Multilabel Classification with Group Testing and Codes

Shashanka Ubaru [1] **Arya Mazumdar** [2]

Abstract

In recent years, the multiclass and mutlilabel classification problems we encounter in many applications have very large ($10^3 - 10^6$) number of classes. However, each instance belongs to only one or few classes, i.e., the label vectors are sparse. In this work, we propose a novel approach based on group testing to solve such large multilabel classification problems with sparse label vectors. We describe various group testing constructions, and advocate the use of concatenated Reed Solomon codes and unbalanced bipartite expander graphs for extreme classification problems. The proposed approach has several advantages theoretically and practically over existing popular methods. Our method operates on the binary alphabet and can utilize the well-established binary classifiers for learning. The error correction capabilities of the codes are leveraged for the first time in the learning problem to correct prediction errors. Even if a linearly growing number of classifiers mis-classify, these errors are fully corrected. We establish Hamming loss error bounds for the approach. More importantly, our method utilizes a simple prediction algorithm and does not require matrix inversion or solving optimization problems making the algorithm very inexpensive. Numerical experiments with various datasets illustrate the superior performance of our method.

1. Introduction

In the multilabel classification problem, we are given a set of labeled training data $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ are the input features for each data instances and $y_i \in \{0,1\}^d$ are vectors indicating the corresponding labels (classes) the data instances belong to. The vector y_i has a one in the jth coordinate if the ith data point belongs to jth class. We wish to learn a mapping (prediction rule) between the features and the labels, such that, we can predict the class label vector y of a new data point x correctly. Such multilabel classification problems occur in many domains such as text mining, computer vision, music, and bioinformatics (Barutcuoglu et al., 2006; Trohidis, 2008; Tai & Lin, 2012), and modern applications involve large number of labels. Popular applications with many labels include image and video annotation (Wang et al., 2009), web page categorization (Agrawal et al., 2013), text and document categorization (Tsoumakas et al., 2008), and others (Bhatia et al., 2015). In most of these applications, the label vectors y_i are sparse (with average sparsity of $k \ll d$), i.e., each data point belongs to a few (average k out of d) classes. The multiclass classification is an instance of the multilabel classification, where all data points belong to only one of the d classes ($k = 1$).

The simple binary classification problem, where $d = 2$ and $k = 1$ is well-studied, and several efficient algorithms have been proposed in the literature. A natural approach used to solve the multiclass ($d > 2, k = 1$) classification problem is to reduce the problem into a set of binary classification problem, and then employ the efficient binary classifiers to solve the individual problems. Popular methods based on this approach are: one-vs-all, all-pairs, and the error-correcting output code (ECOC) (Dietterich & Bakiri, 1995) methods. In ECOC method, m-dimensional binary vectors (typically codewords from an error correcting code with $m \leq d$) are assigned to each class, and m binary classifiers are learned. For the jth classification, the jth coordinate of the corresponding codeword is used as the binary label for each class. In the modern applications, where d is typically very large, this approach is found to be very efficient due to the reduction of the class dimension.

The idea of ECOC approach has been extended to the multilabel classification (MLC) problem. In the multiclass classification, using codewords for each class in ECOC is equivalent to multiplying the code matrix to the label vectors (since the label vectors are basis vectors). Hence, in the multilabel setting, the reduction from d dimensional label vectors to m dimensional can be achieved by multiply-

[1]Department of Computer Science and Engineering, University of Minnesota at Twin Cities, MN USA. [2]College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA, USA.. Correspondence to: Shashanka Ubaru <ubaru001@umn.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, 2017. JMLR: W&CP.

ing a code matrix $A \in \mathbb{R}^{m \times d}$ to the label vector y. This reduction method was analyzed from the compressed sensing point of view in (Hsu et al., 2009), with the assumption of output sparsity, i.e., y is sparse (with average sparsity k). Using compressed sensing (CS) theory, the results in (Hsu et al., 2009) show that for a linear hypothesis class and under the squared loss, a random embedding (random code matrix) of the classes to $m = O(k \log d)$ dimensions does not increase the $L2$ risk of the classifier. Similarly, (Kapoor et al., 2012) discusses MLC using compressed sensing in the Bayesian framework. However, the CS approach requires solving an optimization problem to recover the label vector. Constructions with faster recovery algorithms exist (see, e.g., (Jafarpour et al., 2009)) but we cannot obtain $L2$ norm results with them.

