Locally Non-linear Embeddings for Extreme Multi-label Learning

Kush Bhatia*   Himanshu Jain#   Purushottam Kar*   Prateek Jain*

Manik Varma*

*Microsoft Research, Bangalore, INDIA

{t-kushb, t-purkar, prajain, manik}@microsoft.com

#Indian Institute of Technology, Delhi, INDIA

himanshu.j689@gmail.com

July 10, 2015

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 X1 classifier to address both limitations. The main technical contribution in X1 is a formulation for learning a small ensemble of local distance preserving embeddings which can accurately predict infrequently occurring (tail) labels. This allows X1 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.

We conducted extensive experiments on several real-world as well as benchmark data sets and compared our method against state-of-the-art methods for extreme multi-label classification. Experiments reveal that X1 can make significantly more accurate predictions then the state-of-the-art methods including both embeddings (by as much as 35%) as well as trees (by as much as 6%). X1 can also scale efficiently to data sets with a million labels which are beyond the pale of leading embedding methods.

1   Introduction

Our objective in this paper is to develop an extreme multi-label classifier, referred to as X1, which 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 classification 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 labels. It should be emphasized that multi-label learning is distinct from multi-class classification which aims to predict a single mutually exclusive label.

Extreme multi-label learning is a challenging research problem as one needs to simultaneously deal with hundreds of thousands, or even millions, of labels, features and training points. An obvious baseline is provided by the 1-vs-All technique where an independent classifier is learnt per label. Regrettably, this technique is infeasible due to the prohibitive training and prediction costs. These problems could be ameliorated if a label hierarchy was provided. Unfortunately, such a hierarchy is unavailable in many applications [1, 2].

Embedding based approaches make training and prediction tractable by reducing the effective number of labels. 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

1

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 $\mathbf{z}_i = \mathbf{U}\mathbf{y}_i$, based on a low-rank assumption. 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 compression and decompression techniques such as compressed sensing [3], Bloom filters [4], SVD [5], landmark labels [6, 7], output codes [8], *etc*. The state-of-the-art LEML algorithm [9] directly optimizes for $\mathbf{U}^\dagger$, $\mathbf{V}$ using the following objective: $\mathrm{argmin}_{\mathbf{U}^\dagger,\mathbf{V}} Tr(\mathbf{U}^{\dagger\top}\mathbf{U}^\dagger) + Tr(\mathbf{V}^\top\mathbf{V}) + 2C\sum_{i=1}^{n}\|y_i - \mathbf{U}^\dagger\mathbf{V}\mathbf{x}_i\|^2$.

Embedding approaches have many advantages including simplicity, ease of implementation, strong theoretical foundations, the ability to handle label correlations, the ability to adapt to online and incremental scenarios, *etc*. Consequently, embeddings have proved to be the most popular approach for tackling extreme multi-label problems [6, 7, 10, 4, 11, 3, 12, 9, 5, 13, 8, 14].

Embedding approaches also have some limitations. They are slow at training and prediction even for a small embedding dimension $\widehat{L}$. For instance, on WikiLSHTC [15, 16], a Wikipedia based challenge data set, LEML with $\widehat{L} = 500$ took 22 hours for training even with early termination while prediction took nearly 300 milliseconds per test point. In fact, for WikiLSHTC and other text applications with $\widehat{d}$-sparse feature vectors, LEML's prediction time $\Omega(\widehat{L}(\widehat{d} + L))$ can be an order of magnitude more than even 1-vs-All's prediction time $O(\widehat{d}L)$ (as $\widehat{d} = 42 \ll \widehat{L} = 500$ for WikiLSHTC).

More importantly, the critical assumption made by most 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 from 100 to 500 on the WikiLSHTC data set. As can be seen, even with a 500-dimensional subspace the label matrix still has 90% approximation error. We observe that this limitation arises primarily due to the presence of hundreds of thousands of "tail" labels (see Figure 1(b)) which occur in at most 5 data points each and, hence, cannot be well approximated by any linear low dimensional basis.

This paper develops the X1 algorithm which extends embedding methods in multiple ways to address these limitations. First, instead of projecting onto a linear low-rank subspace, X1 learns embeddings which non-linearly capture label correlations by preserving the pairwise distances between only the closest (rather than all) label vectors, i. e. $d(\mathbf{z}_i, \mathbf{z}_j) \approx d(\mathbf{y}_i, \mathbf{y}_j)$ if $i \in \mathrm{kNN}(j)$ where d is a distance metric. Regressors $\mathbf{V}$ are trained to predict $\mathbf{z}_i = \mathbf{V}\mathbf{x}_i$. During prediction, rather than using a decompression matrix, X1 uses a k-nearest neighbour (kNN) classifier in the learnt embedding space, thus leveraging the fact that nearest neighbour distances have been preserved during training. Thus, for a novel point $\mathbf{x}$, the predicted label vector is obtained as $\mathbf{y} = \sum_{i:\mathbf{V}\mathbf{x}_i\in\mathrm{kNN}(\mathbf{V}\mathbf{x})} \mathbf{y}_i$. Our use of the kNN classifier is also motivated by the observation that kNN outperforms discriminative methods in acutely low training data regimes [17] as in the case of tail labels.

The superiority of X1's proposed embeddings over traditional low-rank embeddings can be determined in two ways. First, as can be seen in Figure 1, the relative approximation error in learning X1's embeddings is significantly smaller as compared to the low-rank approximation error. Second, X1 can improve over state-of-the-art embedding methods' prediction accuracy by as much as 35% (absolute) on the challenging WikiLSHTC data set. X1 also significantly outperforms methods such as WSABIE [13] which also use kNN classification in the embedding space but learn their embeddings using the traditional low-rank assumption.

However, kNN classifiers are known to be slow at prediction. X1 therefore clusters the training data into C clusters, learns a separate embedding per cluster and performs kNN classification within the test point's cluster alone. This reduces X1's prediction costs to $O(\widehat{d}C + \widehat{d}\widehat{L} + N_C\widehat{L})$ for determining the cluster membership of the test point, embedding it and then performing kNN classification respectively, where N_C is the number of points in the cluster to which the test point was assigned. X1 can therefore be more than two orders of magnitude faster at prediction than LEML and other embedding methods on the WikiLSHTC data set where $C = 300, N_C \lesssim 13K, \widehat{L}_{\mathbf{X1}} = 50$ and $\widehat{D} = 42$. Clustering can also reduce X1's training time by almost a factor of C. This allows X1 to scale to the Ads1M data set involving a million labels which is beyond the pale of leading embedding methods.

Of course, clustering is not a significant technical innovation in itself, and could easily have been applied to traditional embedding approaches. However, as our results demonstrate, state-of-the-art methods such as LEML do not benefit much from clustering. Clustered LEML's prediction accuracy continues to lag behind X1's by 14% on WikiLSHTC and the training time on Ads1M continues to be prohibitively large.
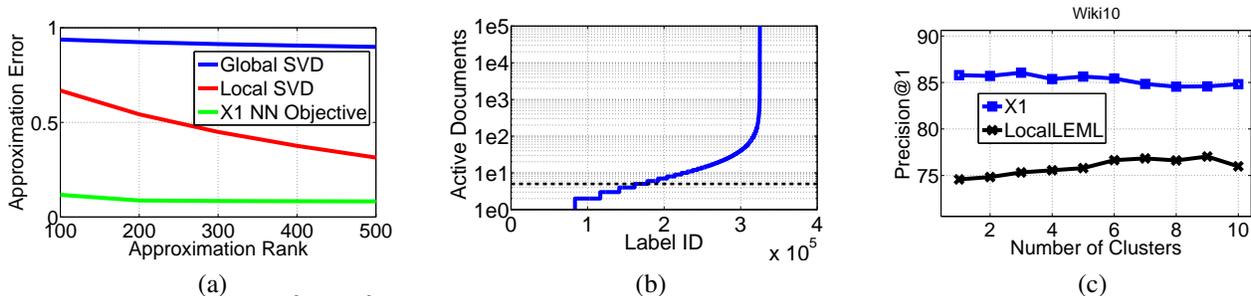
Figure 1: (a) error $\|Y - Y_{\widehat{L}}\|_F^2 / \|Y\|_F^2$ in approximating the label matrix Y. Global SVD denotes the error incurred by computing the rank $\widehat{L}$ SVD of Y. Local SVD computes rank $\widehat{L}$ SVD of Y within each cluster. X1 NN objective denotes X1's objective function. Global 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 X1 and localLEML on the Wiki-10 data set as we vary the number of clusters.

The main limitation of clustering is that it can be unstable in high dimensions. X1 compensates by learning a small ensemble where each individual learner is generated by a different random clustering. This was empirically found to help tackle instabilities of clustering and significantly boost prediction accuracy with only linear increases in training and prediction time. For instance, on WikiLSHTC, X1's prediction accuracy was 56% with an 8 millisecond prediction time whereas LEML could only manage 20% accuracy while taking 300 milliseconds for prediction per test point.

