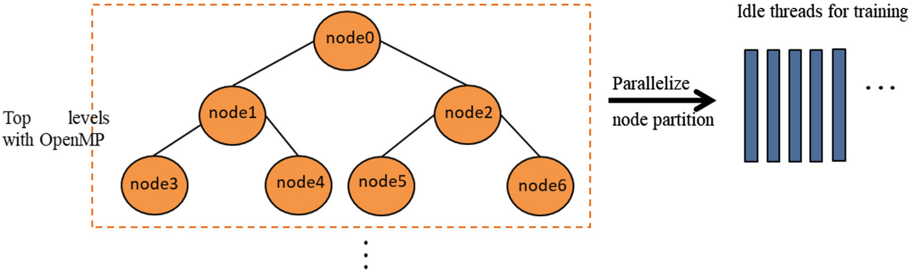

Fig. 1. The main structure of PParabel.

be appeared in two groups. In other words, the split of different nodes is independent. Therefore, we can use its independence to make Parabel parallel. Each node is carried on a single thread.

For each label tree, we load feature matrix $X = x_1, x_2, \ldots, x_n$ and label matrix $Y = y_1, y_2, \ldots, y_n$. Then we represent the label representation in the way Parabel did. We average the feature vectors of instances which are positive to the label as the representation of label. With all labels represented, we put all label representations in the root node and start partitioning. We parallelize the partition process in a two-stage way which will be discussed in Sects. 3.1 and 3.2. For each node partition, we apply k-means to split the node into two nodes. We first randomly choose two label representations as centroids. After that, we calculate the distance between centroids and labels. For nodes on top levels, we parallelize the calculating distance with OpenMP. If it is not converged, we calculate the new centroids and repeat the k-means clustering. When the k-means clustering is converged, we split the labels into two nodes according to the resulting clusters. All these child nodes will be sent to idle threads to further split. The partition process will not stop until the number of labels in any leaf nodes is smaller than a threshold.

We also implement a two-stage parallelization which executes different strategies according to different nodes on different levels. In the following, we will elaborate more details about the two-stage parallelization. The first stage parallelization is applied to all nodes. The second is applied to nodes on top levels.

Algorithm 1. PParabel - Parallel Partitioned Label Trees for Extreme Classification

Input: Feature matrix **X**, label matrix **Y**

Output: Balanced tree with the number of leaf nodes' labels smaller than a threshold

```
Load single copy of feature matrix $\mathbf{X}=x_1,x_2,\ldots,x_n$ and label matrix $\mathbf{Y}=y_1,y_2,\ldots,y_n$
Compute the label representation by averaging the feature vectors of instances
    which are positive to the label.
while the number of labels in any leaf node is larger than the threshold do
    if node is on top levels then
        Calculate the distance between the cluster centroids and labels with OpenMP.
    else
        Execute K-means in its own thread
    end if
    Partition the internal node into new nodes $node_1$,$node_2$,... according to the
        resulting clusters.
    Assign the new nodes $node_1$,$node_2$,... to idle threads
end while
Sort all nodes in an ascending order according to their numbers.
return balanced tree
```

3.1 First-Stage Parallelization

In this stage, training process of each node is carried on a single thread. And we parallelize the training process of nodes on the same level. The notion behind is that each node is split into two nodes with totally different labels. When it comes to splitting these two child nodes, they do not affect each other. In other words, siblings do not have data dependency.

3.2 Second-Stage Parallelization

Since each tree node is halved, the training time of child nodes will also be halved. In other words, the time child nodes take for training should be half of the time their parent takes. We make the analysis to demonstrate this and we will discuss this in Sect. 4.3. To maximize the usage of threads and speedup the training process, we parallelize the k-means, which is used to split the parent node into two parts, for the nodes on top layers with OpenMP. Here, we set the first five layers as top layer. Since calculating the distance between labels and cluster centroids is the most time consuming step in k-means, we parallelize this process with OpenMP. For the rest nodes which are on the sixth or after sixth layers, since their label sets are not large enough and there is no idle thread, they do not need to parallelize the k-means.

4 Experiments

4.1 Dataset Description

Table 1. Dataset Statistics

Dataset	Number of train points	Number of test points	Label dimensionality	Feature dimensionality
EURLex-4K	15,539	3,809	3,993	5,000
WikiLSHTC-325K	1,778,351	587,084	325,056	1,617,899
Wiki-500K	1,813,391	783,743	501,070	2,381,304
Amazon-670K	490,449	153,025	670,091	135,909

We carry out experiments on publicly available datasets from the Extreme Classification repository[4]. The detailed information of these datasets is shown in Table 1. All these datasets are processed from their original sources such as Wikipedia and Amazon. To figure out the effectiveness of the algorithm on different scale dataset, we choose one small dataset (EURLex-4K with 3,993 labels) and three large scale datasets (WikiLSHTC-325K, Wiki-500K and Amazon-670K) which include hundreds of thousands labels along with million train points.

