Matching Neural Network for Extreme Multi-Label Learning

Zhiyun Zhao[1], Fengzhi Li[2], Yuan Zuo[2,*] and Junjie Wu[2]

[1] National Computer Network Emergency Response Technical Team/Coordination Center of China, No. 3 Yumin Road, Chaoyang District, Beijing 100029, China
[2] School of Economics and Management, Beihang University, No. 37 Xueyuan Road, Haidian District, Beijing 100191, China

[*] E-mail: zuoyuan@buaa.edu.cn

Abstract. Multi-label learning involving hundreds of thousands or even millions of labels is referred to as extreme multi-label learning, in which the labels often follow a power-law distribution with the majority occurring in very few data points as tail labels. As a promising solution of multi-label learning, however, the embedding-based methods face a problem that most of them have the basic low-rank assumption but the widespread of tail labels in data violates it. Recently, research efforts have been put on building tail label tolerant embedding-based models, however, for real-life datasets containing substantial data points with only tail labels, simply treating them as label matrix outliers will incur severe information loss, meanwhile accurately computing the pairwise distances between label vectors turns infeasible. In light of this, we present the Matching Neural Network (MNN), which learns two neural mapping functions that encode feature vectors and label vectors into their distributed representations, respectively. A noise contrastive loss is also proposed to guide the training of the functions so as to ensure matched features and labels have similar distributed representation measured by cosine similarity. Extensive experiments on various benchmark datasets with state-of-the-art baselines demonstrate the more accurate predictions of MNN.

1. Introduction

Different from multi-class learning where each data point has one class/label, multi-label learning tries to predict a label vector given the features of a data point. In recent years, along with the rapid development of information technologies, large-scale multi-label applications with huge numbers of labels keep emerging. The problem that extreme multi-label learning wants to address is to train a classifier that can automatically tag a new data point with the most relevant subset of labels from an extremely large label set. To tackle this problem, numerous embedding-based approaches have been recently proposed [1], [2], [3], [4], [5], [6]. However, these approaches still have limitations since the learning of embeddings depends heavily on similarities between label vectors, which might fail when substantial data points are tagged with only tail labels. Under such circumstance, tail labels can no longer be viewed simply as outliers. So it is almost impossible to compute the distance between two label vectors when they are entirely composed of tail labels.

To learn embeddings more robustly, we propose the Matching Neural Network (MNN). Instead of computing similarities between label vectors to reveal low-dimensional subspace, MNN learns two mapping functions $f_x(\cdot)$ and $f_y(\cdot)$, where $f_x(\cdot)$ projects *features* into a low-dimensional subspace and $f_y(\cdot)$ projects *labels* into the same subspace. We propose a loss function that combines contrastive loss [7] with negative sampling, to guide the learning of mapping functions so that matched features and

labels will be projected near to each other in the subspace, and vise versa. In this way, MNN can reduce the reliance on label vector similarities, and thus gains expertise in learning from data points tagged with only tail labels.

Extensive experiments on six benchmark datasets with state-of-the-art baseline methods demonstrate the more accurate predictive power of MNN. Its robustness against tail labels in learning embeddings is also testified in the comparative study with leading embedding-based methods. Specifically, MNN achieves 8% improvement over the best embedding-based baselines on the dataset "Amazon" with abundant tail labels.

2. Related work

2.1. Extreme Multi-label Classification

Various approaches have been proposed to address the extreme multi-label learning, which can be broadly divided into embedding-based and tree-based methods.

Embedding-based methods tackle extreme multi-label learning by reducing the number of effective labels. Generally, they project label vectors into a low dimensional subspace, and learn predictor for embedded label vectors instead of original one. The advantage is training complexity of predictor is largely decreased since dimensionality of embeddings is only up to several hundreds. However, an additional decompression module is needed to lift the embedded label vectors back to the original label space.

