Deep Extreme Multi-label Learning

Wenjie Zhang[1], Junchi Yan[1,2], Xiangfeng Wang[1] and Hongyuan Zha[1]
[1]East China Normal University
[2] IBM Research – China
izhangwenjie@gmail.com
{jcyan,xfwang,zha}@sei.ecnu.edu.cn

Abstract

Extreme multi-label learning (XML) or classification has been a practical and important problem since the boom of big data. The main challenge lies in the exponential label space which involves 2^L possible label sets when the label dimension L is very large, e.g., in millions for Wikipedia labels. This paper is motivated to better explore the label space by building and modeling an explicit label graph. In the meanwhile, deep learning has been widely studied and used in various classification problems including multi-label classification, however it has not been sufficiently studied in this extreme but practical case, where the label space can be as large as in millions. In this paper, we propose a practical deep embedding method for extreme multi-label classification. Our method harvests the ideas of non-linear embedding and modeling label space with graph priors at the same time. Extensive experiments on public datasets for XML show that our method performs competitively against state-of-the-art result.

Introduction

In the field of machine learning, eXtreme Multi-label Learning (XML) addresses the problem of learning a classifier that can automatically tag a data sample with the most relevant subset of labels from a large label set. For instance, there are more than a million labels (i.e. categories) on Wikipedia and one may wish to build a classifier that can annotate a new article or web page with a subset of relevant Wikipedia categories. Extreme multi-label learning or specifically classification, is a very challenging research problem for the need to simultaneously deal with massive labels, dimensions and training points. Compared with traditional multi-label classification methods (Tsoumakas and Katakis 2006), *extreme* multi-label classification methods focus on tackling the problem of extremely high input dimensions for both input feature dimension and label dimension. It should also be emphasized that multi-label learning is distinct from multi-class classification (Wu, Lin, and Weng 2004) whose aim is to predict a single mutually *exclusive* label. In contrast, XML allows for the co-existence of more than one labels for a single data sample.

One straightforward method for classification is to train an independent one-against-all classifier for each label, which is clearly not the optimal for multi-label classification as the dependencies among class labels can not be leveraged. Furthermore, in the context of extreme multi-label classification, this is not feasible since it will be almost computationally intractable to train a massive number of e.g. one million classifiers. The issue could be ameliorated if a label hierarchy was provided. However, such a hierarchy is often unavailable in many applications (Bhatia et al. 2015). The pain also lies in the prediction stage, where all the classifiers need to be evaluated for each testing data sample. To address these challenges, state-of-the-art extreme multi-label classification methods have been proposed recently, which in general can be divided into two categories: tree based methods and embedding based methods.

Tree based methods (Weston, Makadia, and Yee 2013; Agrawal et al. 2013; Prabhu and Varma 2014) have become popular as they enjoy notable accuracy improvement over traditional embedding methods. The idea is to learn a hierarchy from the training data. The root is initialized to contain the whole label set. A node partition formulation is then optimized to determine which labels should be assigned to the left child or to the right. Nodes are recursively partitioned till each leaf contains a small number of labels. In the prediction stage, a testing data sample is passed down the tree until it arrives at the leaf nodes whereby its predicted labels are finally determined.

Another important line of research is the embedding based method (Hsu et al. 2009; Zhang and Schneider 2011; Tai and Lin 2012; Balasubramanian and Lebanon 2012; Bi and Kwok 2013; Cisse et al. 2013). These approaches attempt to make training and prediction tractable by assuming that the training label matrix (of which each column/row corresponds a training sample's label vector) is low-rank and reducing the effective number of labels by projecting the high dimensional label vectors onto a low dimension linear subspace. While for prediction, labels for a novel sample are predicted by post-processing where a decompression matrix lifts the embedded label vectors back to the original extremely high dimensional label space. However, the fundamental problem of the embedding method is the low-rank label space assumption, which is violated in most real world applications (Bhatia et al. 2015). In general, traditional embedding methods are more efficient for computing than tree based method at the expense of lower accuracy.

Notably, the state-of-the-art embedding based method SLEEC (**S**parse **L**ocal **E**mbeddings for **E**xtreme Multi-label

Classification) (Bhatia et al. 2015) achieves significant accuracy gain while still being computationally economical, which is attributed to the non-linearity modeled through *neighbourhood preserving constraints*. This inspires us to take the step further along this direction. It is intriguing to take a deep neural network approach for non-linearity modeling. However, to the best of our knowledge, there is very few prior art on deep learning for XML.

Perhaps more importantly, we make the observation that the label structure is also very important in those tree based methods where the node is each label. However, they seem to be totally ignored in state-of-the-art embedding methods. For instance, SLEEC focuses on dimension reduction on the raw *label matrix* (of which each column/row corresponds a training sample's label vector) rather than modeling of the label graph structure as mentioned above.

In this paper, we propose a deep learning based approach for the extreme multi-label learning. We extend traditional deep learning framework for multi-label classification with millions of labels via building non-linear embedding for both feature and label space. As far as we know, our method is the first that models the feature space non-linearity and label graph structure at the same time in solving this problem.