Alternatively, embedding based approaches have been proposed to reduce the effective number of labels. These methods reduce the label dimension by projecting label vectors onto a low dimensional space, based on the assumption that the label matrix $Y = [y_1, \ldots, y_n]$ is low-rank. The various embedding methods proposed in the literature mainly differ in the way this reduction is achieved. The reduction is achieved using SVD in (Tai & Lin, 2012), while column subset selection is used in (Bi & Kwok, 2013). (Zhang & Schneider, 2011) used canonical correlation analysis and (Chen & Lin, 2012) used an SVD approach that leverages the feature space information. (Yu et al., 2014) discussed multilabel classification with missing entries and used an embedding method with a regularized least squares objective. These embedding methods capture the label correlation, and Euclidean distance error guarantees are established. However, the low rank assumption breaks down in many situations (Bhatia et al., 2015; Xu et al., 2016), e.g., data is power law distributed (Babbar & Schölkopf, 2017).

The state of the art embedding method called SLEEC (Sparse Local Embedding for Extreme Classification) (Bhatia et al., 2015) overcomes the limitations of previous embedding methods by first clustering the data into smaller regions, and then performs local embeddings of label vectors by preserving distances to nearest label vectors. However, this method also has many shortcomings, see (Babbar & Schölkopf, 2017). Moreover, most of these embedding based methods are very expensive. They involve eigenvalue or singular value decompositions and matrix inversions, and may require solving convex optimization problems, all of which become impractical for very large d. In all the embedding methods and the CS method, the reduced label space is a real space (no longer binary). Hence we need to use regressors for training and cannot leverage the efficient binary classifiers for effective training for the model. Prediction will also involve rounding/thresholding of real values. This is additional work, and choosing a right threshold is sometimes problematic.

Proposed Approach. In this paper, we present a novel reduction approach to solve the MLC problem. Our approach assumes output sparsity (sparse label vectors with $k \ll d$) similar to the CS approach, but reduces a large binary label vector to a *binary* vector of smaller size. Since the reduced label vectors are binary, we can use the efficient binary classifiers for effective training for the model. Our prediction algorithm is extremely simple and does not involve any matrix inversion or solving optimization algorithm. The prediction algorithm can also detect and correct errors. Hence, even if a constant fraction of the binary classifiers mis-classify, our prediction error will be *zero*.

Our approach is based on the popular group testing problem (Dorfman, 1943; Du & Hwang, 2000). In the group testing problem, we wish to efficiently identify a small number k of defective elements in a population of large size d. The idea is to test the items in groups with the premise that most tests will return negative results, clearing the entire group. Only few $m \ll d$ tests are needed to detect the k defectives. The items can be grouped in either an adaptive or nonadaptive fashion. In the nonadaptive group testing scheme, the grouping for each test can be described using an $m \times d$ binary ($0/1$ entries) matrix A.