Various compression and decompression techniques have been exploited. Hsu et al. [8] take a three-step approach to handle classification with large number of labels: 1) random transformation is applied to project high-dimensional label vector into lowdimensional one; 2) A regression model is trained as predictor for each dimension of compressed label vector given features of an example; 3) For a test example, its compressed vector is decompressed into original label space. Random transformation causes the decompression needs to solve an optimization problem for each incoming test example, which is time consuming. Therefore, Tai and Lin [9] propose the Principal Label Space Transformation (PLST), which used Principal Component Analysis (PCA) to accomplish the compression operation. Since PCA in PLST only focuses on minimizing the encoding error of label vectors, Chen and Lin [10] propose Conditional Principal Label Space Transformation (CPLST) to further applies Canonical Correlation Analysis (CCA) on feature space, which simultaneously considers the encoding error and prediction error. Zhang and Schneider [11] also take both label and feature matrix into consideration.

Recently, Yu et al. [1] model multi-label classification as a general empirical risk minimization (ERM) problem with low-rank constraint, which generalizes both label and feature dimensionality reduction. However, the low-rank assumption is easily violated due to large number of tail labels in real-world datasets. Xu et al. [12] keep the low-rank assumption, and suppress the influence of tail labels by regarding them as label matrix outliers. Instead of globally projecting into a linear low-rank subspace, Bhatia et al. [2] learn embeddings by preserving the pairwise distances between only the closest label vectors. A k Nearest Neighbor (kNN) classifier is used for prediction, which leverages the fact that distances between closest label vectors have been preserved during training. Rather than changing the embedding methods in LTLS, Evron et al. [4] introduce an efficient loss-based learning and decoding algorithm with better accuracy by adding loss-based decoding methods to it. So as to allow trade-offs between accuracy, model size, and inference time. Gupta et al. [6] leverage word embedding techniques such as word2vec to learn label embeddings and achieve better accuracies and training speed. Besides, the algorithm further improve the performance by joint learning of embedding and regressors through a novel objective. Jalan et al. [5] accelerates extreme classification algorithm by constructing a balanced hierarchy which offers faster and better feature agglomerates than traditional clustering methods. Profiting from feature agglomerates, a relatively large embedding dimension can be used to preserve much of the information of the original vector.

Tree-based methods aim towards faster prediction which can be achieved by recursively partitioning label or feature space. However, due to the cascading effect, the prediction error made at top-level is hard to be corrected at lower levels. As a result, these techniques have to trade-off, prediction accuracy for speed.

The Label Partitioning by Sub-linear Ranking (LPSR) [13] reduces the prediction time by learning a hierarchy over a base classifier or ranker. The prediction accuracy and overall complexity of LPSR is governed by base multi-label classifier. However, classifiers are quick to train tend to have low prediction accuracy. FastXML [14] recursively partitions the feature space, instead of the label space, and observes only small number of labels are active in each region of feature space. PfastreXML [15] improves FastXML by replacing the original nDCG based loss function with propensity scored one, which is able to handle missing but relevant labels and tail labels. SwiftXML [16] improves tree based extreme classifiers by partitioning tree nodes using two hyperplanes learnt jointly in the label and data point feature spaces and can tackle warm-start applications by leveraging label features. More than the use of a forest of decision trees similar to PfastreXML, CRAFTML [17] exploits a random forest strategy to obtain diversity and preserve more information. A novel low-complexity splitting strategy is also proposed to avoid the resolution of a multi-objective optimization problem at each node.

2.2. Metric Learning

Distance metric learning is a representative approach of learning similarities (or dissimilarities) for homogeneous data [18], [19], [20]. Typically, a linear transformation is learned for mapping objects from the same space into a latent space. In that space, dot product or Euclidean distance is often taken as measurement of similarity. Recently, learning a similarity function for a pair of objects from two different spaces has emerged [21], [22], which is also known as learning to match. SLEEC [2] is an instance of distance metric learning while our model is an instance of learning to match, and both models are tailored for extreme multi-label learning by solving problems like scalability and tail labels. Several other methods for multi-label learning [23], [24] are based on distance metric learning, however, they are not suited for extreme multi-label learning, since they are less scalable and aim at predicting all relevant labels, which is not appropriate for data with a large number of labels due to missing labels and tail labels as suggested in [15].