Contribution In a nutshell, the contributions are:

- To the best of our knowledge, this is the first work for explicit label graph structure modeling for the *extreme* multi-label learning. Note the label hierarchy explored by the tree based methods is different from label graph.
- This paper is also the first embedding based model with a deep neural network for XML – see Fig.1. In fact, no prior art is identified for exploring deep learning neither for feature space reduction nor for label space reduction in the XML setting.
- Extensive experiments on various public benchmark show that our method consistently produces the best or the second best results without ensemble.

The paper is organized as follows. Section 2 introduces the related work. The details of the above mentioned concept and idea will be detailed in Section 3. Section 4 presents the main empirical results and Section 5 concludes this paper.

Related Work

Classification methods for XML

As the XML problem is in general algorithmically intractable for one-against-all classifier [1], as mentioned in Section 1, various tree based and embedding based models are devised.

Tree based methods for XML The label partitioning by sub-linear ranking (LPSR) method (Weston, Makadia, and Yee 2013) is focused on reducing the prediction time by learning a hierarchy over a base classifier. While it can also incur quite expensive costs since it needs to learn the hierarchy in addition to the base classifier. The multi-label random forest method (MLRF) (Agrawal et al. 2013) seeks to learn an ensemble of randomized trees instead of relying on the learning of a base classifier. Both MLRF and LPSR suffer the high training costs. FastXML (Prabhu and Varma 2014) is considered the state-of-the-art tree-based method for XML, which is proposed to learn a hierarchy not over the label space but over the feature space. FastXML defines the set of labels active in a region to be the union of the labels of all training points present in that region. Predictions are made by returning the ranked list of the most frequently occurring labels in all the leaf nodes in the ensemble containing the novel point.

Embedding methods for XML To make training and prediction tractable, embedding based approaches aim to reduce the effective number of labels, which is often based on the low-rank assumption. Specifically, given I training samples $(\mathbf{X}_i, \mathbf{Y}_i)$ for the feature vector $\mathbf{X}_i \in \mathbb{R}^d$ and the label vector $\mathbf{Y}_i \in \{0,1\}^L$ whereby d and L are supposed both to be very high, embedding methods linearly project the label vectors onto a lower k-dimension space: $\mathbf{y}_i = \mathbf{U}\mathbf{Y}_i$ for $\mathbf{y}_i \in \mathbb{R}^k$ and $k \ll d, L$. Then a regressor is learned for the mapping between $\mathbf{y}_i$ and $\mathbf{X}_i$. Labels for a testing sample $\bar{\mathbf{X}}$ is predicted by lifting the predicted low-dimension $\bar{\mathbf{y}}$ label to the original label space $\bar{\mathbf{Y}}$ via linear or nonlinear projection.

In fact, the main difference of existing embedding models often lies in the choice of compression and decompression techniques for embedding and lifting, including compressed sensing (Hsu et al. 2009), output codes (Zhang and Schneider 2011), SVD (Tai and Lin 2012), landmark labels (Balasubramanian and Lebanon 2012; Bi and Kwok 2013), Bloom filters (Cisse et al. 2013). The accuracy for embedding methods achieve significant gain by the recently proposed embedding method SLEEC (Bhatia et al. 2015). It differs from the previous embedding methods in two facts: i) for training, instead of training a global projection matrix onto a linear low-rank subspace, it incorporates the nonlinear neighborhood constraints in the low-dimension embedding space; ii) for prediction, rather than using a decompression matrix for dimension lifting, it uses a simple k-nearest neighbor (k-NN) classifier in the embedding space. Our method falls into the embedding approaches, while we take one step further by exploring neural networks to model the label space structure as well as for feature space embedding.

Traditional multi-label classification

Multi-label classification (MLC) is a classic problem in machine learning. Traditionally this problem is tackled with a moderate number of labels (Tsoumakas and Katakis 2006). This makes it different from the XML problem where it involves millions of or more labels for each sample. Early MLC methods (Boutell et al. 2004) transform the MLC problem either into one or more single-label classification or regression problems. Recent approaches (Cheng, Hüllermeier, and Dembczynski 2010; Bi and Kwok 2011) try to solve the multi-label learning directly. However, when the number of labels grows rapidly, these methods can easily

[1] It is also worth noting that the recent effort (Babbar and Shoelkopf 2017) shows that, via intensive system level parallelization, a careful implementation of one-vs-rest mechanism is attainable with competitive accuracy against state-of-the-art FastXML (Prabhu and Varma 2014) and SLEEC (Bhatia et al. 2015).

become computationally unaffordable. For instance, for tree based models for traditional MLC (Zhang and Zhang 2010; Bi and Kwok 2011), with the large feature dimension and the huge samples number, the trees will be giant which leads to the intractability for training. There is also a principled generalization for the naive one-against-all method (Hariharan et al. 2010) which cost-effectively explores the label priors. Same as 1-vs-all, the method is algorithmically not scalable to XML.

Meanwhile, in cross modal retrieval field, deep neural network (Wang, Li, and Lazebnik 2016) has been designed for learning the shared embedding space between images and texts. However, the input features of text are usually with small dimensions and therefore, the existing deep embedding framework can not be directly applied in the extreme multi-label settings.

Deep Learning for XML

Our approach involves the training and prediction stage. In the training stage, via our proposed framework – see Fig.1, we map the high-dimensional feature and high-dimensional label vectors into a common embedding space. In the prediction stage, a standard k-NN based classifier is used in the embedded feature space to determine the final label predictions. Next, we first briefly introduce the three main components of our proposed framework and then describe training and prediction procedure in detail.