We make the crucial observation that, the MLC problem can be solved using the group testing (GT) premise. That is, the problem of estimating the (few) classes of a data instance from a large set of classes, is similar to identifying a small set of items from a large set. We consider a group testing binary matrix A and reduce the label vectors y_i's to smaller binary vectors z_i using the boolean OR operation $z_i = A \vee y_i$ (described later). We can now use binary classifiers on z_i for training. The m classifiers learn to test whether the data belongs to a group (of labels) or not. During prediction, the label vector can be recovered from the predictions of the classifiers using a simple inexpensive algorithm (requiring no matrix inversion or solving optimization algorithms). A low prediction cost is extremely desirable in real time applications. Depending on a certain property called (k, e)-disjunct property of the group testing matrix A, the recovery algorithm can correct up to $\lfloor e/2 \rfloor$ errors in the prediction. We discuss various constructions for the group testing matrix A which have the desired (k, e)-disjunct property. We advocate the use of concatenated Reed Solomon codes (Kautz & Singleton, 1964), and unbalanced bipartite expander graphs (Vadhan, 2012) as the group testing matrix A. The optimal number of binary classifiers required for exact recovery (to form a (k, e)-disjunct matrix) will be $m = \Theta(k^2 \log_k d)$. However, we show how this can be reduced to $m = O(k \log d)$ if we tolerate a small ε error in the labels recovery.

The idea of grouping the labels helps overcome the issues most existing methods encounter; e.g., when the data has

power law distribution (Babbar & Schölkopf, 2017), that is many labels have very few training instances (which is the case in most popular datasets), and tail labels (Xu et al., 2016). Since the classifiers in our approach learn to test for groups of labels, we will have more training instances per group yielding effective classifiers. It is well known that the one-vs-rest is a highly effective method (expensive), and recently a (doubly) parallelized version of this method called DiSMEC (Babbar & Schölkopf, 2017) was shown to be very effective. Our approach is similar to one-vs-rest, but the classifiers test for a group of labels, and we require very few classifiers ($O(\log d)$ instead of d). Our approach also resembles the Bloom filter method (Cisse et al., 2013), which is based on using hash functions to reduce the label size. However, for Bloom filters the lower dimension m can be larger than $O(k \log d)$ (no bounds are established) and they may yield many false positives. For proper encoding this method requires clustering of the labels.

We establish Hamming loss error bounds for the proposed approach. Due to the error correcting capabilities of the algorithm, even if a fraction of classifiers mis-classify, we can achieve zero prediction error. Numerical experiments with various datasets illustrate the superior performance of our group testing approach with different GT matrices. Our method is extremely inexpensive compared to the CS approach and especially compared to the embedding based methods, making it very desirable for real time applications too. The results we obtain using the GT approach are more accurate compared to the other popular methods (in terms of Hamming distance). For many examples, the training errors we obtained were almost zero and the test errors were also quite low.

2. Preliminaries

Group testing. Formally, the group testing problem involves identifying an unknown k-sparse binary vector $y \in \{0,1\}^d$, such that $|\texttt{supp}(y)| \leq k$, where $\texttt{supp}(y) := \{i : y_i \neq 0\}$ is called the support of the vector y, by performing a small number of tests (measurements). In the MLC problem, we can view this vector as the sparse label vector y of the data (indicating the k labels).

A nonadaptive group testing scheme with m tests is described by an $m \times d$ binary matrix A, where each row corresponds to a test, and $A_{ij} = 1$ if and only if the ith test includes the jth element. The measured vector z is the boolean OR operation between the matrix A and the label vector y. The boolean OR operation $z = A \vee y$ can simply be obtained by setting every nonzero entry of the usual matrix-vector product Ay to 1 (and leaving the zero entries as they are). It can also be thought of as coordinate-wise Boolean OR of the columns of A that correspond to the nonzero entries of y.

Definition 1 (Disjunctness). *An $m \times d$ binary matrix A is called* k-disjunct *if the support of any of its columns is not contained in the union of the supports of any other k columns.*

A k-disjunct matrix gives a group testing scheme that identifies any defective set up to size k exactly.

Definition 2 (Error Correction). *An $m \times d$ binary matrix A is called* (k, e)-disjunct, *$e \geq 1$, (k-disjunct and e-error detecting) if for every set S of columns of A with $|S| \leq k$, and $i \notin S$, we have $|\texttt{supp}(A^{(i)}) \setminus \cup_{j \in S} \texttt{supp}(A^{(j)})| > e$, where $A^{(i)}$ denote the ith column of A.*

A (k, e)-disjunct matrix can detect up to e errors in the measurements and can correct up to $\lfloor e/2 \rfloor$ errors.