3. Method

Generally speaking, we propose a distance metric learning approach for the multi-label learning task. But our focus is not on the distances between labels. Rather, we try to learn a distance metric for a pair of heterogeneous objects, i.e., the features and labels, such that both of them can be mapped into a same subspace for more accurate matching. This way of modeling helps us avoid the explicitly computing of distances between label vectors, which is critically important to gaining robustness against tail labels.

Therefore, we propose a Matching Neural Network (MNN) that learns to match features with its corresponding labels, which less depends on similarities between labels. As shown in Figure 1, MNN consists of two parts: 1) the Averaging Neural Network (ANN) that acts as a mapping function (or an encoder) that takes features or labels as input and outputs a real-valued vector which also known as distributed representation that encodes inherent informations within features or labels. 2) A noise contrastive loss function is proposed to provide the signal to guide the learning of the two ANNs. The ANNs of features and labels are identical in architecture, but are slightly different in computation. We will first introduce the ANN, along which we describe the computation of distribution representations for features and labels. Then, we present the feature dropout that prevents MNN from overfitting the training data. At last, we discuss the construction of the noise contrastive loss function.

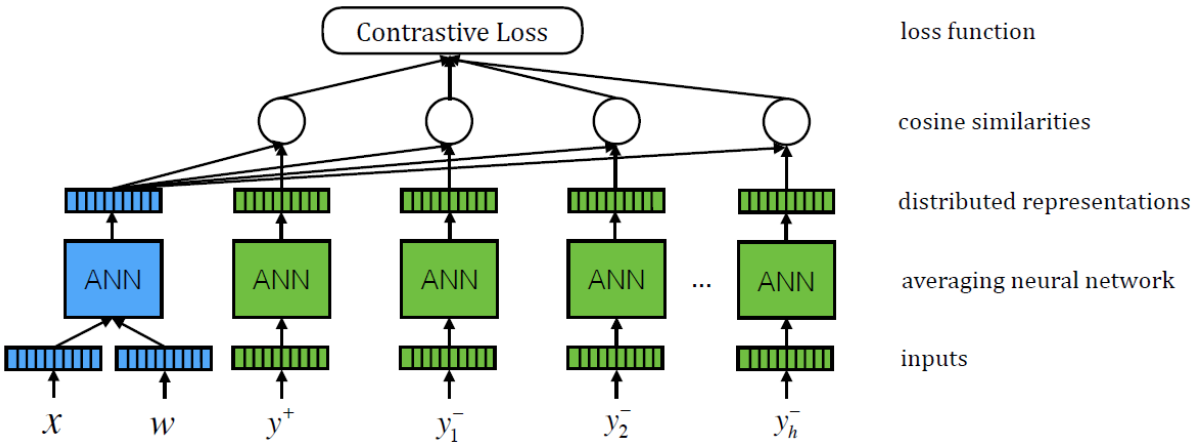

Figure 1. The architecture of the Matching Neural Network (MNN) for extreme multi-label learning.

3.1. Averaging Neural Network

Averaging Neural Network (ANN) consists of three layers namely input layer, embedding layer and averaging layer. By applying the feed-forward process of ANN to input features or labels, one can obtain its distributed representation.

3.1.1. Input Layer and Embedding Layer. The label input to the ANN is $y = \{y_1, \cdots, y_T\}$, where $y_t \in \mathbb{R}^L$ is one-hot-vector representation of the *t*-th label and *L* is the dimensionality of label space. *T* is the length of the labels, which varies for different data points. The feature input to the ANN is $x = \{x_1, \cdots, x_K\}$ and $w = \{w_1, \cdots, w_K\}$, where $x_k \in \mathbb{R}^d$ is one-hot-vector representation of the *k*-th feature and *d* is the dimensionality of feature space, and w_k is a real value that indicates the weight of *k*-th feature. *K* is the length of the features, which also varies for different data points.

Then the embedding layer transforms the one-hot vector of *t*-th label $\mathrm{y_t}$ and the one-hot vector of *k*-th feature $\mathrm{x_k}$ into a low-dimensional dense vector e_t,e'_k.