Recently, tree based methods [1, 15, 2] have also become popular for extreme multi-label learning as they enjoy significant accuracy gains over the existing embedding methods. For instance, FastXML [15] can achieve a prediction accuracy of 49% on WikiLSHTC using a 50 tree ensemble. However, X1 is now able to extend embedding methods to outperform tree ensembles, achieving 49.8% with 2 learners and 55% with 10. Thus, by learning local distance preserving embeddings, X1 can now obtain the best of both worlds. In particular, X1 can achieve the highest prediction accuracies across all methods on even the most challenging data sets while retaining all the benefits of embeddings and eschewing the disadvantages of large tree ensembles such as large model size and lack of theoretical understanding.

Our contributions in this paper are: First, we identify that the low-rank assumption made by most embedding methods is violated in the real world and that local distance preserving embeddings can offer a superior alternative. Second, we propose a novel formulation for learning such embeddings and show that it has sound theoretical properties. In particular, we prove that X1 consistently preserves nearest neighbours in the label space and hence learns good quality embeddings. Third, we build an efficient pipeline for training and prediction which can be orders of magnitude faster than state-of-the-art embedding methods while being significantly more accurate as well.

2 Method

Let $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1) \ldots (\mathbf{x}_n, \mathbf{y}_n)\}$ 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 $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ be the data matrix and $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$ 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 extreme multi-label 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 X1 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 $[V x_1, V x_2, \ldots, V x_n]$. 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.

Algorithm 1 X1: Train Algorithm
***
Require: $\mathcal{D} = \{(x_1, y_1) \dots (x_n, y_n)\}$, embedding dimensionality: $\widehat{L}$, no. of neighbors: $\bar{n}$, no. of clusters: C, regularization parameter: λ, μ, L1 smoothing parameter ρ
1: Partition X into $Q^1, .., Q^C$ using k-means
2: **for** each partition Q^j **do**
3:      Form Ω using $\bar{n}$ nearest neighbors of each label vector $\mathbf{y}_i \in Q^j$
4:      $[U \; \Sigma] \leftarrow \text{SVP}(P_\Omega(Y^j Y^{j^T}), \; \widehat{L})$
5:      $Z^j \leftarrow U \Sigma^{\frac{1}{2}}$
6:      $V^j \leftarrow ADMM(X^j, Z^j, \lambda, \; \mu, \; \rho)$
7:      $Z^j = V^j X^j$
8: **end for**
9: **Output**: $\{(Q^1, V^1, Z^1), \dots, (Q^C, V^C, Z^C\}$

Algorithm 2 X1: Test Algorithm
***
Require: Test point: $\mathbf{x}$, no. of NN: $\bar{n}$, no. of desired labels: p
1: Q_τ: partition closest to $\mathbf{x}$
2: $\mathbf{z} \leftarrow V^\tau \mathbf{x}$
3: $\mathcal{N}_z \leftarrow \bar{n}$ nearest neighbors of z in Z^τ
4: $P_x \leftarrow$ empirical label dist. for points $\in \mathcal{N}_z$
5: $y_{pred} \leftarrow Top_p(P_x)$

Sub-routine 3 X1: SVP
***
Require: Observations: G, index set: Ω, dimensionality: $\widehat{L}$
1: $M_1 := 0, \eta = 1$
2: **repeat**
3:      $\widehat{M} \leftarrow M + \eta(G - P_\Omega(M))$
4:      $[U \; \Sigma] \leftarrow \text{Top-EigenDecomp}(\widehat{M}, \; \widehat{L})$
5:      $\Sigma_{ii} \leftarrow \max(0, \Sigma_{ii}), \forall i$
6:      $M \leftarrow U \cdot \Sigma \cdot U^T$
7: **until** Convergence
8: **Output**: U, Σ

Sub-routine 4 X1: ADMM
***
Require: Data Matrix : X, Embeddings : Z, Regularization Parameter : λ, μ, Smoothing Parameter : ρ
1: $\beta := 0, \alpha := 0$
2: **repeat**
3:      $Q \leftarrow (Z + \rho(\alpha - \beta))X^\top$
4:      $V \leftarrow Q(XX^\top(1+\rho) + \lambda I)^{-1}$
5:      $\alpha \leftarrow (VX + \beta)$
6:      $\alpha_i = \text{sign}(\alpha_i) \cdot \max(0, |\alpha_i| - \frac{\mu}{\rho}), \forall i$
7:      $\beta \leftarrow \beta + VX - alpha$
8: **until** Convergence
9: **Output**: V

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 modeled 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:

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

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 $\mathbf{y}_i$. $P_\Omega : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$ is defined as:

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

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 $Z = [\mathbf{z}_1, \dots, \mathbf{z}_n] \in \mathbb{R}^{\widehat{L} \times n}$, 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:

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

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 [18, 19]. However, in contrast to our approach,

4

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.

Optimization: We first note that optimizing (3) is a significant challenge as the objective function is non-convex as well as non-differentiable. Furthermore, our goal is to perform optimization for data sets where $L, n, d \gg 100,000$. To this end, we divide the optimization into two phases. We first learn embeddings $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_n]$ and then learn regressors V in the second stage. That is, Z is obtained by directly solving (1) but without the L_1 penalty term:

$$\min_{Z, Z \in \mathbb{R}^{\widehat{L} \times n}} \|P_\Omega(Y^T Y) - P_\Omega(Z^T Z)\|_F^2 \equiv \min_{\substack{M \succeq 0, \\ rank(M) \leq \widehat{L}}} \|P_\Omega(Y^T Y) - P_\Omega(M)\|_F^2, \tag{4}$$

where $M = Z^T Z$. Next, V is obtained by solving the following problem:

$$\min_{V \in \mathbb{R}^{\widehat{L} \times d}} \|Z - VX\|_F^2 + \lambda \|V\|_F^2 + \mu \|VX\|_1. \tag{5}$$

Note that the Z matrix obtained using (4) need not be sparse. However, we store and use VX as our embeddings, so that sparsity is still maintained.

Optimizing (4): Note that even the simplified problem (4) is an instance of the popular low-rank matrix completion problem and is known to be NP-hard in general. The main challenge arises due to the non-convex rank constraint on M. However, using the Singular Value Projection (SVP) method [20], a popular matrix completion method, we can guarantee convergence to a local minima.

SVP is a simple projected gradient descent method where the projection is onto the set of low-rank matrices. That is, the t-th step update for SVP is given by:

$$M_{t+1} = P_{\widehat{L}}(M_t + \eta P_\Omega(Y^T Y - M_t)), \tag{6}$$

where M_t is the t-th step iterate, $\eta > 0$ is the step-size, and $P_{\widehat{L}}(M)$ is the projection of M onto the set of rank-$\widehat{L}$ positive semi-definite definite (PSD) matrices. Note that while the set of rank-$\widehat{L}$ PSD matrices is non-convex, we can still project onto this set efficiently using the eigenvalue decomposition of M. That is, if $M = U_M \Lambda_M U_M^T$ be the eigenvalue decomposition of M. Then,

$$P_{\widehat{L}}(M) = U_M(1:r) \cdot \Lambda_M(1:r) \cdot U_M(1:r)^T,$$

where $r = \min(\widehat{L}, \widehat{L}_M^+)$ and $\widehat{L}_M^+$ is the number of positive eigenvalues of M. $\Lambda_M(1:r)$ denotes the top-r eigenvalues of M and $U_M(1:r)$ denotes the corresponding eigenvectors.

While the above update restricts the rank of all intermediate iterates M_t to be at most $\widehat{L}$, computing rank-$\widehat{L}$ eigenvalue decomposition can still be fairly expensive for large n. However, by using special structure in the update (6), one can significantly reduce eigenvalue decomposition's computation complexity as well. In general, the eigenvalue decomposition can be computed in time $O(\widehat{L}\zeta)$ where ζ is the time complexity of computing a matrix-vector product. Now, for SVP update (6), matrix has special structure of $\hat{M} = M_t + \eta P_\Omega(Y^T Y - M_t)$. Hence $\zeta = O(n\widehat{L} + n\bar{n})$ where $\bar{n} = |\Omega|/n^2$ is the average number of neighbors preserved by X1. Hence, the per-iteration time complexity reduces to $O(n\widehat{L}^2 + n\widehat{L}\bar{n})$ which is linear in n, assuming $\bar{n}$ is nearly constant.

Optimizing (5): (5) contains an L_1 term which makes the problem non-smooth. Moreover, as the L_1 term involves both V and X, we cannot directly apply the standard prox-function based algorithms. Instead, we use the ADMM method to optimize (5). See Sub-routine 4 for the updates and [21] for a detailed derivation of the algorithm.