Several random and deterministic construction of k-disjunct matrices have been proposed in the literature (Kautz & Singleton, 1964; Du & Hwang, 2000). Matrices from error correcting codes and expander graphs have also been designed (Dyachkov et al., 2000; Ubaru et al., 2016; Cheraghchi, 2010; Mazumdar, 2016).

3. MLC via Group testing

In this section, we present our main idea of adapting the group testing scheme to the multilabel classification problem (MLGT).

Training. Suppose we are given n training instances $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ are the input features for each instances and $y_i \in \{0,1\}^d$ are corresponding label vectors. We begin by assuming that each data instance belongs to at most k classes (the label vector y is k sparse). We consider a (k, e)-disjunct matrix $A \in \mathbb{R}^{m \times d}$. We then compute the reduced measured (label) vectors z_i for each label vectors $y_i, i = 1, \ldots, n$ using the boolean OR operation $z_i = A \vee y_i$. We can now train m binary classifiers $\{w_j\}_{j=1}^m$ based on $\{x_i, z_i\}_{i=1}^n$ with jth entry of z_i indicating which class $(1/0)$ the ith instance belongs to for the jth classifier. Algorithm 1 summarizes our training algorithm.

Algorithm 1 MLGT: Training Algorithm

Input: Training data $\{(x_i, y_i)\}_{i=1}^n$, group testing matrix $A \in \mathbb{R}^{m \times d}$, a binary classifier algorithm $\mathcal{C}$.
Output: m classifiers $\{w_j\}_{j=1}^m$.
for $i = 1, \ldots, n$. **do**
  $z_i = A \vee y_i$.
end for
for $j = 1, \ldots, m$. **do**
  $w_j = \mathcal{C}(\{(x_i, z_{ij})\}_{i=1}^n)$.
end for

Prediction. In the prediction stage, given a test data $x \in \mathbb{R}^p$, we use the m classifiers $\{w_j\}_{j=1}^m$ to obtain a predicted

reduced label vector $\hat{z}$. We know that a k sparse label vector can be recovered exactly, if the group testing matrix A is a k-disjunct matrix. With a (k, e)-disjunct matrix, $e \geq 1$, we can recover the k sparse label vector exactly, even if $\lfloor e/2 \rfloor$ binary classifiers mis-classify, using the following decoder.

Decoder : Given a predicted reduced label vector $\hat{z}$, and a group testing matrix A, set the coordinate position of $\hat{y}$ corresponding to $l \in [1, \ldots, d]$ to 1 if and only if $|\texttt{supp}(A^{(l)}) \backslash \texttt{supp}(\hat{z})| < e/2$.

That is, we set the lth coordinate of $\hat{y}$ to 1, if the number of coordinates that are in the support of the corresponding column $A^{(l)}$ but are not in the predicted reduced vector $\hat{z}$, is less than $e/2$. The decoder returns the exact label vector even if up to $e/2$ binary classifiers make errors. Algorithm 2 summarizes our prediction algorithm.

Algorithm 2 MLGT: Prediction Algorithm

Input: Test data $x \in \mathbb{R}^p$, the GT matrix $A \in \mathbb{R}^{m \times d}$ which is (k, e)-disjunct ($e \geq 1$), m classifiers $\{w_j\}_{j=1}^m$.
Output: predicted label $\hat{y}$.
Compute $\hat{z} = [w_1(x), \ldots, w_m(x)]$.
Set $\hat{y} \leftarrow 0$.
for $l = 1, \ldots, d$ **do**
  if $|\texttt{supp}(A^{(l)}) \backslash \texttt{supp}(\hat{z})| < e/2$ **then**
    $\hat{y}_l = 1$.
  end if
end for

Note that the prediction algorithm is very inexpensive (requires no matrix inversion or solving optimization). It is equivalent to an AND operation between a binary sparse matrix and a binary (likely sparse) vector, which should cost less than a sparse matrix vector product $O(nnz(A)) \approx O(kd)$, where $nnz(A)$ is the number of nonzero entries of A. It is an interesting future work to design an even faster prediction algorithm.

