Sparse Local Embeddings for Extreme Multi-label Classification

Kush Bhatia T-KUSHB@MICROSOFT.COM
Microsoft Research, India

Himanshu Jain HIMANSHU.J689@GMAIL.COM
Indian Institute of Technology Delhi, India

Purushottam Kar PURUSHOT@CSE.IITK.AC.IN
Indian Institute of Technology Kanpur, India

Manik Varma MANIK@MICROSOFT.COM
Prateek Jain PRAJAIN@MICROSOFT.COM
Microsoft Research, India

Abstract

The objective in extreme multi-label learning is to train a classifier that can automatically tag a novel data point with the most relevant *subset* of labels from an extremely large label set. Embedding based approaches make training and prediction tractable by assuming that the training label matrix is low-rank and hence the effective number of labels can be reduced by projecting the high dimensional label vectors onto a low dimensional linear subspace. Still, leading embedding approaches have been unable to deliver high prediction accuracies or scale to large problems as the low rank assumption is violated in most real world applications.

This paper develops the SLEEC classifier to address both limitations. The main technical contribution in SLEEC is a formulation for learning a small ensemble of local distance preserving embeddings which can accurately predict infrequently occurring (tail) labels. This allows SLEEC to break free of the traditional low-rank assumption and boost classification accuracy by learning embeddings which preserve pairwise distances between only the nearest label vectors.

Preliminary work. Under review by the Indian Workshop on Machine Learning (*i*WML). Do not distribute.

1. Introduction

In this paper we develop SLEEC (Sparse Local Embeddings for Extreme Classification), an extreme multi-label classifier that can make significantly more accurate and faster predictions, as well as scale to larger problems, as compared to state-of-the-art embedding based approaches.

eXtreme Multi-label Learning (**XML**) addresses the problem of learning a classifier that can automatically tag a data point with the most relevant *subset* of labels from a large label set. For instance, there are more than a million labels (categories) on Wikipedia and one might wish to build a classifier that annotates a new article or web page with the subset of most relevant Wikipedia categories. It should be emphasized that multi-label learning is distinct from multi-class classification where the aim is to predict a single mutually exclusive label.

Challenges: XML is a hard problem that involves learning with hundreds of thousands, or even millions, of labels, features and training points. Although, some of these problems can be ameliorated using a label hierarchy, such hierarchies are unavailable in many applications (Agrawal et al., 2013; Weston et al., 2013). In this setting, an obvious baseline is thus provided by the *1-vs-All* technique which seeks to learn an an independent classifier per label. As expected, this technique is infeasible due to the prohibitive training and prediction costs given the large number of labels.

Embedding-based approaches: A natural way of overcoming the above problem is to reduce the effective number of labels. Embedding based approaches try to do so by projecting label vectors onto a low dimensional space, based on an assumption that the label matrix is low-rank. More specifically, given a set of n training points $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n\}$ with d-dimensional feature vectors $\mathbf{x}_i \in \mathbb{R}^d$ and L-dimensional label vectors $\mathbf{y}_i \in \{0,1\}^L$, state-of-the-art embedding approaches project the label vectors onto a lower $\widehat{L}$-dimensional linear subspace as $\mathbf{z}_i = \mathbf{U}\mathbf{y}_i$. Regressors are then trained to predict $\mathbf{z}_i$ as $\mathbf{V}\mathbf{x}_i$. Labels for a novel point $\mathbf{x}$ are predicted by post-processing $\mathbf{y} = \mathbf{U}^\dagger\mathbf{V}\mathbf{x}$ where $\mathbf{U}^\dagger$ is a decompression matrix which lifts the embedded label vectors back to the original label space.

Embedding methods mainly differ in the choice of their compression and decompression techniques such as compressed sensing (Hsu et al., 2009), Bloom filters (Cissé et al., 2013), SVD (Tai & Lin, 2010), landmark labels (Balasubramanian & Lebanon, 2012; Bi & Kwok, 2013), output codes (Zhang & Schneider, 2011), *etc.* The state-of-the-art LEML algorithm (Yu et al., 2014) directly optimizes for $\mathbf{U}^\dagger$, $\mathbf{V}$ using a regularized least squares objective.

Embedding approaches also have limitations – they are slow at training and prediction even for small embedding dimensions $\widehat{L}$. More importantly, the critical assumption made by embedding methods, that the training label matrix is low-rank, is violated in almost all real world applications. Figure 1(a) plots the approximation error in the label matrix as $\widehat{L}$ is varied on the WikiLSHTC data set. As is clear, even with a 500-dimensional subspace the label matrix still has 90% approximation error. This happens primarily due to the presence of hundreds of thousands of "tail" labels (Figure 1(b)) which occur in at most 5 data points each and, hence, cannot be well approximated by any linear low dimensional basis.