Deep Convolutional Neural Network

Over the past years, convolutional neural networks (CNNs) have significantly boosted state-of-the-art performance in many visual classification tasks such as object recognition (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016) and hand-written digit recognition (Wan et al. 2013), while it has also been successful in natural language processing (NLP) tasks such as text classification (Kim 2014) and machine translation (Gehring et al. 2017). Generally, CNNs consist of several basic components, such as convolution layer used for extract low-level or higher-level spatial features, pooling layer for reducing the complexity of the model by down-sampling and batch normalization (BN) layer for accelerating convergence and reducing overfitting. In general, the layered learning architecture, together with convolution and pooling which carefully extract features from local to global, render the strong data representation ability of CNNs as well as their current significant positions in computer vision and natural language processing. Compared simply multi-layer perceptron (MLP), deep CNNs has more model capacity and less model parameters and it has shown potential for learning state-of-the-art feature embeddings or deep metrics, so we use CNNs as *backbone* network in Fig.1.

DeepWalk

Recently, network representation learning (NRL) or network embedding has attracted considerable attention. It studies the problem of embedding information networks into low-dimensional spaces, in which every vertex is represented as a low-dimensional vector. Such a low-dimensional embedding is very useful in a variety of applications such as node classification (Bhagat, Cormode, and Muthukrishnan 2011), link prediction (Liben-Nowell and Kleinberg 2007) and recommendation (Yu et al. 2014b).

DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) belongs to one of the NRLs, which is an algorithmic framework for learning continuous feature representations for nodes in networks. It shows that the distribution of nodes appearing in short random walks is similar as the distribution of words in natural language, so it generalizes recent advancements in language modeling from sequences of words to graphs and it employes Skip-Gram, a word representation learning model (Mikolov et al. 2013) to learn the representations of nodes. Because deepwalk has good computational efficiency and scalability, it is chosen in our framework – see Fig.1

Embedding Loss

Embedding loss refers to a class of deep metric learning loss such as contrastive loss (Chopra, Hadsell, and LeCun 2005) and triplet loss (Schroff, Kalenichenko, and Philbin 2015; Hoffer and Ailon 2015) as well as their variants are widely used in image retrieval, face recognition and person re-identification. They are usually able to achieve good performance, but the training is difficult and its overhead is slightly larger, because we need to sample lots of appropriate tuples or triplets. In this paper, we aim to propose a simple and effective method. Hence we use the ℓ_2 norm: $\|\mathbf{x} - \mathbf{y}\|_2$ in our proposed framework where **x** and **y** are multi-dimensional variable.

Formal Definition

Let $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ be the given training dataset, $\mathbf{x}_i \in \mathbb{R}^d$ be the input feature vector, $\mathbf{y}_i \in \{0,1\}^L$ be the corresponding label vector, and $y_{ij} = 1$ iff the j-th label is turned on for $\mathbf{x}_i$. Let $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ be the data matrix and $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$ be the label matrix. For any unseen instance **x**, the multi-label classifier $h(\cdot)$ predicts $h(x) : \mathbb{R}^d \to \{0,1\}^L$ as the set of proper labels for **x**. Let $M \in \mathbb{R}^{L \times L}$ be the label adjacency matrix and $P \in \mathbb{R}^{L \times k}$ be label embedded matrix.

Training stage

Label space embedding We first build the label adjacency matrix M according to the label graph structure in Fig.1. Each node in the label graph denotes a label and there is an edge if the two connecting labels co-appear on at least one sample, Fig.2 shows one example of how to build the label graph from the label space. Then, we use DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) to learn low-dimensional continuous feature representations, which is P for nodes in the label graph.

Feature embedding Given the low-dimensional representation of $\mathbf{y} \in \{0,1\}^{1 \times L}$ by $\mathbf{y}'=\frac{1}{nnz(\mathbf{y})}\mathbf{y}P$, where $P \in \mathbb{R}^{L \times k}$ and $nnz(\cdot)$ means the number of non-zero elements in the original binary vector **y**, the deep convolutional network is trained to find a mapping that the sample's feature vector is embedded into the same space with the low-dimensional