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

4.2 Evaluation Metrics

We use precision at k and speedup as the metrics for comparison. Precision at k is a commonly used metrics in extreme multi-label classification to show the classification accuracy. And speedup is used to show the effectiveness of the parallelization. For a predicted score vector $\hat{y} \in R^L$ and the ground truth label vector $y \in \{0,1\}^L$, the precision at k is defined as:

$$P@k := \frac{1}{k} \sum_{l \in rank_k \hat{(y)}} y_l \tag{1}$$

The speedup is defined as:

$$S = \frac{T_s}{T_p} \tag{2}$$

where T_s is the time that the experiment takes in a serial way and T_p is the time that the experiment takes in a parallel way. Higher value of S means more effectiveness of the algorithm.

4.3 Results

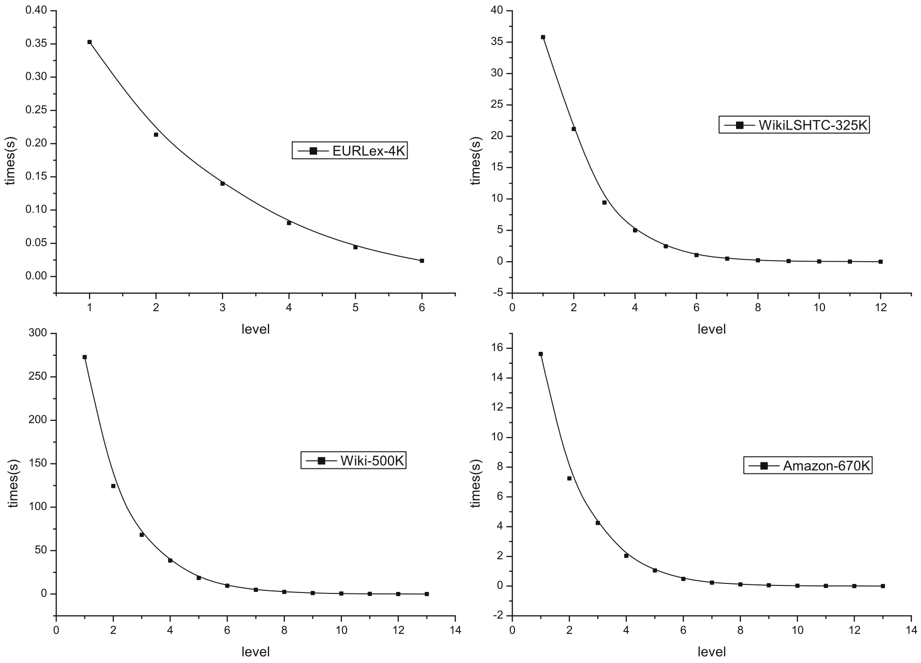

Fig. 2. Average training time of each tree level for four datasets.

Figure 2 shows the average split time of each level. As we can see, the average split time of second-layer nodes is about half of the root node. The average split time of third-layer nodes is about half of the second-layer nodes. So are the forth layer and fifth layer. When it comes to layers after fifth layer, the split time is almost the same. The reason is that the original label set has been divided into more than 32 parts and the k-means process in these nodes can be quickly converged. As we can see, k-means process is the most time consuming step in top level nodes. When it comes to other levels, a large number of node partition is the most time consuming step. Therefore, we can parallelize the k-means process with OpenMP on top levels to accelerate the training.

All experiments are run on two Intel Xeon E5-2620 v4 2.10 GHz CPUs. Each CPU has 8 physical cores. There are no hyper threads per core. While the proposed method is based on Parabel, the precision@k and speedup are calculated between the Parabel, PParabel and fastXML. For Parabel and PParabel, the number of balanced trees trained is three and two algorithms use squared hinge loss. But fastXML trains fifty trees in order to achieve high accuracy. Table 2 shows the results on extreme classification datasets. It turns out that the prediction accuracy of PParabel is almost the same as Parabel. It is much better than fastXML which trains much more trees to increase the accuracy. Since we