Tree-based approaches: Recently, tree based methods (Agrawal et al., 2013; Prabhu & Varma, 2014; Weston et al., 2013) have also become popular for XML as they enjoy significant accuracy gains over the existing embedding methods. For instance, FastXML (Prabhu & Varma, 2014) can achieve a prediction accuracy of 49% on WikiLSHTC using a 50 tree ensemble. However, using SLEEC, we are now able to extend embedding methods to outperform tree ensembles, achieving 49.8% with 2 learners and 55% with 10. Thus, SLEEC obtains the best of both worlds – achieving the highest prediction accuracies across all methods on even the most challenging data sets, as well as retaining all the benefits of embeddings and eschewing the disadvantages of large tree ensembles such as large model size and lack of theoretical understanding.

2. Method

Let $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1) \dots (\mathbf{x}_n, \mathbf{y}_n)\}$ be the given training data set, $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ be the input feature vector, $\mathbf{y}_i \in \mathcal{Y} \subseteq \{0,1\}^L$ be the corresponding label vector, and $y_{ij} = 1$ iff the j-th label is turned on for $\mathbf{x}_i$. Let $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ be the data matrix and $Y = [\mathbf{y}_1, \dots, \mathbf{y}_n]$ be the label matrix. Given $\mathcal{D}$, the goal is to learn a multi-label classifier $f : \mathbb{R}^d \to \{0,1\}^L$ that accurately predicts the label vector for a given test point. Recall that in XML settings, L is very large and is of the same order as n and d, ruling out several standard approaches such as 1-vs-All.

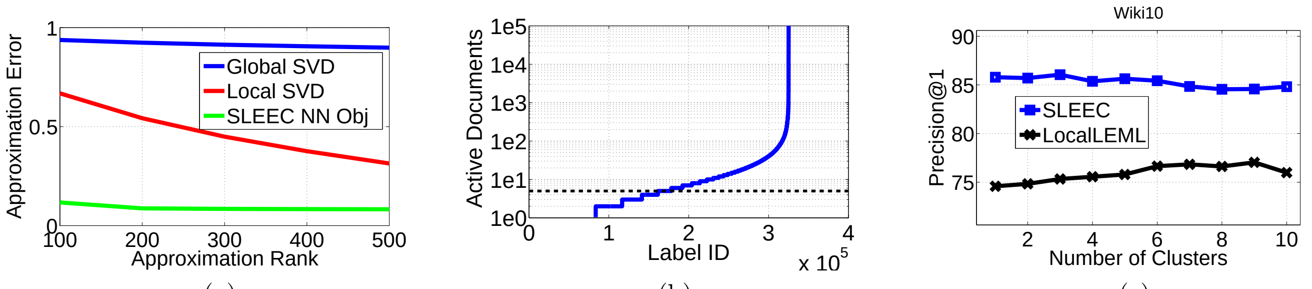
We now present our algorithm SLEEC which is designed primarily to scale efficiently for large L. Our algorithm is an embedding-style algorithm, i.e., during training we map the label vectors $\mathbf{y}_i$ to $\widehat{L}$-dimensional vectors $\mathbf{z}_i \in \mathbb{R}^{\widehat{L}}$ and learn a set of regressors $V \in \mathbb{R}^{\widehat{L} \times d}$ s.t. $\mathbf{z}_i \approx V\mathbf{x}_i, \forall i$. During the test phase, for an unseen point $\mathbf{x}$, we first compute its embedding $V\mathbf{x}$ and then perform kNN over the set $[Vx_1, Vx_2, \dots, Vx_n]$. To scale our algorithm, we perform a clustering of all the training points and apply the above mentioned procedures in each of the cluster separately. Below, we first discuss our method to compute the embeddings $\mathbf{z}_i$s and the regressors V. Section 2.2 then discusses our approach for scaling the method to large data sets.

2.1. Learning Embeddings

As mentioned earlier, our approach is motivated by the fact that a typical real-world data set tends to have a large number of tail labels that ensure that the label matrix Y cannot be well-approximated using a low-dimensional linear subspace (see Figure 1). However, Y can still be accurately modelled using a low-dimensional non-linear manifold. That is, instead of preserving distances (or inner products) of a given label vector to all the training points, we attempt to preserve the distance to only a few nearest neighbors. That is, we wish to find a $\widehat{L}$-dimensional embedding matrix $Z = [\mathbf{z}_1, \dots, \mathbf{z}_n] \in \mathbb{R}^{\widehat{L} \times n}$ which minimizes the following objective:

$$\min_{Z \in \mathbb{R}^{\widehat{L} \times n}} \|P_\Omega(Y^TY) - P_\Omega(Z^TZ)\|_F^2 + \lambda\|Z\|_1, \qquad (1)$$