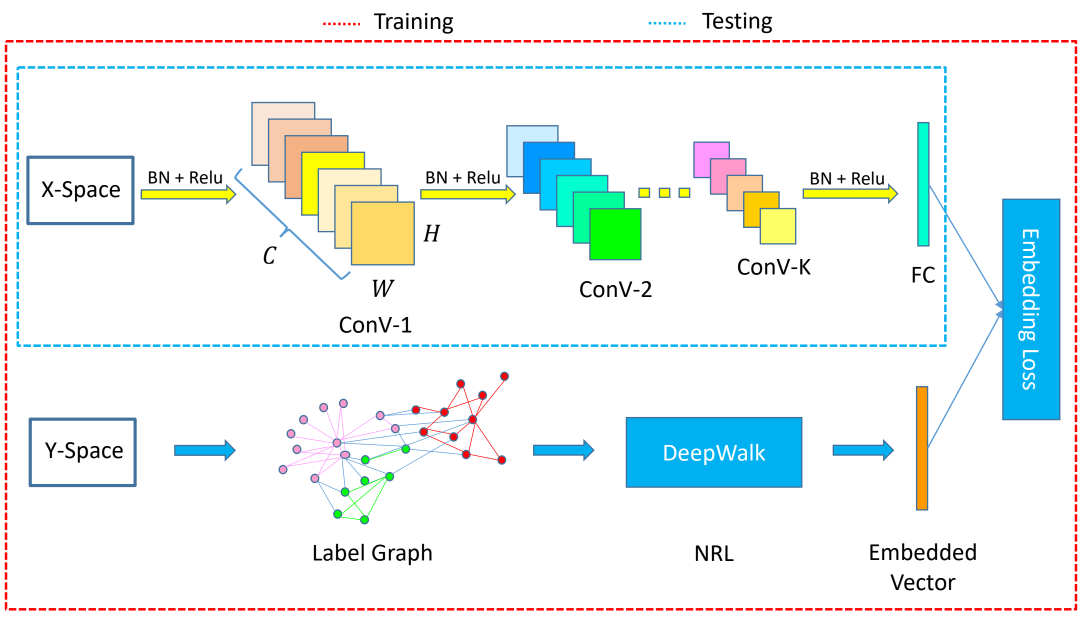

Figure 1: The proposed framework: C denotes the number of channel, W denotes the width of feature map, H denotes the height of feature map, BN denotes Batch Normalization layer, Relu denotes non-linear activation function, ConV-K denotes the k-th convolution layer, FC denotes the fully-connected layer, NRL stands for network representation learning.

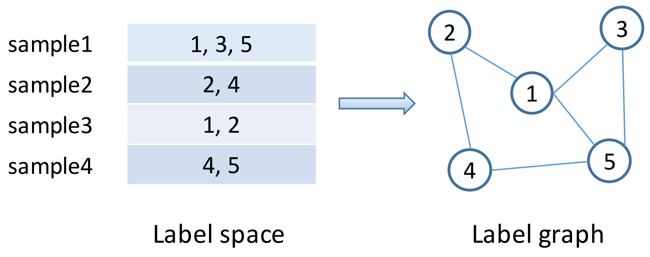

Figure 2: Example for build the label graph.

representation of $\mathbf{y}$. Moreover x and y are close to each other as measured by the embedding loss. The hope is that such an embedding will be effective as it involves the information from the labels.

Prediction stage

The prediction is relatively simple and standard, which is in line with SLEEC (Bhatia et al. 2015): given a test sample, we perform k-NN search in its low-dimensional feature representation to find its similar samples from the training dataset. Then the average (or simply sum) of the k-nearest neighbors' labels is set as final label prediction.

The working flow of our method is shown in Algorithm 1, termed by DXML (**D**eep embedding for **XML** classification).

Algorithm 1 Deep embedding for XML classification.

```
1: Training stage
2: Build $M$ by label graph;
3: Use DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) to obtain $P$;
4: Project the sample label vector $\mathbf{y}$ by $\mathbf{y}' = \frac{1}{nnz(\mathbf{y})}\mathbf{y}P$;
5: Train the joint embedding deep neural network shown in Fig.1 to obtain the mapping from raw feature $\mathbf{x}$ to embedded feature $\mathbf{x}'$. The loss is the $\ell_2$ norm linear regression loss in the embedding space $\|\mathbf{x}' - \mathbf{y}'\|_2$;
6: Prediction stage
7: $\mathbf{y} = \sum_{i:\mathbf{x}_i \in \text{kNN}(\mathbf{x})} \mathbf{y}_i$
```

Experiments

Experiments are carried out on publicly available XML benchmark dataset from the Extreme Classification Repository[2]. It includes both small-scale dataset (Prabhu and Varma 2014) and large-scale dataset (Bhatia et al. 2015), in comparison with state-of-the-art peer methods for both embedding based and tree based models.

Protocol

Platform The experiments are implemented under CentOS-6.5 64-bit system, with Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz $\times$6 CPU, GeForce GTX TITAN-X and 128G RAM. The code is written in Python 2.7 and the network is built by MXNet[3] (Chen et al. 2015), which is a flexible and

[2] http://research.microsoft.com/en-us/um/people/manik/downloads/XC/XMLRepository.html
[3] https://github.com/dmlc/mxnet

Table 1: Statistics of the datasets used in experiments. Point denotes the data sample.

Scale	Dataset	Train	Test	Features	Labels	Avg. points per label	Avg. labels per point
Small	MediaMill	30,993	12,914	120	101	1902.15	4.38
	Bibtex	4,880	2,515	1,836	159	111.71	2.40
	Delicious	12,920	3,185	500	983	311.61	19.03
	EURLex	15,539	3,809	5,000	3,993	25.73	5.31
Large	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

Table 2: P@k on small scale datasets.