Generalization Error Analysis: Let $\mathcal{P}$ be a fixed (but unknown) distribution over $\mathcal{X} \times \mathcal{Y}$. Let each training point $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}$ be sampled i.i.d. from $\mathcal{P}$. Then, the goal of our non-linear embedding method (3) is to learn an embedding matrix $A = V^T V$ that preserves nearest neighbors (in terms of label distance/intersection) of any $(\mathbf{x}, \mathbf{y}) \sim \mathcal{P}$. The above requirements can be formulated as the following stochastic optimization problem:

$$\min_{\substack{A \succeq 0 \\ rank(A) \leq k}} \mathcal{L}(A) = \mathbb{E}_{(\mathbf{x}, \mathbf{y}), (\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}) \sim \mathcal{P}} \ell(A; (\mathbf{x}, \mathbf{y}), (\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})), \tag{7}$$

5

where the loss function $\ell(A; (\mathbf{x}, \mathbf{y}), (\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})) = g(\langle \widetilde{\mathbf{y}}, \mathbf{y} \rangle)(\langle \widetilde{\mathbf{y}}, \mathbf{y} \rangle - \widetilde{\mathbf{x}}^T A \mathbf{x})^2$, and $g(\langle \widetilde{\mathbf{y}}, \mathbf{y} \rangle) = \mathbb{I}[\langle \widetilde{\mathbf{y}}, \mathbf{y} \rangle \geq \tau]$, where $\mathbb{I}[\cdot]$ is the indicator function. Hence, a loss is incurred only if $\mathbf{y}$ and $\tilde{\mathbf{y}}$ have a large inner product. For an appropriate selection of the neighborhood selection operator Ω, (3) indeed minimizes a regularized empirical estimate of the loss function (7), i.e., it is a regularized ERM w.r.t. (7).

We now show that the optimal solution $\widehat{A}$ to (3) indeed minimizes the loss (7) upto an additive approximation error. The existing techniques for analyzing excess risk in stochastic optimization require the empirical loss function to be decomposable over the training set, and as such do not apply to (3) which contains loss-terms with two training points. Still, using techniques from the AUC maximization literature [22], we can provide interesting excess risk bounds for Problem (7).

Theorem 1. *With probability at least $1 - \delta$ over the sampling of the dataset $\mathcal{D}$, the solution $\hat{A}$ to the optimization problem* (3) *satisfies*

$$\mathcal{L}(\hat{A}) \leq \inf_{A^* \in \mathcal{A}} \left\{ \mathcal{L}(A^*) + \overbrace{C\left(\bar{L}^2 + \left(r^2 + \|A^*\|_F^2\right) R^4\right) \sqrt{\frac{1}{n} \log \frac{1}{\delta}}}^{E\text{-}Risk(n)} \right\},$$

where $\hat{A}$ is the minimizer of (3)*, $r = \frac{\bar{L}}{\lambda}$ and $\mathcal{A} := \left\{ A \in \mathbb{R}^{d \times d} : A \succeq 0, rank(A) \leq \widehat{L} \right\}$.*

See Appendix A for a proof of the result. Note that the generalization error bound is *independent* of both d and L, which is critical for extreme multi-label classification problems with large d, L. In fact, the error bound is only dependent on $\bar{L} \ll L$, which is the average number of positive labels per data point. Moreover, our bound also provides a way to compute best regularization parameter λ that minimizes the error bound. However, in practice, we set λ to be a fixed constant.

Theorem 1 only preserves the *population* neighbors of a test point. Theorem 7, given in Appendix A, extends Theorem 1 to ensure that the neighbors in the *training* set are also preserved. We would also like to stress that our excess risk bound is universal and hence holds even if $\hat{A}$ does not minimize (3), i.e., $\mathcal{L}(\hat{A}) \leq \mathcal{L}(A^*) + \text{E-Risk}(n) + (\mathcal{L}(\hat{A}) - \mathcal{L}(\hat{(A^*)}))$, where E-Risk$(n)$ is given in Theorem 1.

2.2 Scaling to Large-scale Data sets

For large-scale data sets, one might require the embedding dimension $\widehat{L}$ to be fairly large (say a few hundreds) which might make computing the updates (6) infeasible. Hence, to scale to such large data sets, X1 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.

We would like to stress that despite clustering datapoints in homogeneous regions, the label matrix of any given cluster is still not close to low-rank. Hence, applying a state-of-the-art linear embedding method, such as LEML, to each cluster is still significantly less accurate when compared to our method (see Figure 1). Naturally, one can cluster the data set into an extremely large number of regions, so that eventually the label matrix is low-rank in each cluster. However, increasing the number of clusters beyond a certain limit might decrease accuracy as the error incurred during the cluster assignment phase itself might nullify the gain in accuracy due to better embeddings. Figure 1 illustrates this phenomenon where increasing the number of clusters beyond a certain limit in fact decreases accuracy of LEML.

Algorithm 1 provides a pseudo-code of our training algorithm. We first cluster the datapoints into C partitions. Then, for each partition we learn a set of embeddings using Sub-routine 3 and then compute the regression parameters $V^\tau, 1 \leq \tau \leq C$ using Sub-routine 4. For a given test point $\mathbf{x}$, we first find out the appropriate cluster τ. Then, we find the embedding $\mathbf{z} = V^\tau \mathbf{x}$. The label vector is then predicted using k-NN in the embedding space. See Algorithm 2 for more details.

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. Our empirical results demonstrate that using such ensembles leads to significant increase in accuracy of X1 (see Figure 2) and also leads to stable solutions with small variance (see Table 4).
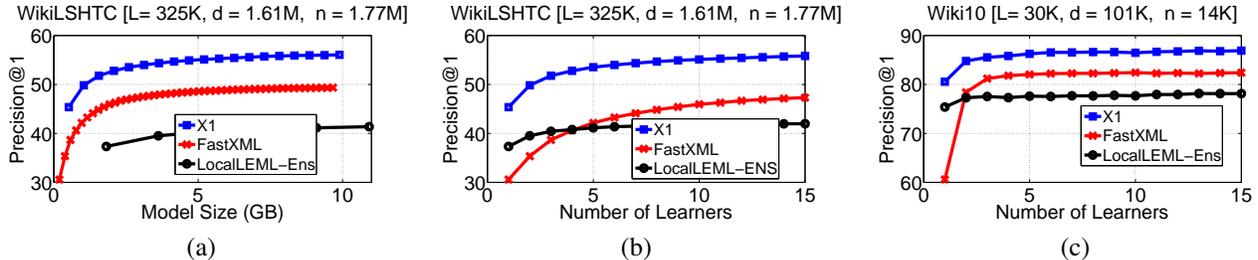
6

Figure 2: Variation in Precision@1 accuracy with model size and the number of learners on large-scale data sets. Clearly, X1 achieves better accuracy than FastXML and LocalLEML-Ensemble at every point of the curve. For WikiLSTHC, X1 with a single learner is more accurate than LocalLEML-Ensemble with even 15 learners. Similarly, X1 with 2 learners achieves more accuracy than FastXML with 50 learners.

3 Experiments

Experiments were carried out on some of the largest extreme multi-label benchmark data sets demonstrating that X1 could achieve significantly higher prediction accuracies as compared to the state-of-the-art. It is also demonstrated that X1 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 [15] (1M labels), Amazon [23] (670K labels), WikiLSHTC (320K labels), DeliciousLarge [24] (200K labels) and Wiki10 [25] (30K labels). All the data sets are publically available except Ads1M which is proprietary and is included here to test the scaling capabilities of X1.

Unfortunately, most of the existing embedding techniques do not scale to such large data sets. We therefore also present comparisons on publically available small data sets such as BibTeX [26], MediaMill [27], Delicious [28] and EURLex [29]. Table 2 in the supplementary material lists the statistics of each of these data sets.

Baseline algorithms: This paper's primary focus is on comparing X1 to state-of-the-art methods which can scale to the large data sets such as embedding based LEML [9] and tree based FastXML [15] and LPSR [2]. Naïve Bayes was used as the base classifier in LPSR as was done in [15]. Techniques such as CS [3], CPLST [30], ML-CSSP [7], 1-vs-All [31] could only be trained on the small data sets given standard resources. Comparisons between X1 and such techniques are therefore presented in the supplementary material. The implementation for LEML and FastXML was provided by the authors. We implemented the remaining algorithms and ensured that the published results could be reproduced and were verified by the authors wherever possible.

Hyper-parameters: Most of X1's hyper-parameters were kept fixed including the number of clusters in a learner $\left(\lfloor N_{\text{Train}}/6000 \rfloor\right)$, embedding dimension (100 for the small data sets and 50 for the large), number of learners in the ensemble (15), and the parameters used for optimizing (3). The remaining two hyper-parameters, the k in kNN and the number of neighbours considered during SVP, were both set by limited validation on a validation set.

The hyper-parameters for all the other algorithms were set using fine grained validation on each data set so as to achieve the highest possible prediction accuracy for each method. In addition, all the embedding methods were allowed a much larger embedding dimension ($0.8L$) than X1 (100) to give them as much opportunity as possible to outperform X1.