3.1.2. Averaging Layer. Given label embeddings $e = \{e_1, \cdots, e_T\}$, we apply a composition function g to get distributed representation $\mathbf{z}$ of label sequence $\mathbf{y}$. In our model, g is an instantiation of Neural Bag-of-Words (NBOW) [25], which averages label embeddings

$$\mathbf{z} = g(\mathbf{e}) = \frac{1}{T}\sum_{t=1}^{T} e_t \tag{1}$$

Given feature embeddings $\mathbf{e}'$, g computes the features' distributed representation $\mathbf{z}'$ by weighted averaging feature embeddings $\mathbf{e}'$ due to the existing of weights $\mathbf{w}$

$$\mathbf{z}' = g(\mathbf{e}', \mathbf{w}) = \frac{1}{K}\sum_{k=1}^{K} e'_k w_k \tag{2}$$

In summary, ANN is actually a mapping function that projects labels or features into their low-dimensional representations, i.e., distributed representations. We denote $f_x(x, w)$ and $f_y(y)$ as the mapping function for features and labels respectively:

$$f_y(\mathbf{y}) = g(W_e\mathbf{y}) = \frac{1}{T}\sum_{t=1}^{T} W_e y_t \tag{3}$$

$$f_x(\mathbf{x}, \mathbf{w}) = g(W'_e\mathbf{x}, \mathbf{w}) = \frac{1}{K}\sum_{k=1}^{K} W'_e x_k w_k \tag{4}$$

3.2. Feature Dropout

To regularize neural networks, instead of dropping hidden units, a natural extension for the ANN is to randomly drop feature's entire embedding before averaging features' embeddings. By doing so, our ANN theoretically observes $2^{|x|}$ different feature sequence for each input x.

Assume there is a vector $\mathbf{r}$ for feature sequence $\mathbf{x}$ with $|x|$ independent Bernoulli trials, each of which equals 1 with probability p. The embedding e_k for feature x_k in $\mathbf{x}$ is dropped from the average if $r_k = 0$, which exponentially increases the number of unique examples the network observes during training. This allows us to modify Equation 2:

$$r_k \sim Bernoulli(p) \tag{5}$$

$$\mathbf{z}' = g(\mathbf{e}', \mathbf{w}, \mathbf{r}) = \frac{1}{\|\mathbf{r}\|_1} \sum_{k=1}^{K} e'_k r_k w_k \tag{6}$$

The mapping function $f_x(x, w)$ now can be written as

$$f_x(\mathbf{x}, \mathbf{w}, \mathbf{r}) = g(W'_e \mathbf{x}, \mathbf{w}, \mathbf{r}) = \frac{1}{\|\mathbf{r}\|_1} \sum_{k=1}^{K} W'_e x_k r_k w_k \tag{7}$$

3.3. Noise Contrastive Loss

Since our objective is to ensure distributed representation of features be similar to the one of its corresponding labels and vise versa, we need an appropriate loss function to provide the signal to guide the learning of our network. Here we propose a noise contrastive loss function which tries to maximize the cosine similarity of distributed representations between matched features and labels, meanwhile, tries to minimize the cosine similarity of distributed representations between closest mismatched pair.

Contrastive loss [7] is often applied to learn a low-dimensional space by preserving the distance between a pair of homogeneous objects in their original space. However, it is not well suited for MNN, since our target is preserving the distance between a pair of heterogeneous objects. Therefore, we extend the contrastive loss to scenarios of learning to match by combining it with negative sampling.

Formally, we take $(\mathrm{x_i}, \mathrm{y_i})$ as a matched pair, since they belong to the same data point. We randomly sample h negative samples $\{\mathrm{y}_1, \cdots, \mathrm{y_h}\}$ from training data for $\mathrm{x_i}$. Therefore, we construct a positive pair $(\mathrm{x_i}, \mathrm{y_i})^+$ and h negative pairs $\{(\mathrm{x_i}, \mathrm{y_{i1}})^-, \cdots, (\mathrm{x_i}, \mathrm{y_{ih}})^-\}$. Note that, for brevity of notation, we ignore the weights $\mathrm{w_i}$ for $\mathrm{x_i}$.