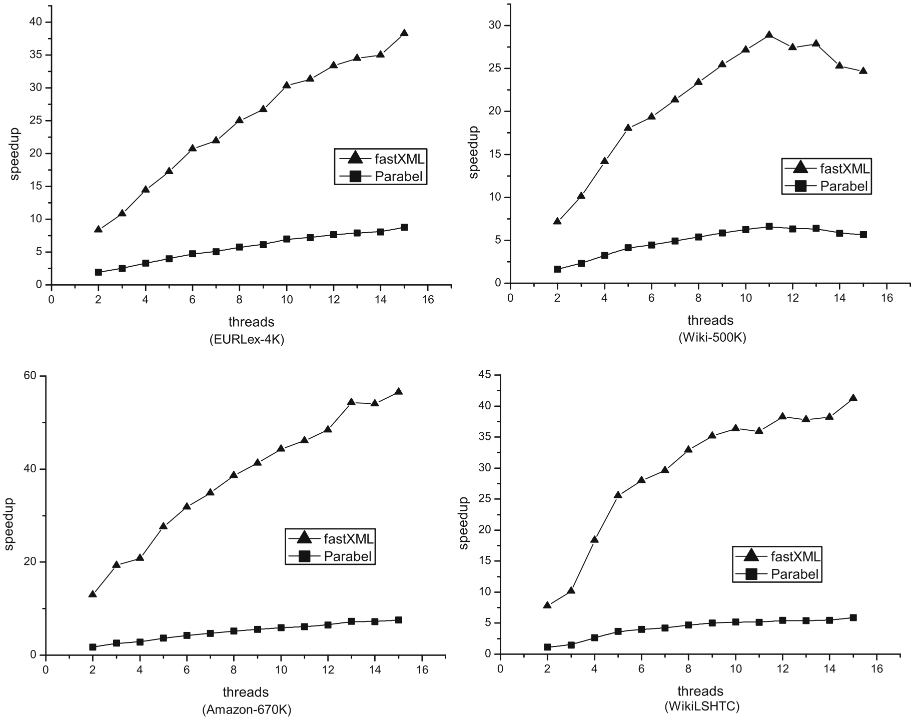

Fig. 3. Speedup of different threads on four datasets.

Table 2. Results on extreme classification datasets.

Method	P1(%)	P3(%)	P5(%)	Training time (hr)	Method	P1(%)	P3(%)	P5(%)	Training time (hr)
EURLex-4K					*Wiki-500K*				
fastXML	71.36	59.90	50.39	0.0590	fastXML	49.27	33.30	25.63	27.72
Parabel	82.25	68.71	57.53	0.0136	Parabel	68.52	48.42	38.55	6.37
PParabel-t=2	81.31	68.33	57.04	0.0070	PParabel-t=2	67.97	48.83	38.02	3.88
PParabel-t=3				0.0054	PParabel-t=3				2.75
PParabel-t=4				0.0041	PParabel-t=4				1.96
PParabel-t=5				0.0034	PParabel-t=5				1.53
PParabel-t=6				0.0029	PParabel-t=6				1.43
PParabel-t=7				0.0027	PParabel-t=7				1.30
PParabel-t=8				0.0024	PParabel-t=8				1.19
PParabel-t=9				0.0022	PParabel-t=9				1.09
PParabel-t=10				0.0019	PParabel-t=10				1.02
PParabel-t=11				0.0019	PParabel-t=11				0.96
PParabel-t=12				0.0018	PParabel-t=12				1.01
PParabel-t=13				0.0017	PParabel-t=13				0.995
PParabel-t=14				0.016	PParabel-t=14				1.09
PParabel-t=15				0.0015	PParabel-t=15				1.12
WikiLSHTC-325K					*Amazon-670K*				
fastXML	49.75	33.10	24.45	4.556	fastXML	36.99	33.28	30.53	2.263
Parabel	65.04	43.23	32.05	0.651	Parabel	44.90	39.81	35.99	0.302
PParabel-t=2	64.08	42.54	31.46	0.585	PParabel-t=2	44.38	39.28	35.44	0.174
PParabel-t=3				0.458	PParabel-t=3				0.117
PParabel-t=4				0.248	PParabel-t=4				0.109
PParabel-t=5				0.178	PParabel-t=5				0.082
PParabel-t=6				0.163	PParabel-t=6				0.071
PParabel-t=7				0.154	PParabel-t=7				0.065
PParabel-t=8				0.138	PParabel-t=8				0.059
PParabel-t=9				0.129	PParabel-t=9				0.055
PParabel-t=10				0.125	PParabel-t=10				0.051
PParabel-t=11				0.126	PParabel-t=11				0.049
PParabel-t=12				0.119	PParabel-t=12				0.047
PParabel-t=13				0.120	PParabel-t=13				0.0426
PParabel-t=14				0.119	PParabel-t=14				0.0428
PParabel-t=15				0.110	PParabel-t=15				0.040

just parallelize the partition process before learning classifiers, the precision@k should be the same theoretically. However, choosing random starting points may result in different clustering results. This may explain why precision@k of Parabel and PParabel are slightly different.

Table 2 also shows the training time of three algorithms which run on these datasets. It can be seen that as the number of threads increases, the training time of PParabel gets shorter and shorter. But when the number of threads increases more than 10, the training time will not decrease much. We can also reduce the training time from 27 h to 1 h just using one machine. It is great to shorten the training time while using much fewer machines.