Evaluation Metric: Precision at k (P@k)has been widely adopted as the metric of choice for evaluating extreme multi-label algorithms [1, 3, 15, 13, 2, 9]. This is motivated by real world application scenarios such as tagging and recommendation. Formally, the precision at k for a predicted score vector $\hat{\mathbf{y}} \in \mathcal{R}^L$ is the fraction of correct positive predictions in the top k scores of $\hat{\mathbf{y}}$.

Results on large data sets with more than 100K labels:. Table 1a compares X1'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. X1 could improve over the leading embedding method, LEML, by as much as 35% and 15% in terms of P@1 and P@5 on the WikiLSHTC data set. Similarly, X1 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. X1 could also outperform 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 X1's overall pipeline constructed using local distance preserving embeddings followed by kNN classification.

7

Table 1: **Precision Accuracies** (a) Large-scale data sets : Our proposed method X1 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. Other embedding based methods do not scale to the large-scale data sets; we compare against them on small-scale data sets in Table 3. X1 is also 6% more accurate (w.r.t. P@1 and P@5) than FXML, a state-of-the-art tree method. '-' indicates LEML could not be run with the standard resources. (b) Small-scale data sets : X1 consistently outperforms state of the art approaches. WSABIE, which also uses kNN classifier on its embeddings is significantly less accurate than X1 on all the data sets, showing the superiority of our embedding learning algorithm.

(a)

Data set		X1	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	27.91
	P@3	**33.84**	11.43	32.69	16.04
	P@5	**24.07**	8.39	24.03	11.57
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

(b)

Data set		X1	LEML	FastXML	WSABIE	OneVsAll
BibTex	P@1	**65.57**	62.53	63.73	54.77	61.83
	P@3	**40.02**	38.4	39.00	32.38	36.44
	P@5	**29.30**	28.21	28.54	23.98	26.46
Delicious	P@1	68.42	65.66	**69.44**	64.12	65.01
	P@3	61.83	60.54	**63.62**	58.13	58.90
	P@5	56.80	56.08	**59.10**	53.64	53.26
MediaMill	P@1	**87.09**	84.00	84.24	81.29	83.57
	P@3	**72.44**	67.19	67.39	64.74	65.50
	P@5	**58.45**	52.80	53.14	49.82	48.57
EurLEX	P@1	**80.17**	61.28	68.69	70.87	74.96
	P@3	**65.39**	48.66	57.73	56.62	62.92
	P@5	**53.75**	39.91	48.00	46.2	53.42

X1 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 X1 learner could be learnt on WikiLSHTC in 4 hours on a single core and already gave ∼ 20% improvement in P@1 over LEML (see Figure 2 for the variation in accuracy vs X1 learners). In fact, X1's training time on WikiLSHTC was comparable to that of tree based FastXML. FastXML trains 50 trees in 13 hours on a single core to achieve a P@1 of 49.35% whereas X1 could achieve 51.78% by training 3 learners in 12 hours. Similarly, X1's training time on Ads1M was 7 hours per learner on a single core.

X1's predictions could also be up to 300 times faster than LEMLs. For instance, on WikiLSHTC, X1 made predictions in 8 milliseconds per test point as compared to LEML's 279. X1 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.

Effect of clustering and multiple learners: As mentioned in the introduction, other embedding methods could also be extended by clustering the data and then learning a local embedding in each cluster. Ensembles could also be learnt from multiple such clusterings. We extend LEML in such a fashion, and refer to it as LocalLEML, by using exactly the same 300 clusters per learner in the ensemble as used in X1 for a fair comparison. As can be seen in Figure 2, X1 significantly outperforms LocalLEML with a single X1 learner being much more accurate than an ensemble of even 10 LocalLEML learners. Figure 2 also demonstrates that X1's ensemble can be much more accurate at prediction as compared to the tree based FastXML ensemble (the same plot is also presented in the appendix depicting the variation in accuracy with model size in RAM rather than the number of learners in the ensemble). The figure also demonstrates that very few X1 learners need to be trained before accuracy starts saturating. Finally, Table 4 shows that the variance in X1 s prediction accuracy (w.r.t. different cluster initializations) is very small, indicating that the method is stable even though clustering in more than a million dimensions.

Results on small data sets: Table 3, in the appendix, compares the performance of X1 to several popular methods including embeddings, trees, kNN and 1-vs-All SVMs. Even though the tail label problem is not acute on these data sets, and X1 was restricted to a single learner, X1's predictions could be significantly more accurate than all the other methods (except on Delicious where X1 was ranked second). For instance, X1 could outperform the closest competitor on EurLex by 3% in terms of P1. Particularly noteworthy is the observation that X1 outperformed WSABIE [13], which performs kNN classification on linear embeddings, by as much as 10% on multiple data sets. This demonstrates the superiority of X1's local distance preserving embeddings over the traditional low-rank embeddings.

References

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

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

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

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

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

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

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

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

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

[10] Y.-N. Chen and H.-T. Lin. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, pages 1538–1546, 2012.

[11] C.-S. Feng and H.-T. Lin. Multi-label classification with error-correcting codes. *JMLR*, 20, 2011.

[12] S. Ji, L. Tang, S. Yu, and J. Ye. Extracting shared subspace for multi-label classification. In *KDD*, 2008.

[13] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011.

[14] Z. Lin, G. Ding, M. Hu, and J. Wang. Multi-label classification via feature-aware implicit label space encoding. In *ICML*, pages 325–333, 2014.

[15] Yashoteja Prabhu and Manik Varma. FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, pages 263–272, 2014.

[16] Wikipedia dataset for the 4th large scale hierarchical text classification challenge, 2014.

[17] A. Ng and M. Jordan. On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes. In *NIPS*, 2002.

[18] Kilian Q. Weinberger and Lawrence K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, pages 1683–1686, 2006.

[19] Blake Shaw and Tony Jebara. Minimum volume embedding. In *AISTATS*, pages 460–467, 2007.

[20] Prateek Jain, Raghu Meka, and Inderjit S. Dhillon. Guaranteed rank minimization via singular value projection. In *NIPS*, pages 937–945, 2010.

[21] Pablo Sprechmann, Roee Litman, Tal Ben Yakar, Alex Bronstein, and Guillermo Sapiro. Efficient Supervised Sparse Analysis and Synthesis Operators. In *27th Annual Conference on Neural Information Processing Systems (NIPS)*, 2013.

[22] Purushottam Kar, Bharath K Sriperumbudur, Prateek Jain, and Harish Karnick. On the Generalization Ability of Online Learning Algorithms for Pairwise Loss Functions. In *ICML*, 2013.

[23] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection, 2014.

[24] R. Wetzker, C. Zimmermann, and C. Bauckhage. Analyzing social bookmarking systems: A del.icio.us cookbook. In *Mining Social Data (MSoDa) Workshop Proceedings, ECAI*, pages 26–30, July 2008.

[25] A. Zubiaga. Enhancing navigation on wikipedia with social tags, 2009.

[26] I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge*, 2008.

[27] C. Snoek, M. Worring, J. van Gemert, J.-M. Geusebroek, and A. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *ACM Multimedia*, 2006.

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

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

[30] Yao-Nan Chen and Hsuan-Tien Lin. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, pages 1538–1546, 2012.

[31] B. Hariharan, S. V. N. Vishwanathan, and M. Varma. Efficient max-margin multi-label classification with applications to zero-shot learning. *ML*, 2012.

A Generalization Error Analysis

To present our results, we first introduce some notation: for any embedding matrix A and dataset $\mathcal{D}$, let

$$\hat{\mathcal{L}}(A; \mathcal{D}) := \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i} \ell(A; (\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j))$$

$$\tilde{\mathcal{L}}(A; \mathcal{D}) := \frac{1}{n} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{P}} \ell(A; (\mathbf{x}, \mathbf{y}), (\mathbf{x}_i, \mathbf{y}_i))$$

$$\mathcal{L}(A) := \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y}),(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) \sim \mathcal{P}} \ell(A; (\mathbf{x}, \mathbf{y}), (\tilde{\mathbf{x}}, \tilde{\mathbf{y}}))$$

We assume, without loss of generality that the data points are confined to a unit ball i.e. $\|\mathbf{x}\|_2 \leq 1$ for all $\mathbf{x} \in \mathcal{X}$. Also let $Q = C \cdot \left(\bar{L}(r + \bar{L}) \right)$ where $\bar{L}$ is the average number of labels active in a data point, $r = \frac{\bar{L}}{\lambda}$, λ and μ are the regularization constants used in (3), and C is a *universal constant*.