By applying the feed-forward process of ANN, we can obtain the distributed representations $f_x(\mathrm{x})$ and $f_y(\mathrm{y})$ of features and labels. We define $c(\mathrm{x}, \mathrm{y})$ as cosine similarity between distributed representations of features and labels:

$$c(\mathbf{x}, \mathbf{y}) = \frac{\langle f_x(\mathbf{x}), f_y(\mathbf{y}) \rangle}{\|f_x(\mathbf{x})\|_2 \|f_y(\mathbf{y})\|_2} \tag{8}$$

For a positive pair $(\mathrm{x_i}, \mathrm{y_i})^+$, we define loss function as

$$\mathcal{L}_+(\mathrm{x_i}, \mathrm{y_i}) = \frac{1}{4}(1 - \mathrm{c}(\mathrm{x_i}, \mathrm{y_i})^2) \tag{9}$$

For a negative pair $(\mathrm{x_i}, \mathrm{y_{ij}})^-$, we define loss function as

$$\mathcal{L}_-(\mathrm{x_i}, \mathrm{y_{ij}}) = \begin{cases} c(\mathrm{x_i}, \mathrm{y_{ij}})^2 & if\ c(\mathrm{x_i}, \mathrm{y_{ij}}) > m \\ 0 & otherwise \end{cases} \tag{10}$$

m is margin that loss is zero when cosine similarity between negative pairs is small than m. The loss function for training set $\mathcal{D}$ is defined as

$$\mathcal{L}(\mathcal{D}) = \sum_{i=1}^{n} \mathcal{L}_+(\mathrm{x_i}, \mathrm{y_i}) + \max_{1 \le j \le h} \left(\mathcal{L}_-(\mathrm{x_i}, \mathrm{y_{ij}}) \right) \tag{11}$$

where we take the maximum $\mathcal{L}_-$ among h negative samples instead of taking the sum or the mean of them, which prevents the whole loss from leaning to negative samples during the optimizing or prevents the whole loss from being more easily influenced by those noisy negative samples.

4. Experiment

4.1. Experimental Setup

4.1.1. Dataset description. We evaluate the proposed model on six publicly available benchmark multi-label datasets from the Extreme Classification Repository [26] namely *Bibtex, Delicious, Eurlex, Wiki10, DeliciousLarge* and *Amazon*. The first three datasets with less than 5,000 labels are regarded as small datasets, while the last three are large datasets since their label dimensionalities range upto 670,091. The detailed statistics for individual datasets are shown in Table 1. We keep the split between training and test datasets the same as given on repository page.

Table 1. Dataset statistics.

Dataset	Number of Train Points	Number of Test Points	Feature Dimensionality	Label Dimensionality	Avg. Points per Label	Avg. Labels per Point
Bibtex	4,880	2,515	1,836	159	111.71	2.4
Delicious	12,920	3,185	500	983	311.61	19.03
Eurlex	15,539	3,809	5,000	3,993	25.73	5.31
Wiki10	14,146	6,616	101,938	30,938	8.52	18.64
DeliciousLarge	196,606	100,095	782,585	205,443	72.29	75.54
Amazon	490,449	153,025	135,909	670,091	3.99	5.45

4.1.2. Evaluation Metrics and Baseline Methods. Like most state-of-the-art methods in the extreme multi-label learning, we apply precision at k, denoted as *precision@k*, and normalized Discounted Cumulative Gain at k, denoted as *nDCG@k* as the metrics for comparison.

Then we compare our model with four state-of-the-art methods on large datasets. These are listed below:

Embedding-based methods that project the labels from high-dimensional space into a low-dimensional sub-space: SLEEC [2], REML [12].

Tree-based methods that are designed for faster prediction by cutting the search space of labels. Since wrong predictions at top level is hard to be recovered, there methods typically trade-off prediction accuracy with speed: PfastreXML [15], LPSR [13].

Apart from above four scalable methods, other five representative multi-learning methods namely as 1-vs-all [27], ML-CSSP [28], CS [8], CPLST [10] and WSABIE [29] are further included in comparison on small datasets.