Figure 3 shows the speedup of different threads on four datasets. The number of threads for PParabel varies from 2 to 15. Both Parabel and fastXML are run with a single thread. We set the maximum threads 15 since we want to make sure that every thread can work on different processor separately which make parallelization happen. PParabel uses multi-threads, while Parabel and fastXML use just one thread. And all these experiments are run in this situation.

The maximum speedup of Parabel is around 9 which is achieved in EURLex-4K dataset. And the maximum speedup of fastXML is around 57 which is achieved in Amazon-670K dataset. As it can be seen, the speedup gain per thread is getting down with the number of threads increasing. The reason is that to protect the data consistency, we need to block other completed threads until the current thread finishes writing. When the number of threads increases, the chance of being blocked is getting bigger and bigger which wastes a lot of time to communicate. Therefore, the speedup gain per thread is getting down with the number of threads increasing. But the speedup is achieved almost linearly with the threads number increasing. In this consideration, to maximize the performance, the optimal number of threads is around 15.

5 Conclusion

In this paper, we have discussed the hardware cost and training time of four typical kinds of approaches to extreme multi-label classification. In order to reduce the hard-ware cost and speedup the training process, we have proposed PParabel algorithm based on Parabel. Our main contribution is employing a two-stage thread-level parallelism. Moreover, we analyze the data independence of nodes on the same level to make sure the training process can be successfully parallelized. The experiment results show that our method is successful to accelerate the training process. All our experiments are conducted on a standard desktop. However, the speedup is achieved almost linearly with the thread number increasing. In the future work, we will study more sufficient approaches in thread level.

Acknowledgement. We thank all viewers who provided the thoughtful and constructive comments on this paper. The third author is the corresponding author. We are grateful to Dr. Manik Varma and his group for their preprocessed datasets. We also thank ECNU Public Platform for Innovation (001) for their equipment to carry out our experiments.

References

1. Agrawal, R., Gupta, A., Prabhu, Y., Varma, M.: 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 (WWW), pp. 13–24. ACM, New York (2013)

2. Babbar, R., Schölkopf, B.: DiSMEC: distributed sparse machines for extreme multi-label classification. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM), pp. 721–729. ACM, New York (2017)
3. Bhatia, K., Jain, H., Kar, P., Varma, M., Jain, P.: Sparse local embeddings for extreme multi-label classification. In: Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS), vol. 1, pp. 730–738. MIT Press Cambridge, MA (2015)
4. Choromanska, A.E., Langford, J.: Logarithmic time online multi-class prediction. In: Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS), vol. 1, pp. 55–63. MIT Press Cambridge, MA (2015)
5. Jain, H., Prabhu, Y., Varma, M.: 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 (KDD), pp. 935–944. ACM, New York (2016)
6. Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., Hullermeier, E.: Extreme F-measure maximization using sparse probability estimates. In: Proceedings of the 33rd International Conference on Machine Learning(ICML), vol. 48, pp. 1435–1444 (2016)
7. Liu, J., Chang, W.C., Wu, Y., Yang, Y.: Deep learning for extreme multi-label text classification. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 115–124. ACM, New York (2017)
8. Mouhamadou, M.C.: Efficient extreme classification. Data Structures and Algorithms. [cs.DS]. Université Pierre et Marie Curie - Paris VI (2014)
9. Niculescu-Mizil, A., Abbasnejad, E.: Label filters for large scale multilabel classification. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 1448–1457 (2017)
10. Prabhu, Y., Varma, M.: 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 (KDD), pp. 263–272. ACM, New York (2014)
11. Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., Varma, M.: Parabel: partitioned label trees for extreme classification with application to dynamic search advertising. In: Proceedings of the 2018 World Wide Web Conference (WWW), pp. 993–1002. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva (2018)
12. Yen, I.E.H., Huang, X., Dai, W., Ravikumar, P., Dhillon, I., Xing, E.: 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 (KDD), pp. 545–553. ACM, New York (2017)
13. Yen, I.E.H., Huang, X., Zhong, K., Ravikumar, P., Dhillon, I.S.: PD- Sparse: a primal and dual sparse approach to extreme multiclass and multilabel classification. In: Proceedings of the 33rd International Conference on Machine Learning (ICML), vol. 48, pp. 3069–3077. JMLR.org (2016)
14. Yu, H., Jain, P., Kar, P., Dhillon, I.S.: Large-scale multi-label learning with missing labels. In: Proceedings of the 31st International Conference on Machine Learning (ICML), vol. 32, pp. I-592–I-601. JMLR.org (2014)
15. Zhang, W., Yan, J., Wang, X., Zha, H.: Deep extreme multi-label learning. In: proceedings of the 2018 ACM on International Conference on Multimedia Retrieval (ICMR), pp. 100–107. ACM, New York (2018)