Theorem 1. *Assume that all data points are confined to a ball of radius R i.e $\|x\|_2 \leq R$ for all $x \in \mathcal{X}$. Then with probability at least $1 - \delta$ over the sampling of the data set $\mathcal{D}$, the solution $\hat{A}$ to the optimization problem (3) satisfies,*

$$\mathcal{L}(\hat{A}) \leq \inf_{A^* \in \mathcal{A}} 8 \left\{ \mathcal{L}(A^*) + C \left(\bar{L}^2 + \left(r^2 + \|A^*\|_F^2 \right) R^4 \right) \sqrt{\frac{1}{n} \log \frac{1}{\delta}} \right\},$$

where $r = \frac{\bar{L}}{\lambda}$, and C and C' are universal constants.

Proof. Our proof will proceed in the following steps. Let A^* be the population minimizer of the objective in the statement of the theorem.

1. **Step 1** (Capacity bound): we will show that for some r, we have $\|\hat{A}\|_F \leq r$

2. **Step 2** (Uniform convergence): we will show that w.h.p., $\sup_{\substack{A \in \mathcal{A} \\ \|A\| \leq r}} \left\{ \mathcal{L}(A) - \hat{\mathcal{L}}(A; \mathcal{D}) \right\} \leq \mathcal{O}\left(\sqrt{\frac{1}{n} \log \frac{1}{\delta}} \right)$

3. **Step 3** (Point convergence): we will show that w.h.p., $\hat{\mathcal{L}}(A^*; \mathcal{D}) - \mathcal{L}(A^*) \leq \mathcal{O}\left(\sqrt{\frac{1}{n} \log \frac{1}{\delta}} \right)$

Having these results will allow us to prove the theorem in the following manner

$$\mathcal{L}(\hat{A}) \leq \hat{\mathcal{L}}(\hat{A}, \mathcal{D}) + \sup_{\substack{A \in \mathcal{A} \\ \|A\| \leq r}} \left\{ \hat{\mathcal{L}}(A; \mathcal{D}) - \mathcal{L}(A) \right\} \leq \hat{\mathcal{L}}(A^*, \mathcal{D}) + \mathcal{O}\left(\sqrt{\frac{1}{n} \log \frac{1}{\delta}} \right) \leq \mathcal{L}(A^*) + \mathcal{O}\left(\sqrt{\frac{1}{n} \log \frac{1}{\delta}} \right),$$

where the second step follows from the fact that $\hat{A}$ is the empirical risk minimizer.

We will now prove these individual steps as separate lemmata, where we will also reveal the exact constants in these results.

Lemma 2 (Capacity bound). *For the regularization parameters chosen for the loss function $\ell(\cdot)$, the following holds for the minimizer $\hat{A}$ of* (3)

$$\|\hat{A}\|_F \leq Tr(A) \leq \frac{1}{\lambda} \bar{L}.$$

Proof. Since, $\hat{A}$ minimizes (3), we have:

$$\|A\|_F \leq Tr(A) \leq \frac{1}{\lambda} \frac{1}{n(n-1)} \sum_{ij} (\langle \mathbf{y}_i, \mathbf{y}_j \rangle)^2 \leq \frac{1}{\lambda} \max_{ij} \langle \mathbf{y}_i, \mathbf{y}_j \rangle .$$

$\square$

The above result shows that we can, for future analysis, restrict our hypothesis space to

$$\widetilde{\mathcal{A}}(r) := \left\{ A \in \mathcal{A} : \|A\|_F^2 \leq r^2 \right\},$$

where we set $r = \frac{\bar{L}}{\lambda}$. This will be used to prove the following result.

Lemma 3 (Uniform convergence). *With probability at least $1 - \delta$ over the choice of the data set $\mathcal{D}$, we have*

$$\hat{\mathcal{L}}(\hat{A}; \mathcal{D}) - \mathcal{L}(\hat{A}) \leq 6 \left(rR^2 + \bar{L} \right)^2 \sqrt{\frac{1}{2n} \log \frac{1}{\delta}}$$

Proof. For notional simplicity, we will denote a labeled sample as $\mathbf{z} = (\mathbf{x}, \mathbf{y})$. Given any two points $\mathbf{z}, \mathbf{z}' \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and any $A \in \widetilde{\mathcal{A}}(r)$, we will then write

$$\ell(A; \mathbf{z}, \mathbf{z}') = g(\langle \mathbf{y}, \mathbf{y}' \rangle) \left(\langle \mathbf{y}, \mathbf{y}' \rangle - \mathbf{x}^T A \mathbf{x}' \right)^2 + \lambda Tr(A),$$

so that, for the training set $\mathcal{D} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, we have

$$\hat{\mathcal{L}}(A; \mathcal{D}) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i} \ell(A; \mathbf{z}_i, \mathbf{z}_j)$$

as well as

$$\mathcal{L}(A) = \mathbb{E}_{z, z' \sim \mathcal{P}} \ell(A; \mathbf{z}, \mathbf{z}').$$

Note that we ignore the $\|V\mathbf{x}\|_1$ term in (3) completely because it is a regularization term and it won't increase the excess risk.

Suppose we draw a fresh data set $\widetilde{\mathcal{D}} = \{\tilde{\mathbf{z}}_1, \ldots, \tilde{\mathbf{z}}_n\} \sim \mathcal{P}$, then we have, by linearity of expectation,

$$\mathbb{E}_{\widetilde{\mathcal{D}} \sim \mathcal{P}} \hat{\mathcal{L}}(\hat{A}; \tilde{\mathcal{D}}) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i} \mathbb{E}_{\widetilde{\mathcal{D}} \sim \mathcal{P}} \ell(A; \tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) = \mathcal{L}(\hat{A}).$$

Now notice that for any $A \in \widetilde{\mathcal{A}}$, suppose we perturb the data set $\mathcal{D}$ at the ith location to get a perturbed data set $\mathcal{D}^i$, then the following holds

$$\left| \hat{\mathcal{L}}(A; \mathcal{D}) - \hat{\mathcal{L}}(A; \mathcal{D}^i) \right| \leq \frac{4(\bar{L}^2 + r^2 R^4)}{n}.$$

which allows us to bound the excess risk as follows

$$\mathcal{L}(\hat{A}) - \hat{\mathcal{L}}(\hat{A}; \mathcal{D}) = \mathbb{E}_{\widetilde{\mathcal{D}} \sim \mathcal{P}} \hat{\mathcal{L}}(\hat{A}; \tilde{\mathcal{D}}) - \hat{\mathcal{L}}(\hat{A}; \mathcal{D}) \leq \sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \mathbb{E}_{\widetilde{\mathcal{D}} \sim \mathcal{P}} \hat{\mathcal{L}}(A; \tilde{\mathcal{D}}) - \hat{\mathcal{L}}(A; \mathcal{D}) \right\}$$

$$\leq \mathbb{E}_{\mathcal{D} \sim \mathcal{P}} \sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \mathbb{E}_{\widetilde{\mathcal{D}} \sim \mathcal{P}} \hat{\mathcal{L}}(A; \tilde{\mathcal{D}}) - \hat{\mathcal{L}}(A; \mathcal{D}) \right\} + 4(\bar{L}^2 + r^2 R^4) \sqrt{\frac{1}{2n} \log \frac{1}{\delta}}$$

$$\leq \underbrace{\mathbb{E}_{\mathcal{D}, \widetilde{\mathcal{D}} \sim \mathcal{P}} \sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \hat{\mathcal{L}}(A; \tilde{\mathcal{D}}) - \hat{\mathcal{L}}(A; \mathcal{D}) \right\}}_{Q_n(\widetilde{\mathcal{A}}(r))} + 4(\bar{L}^2 + r^2 R^4) \sqrt{\frac{1}{2n} \log \frac{1}{\delta}},$$

where the third step follows from an application of McDiarmid's inequality and the last step follows from Jensen's inequality. We now bound the quantity $Q_n(\widetilde{\mathcal{A}}(r))$ below. Let $\bar{\ell}(A, \mathbf{z}, \mathbf{z}') := \ell(A, \mathbf{z}, \mathbf{z}') - \lambda \cdot Tr(A)$. Then we have

$$Q_n(\widetilde{\mathcal{A}}(r)) = \mathbb{E}_{\mathcal{D}, \widetilde{\mathcal{D}} \sim \mathcal{P}} \sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \hat{\mathcal{L}}(A; \tilde{\mathcal{D}}) - \hat{\mathcal{L}}(A; \mathcal{D}) \right\}$$

$$= \frac{1}{n(n-1)} \mathbb{E}_{\mathbf{z}_i, \tilde{\mathbf{z}}_i \sim \mathcal{P}} \left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n} \sum_{j \neq i} \ell(A; \tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) - \ell(A; \mathbf{z}_i, \mathbf{z}_j) \right\} \right]$$

$$= \frac{1}{n(n-1)} \mathbb{E}_{\mathbf{z}_i, \tilde{\mathbf{z}}_i \sim \mathcal{P}} \left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n} \sum_{j \neq i} \bar{\ell}(A; \tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) - \bar{\ell}(A; \mathbf{z}_i, \mathbf{z}_j) \right\} \right]$$