4.2. Experimental Results

4.2.1. Results on Large Datasets. We compare the prediction performance of our model with leading algorithms on three large datasets. The prediction results of baseline methods, except for REML, are provided by the extreme classification repository [26]. Results of REML are from its paper [12]. All results are measured in precision@k and nDCG@k for k = 1, 3 and 5.

We first discuss the results on Wiki10 and DeliciousLarge as illustrated in Figure 2 and 3. Wiki10 has a relative large average number of labels per point, as shown in Table 1, which implies it contains few data points filled with tail labels. DeliciousLarge has the largest average number of labels per training point and the largest average number of points per label, which implies correlations between labels in this dataset are stronger and number of data points filled with more tail labels than frequent ones is smaller as compared to other datasets. As a result, with sufficient correlations between labels,

embedding-based methods perform better than tree-based ones on both Wiki10 and DeliciousLarge. Among embedding-based methods, our model outperforms SLEEC and REML on Wiki10, and outperforms REML while performs quite similar with SLEEC on DeliciousLarge. The less competitive performance of REML on DeliciousLarge might due to its linear sparse function designed to handle label matrix outliers has difficulty in scaling to hundreds of thousands of tail labels. The good performance of our method indicates learning to match features and labels is able to achieve or outperform leading embedding-based methods on large datasets, even when similarities between labels can be effectively utilized to learn embeddings.

We then highlight the results on Amazon as illustrated in Figure 4. Our method significantly outperforms SLEEC and REML according to results in Figure 4. Amazon contains tens of thousands of data points filled with tail labels, which severely violates the assumption of leading embedding-based models. The outperformance of our method indicates learning embeddings by matching features and labels is more robust against low-rank dimension reduction of REML and sparse locally label embedding of SLEEC on datasets like Amazon. Our method bridges the performance gap between embedding-based approaches with the leading tree-based one.

Interestingly, REML becomes less and less competitive with MNN and SLEEC with the increasing of label dimensionality. REML applies divide-and-conquer strategy to scale to large datasets by splitting the original problem into several subproblems. However, with the increasing number of subproblems, its performance continuously degrades. Thus, REML is less practical than MNN and SLEEC on datasets with a huge set of labels.

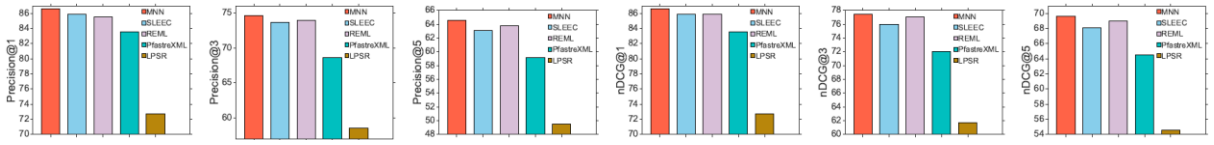

Figure 2. Precision@k and nDCG@k results on the Wiki10 dataset.

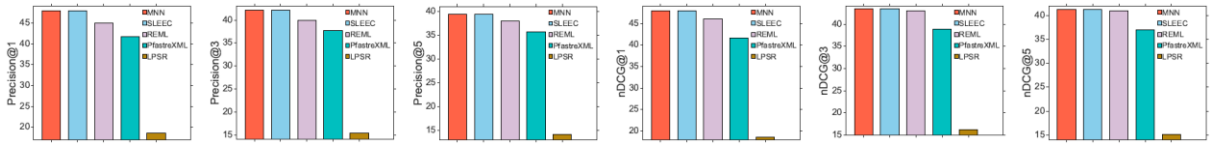

Figure 3. Precision@k and nDCG@k results on the DeliciousLarge dataset.

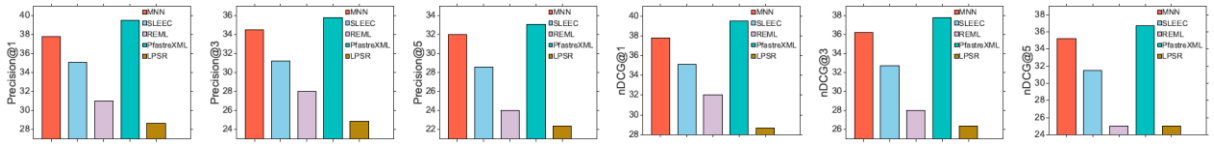

Figure 4. Precision@k and nDCG@k results on the Amazon dataset.