Dataset	P@k	Embedding based							Tree based		Other	
		DXML	SLEEC	LEML	WSABIE	CPLST	CS	ML-CSSP	FastXML	LPSR	1-vs-All	KNN
MediaMill	P@1	88.71	87.82	84.01	81.29	83.35	83.82	78.95	84.22	83.57	83.57	82.97
	P@3	72.35	73.45	67.20	64.74	66.18	67.32	60.93	67.33	65.78	65.50	67.91
	P@5	59.81	59.17	52.80	49.83	51.46	52.80	44.27	53.04	49.97	48.57	54.23
Bibtex	P@1	63.71	65.08	62.54	54.78	62.38	58.87	44.98	63.42	62.11	62.62	57.04
	P@3	38.33	39.64	38.41	32.39	37.84	33.53	30.43	39.23	36.65	39.09	34.38
	P@5	27.51	28.87	28.21	23.98	27.62	23.72	23.53	28.86	26.53	28.79	25.44
Delicious	P@1	68.17	67.59	65.67	64.13	65.31	61.36	63.04	69.61	65.01	65.02	64.95
	P@3	62.02	61.38	60.55	58.13	59.95	56.46	56.26	64.12	58.96	58.88	58.89
	P@5	57.45	56.56	56.08	53.64	55.31	52.07	50.16	59.27	53.49	53.28	54.11
EurLEX	P@1	78.67	79.26	63.40	68.55	72.28	58.52	62.09	71.36	76.37	79.89	—
	P@3	64.76	64.30	50.35	55.11	58.16	45.51	48.39	59.90	63.36	66.01	—
	P@5	52.47	52.33	41.28	45.12	47.73	32.47	40.11	50.39	52.03	53.80	—

Table 3: nDCG@k on small scale datasets.

Dataset	nDCG@k	Embedding based							Tree based		Other	
		DXML	SLEEC	LEML	WSABIE	CPLST	CS	ML-CSSP	FastXML	LPSR	1-vs-All	KNN
MediaMill	nDCG@1	88.71	87.82	84.01	81.29	83.35	83.82	78.95	84.22	83.57	83.57	82.97
	nDCG@3	80.78	81.50	75.23	72.92	74.21	75.29	68.97	75.41	74.06	73.84	75.44
	nDCG@5	80.01	79.22	71.96	69.37	70.55	71.92	62.88	72.37	69.34	68.18	72.83
Bibtex	nDCG@1	63.71	65.08	62.54	54.78	62.38	58.87	44.98	63.42	62.11	62.62	57.04
	nDCG@3	57.87	60.47	58.22	50.11	57.63	52.19	44.67	59.51	56.50	59.13	52.29
	nDCG@5	62.03	62.64	60.53	52.39	59.71	53.25	47.97	61.70	58.23	61.58	54.64
Delicious	nDCG@1	68.17	67.59	65.67	64.13	65.31	61.36	63.04	69.61	65.01	65.02	64.95
	nDCG@3	63.45	62.87	61.77	59.59	61.16	57.66	57.91	65.47	60.45	60.43	60.32
	nDCG@5	59.89	59.28	58.47	56.25	57.80	54.44	63.36	61.90	56.38	56.28	56.77
EurLEX	nDCG@1	78.67	79.26	63.40	68.55	72.28	58.52	62.09	71.36	76.37	79.89	—
	nDCG@3	69.70	68.13	53.56	58.44	61.64	48.67	51.63	62.87	66.63	69.62	—
	nDCG@5	62.32	61.60	48.47	53.03	55.92	40.79	47.11	58.06	60.61	63.04	—

efficient library for deep learning and has been chosen by Amazon as the official deep learning framework for its web service.

Datasets The tested extreme multi-label datasets include WikiLSHTC-325K, Delicious-200K (Wetzker, Zimmermann, and Bauckhage 2008) and Wiki10-31K (Zubiaga 2012). All the datasets are publicly available. It should be noted that, some other methods do not scale well on such large datasets. Therefore, we also present comparisons on public relatively small datasets such as BibTex (Katakis, Tsoumakas, and Vlahavas 2008; Prabhu and Varma 2014), MediaMill (Snoek et al. 2006; Prabhu and Varma 2014), Delicious (Tsoumakas, Katakis, and Vlahavas 2008) and EURLex (Menca and Furnkranz 2008). The statistics of the benchmarks are listed in Table 1.

Baseline algorithms for comparison Our primary focus is to compare with those state-of-the-art extreme multi-label classification methods, such as embedding based methods SLEEC (Bhatia et al. 2015), LEML (Yu et al. 2014a) and tree based like FastXML (Prabhu and Varma 2014) and LPSR (Weston, Makadia, and Yee 2013). Our method can be considered as natural combination of label graph and deep embedding. Techniques such as compressed sensing (CS) (Hsu et al. 2009), CPLST (Chen and Lin 2012), ML-CSSP (Bi and Kwok 2013), one-vs-all (Hariharan, Vishwanathan, and Varma 2012) can only be trained on small datasets using commodity computational hardware.

Network structure We propose to learn a non-linear embedding in a deep convolutional neural network framework. As shown in the top half of Fig.1, there are three 1×1 convolution layers, which $C \in \{1024, 512, 512\}$, $W = H = 1$ and one fully-connected layer with $d \in \{150, 300\}$ di-

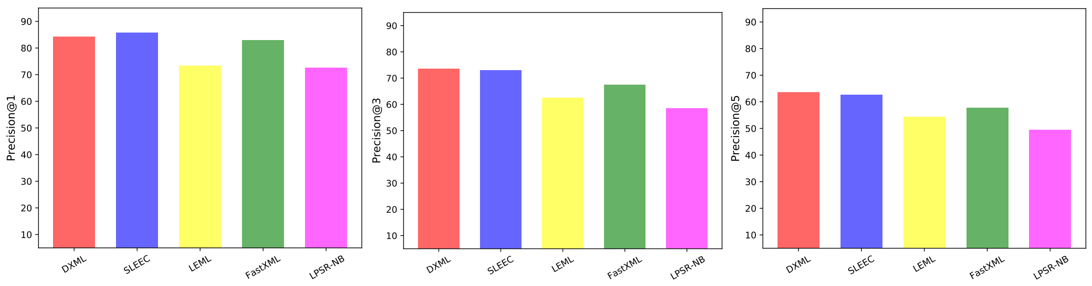

Figure 3: Top k precision of multi-label learning algorithms on the Wiki10-31K dataset.