$$\leq \frac{2}{n} \mathbb{E}_{\mathbf{z}_i, \tilde{\mathbf{z}}_i} \left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \bar{\ell}(A; \tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_{n/2+i}) - \bar{\ell}(A; \mathbf{z}_i, \mathbf{z}_{n/2+i}) \right\} \right]$$

$$\leq 2 \cdot \underbrace{\frac{2}{n} \mathbb{E}_{z_i, \epsilon_i} \left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \epsilon_i \bar{\ell}(A; \mathbf{z}_i, \mathbf{z}_{n/2+i}) \right\} \right]}_{\mathcal{R}_n(\ell \circ \widetilde{\mathcal{A}}(r))} = 2 \cdot \mathcal{R}_{n/2}(\ell \circ \widetilde{\mathcal{A}}(r))$$

where the last step uses a standard symmetrization argument with the introduction of the Rademacher variables $\epsilon_i \sim -1, +1$. The second step presents a stumbling block in the analysis since the interaction between the pairs of the points means that traditional symmetrization can no longer done. Previous works analyzing such "pairwise" loss functions face similar problems [22]. Consequently, this step uses a powerful alternate representation for U-statistics to simplify the expression. This technique is attributed to Hoeffding. This, along with the Hoeffding decomposition, are two of the most powerful techniques to deal with "coupled" random variables as we have in this situation.

Theorem 4. *For any set of real valued functions* $q_\tau : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *indexed by* $\tau \in T$, *if* $X_1, \ldots, X_n$ *are i.i.d. random variables then we have*

$$\mathbb{E} \left[\sup_{\tau \in T} \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} q_\tau(X_i, X_j) \right] \leq \mathbb{E} \left[\sup_{\tau \in T} \frac{2}{n} \sum_{i=1}^{n/2} q_\tau(X_i, X_{n/2+i}) \right]$$

Applying this decoupling result to the random variables $X_i = (\tilde{\mathbf{z}}_i, \mathbf{z}_i)$, the index set $\widetilde{\mathcal{A}}(r)$ and functions $q_A(X_i, X_j) = \ell(A; \tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) - \ell(A; \mathbf{z}_i, \mathbf{z}_j) = \bar{\ell}(A; \tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) - \bar{\ell}(A; \mathbf{z}_i, \mathbf{z}_j)$ gives us the second step. We now concentrate on bounding the resulting Rademacher average term $\mathcal{R}_n(\ell \circ \widetilde{\mathcal{A}}(r))$. We have

$$\mathcal{R}_{n/2}(\ell \circ \widetilde{\mathcal{A}}(r)) = \frac{2}{n} \underset{z_i, \epsilon_i}{\mathbb{E}} \left[\left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \epsilon_i \bar{\ell}(A; \mathbf{z}_i, \mathbf{z}_{n/2+i}) \right\} \right]\right]$$

$$= \frac{2}{n} \underset{z_i, \epsilon_i}{\mathbb{E}} \left[\left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \epsilon_i g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle) \left(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle - \mathbf{x}_i^T A \mathbf{x}_{n/2+i} \right)^2 \right\} \right]\right].$$

That is,

$$\mathcal{R}_{n/2}(\ell \circ \widetilde{\mathcal{A}}(r)) \leq \underbrace{\frac{2}{n} \underset{z_i, \epsilon_i}{\mathbb{E}} \left[\left[\sum_{i=1}^{n/2} \epsilon_i g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle) \langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle^2 \right]\right]}_{(A)}$$

$$+ \underbrace{\frac{2}{n} \underset{z_i, \epsilon_i}{\mathbb{E}} \left[\left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \epsilon_i g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle) \left(\mathbf{x}_i^T A \mathbf{x}_{n/2+i} \right)^2 \right\} \right]\right]}_{B_n(\ell \circ \widetilde{\mathcal{A}}(r))}$$

$$+ \underbrace{\frac{4}{n} \underset{z_i, \epsilon_i}{\mathbb{E}} \left[\left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \epsilon_i g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle) \langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle \left(\mathbf{x}_i^T A \mathbf{x}_{n/2+i} \right) \right\} \right]\right]}_{C_n(\ell \circ \widetilde{\mathcal{A}}(r))}$$

Now since the random variables ϵ_i are zero mean and independent of z_i, we have $\underset{\epsilon_i | z_i, z_{n/2+i}}{\mathbb{E}} \epsilon_i = 0$ which we can use to show that $\underset{\epsilon_i | z_i, z_{n/2+i}}{\mathbb{E}} \left[\left[\epsilon_i g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle) \langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle^2 \right]\right] = 0$ which gives us, by linearity of expectation, $(A) = 0$. To bound the next two terms we use the following standard contraction inequality:

Theorem 5. *Let $\mathcal{H}$ be a set of bounded real valued functions from some domain $\mathcal{X}$ and let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be arbitrary elements from $\mathcal{X}$. Furthermore, let $\phi_i : \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, n$ be L-Lipschitz functions such that $\phi_i(0) = 0$ for all i. Then we have*

$$\mathbb{E} \left[\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i \phi_i(h(\mathbf{x}_i)) \right]\right] \leq L \mathbb{E} \left[\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i h(\mathbf{x}_i) \right]\right].$$

Now define

$$\phi_i(w) = g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle) w^2$$

Clearly $\phi_i(0) = 0$ and $0 \leq g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle) \leq 1$. Moreover, in our case $w = x^T A x'$ for some $A \in \widetilde{\mathcal{A}}(r)$ and $\|x\|, \|x'\| \leq R$. Thus, the function $\phi_i(\cdot)$ is rR^2-Lipschitz. Note that here we exploit the fact that the contraction inequality is actually proven for the empirical Rademacher averages due to which we can take $g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle)$ to be a constant dependent only on i. This allows us to bound the term $B_n(\ell \circ \widetilde{\mathcal{A}}(r))$ as follows

$$B_n(\ell \circ \widetilde{\mathcal{A}}(r)) = \frac{2}{n} \underset{z_i, \epsilon_i}{\mathbb{E}} \left[\left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \epsilon_i g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle) \left(\mathbf{x}_i^T A \mathbf{x}_{n/2+i} \right)^2 \right\} \right]\right]$$

$$\leq rR^2 \cdot \underbrace{\frac{2}{n} \underset{z_i, \epsilon_i}{\mathbb{E}} \left[\left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \epsilon_i \left(\mathbf{x}_i^T A \mathbf{x}_{n/2+i} \right) \right\} \right]\right]}_{\mathcal{R}_{n/2}(\widetilde{\mathcal{A}}(r))} \leq rR^2 \cdot \mathcal{R}_{n/2}(\widetilde{\mathcal{A}}(r)).$$

13

Similarly, we can show that

$$C_n(\ell \circ \widetilde{\mathcal{A}}(r)) = \frac{4}{n} \mathop{\mathbb{E}}_{z_i, \epsilon_i} \left[\left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \epsilon_i g(\langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle) \langle \mathbf{y}_i, \mathbf{y}_{n/2+i} \rangle \left(\mathbf{x}_i^T A \mathbf{x}_{n/2+i} \right) \right\} \right]\right]$$

$$\leq \frac{4\bar{L}}{n} \mathop{\mathbb{E}}_{z_i, \epsilon_i} \left[\left[\sup_{A \in \widetilde{\mathcal{A}}(r)} \left\{ \sum_{i=1}^{n/2} \epsilon_i \left(\mathbf{x}_i^T A \mathbf{x}_{n/2+i} \right) \right\} \right]\right]$$

$$\leq 2\bar{L} \cdot \mathcal{R}_{n/2}(\widetilde{\mathcal{A}}(r)).$$

Thus, we have

$$\mathcal{R}_{n/2}(\ell \circ \widetilde{\mathcal{A}}(r)) \leq \left(rR^2 + 2\bar{L} \right) \cdot \mathcal{R}_{n/2}(\widetilde{\mathcal{A}}(r))$$

Now all that remains to be done is bound $\mathcal{R}_n(\ell \circ \widetilde{\mathcal{A}}(r))$. This can be done by invoking standard bounds on Rademacher averages for regularized function classes. In particular, using the two stage proof technique outlined in [22], we can show that

$$\mathcal{R}_{n/2}(\widetilde{\mathcal{A}}(r)) \leq rR^2 \sqrt{\frac{2}{n}}$$

Putting it all together gives us the following bound: with probability at least $1 - \delta$, we have

$$\mathcal{L}(\hat{A}) - \hat{\mathcal{L}}(\hat{A}; \mathcal{D}) \leq 2(rR^2 + 2\bar{L})rR^2 \sqrt{\frac{2}{n}} + 4(\bar{L}^2 + r^2 R^4)\sqrt{\frac{1}{2n} \log \frac{1}{\delta}}$$

as claimed $\qquad\qquad\square$

The final part shows pointwise convergence for the population risk minimizer.