4.2.2. Results on Small Datasets. Here we compare the prediction performance on three small datasets, which can be handled by more state-of-the-art multi-label learning methods such as CPLST and WSABIE etc. The prediction results measured in precision@k and nDCG@k for k = 1, 3 and 5 are presented in Table 2-3, where the results of REML on Eurlex are marked as – denoting the lacking of reported results and the code.

On these datasets, tail label problem is not acute, still our model consistently outperforms embedding-based methods like WSABIE, REML and SLEEC and gains overall best performance against all other methods according to precisions in Table 2. As to results of nDCGs, we can find our model outperforms other embedding-based methods again, except for the nDCG@5 on Bibtex. The state-of-the-art performance of our model on small datasets indicates learning embeddings by matching features and labels is well suited for multilabel learning.

Table 2. Precision@k results on small-scale datasets (with the best results in bold and the second best underlined).

Method	Bibtex			Delicious			Eurlex		
	precision@1	precision@3	precision@5	precision@1	precision@3	precision@5	precision@1	precision@3	precision@5
1-vs-all	62.62	39.09	28.79	65.02	58.88	53.28	<u>79.89</u>	**66.01**	<u>53.80</u>
LPSR	62.11	36.65	26.53	65.01	58.96	53.49	76.37	63.36	52.03
PfastreXML	63.46	39.22	29.14	67.13	<u>62.33</u>	**58.62**	75.45	62.70	52.51
ML-CSSP	44.98	30.43	23.53	63.04	56.26	50.16	62.09	48.39	40.11
CS	58.87	33.53	23.72	61.36	56.46	52.07	58.52	45.51	32.47
CPLST	62.38	37.84	27.62	65.31	59.95	55.31	72.28	58.16	47.73
WSABIE	54.78	32.39	23.98	64.13	58.13	53.64	68.55	55.11	45.12
REML	<u>65.13</u>	<u>41.35</u>	<u>29.89</u>	66.30	61.75	56.67	–	–	–
SLEEC	65.08	39.64	28.87	<u>67.59</u>	61.38	56.56	79.26	64.30	52.33
MNN	**65.21**	**43.61**	**33.48**	**68.88**	**62.44**	<u>57.85</u>	**81.02**	<u>65.17</u>	**53.81**

Table 3. nDCG@k results on small-scale datasets (with the best results in bold and the second best underlined).

Method	Bibtex			Delicious			Eurlex		
	nDCG@1	nDCG@3	nDCG@5	nDCG@1	nDCG@3	nDCG@5	nDCG@1	nDCG@3	nDCG@5
1-vs-all	62.62	59.13	61.58	65.02	60.43	56.28	<u>79.89</u>	**69.62**	**63.04**
LPSR	62.11	56.5	58.23	65.01	60.45	56.38	76.37	66.63	60.61
PfastreXML	63.46	59.61	62.12	67.13	**63.48**	**60.74**	75.45	65.97	60.78
ML-CSSP	44.98	44.67	47.97	63.04	57.91	53.36	62.09	51.63	47.11
CS	58.87	52.19	53.25	61.36	57.66	54.44	58.52	48.46	40.79
CPLST	62.38	57.63	59.71	65.31	61.16	57.8	72.28	61.64	55.92
WSABIE	54.78	50.11	52.39	64.13	59.59	56.25	68.55	58.44	53.03
REML	<u>65.13</u>	60.01	<u>62.46</u>	66.3	62.65	59.1	–	–	–
SLEEC	65.08	<u>60.47</u>	**62.64**	<u>67.59</u>	62.87	59.28	79.26	68.13	61.6
MNN	**65.21**	**60.76**	62.31	**68.88**	<u>62.98</u>	<u>60.53</u>	**81.02**	<u>69.12</u>	<u>62.99</u>

5. Conclusion

In this paper, we propose the Matching Neural Network (MNN) for extreme multi-label learning. By learning two mapping functions to project features and its labels near to each other in a latent space, MNN gains robustness in learning embedding against tail labels. A loss function extends the contrastive loss with negative sampling is proposed to guide the training of MNN. Feature dropout is also applied as a regularization technique, which prevents MNN from overfitting the data and improves the prediction accuracy. Extensive experiments on benchmark datasets demonstrate the superior of MNN to leading embedding-based methods, particularly with the presence of massive tail labels in data.