4. Constructions

In order to recover a k sparse label vector exactly, we know that the group testing matrix A must be a k-disjunct matrix. With a (k, e)-disjunct matrix, our algorithm can extract the sparse label vector exactly even if $e/2$ binary classifiers make errors (mis-classify). Here, we present the results that will help us construct specific GT matrices with the desired properties.

4.1. Random Constructions

Proposition 1 (Random Construction)**.** *An $m \times d$ random binary $\{0, 1\}$ matrix A where each entry is 1 with probability $\rho = \frac{1}{k+1}$, is $(k, 3k \log d)$-disjunct with very high probability, if $m = O(k^2 \log d)$.*

If we tolerate a small ε fraction of sparsity label misclassifications (i.e., εk errors in the recovered label vector), which we call ε-tolerance group testing, then we can follow the analysis of Theorem 8.1.1 in (Du & Hwang, 2000), to show that it is sufficient to have $m = O(k \log d)$ number of classifiers. Further, we can derive the following result.

Theorem 1. *Suppose we wish to recover a k sparse binary vector $y \in \mathbb{R}^d$. A random binary $\{0, 1\}$ matrix A where each entry is 1 with probability $\rho = 1/k$ recovers $1 - \varepsilon$ proportion of the support of y correctly with high probability, for any $\varepsilon > 0$, with $m = O(k \log d)$. This matrix will also detect $e = \Omega(m)$ errors.*

The proofs of the proposition and the theorem can be found in the supplementary.

4.2. Concatenated code based constructions

Kautz and Singleton (Kautz & Singleton, 1964) introduced a two-level construction in which a q-ary ($q > 2$) Reed-Solomon (RS) code is concatenated with a unit-weight binary code. The construction starts with a q-ary ($q > 2$) RS code of length $q - 1$, and replaces the q-ary symbols in the codewords by unit weight binary vectors of length q. That is, the q-ary symbols are replaced as $0 \to 100 \ldots 0; 1 \to 010 \ldots 0; q - 1 \to 0 \ldots 01$. This gives us a binary matrix with $m = q(q - 1)$ rows. This matrix belongs to a broad class of error correcting codes called the *constant weight codes* (each codeword/column has a constant number of ones w). For this Kautz-Singleton construction, $w = q - 1$.

Proposition 2 (Kautz-Singleton construction)**.** *A Kautz-Singleton construction with $(k \log_k d)$-ary Reed-Solomon (RS) code is a $(k, (k - 1) \log_k d)$-disjunct matrix with $m = \Theta(k^2 \log_k^2 d)$.*

Proof. A constant weight code matrix is k disjunct matrix with $k = \lfloor \frac{w-1}{w-h/2} \rfloor$, where w is the weight and h is the distance of the code, (see, Theorem 7.3.3 in (Du & Hwang, 2000)). The distance of the q-ary RS code is $h = 2(q - \log_q(d))$. Hence, we get $k = \frac{q-2}{\log_q d - 1}$. So, for a k-disjunct matrix, we choose $q = k \log_k d$. A code with distance h will have $e = h/2$ (by using Corollary 8.3.2 in (Du & Hwang, 2000)). Thus, $e = q - \log_q d \approx (k - 1) \log_k d$. $m = q(q - 1) = \Theta(k^2 \log_k^2 d)$. □

Other code based constructions are given in supplementary.

4.3. Expander graphs

Expander graphs have popularly been used in many applications, for example, in coding theory (Sipser & Spielman, 1996), in compressed sensing (Jafarpour et al., 2009), etc. In an expander graph, every small set of vertices "expands": the are "sparse" yet very "well-connected" (see

formal definition below). With high probability a random graph is a good expander. Construction of "lossless" expanders have been notoriously difficult.