where the index set Ω denotes the set of neighbors that we wish to preserve, i.e., $(i,j) \in \Omega$ iff $j \in \mathcal{N}_i$. $\mathcal{N}_i$ denotes a set of nearest neighbors of i. We select $\mathcal{N}_i = \arg\max_{S, |S| \le \alpha \cdot n} \sum_{j \in S}(\mathbf{y}_i^T\mathbf{y}_j)$, which is the set of $\alpha \cdot n$ points with the largest inner products with $\mathbf{y}_i$. $|\mathcal{N}|$ is always chosen large enough so that distances (inner products) to a few far away points are also preserved while optimizing for our objective function. This prohibits non-neighboring points from en-



(a) (b) (c)

Figure 1. (a) error $\|Y - Y_{\widehat{L}}\|_F^2/\|Y\|_F^2$ in approximating the label matrix Y. Global SVD denotes the error incurred by computing the rank $\widehat{L}$ SVD of Y. Local SVD computes rank $\widehat{L}$ SVD of Y within each cluster. SLEEC NN objective denotes SLEEC's objective function. Global SVD incurs 90% error and the error is decreasing at most linearly as well. (b) shows the number of documents in which each label is present for the WikiLSHTC data set. There are about $300K$ labels which are present in < 5 documents lending it a 'heavy tailed' distribution. (c) shows Precision@1 accuracy of SLEEC and localLEML on the Wiki-10 data set as we vary the number of clusters.

tering the immediate neighborhood of any given point. $P_\Omega : \mathbb{R}^{n\times n} \rightarrow \mathbb{R}^{n\times n}$ is defined as:

$$(P_\Omega(Y^TY))_{ij} = \begin{cases} \langle \mathbf{y}_i, \mathbf{y}_j \rangle, & \text{if } (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Also, we add L_1 regularization, $\|Z\|_1 = \sum_i \|\mathbf{z}_i\|_1$, to the objective function to obtain sparse embeddings. Sparse embeddings have three key advantages: a) they reduce prediction time, b) reduce the size of the model, and c) avoid overfitting. Now, given the embeddings $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_n] \in \mathbb{R}^{\widehat{L}\times n}$, we wish to learn a multi-regression model to predict the embeddings Z using the input features. That is, we require that $Z \approx VX$ where $V \in \mathbb{R}^{\widehat{L}\times d}$. Combining the two formulations and adding an L_2-regularization for V, we get:

$$\min_{V\in\mathbb{R}^{\widehat{L}\times d}} \|P_\Omega(Y^TY) - P_\Omega(X^TV^TVX)\|_F^2 + \lambda\|V\|_F^2 + \mu\|VX\|_1. \tag{3}$$

Note that the above problem formulation is somewhat similar to a few existing methods for non-linear dimensionality reduction that also seek to preserve distances to a few near neighbors (Weinberger & Saul, 2006). However, in contrast to our approach, these methods do not have a direct out of sample generalization, do not scale well to large-scale data sets, and lack rigorous generalization error bounds. Details of the algorithm and generalization error analysis can be found in full paper (Bhatia et al., 2015).

2.2. Scaling to Large-scale Data sets

For large-scale data sets, with millions of training points, finding the neighborhood set Ω might become computationally very expensive. Hence, to scale to such large data sets, SLEEC clusters the given datapoints into smaller local region. Several text-based data sets indeed reveal that there exist small local regions in the feature-space where the number of points as well as the number of labels is reasonably small. Hence, we can train our embedding method over such local regions without significantly sacrificing overall accuracy.

Owing to the curse-of-dimensionality, clustering turns out to be quite unstable for data sets with large d and in many cases leads to some drop in prediction accuracy. To safeguard against such instability, we use an ensemble of models generated using different sets of clusters. We use different initialization points in our clustering procedure to obtain different sets of clusters.

3. Experiments

Experiments were carried out on some of the largest XML benchmark data sets demonstrating that SLEEC could achieve significantly higher prediction accuracies as compared to the state-of-the-art. It is also demonstrated that SLEEC could be faster at training and prediction than leading embedding techniques such as LEML.

Data sets: Experiments were carried out on multi-label data sets including Ads1M (1M labels), Amazon (670K labels), WikiLSHTC (320K labels), Delicious-Large (200K labels) and Wiki10 (30K labels). All the data sets are publically available except Ads1M which is proprietary and is included here to test the scaling capabilities of SLEEC.

Baseline algorithms: This paper's primary focus is on comparing SLEEC to state-of-the-art methods which can scale to the large data sets such as embedding based LEML (Yu et al., 2014) and tree based FastXML (Prabhu & Varma, 2014) and LPSR (Weston et al., 2013).