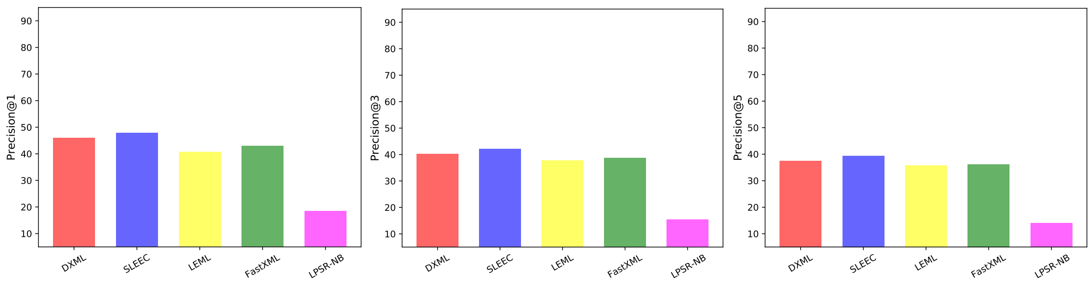

Figure 4: Top k precision of multi-label learning algorithms on the Delicious-200K dataset.

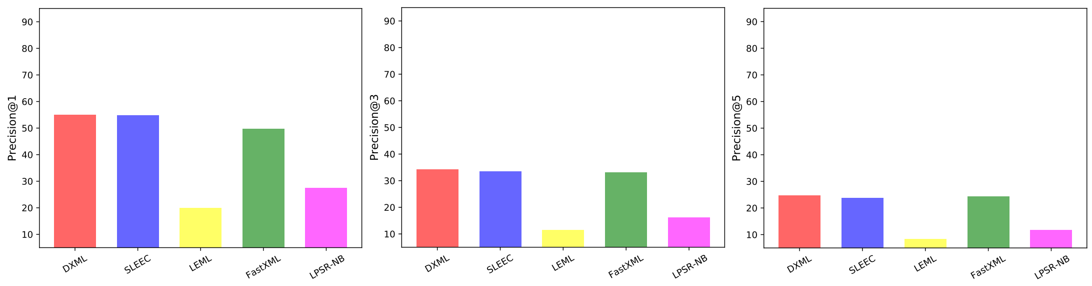

Figure 5: Top k precision of multi-label learning algorithms on the WikiLSHTC-325K dataset.

mension. We apply batch normalization (Ioffe and Szegedy 2015) and Rectified Linear Unit (Relu) nonlinearities right before per convolution layer to improve the convergence and the performance during training.

Hyper-parameter setting For our method, we set the embedding dimension to be 150 for small and 300 for large datasets. We train our networks using SGD (stochastic gradient descent) with momentum 0.9, weight decay 0.0005 and the fixed learning rate 0.0015. The remaining one significant hyper-parameter, the k in k-NN is set by cross-validation on a validation set.

Evaluation metrics The evaluation metric in (Bhatia et al. 2015) is $precison@k$, Precision at k ($P@k$) has been widely adopted as the choice of metric for evaluating extreme multi-label algorithms. The $precision@k$ is the fraction of correct positive labels, which is the number of correct predictions over k. It decreases as k increases. Such a metric encourages the correct label to be ranked higher: $P@k = \frac{\#correct}{k}$. We use the ranking measure $nDCG@k$ as another evaluation metric.

Results and discussion

In general, our method DXML produces the best or the second best results without ensemble.

Results on small datasets The results in Table 2 and 3 are averaged over 10 random train-test split for each dataset. From two tables, one can see that our method DXML can mostly be ranked as top 2 on all the four datasets. On MediaMill, Delicious and EurLEX, DXML, our method almost outperforms all the other baseline algorithms. DXML outperforms LEML and FastXML by nearly 4% for $\{P, nDCG\}@\{3, 5\}$ on MediaMill. On the EurLEX, DXML outperforms LEML and FastXML by around 15% and 7% respectively. While on Bibtex, it slightly underperforms SLEEC. We conjecture the main reason is that the Bibtex dataset is not large enough and has only 4,880 samples, which becomes an issue for training our deep neural network.

Results on large datasets As observed from Fig.3,4,5,6,7 and 8, DXML's predictions could be significantly more accurate than all the other baseline methods (except on

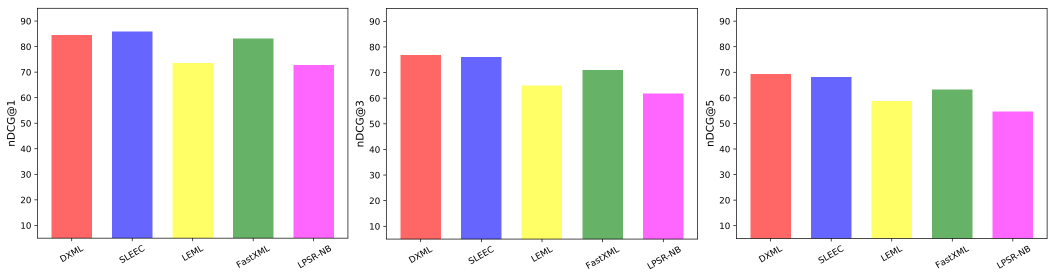

Figure 6: Top k nDCG of multi-label learning algorithms on the Wiki10-31K dataset.