Definition 3 (Unbalanced Lossless Expander Graphs)**.** *A (k, ϵ)-unbalanced bipartite expander graph is a bipartite graph $G(L, R, E), |L| = d, |R| = m$, where L is the set of left nodes and R is the set of right nodes, with regular left degree ℓ such that for any $S \subset L$, if $|S| \leq k$ then the set of neighbors $N(S)$ of S has the size $N(S) > \epsilon\ell|S|$.*

The following proposition describe the expander property of random graphs.

Proposition 3. *A random construction of bipartite graphs $G(L, R, E)$ with $|L| = d$ with overwhelming probability, is (k, ϵ)-lossless ℓ-regular expander where $\ell = O(\log d/\epsilon)$ with $|R| = m = O(k\ell/\epsilon)$.*

The trade-off of this proposition is close to the best we can hope for. The proof can be shown by simple random choice and can be found in (Vadhan, 2012) or in (Cheraghchi, 2010).

The next definition and the subsequent two claims are from (Cheraghchi, 2010). First, let us now connect a lossless expander with disjunct matrix.

Definition 4. *A bipartite graph $G(L, R, E)$ is called (k, e)-disjunct if, for every left vertex $i \in L$ and every set $S \subseteq L$ such that $|S| \leq k$ and $i \notin S$, we have $|N(i) \backslash N(S)| > e$.*

It can be seen that the bipartite adjacency matrix A of a disjunct graph G is a disjunct matrix.

Proposition 4. *Let G be a (k, e)-disjunct graph with adjacency matrix A. Then for every pair of $y, y' \in \{0, 1\}^d$ of k -sparse vectors, we have $\Delta(A \vee y, A \vee y') > e$, where $\Delta(\cdot)$ denotes the Hamming distance between vectors.*

The following proposition relates expander graphs with disjunct graphs.

Proposition 5. *Let G be a ℓ-regular (k, ϵ)-lossless expander. Then, for every $\alpha \in [0, 1)$, G is $(k-1, \alpha\ell)$-disjunct provided that $\epsilon < \frac{1-\alpha}{\ell}$.*

Combining these comments, we get the following:

Proposition 6 (Random Graphs)**.** *The adjacency matrix of a randomly constructed bipartite graph is, with overwhelming probability, k-disjunct with $m = O(k^2 \log(d/k))$. More generally, for every $\alpha \in [0, 1)$, random graphs are (k, e)-disjunct, with $e = \Omega(\alpha k \log d/(1 - \alpha^2))$ with $m = \Omega(\alpha k^2 \log(d/k)/(1 - \alpha^2))$.*

There is an explicit construction of unbalanced (k, ϵ)-lossless expanders for any setting of d and m and is, to our knowledge, the best possible, in (Capalbo et al., 2002). These constructions yield explicit k-disjunct graphs with $m = O(k^2 \text{quasipoly}(\log d))$. Other random constructions are discussed in the supplementary.

With all the above constructions, we can correct a reasonably large number of e errors by the binary classifiers. The number of classifiers required for MLGT will be $m = O(k^2 \log d)$ which is more than the CS approach where $m = O(k \log d)$. However, our analysis is for the worst case: as we saw in Theorem 1, if we tolerate a small ε fraction of error in recovery, we can achieve $m = O(k \log d)$ for MLGT as well. Moreover, MLGT yields *zero* prediction error for a k sparse label vector even if up to $e/2$ classifiers mis-classify. With MLCS, we only get an ϵ error guarantees and with respect to 2-norm (not Hamming distance which is more natural for classification).

5. Error Analysis

Here we summarize the theoretical error guarantees for multilabel classification using group testing (MLGT).