Lemma 6 (Point convergence). *With probability at least $1 - \delta$ over the choice of the data set $\mathcal{D}$, we have*

$$\hat{\mathcal{L}}(A^*; \mathcal{D}) - \mathcal{L}(A^*) \leq 4(\bar{L}^2 + \|A^*\|_F^2 R^4)\sqrt{\frac{1}{2n} \log \frac{1}{\delta}},$$

where A^* is the population minimizer of the objective in the theorem statement.

Proof. We note that, as before

$$\mathop{\mathbb{E}}_{\mathcal{D} \sim \mathcal{P}} \hat{\mathcal{L}}(A^*, \mathcal{D}) = \mathcal{L}(A^*)$$

Let $\mathcal{D}$ be a realization of the sample and $\mathcal{D}^i$ be a perturbed data set where the ith data point is arbitrarily perturbed. Then we have

$$\left| \hat{\mathcal{L}}(A^*; \mathcal{D}) - \hat{\mathcal{L}}(A^*; \mathcal{D}^i) \right| \leq \frac{4 \left(\bar{L}^2 + \|A^*\|_F^2 R^4 \right)}{n}.$$

Thus, an application of McDiarmid's inequality shows us that with probability at least $1 - \delta$, we have

$$\hat{\mathcal{L}}(A^*; \mathcal{D}) - \mathcal{L}(A^*) = \hat{\mathcal{L}}(A^*; \mathcal{D}) - \mathop{\mathbb{E}}_{\mathcal{D} \sim \mathcal{P}} \hat{\mathcal{L}}(A^*; \mathcal{D}) \leq 4 \left(\bar{L}^2 + \|A^*\|_F^2 R^4 \right) \sqrt{\frac{1}{2n} \log \frac{1}{\delta}},$$

which proves the claim. $\qquad\qquad\square$

Putting the three lemmata together as shown above concludes the proof of the theorem. $\qquad\qquad\square$

Although the above result ensures that the embedding provided by $\hat{A}$ would preserve neighbors over the population, in practice, we are more interested in preserving the neighbors of test points among the training points, as they are used to predict the label vector. The following extension of our result shows that $\hat{A}$ indeed accomplishes this as well.

Theorem 7. *Assume that all data points are confined to a ball of radius R i.e $\|x\|_2 \leq R$ for all $x \in \mathcal{X}$. Then with probability at least $1 - \delta$ over the sampling of the data set $\mathcal{D}$, the solution $\hat{A}$ to the optimization problem* (3) *ensures that,*

$$\tilde{\mathcal{L}}(\hat{A}; \mathcal{D}) \leq \inf_{A^* \in \mathcal{A}} \left\{ \tilde{\mathcal{L}}(A^*; \mathcal{D}) + C \left(\bar{L}^2 + \left(r^2 + \|A^*\|_F^2 \right) R^4 \right) \sqrt{\frac{1}{n} \log \frac{1}{\delta}} \right\},$$

where $r = \frac{\bar{L}}{\lambda}$, and C is a universal constant.

Note that the loss function $\tilde{\mathcal{L}}(A; \mathcal{D})$ exactly captures the notion of how well an embedding matrix A can preserve the neighbors of an unseen point among the training points.

Proof. We first recall and rewrite the form of the loss function considered here. For any data set $\mathcal{D} = \{z_1, \ldots, z_n\}$. For any $A \in \mathcal{A}$ and $z \in \mathbb{Z}$, let $\wp(A; z) := \mathbb{E}_{z' \sim \mathcal{P}} \ell(A; z, z')$. This allows us to write

$$\tilde{\mathcal{L}}(A; \mathcal{D}) := \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{z \sim \mathcal{P}} \ell(A; z, z_i) = \frac{1}{n} \sum_{i=1}^n \wp(A; z_i)$$

Also note that for any fixed A, we have

$$\mathbb{E}_{\mathcal{D} \sim \mathcal{P}} \tilde{\mathcal{L}}(A; \mathcal{D}) = \mathcal{L}(A).$$

Now, given a perturbed data set $\mathcal{D}^i$, we have

$$\left| \tilde{\mathcal{L}}(A; \mathcal{D}) - \tilde{\mathcal{L}}(A; \mathcal{D}^i) \right| \leq \frac{4(\bar{L}^2 + r^2 R^4)}{n},$$

as before. Since this problem does not have to take care of pairwise interactions between the data points (since the "other" data point is being taken expectations over), using standard Rademacher style analysis gives us, with probability at least $1 - \delta$,

$$\tilde{\mathcal{L}}(\hat{A}; \mathcal{D}) - \mathcal{L}(\hat{A}) \leq 2 \left(rR^2 + 2\bar{L} \right) rR^2 \sqrt{\frac{2}{n}} + 4(\bar{L}^2 + r^2 R^4) \sqrt{\frac{1}{2n} \log \frac{1}{\delta}}$$

A similar analysis also gives us with the same confidence

$$\mathcal{L}(A^*) - \hat{\mathcal{L}}(A^*; \mathcal{D}) \leq 4(\bar{L}^2 + \|A^*\|_F^2 R^4) \sqrt{\frac{1}{2n} \log \frac{1}{\delta}}$$

However, an argument similar to that used in the proof of Theorem 1 shows us that

$$\mathcal{L}(\hat{A}) \leq \mathcal{L}(A^*) + C \left(\bar{L}^2 + \left(r^2 + \|A^*\|_F^2 \right) R^4 \right) \sqrt{\frac{1}{n} \log \frac{1}{\delta}}$$

Combining the above inequalities yields the desired result. $\qquad \square$

B Experiments

Table 2: Data set Statistics: n and m are the number of training and test points respectively, d and L are the number of features and labels, respectively, and $\bar{d}$ and $\bar{L}$ are the average number of nonzero features and positive labels in an instance, respectively.

Data set	d	L	n	m	$\bar{d}$	$\bar{L}$
MediaMill	120	101	30993	12914	120.00	4.38
BibTeX	1836	159	4880	2515	68.74	2.40
Delicious	500	983	12920	3185	18.17	19.03
EURLex	5000	3993	17413	1935	236.69	5.31
Wiki10	101938	30938	14146	6616	673.45	18.64
DeliciousLarge	782585	205443	196606	100095	301.17	75.54
WikiLSHTC	1617899	325056	1778351	587084	42.15	3.19
Amazon	135909	670091	490449	153025	75.68	5.45
Ads1M	164592	1082898	3917928	1563137	9.01	1.96

Table 3: Results on Small Scale data sets : Comparison of precision accuracies of X1 with competing baseline methods on small scale data sets. The results reported are average precision values along with standard deviations over 10 random train-test split for each Data set. X1 outperforms all baseline methods on all data sets (except Delicious, where it is ranked 2nd after FastXML)

Data set		Proposed	Embedding					Tree Based			Other	
		X1	LEML	WSABIE	CPLST	CS	ML-CSSP	FastXML-1	FastXML	LPSR	OneVsAll	KNN
Bibtex	P@1	**65.57 ±0.65**	62.53±0.69	54.77±0.68	62.38 ±0.42	58.87 ±0.64	44.98 ±0.08	37.62 ±0.91	63.73±0.67	62.09±0.73	61.83 ±0.77	57.00 ±0.85
	P@3	**40.02 ±0.39**	38.40 ±0.47	32.38 ±0.26	37.83 ±0.52	33.53 ±0.44	30.42 ±2.37	24.62 ±0.68	39.00 ±0.57	36.69 ±0.49	36.44 ±0.38	36.32 ±0.47
	P@5	**29.30 ±0.32**	28.21 ±0.29	23.98 ±0.18	27.62 ±0.28	23.72 ±0.28	23.53 ±1.21	21.92 ±0.65	28.54 ±0.38	26.58 ±0.38	26.46 ±0.26	28.12 ±0.39
Delicious	P@1	68.42 ±0.53	65.66 ±0.97	64.12 ±0.77	65.31 ±0.79	61.35 ±0.77	63.03 ±1.10	55.34 ±0.92	**69.44 ±0.58**	65.00±0.77	65.01 ±0.73	64.95 ±0.68
	P@3	61.83 ±0.59	60.54 ±0.44	58.13 ±0.58	59.84 ±0.5	56.45 ±0.62	56.26 ±1.18	50.69 ±0.58	**63.62 ±0.75**	58.97 ±0.65	58.90 ±0.60	58.90 ±0.70
	P@5	56.80 ±0.54	56.08 ±0.56	53.64 ±0.55	55.31 ±0.52	52.06 ±0.58	50.15 ±1.57	45.99 ±0.37	**59.10 ±0.65**	53.46 ±0.46	53.26 ±0.57	54.12 ±0.57
MediaMill	P@1	**87.09 ±0.33**	84.00±0.30	81.29 ±1.70	83.34 ±0.45	83.82 ±0.36	78.94 ±10.1	61.14±0.49	84.24 ±0.27	83.57 ±0.26	83.57 ±0.25	83.46 ±0.19
	P@3	**72.44 ±0.30**	67.19 ±0.29	64.74 ±0.67	66.17 ±0.39	67.31 ±0.17	60.93 ±8.5	53.37 ±0.30	67.39 ±0.20	65.78 ±0.22	65.50 ±0.23	67.91 ±0.23
	P@5	**58.45 ±0.34**	52.80 ±0.17	49.82 ±0.71	51.45 ±0.37	52.80 ±0.18	44.27 ±4.8	48.39 ±0.19	53.14 ±0.18	49.97 ±0.48	48.57 ±0.56	54.24 ±0.21
EurLEX	P@1	**80.17 ±0.86**	61.28±1.33	70.87 ±1.11	69.93±0.90	60.18 ±1.70	56.84±1.5	49.18 ±0.55	68.69 ±1.63	73.01 ±1.4	74.96 ±1.04	77.2 ±0.79
	P@3	**65.39 ±0.88**	48.66 ±0.74	56.62 ±0.67	56.18 ±0.66	48.01 ±1.90	45.4 ±0.94	42.72 ±0.51	57.73 ±1.58	60.36 ±0.56	62.92 ±0.53	61.46 ±0.96
	P@5	**53.75 ±0.80**	39.91 ±0.68	46.20 ±0.55	45.74 ±0.42	38.46 ±1.48	35.84 ±0.74	37.35 ±0.42	48.00 ±1.40	50.46 ±0.50	53.42 ±0.37	50.45 ±0.64