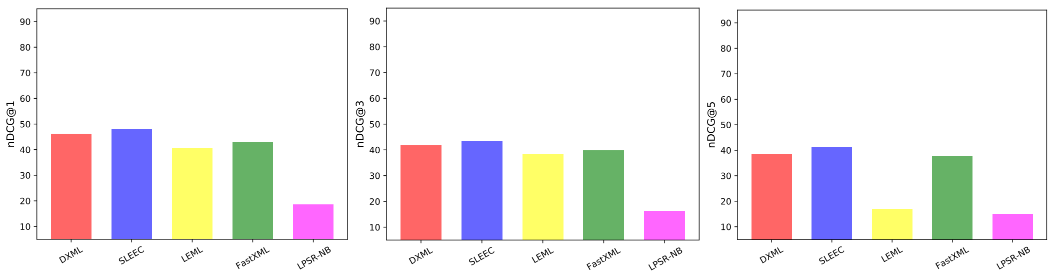

Figure 7: Top k nDCG of multi-label learning algorithms on the Delicious-200K dataset.

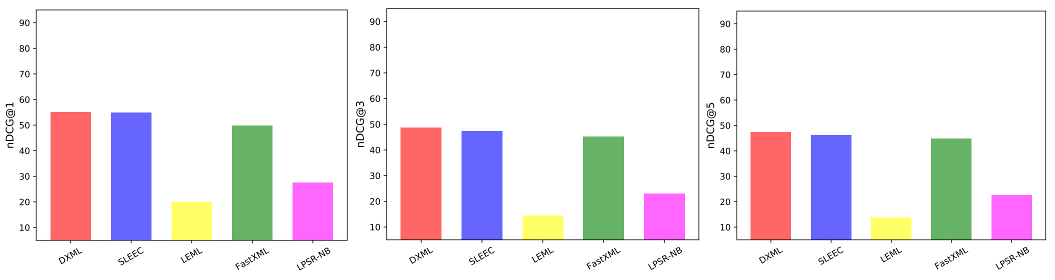

Figure 8: Top k nDCG of multi-label learning algorithms on the WikiLSHTC-325K dataset.

Delicious-200K where DXML was ranked second). On the WikiLSHTC-325K dataset, our method outperforms LFML and LPSR-NB by around 34% and 25% respectively for $\{P, nDCG\}\,@\,\{1,3,5\}$.

Our deep learning based model DXML shows competitiveness not only on large datasets but also on smaller ones. In particular, we achieve comparable results with the state-of-the-art method SLEEC. While our method exhibits certain nice properties over SLEEC: i) Compared SLEEC, DXML does not have the risk of high-dimensional clustering and the extra overhead of ensemble learning, while SLEEC need to cluster high-dimensional feature in the pipeline and SLEEC (with one learner) underperforms our method by around 9% for $\{P, nDCG\}\,@1$ on WikiLSHTC-325K; ii) DXML has better flexibility and scalability. More specically, the *backbone* network in Fig.1 can be replaced by other vogue deep network such as ResNet (He et al. 2016) and DenseNet (Huang et al. 2017) for image and adapt to other domains by using RNN for time series; iii) For new arrival data, SLEEC has to be trained from scratch, while in contrast, DXML allows for incremental learning too.

Conclusion

For the extreme multi-label learning (XML) problem, this paper starts with modeling the large-scale label space via the label graph. In contrast, existing XML methods either explore the label hierarchy as done by many tree based method or perform dimension reduction on the raw label/sample matrix. Moreover, a deep neural network is devised to explore the label space effectively. We also explore deep neural network for learning the embedding function for the feature space as induced by the embedding label space. Extensive experimental results corroborate the efficacy of our method. We leave for future work for more advanced training mechanism for end-to-end deep learning paradigm for XML classification.

Acknowledgments

This material is supported by the NSCF No.11501210, 61602176 and 61672231. We also would like to thank Liwei Wang for discussions on label graph modeling.

References

[Agrawal et al. 2013] Agrawal, R.; Gupta, A.; Prabhu, Y.; and Varma, M. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*.

[Babbar and Shoelkopf 2017] Babbar, R., and Shoelkopf, B. 2017. Dismec-distributed sparse machines for extreme multi-label classification. *WSDM*.

[Balasubramanian and Lebanon 2012] Balasubramanian, K., and Lebanon, G. 2012. The landmark selection method for multiple output prediction. In *ICML*.

[Bhagat, Cormode, and Muthukrishnan 2011] Bhagat, S.; Cormode, G.; and Muthukrishnan, S. 2011. Node classification in social networks. In *Social network data analytics*. Springer.

[Bhatia et al. 2015] Bhatia, K.; Jain, H.; Kar, P.; Jain, P.; and Varma, M. 2015. Sparse local embeddings for extreme multi-label classification. In *NIPS*.