Theorem 2. *Consider MLGT with an $m \times d$ binary matrix A, and a label vector y with sparsity at most k. Suppose A is (k, e)-disjunct, and we use Algorithm 2 during prediction. Let $\hat{y}$ be the predicted label vector and $\Delta(\cdot)$ denote the Hamming distance between vectors. If t number of binary classifiers that make errors in prediction, then we have*

- *If $t \leq \lfloor e/2 \rfloor$, then the prediction error $\Delta(y, \hat{y}) = 0$.*
- *If $t > \lfloor e/2 \rfloor$, $\Delta(y, \hat{y}) \leq w(t - e/2)$ (Hamming error), where w is the maximum weight of rows in A. In particular, the error rate (average error per class) will be $\frac{w}{d}(t - e/2)$.*

If A is a k-disjunct with ε error tolerance, then the prediction error will be at most $(w(t - e/2) + \varepsilon k)$.

Proof. When, $t \leq e/2$, we know that the decoding algorithm will still recover the exact label vector due to the error correcting property of the (k, e)-disjunct matrix. When, $t > e/2$, $e/2$ of the errors are corrected. For every remaining $t - e/2$ errors, if w is the maximum weight of rows in A, a maximum of w errors can occur in the predicted label. This is because, the support different $|A^{(l)} \backslash \hat{z}|$ can change for a maximum of w columns. Hence, the error can be at most $w(t - e/2)$, and the error rate will be $\frac{w}{d}(t - e/2)$. For the k-disjunct matrix with ε error tolerance, the decoding algorithm can make up to εk errors in addition to $w(t - e/2)$. □

Let us see how the error-rate of various group testing constructions translate to MLGT. In the case of a random matrix construction, we have $w \approx d/k$. So, the error rate for this matrix will be $(t - e/2)/k$. From proposition 1, we can take $m = k^2 \log d$, and $e = 3k \log d$. Hence, the error rate

for a random (k, e) disjunct matrix will be $t/k - 3/2 \log d$, for any $t > 3/2k \log d$. For any t less than this the error rate will be zero. Similarly, we can see that the randomized construction of Thm. 1 with $m = O(k \log d)$ rows, gives the average error rate is $(t/k - O(\log d) + \varepsilon k/d)$ for $t > k \log d$. The error rates of other constructions can be calculated in the same way, see supplementary.

The above theorem also shows that a Hamming error regret R ($R \equiv$ |error in the method - least error possible|) in the binary classifiers will transform linearly to the overall regret of at most $w(R - e/2)$ for the MLGT. For the CS approach, the results in (Hsu et al., 2009) show that a L_2 regret R_2 (an L_2-norm error regret) in the regressor will translate as $\sqrt{R_2}$ to the overall regret (which is worse). This is because, a CS based analysis involves, L_2-norm errors and Restricted Isometric Property (RIP) of the compression matrix with respect to L_2 norm. However, $L2$ error metric is never used for evaluation in practice.

Hamming Loss: Since we operate in the binary field, we derive error (regret) bounds, as well as present experimental results with respect to Hamming loss. In certain applications we may be interested in only predicting the top $k_1 < k$ labels correctly, e.g., tagging and recommendation. In such situations, it has been argued that the Hamming loss is not a perfect measure (Jain et al., 2016), and alternate measures such as $Precison@k$ (defined later) for $k = 1, 3, 5$ have been used (Agrawal et al., 2013). However, these measures assume there is a ranking amongst the labels, which can be obtained only when operating in the real space. Also, these measures ignore the false labels. Our approach considers labels as just binary vectors (cannot rank) and attempts to predict all labels correctly (hence Hamming loss). Almost all available mutlilabel datasets have binary label vectors and do not come with the priority information of labels within the large output classes. There are recent works which try to rank the labels first and then classify, e.g. (Jain et al., 2016; Chzhen et al., 2017). Hamming loss is nonetheless an interesting error metric for applications where we need to predict all labels correctly and require few/no false labels, hence is worth analyzing. Most of the recent popular (embedding) methods tend to give good results with respect to $Precison@k$, but give poor Hamming errors due to large number of false labels. Our approach gives very low Hamming loss both theoretically and practically.