Table 4: Stability of X1 learners. We show mean precision values over 10 runs of X1 on WikiLSHTC with varying number of learners. Each individual learner as well as ensemble of X1 learners was found to be extremely stable with with standard deviation ranging from 0.16% on P1 to 0.11% on P5.

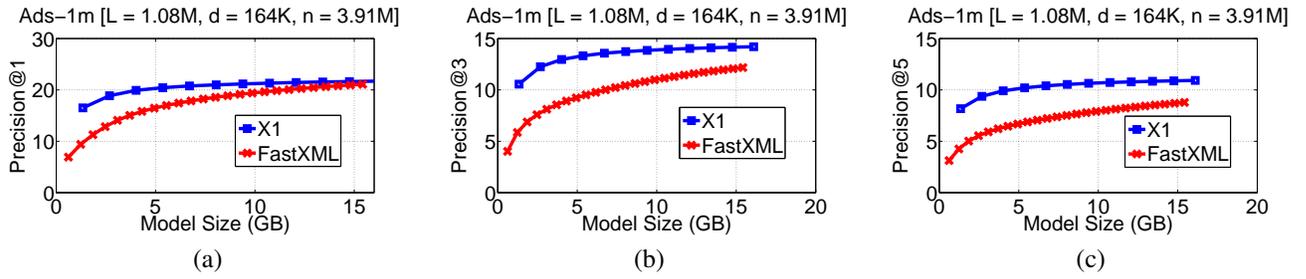| # Learners	1	2	3	4	5	6	7	8	9	10
P@1	46.04 ±0.1659	50.04 ±0.0662	51.65 ±0.074	52.62 ±0.0878	53.28 ±0.0379	53.63 ±0.083	54.03 ±0.0757	54.28 ±0.0699	54.44 ±0.048	54.69 ±0.035
P@3	26.15 ±0.1359	29.32 ±0.0638	30.70 ±0.052	31.55 ±0.067	32.14 ±0.0351	32.48 ±0.0728	32.82 ±0.0694	33.07 ±0.0503	33.24 ±0.023	33.45 ±0.0127
P@5	18.14 ±0.1045	20.58 ±0.0517	21.68 ±0.0398	22.36 ±0.0501	22.85 ±0.0179	23.12 ±0.0525	23.4 ±0.0531	23.60 ±0.0369	23.74 ±0.0172	23.92 ±0.0115

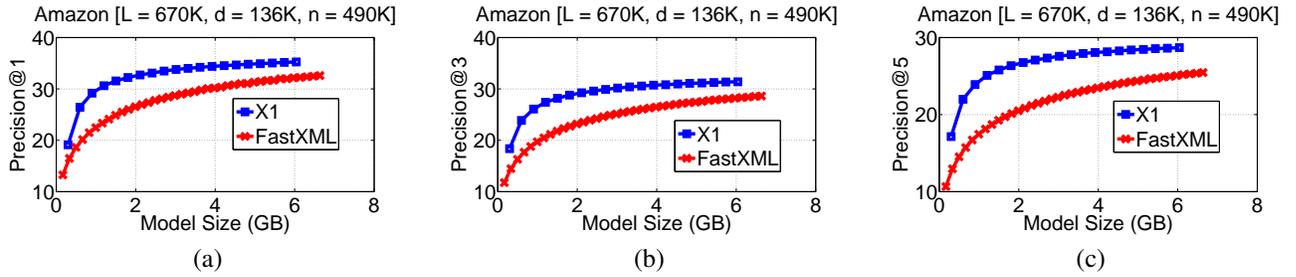Figure 3: Variation of precision accuracy with model size on Ads-1m Data set

Figure 4: Variation of precision accuracy with model size on Amazon Data set
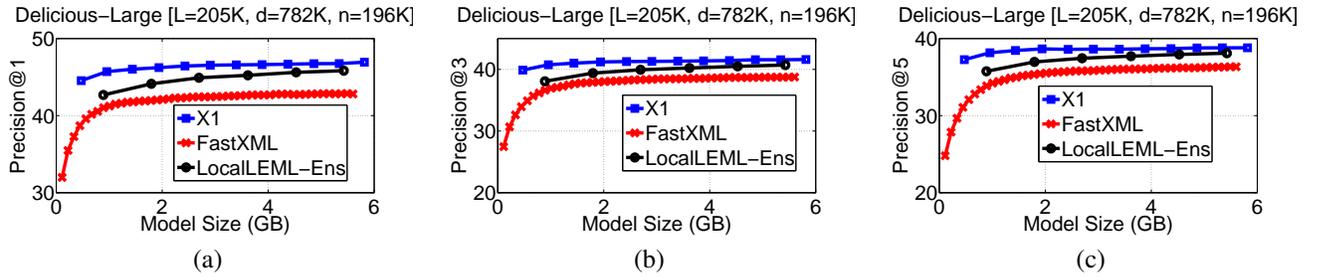

Figure 5: Variation of precision accuracy with model size on Delicious-Large Data set
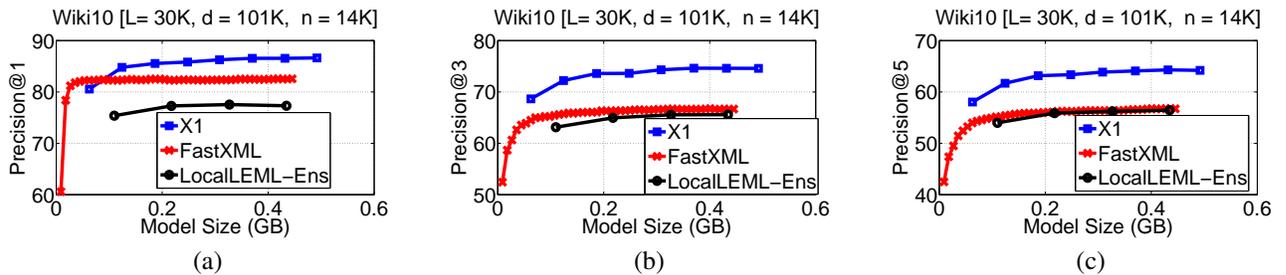

Figure 6: Variation of precision accuracy with model size on Wiki10 Data set
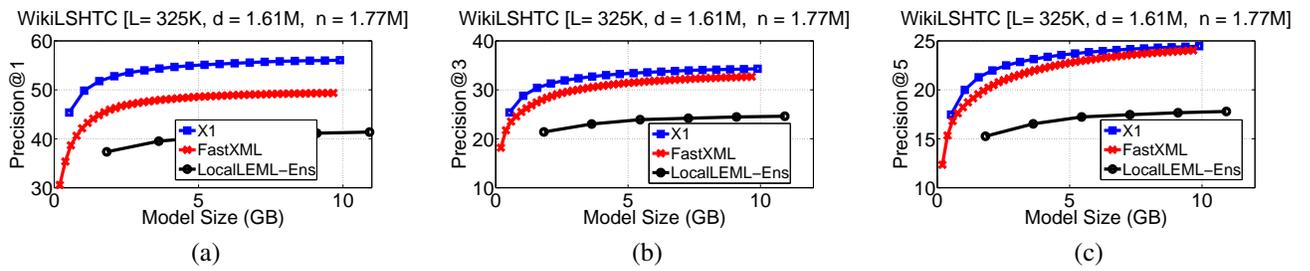
Figure 7: Variation of precision accuracy with model size on WikiLSHTC Data set