[Bi and Kwok 2011] Bi, W., and Kwok, J. T. 2011. Multi-label classification on tree-and dag-structured hierarchies. In *ICML*.

[Bi and Kwok 2013] Bi, W., and Kwok, J. T.-Y. 2013. Efficient multi-label classification with many labels. In *ICML*.

[Boutell et al. 2004] Boutell, M. R.; Luo, J.; Shen, X.; and Brown, C. M. 2004. Learning multi-label scene classification. *Pattern recognition* 37(9).

[Chen and Lin 2012] Chen, Y.-N., and Lin, H.-T. 2012. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*.

[Chen et al. 2015] Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; and Zhang, Z. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

[Cheng, Hüllermeier, and Dembczynski 2010] Cheng, W.; Hüllermeier, E.; and Dembczynski, K. J. 2010. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*.

[Chopra, Hadsell, and LeCun 2005] Chopra, S.; Hadsell, R.; and LeCun, Y. 2005. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1. IEEE.

[Cisse et al. 2013] Cisse, M. M.; Usunier, N.; Artieres, T.; and Gallinari, P. 2013. Robust bloom filters for large multilabel classification tasks. In *NIPS*.

[Gehring et al. 2017] Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

[Hariharan et al. 2010] Hariharan, B.; Zelnik-Manor, L.; Varma, M.; and Vishwanathan, S. 2010. Large scale max-margin multi-label classification with priors. In *ICML*.

[Hariharan, Vishwanathan, and Varma 2012] Hariharan, B.; Vishwanathan, S.; and Varma, M. 2012. Efficient max-margin multi-label classification with applications to zero-shot learning. *Machine learning* 88(1-2).

[He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

[Hoffer and Ailon 2015] Hoffer, E., and Ailon, N. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*. Springer.

[Hsu et al. 2009] Hsu, D.; Kakade, S.; Langford, J.; and Zhang, T. 2009. Multi-label prediction via compressed sensing. In *NIPS*.

[Huang et al. 2017] Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*.

[Ioffe and Szegedy 2015] Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

[Katakis, Tsoumakas, and Vlahavas 2008] Katakis, I.; Tsoumakas, G.; and Vlahavas, I. 2008. Multilabel text classification for automated tag suggestion. *ECML PKDD discovery challenge*.

[Kim 2014] Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

[Krizhevsky, Sutskever, and Hinton 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

[Liben-Nowell and Kleinberg 2007] Liben-Nowell, D., and Kleinberg, J. 2007. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology* 58(7).

[Menca and Furnkranz 2008] Menca, E., and Furnkranz, J. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *ECML/PKDD*.

[Mikolov et al. 2013] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

[Perozzi, Al-Rfou, and Skiena 2014] Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*.

[Prabhu and Varma 2014] Prabhu, Y., and Varma, M. 2014. Fastxml: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*.

[Schroff, Kalenichenko, and Philbin 2015] Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*.

[Snoek et al. 2006] Snoek, C. G.; Worring, M.; Van Gemert, J. C.; Geusebroek, J.-M.; and Smeulders, A. W. 2006. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *ACM-MM*.

[Tai and Lin 2012] Tai, F., and Lin, H.-T. 2012. Multilabel classification with principal label space transformation. *Neural Computation* 24(9).

[Tsoumakas and Katakis 2006] Tsoumakas, G., and Katakis, I. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3(3).

[Tsoumakas, Katakis, and Vlahavas 2008] Tsoumakas, G.; Katakis, I.; and Vlahavas, I. 2008. Effective and effcient multilabel classification in domains with large number of labels. In *ECML/PKDD*.

[Wan et al. 2013] Wan, L.; Zeiler, M.; Zhang, S.; Cun, Y. L.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *ICML*.

[Wang, Li, and Lazebnik 2016] Wang, L.; Li, Y.; and Lazebnik, S. 2016. Learning deep structure-preserving image-text embeddings. In *CVPR*.

[Weston, Makadia, and Yee 2013] Weston, J.; Makadia, A.; and Yee, H. 2013. Label partitioning for sublinear ranking. In *ICML*.

[Wetzker, Zimmermann, and Bauckhage 2008] Wetzker, R.; Zimmermann, C.; and Bauckhage, C. 2008. Analyzing social bookmarking systems: A del. icio. us cookbook. In *ECAI Mining Social Data Workshop*.

[Wu, Lin, and Weng 2004] Wu, T.-F.; Lin, C.-J.; and Weng, R. C. 2004. Probability estimates for multi-class classification by pairwise coupling. *JMLR* 5(Aug).

[Yu et al. 2014a] Yu, H.-F.; Jain, P.; Kar, P.; and Dhillon, I. S. 2014a. Large-scale multi-label learning with missing labels. In *ICML*.

[Yu et al. 2014b] Yu, X.; Ren, X.; Sun, Y.; Gu, Q.; Sturt, B.; Khandelwal, U.; Norick, B.; and Han, J. 2014b. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*. ACM.

[Zhang and Schneider 2011] Zhang, Y., and Schneider, J. G. 2011. Multi-label output codes using canonical correlation analysis. In *AISTATS*.

[Zhang and Zhang 2010] Zhang, M.-L., and Zhang, K. 2010. Multi-label learning by exploiting label dependency. In *KDD*.

[Zubiaga 2012] Zubiaga, A. 2012. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv:1202.5469*.