Evaluation Metrics: We evaluated algorithms using metrics that have been widely adopted for XML and ranking tasks. Precision at k (P@k) is one such metric that counts the fraction of correct predictions in the top k scoring labels in $\hat{\mathbf{y}}$

Results: Table 1 compares SLEEC's prediction accuracy, in terms of P@k (k= $\{1, 3, 5\}$), to all the leading methods that could be trained on five such data sets. SLEEC could improve over the leading embedding method, LEML, by as much as 35% and 15% in terms of P@1 and P@5 on WikiLSHTC. Similarly, SLEEC outperformed LEML by 27% and 22% in terms of P@1 and P@5 on the Amazon data set which also has many tail labels. The gains on the other data sets are consistent, but smaller, as the tail label problem is not so acute. SLEEC also outperforms the leading tree method, FastXML, by 6% in terms of both P@1 and P@5 on WikiLSHTC and Wiki10 respectively. This demonstrates the superiority of SLEEC's overall pipeline constructed using local distance preserving embeddings followed by kNN classification.

SLEEC also has better scaling properties as compared to all other embedding methods. In particular, apart from LEML, no other embedding approach could scale to the large data sets and, even LEML could not scale to Ads1M with a million labels. In contrast, a single SLEEC learner could be learnt on WikiLSHTC in 4 hours on a single core and already gave $\sim$ 20% improvement in P@1 over LEML. In fact, SLEEC's training time on WikiLSHTC was comparable to that of tree based FastXML. FastXML trains 50 trees in 7 hours on a single core to achieve a P@1 of 49.37% whereas SLEEC could achieve 49.98% by training 2 learners in 8 hours. Similarly, SLEEC's training time on Ads1M was 6 hours per learner on a single core.

SLEEC's predictions could also be up to 300 times faster than LEMLs. For instance, on WikiLSHTC, SLEEC made predictions in 8 milliseconds per test point as compared to LEML's 279. SLEEC therefore brings the prediction time of embedding methods to be much closer to that of tree based methods (FastXML took 0.5 milliseconds per test point on WikiLSHTC) and within the acceptable limit of most real world applications.

Dual Submissions

This work was published in conference proceedings of Neural Information Processing Systems 2015 (NIPS 2015), held in Montreal, Canada in Dec-2015. To read the full version submitted to NIPS, click here.To visit the conference website, click here.

Table 1. **Precision Accuracies** Our proposed method SLEEC is as much as 35% more accurate in terms of P@1 and 22% in terms of P@5 than LEML, a leading embedding method. SLEEC is also 6% more accurate (w.r.t. P@1 and P@5) than FastXML, a state-of-the-art tree method. '-' indicates LEML could not be run with the standard resources.

Data set		SLEEC	LEML	FastXML	LPSR-NB
Wiki10	P@1	**85.54**	73.50	82.56	72.71
	P@3	**73.59**	62.38	66.67	58.51
	P@5	**63.10**	54.30	56.70	49.40
Delicious-Large	P@1	**47.03**	40.30	42.81	18.59
	P@3	**41.67**	37.76	38.76	15.43
	P@5	**38.88**	36.66	36.34	14.07
WikiLSHTC	P@1	**55.57**	19.82	49.35	27.43
	P@3	**33.84**	11.43	32.69	16.38
	P@5	**24.07**	8.39	24.03	12.01
Amazon	P@1	**35.05**	8.13	33.36	28.65
	P@3	**31.25**	6.83	29.30	24.88
	P@5	**28.56**	6.03	26.12	22.37
Ads-1m	P@1	21.84	-	**23.11**	17.08
	P@3	**14.30**	-	13.86	11.38
	P@5	**11.01**	-	10.12	8.83

References

Agrawal, R., Gupta, A., Prabhu, Y., and Varma, M. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, pp. 13–24, 2013.

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

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

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

Cissé, M., Usunier, N., Artières, T., and Gallinari, P. Robust bloom filters for large multilabel classification tasks. In *NIPS*, pp. 1851–1859, 2013.

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

Prabhu, Y. and Varma, M. FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, pp. 263–272, 2014. doi: 10.1145/2623330.2623651.

Tai, F. and Lin, H.-T. Multi-label classification with principal label space transformation. In *Workshop proceedings of learning from multi-label data*, 2010.

Weinberger, K. Q. and Saul, L. K. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, pp. 1683–1686, 2006.

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

Yu, H.-F., Jain, P., Kar, P., and Dhillon, I. S. Large-scale multi-label learning with missing labels. *ICML*, 2014.

Zhang, Y. and Schneider, J. G. Multi-label output codes using canonical correlation analysis. In *AISTATS*, pp. 873–882, 2011.