References

[1] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon, 2014. Large-scale multi-label learning with missing labels. In *International conference on machine learning*, pages 593–601.

[2] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain, 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*, pages 730–738.

[3] Yukihiro Tagami, 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*, pages 455–464.

[4] Itay Evron, Edward Moroshko, and Koby Crammer, 2018. Efficient loss-based decoding on graphs for extreme classification. In *Advances in Neural Information Processing Systems*, pages 7233–7244.

[5] Ankit Jalan and Purushottam Kar, 2019. Accelerating extreme classification via adaptive feature agglomeration. *arXiv preprint arXiv:1905.11769*.

[6] Vivek Gupta, Rahul Wadbude, Nagarajan Natarajan, Harish Karnick, Prateek Jain, and Piyush Rai, 2019. Distributional semantics meets multi-label learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3747–3754.

[7] Raia Hadsell, Sumit Chopra, and Yann LeCun, 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

[8] Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang, 2009. Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*, pages 772–780.

[9] Farbound Tai and Hsuan-Tien Lin, 2012. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542.

[10] Yao-Nan Chen and Hsuan-Tien Lin, 2012. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems*, pages 1529–1537.

[11] Yi Zhang and Jeff Schneider, 2011. Multi-label output codes using canonical correlation analysis. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 873–882.

[12] Chang Xu, Dacheng Tao, and Chao Xu, 2016. Robust extreme multi-label learning. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1275–1284.

[13] Jason Weston, Ameesh Makadia, and Hector Yee, 2013. Label partitioning for sublinear ranking. In *International conference on machine learning*, pages 181–189.

[14] Yashoteja Prabhu and Manik Varma, 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*, pages 263–272.

[15] Himanshu Jain, Yashoteja Prabhu, and Manik Varma, 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*, pages 935–944.

[16] Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma, 2018. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 441–449.

[17] Wissam Siblini, Pascale Kuntz, and Frank Meyer, 2018. Craftml, an efficient clustering-based random forest for extreme multi-label learning.

[18] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Russ R Salakhutdinov, 2005. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520.

[19] Kilian Q Weinberger and Lawrence K Saul, 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2).

[20] Yiming Ying, Kaizhu Huang, and Colin Campbell, 2009. Sparse metric learning via smooth optimization. In *Advances in neural information processing systems*, pages 2214–2222.

[21] Jacob Abernethy, Francis Bach, Theodoros Evgeniou, and Jean-Philippe Vert, 2009. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10(3).

[22] David Grangier and Samy Bengio, 2008. A discriminative kernel-based approach to rank images from text queries. *IEEE transactions on pattern analysis and machine intelligence*, 30(8):1371–1384.

[23] Rong Jin, Shijun Wang, and Zhi-Hua Zhou, 2009. Learning a distance metric from multi-instance multi-label data. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 896–902. IEEE.

[24] Weiwei Liu and Ivor W Tsang, 2015. Large margin metric learning for multi-label prediction. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[25] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III, 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 1681–1691.

[26] Kush Bhatia, Kunal Dahiya, Himanshu Jain, Anshul Mittal, Yashoteja Prabhu, and Manik Varma, 2016. The extreme classification repository: Multi-label datasets and code.

[27] Bharath Hariharan, SVN Vishwanathan, and Manik Varma, 2012. Efficient max-margin multi-label classification with applications to zero-shot learning. *Machine learning*, 88(1-2):127–155.

[28] Wei Bi and James Kwok, 2013. Efficient multi-label classification with many labels. In *International Conference on Machine Learning*, pages 405–413.

[29] Jason Weston, Samy Bengio, and Nicolas Usunier, 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.