6. Numerical Experiments

In this section, we illustrate the performance of the proposed group testing approach in the multilabel classification problems (MLGT) via several numerical experiments on various datasets.

Datasets: We use some popular publicly available multilabel datasets in our experiments. All datasets were obtained from The Extreme Classification Repository (Bhatia et al., 2015). Details about the datasets and the references for their original sources can be found in the repository. Table 1 in the supplementary gives data details.

Constructions: For MLGT, we consider three different group testing constructions. The Kautz-Singleton construction with q-ary Reed-Solomon (RS) codes, where we use RS codes (MacWilliams & Sloane, 1977) with $q = 16$ and $m = 240$; and $q = 8$ and $m = 56$. To get desired number of codewords (equal to number of labels), we use appropriate message length. For example, if $d \leq 4096, q = 16$, then we use message length of 3, and if $d \leq 65536$, we use message length of 5. We also use two random GT constructions, namely, the random expander graphs and the sparse random constructions discussed in sec. 4. For MLCS (compressed sensing approach), we again consider three different types compression matrices, namely, random Gaussian matrices, compressed Hadamard matrices and random expander graphs (expander graphs have been used for CS too (Jafarpour et al., 2009)).

Evaluation metrics: Two evaluation metrics are used to analyze the performances of the different methods. First is the Hamming loss error, the Hamming distance between the predicted vector $\hat{y}$ and the actual label vector y, $\Delta(y, \hat{y})$. This metric tells us how close is the recovered vector $\hat{y}$ is from the exact label vector y, and is more suitable for binary vectors. Hamming loss captures the information of both correct predictions and false labels. All prediction errors reported (training and test) are Hamming loss errors. The second metric used is $Precison@k$ (P@k), which is a popular metric used in MLC literature (Agrawal et al., 2013). This measures the precision of predicting the first k coordinates $|\texttt{supp}(\hat{y}_{1:k}) \cap \texttt{supp}(y)|/k$. Since we cannot score the labels, we use $k = nnz(y)$ the output sparsity of the true label for this measure. This is equivalent to checking whether the method predicted all the labels the data belongs to correctly or not (ignoring misclassification). When $Precision@k$ for $k = 1, 3, 5$ are used, one is checking whether the top 1, 3 or 5 labels are predicted correctly (ignoring other and false labels).

MLGT vs MLCS: In the first set of experiments, we compare the performances of the group testing approach (MLGT) and the compressed sensing approach (MLCS) using different group testing constructions and different compression matrices. A least squares binary classifier was used $\{w_j\}_{j=1}^m$ for MLGT. Least squares regression with ℓ_2 regularization (ridge regression) is used as the regressors for MLCS and other embedding based methods. Orthogo-

https://manikvarma.github.io/downloads/XC/XMLRepository.html

Table 1. Comparison between MLGT and MLCS: Average training and test errors and $Precison@k$.

Data	Method	Size m	Training error	Train P@k	Test Error	Test P@k
RCV1-2K, $d = 2456, k_{\max} = 10, n = 2000, nt = 501, p = 6000$.	GT: RS code q=16	240	**0.0280**	**0.9972**	2.106	0.59
	GT: expander	240	0.0375	0.9962	2.075	**0.624**
	GT: sparse rand	240	0.0385	0.9961	**2.059**	0.616
	CS: Gaussian	240	1.9970	0.8003	2.84	0.1875
	CS: Hadamard	240	1.9640	0.8036	2.60	0.1858
	CS: Expander	240	0.762	0.9238	3.52	0.1738
EURLex-4K, $d = 3993, \bar{k} = 5.03, n = 2500, nt = 500, p = 5000$.	GT: RS code q=16	240	0.0320	0.9949	**5.420**	**0.4573**
	GT: expander	240	**0.0284**	**0.9959**	5.526	0.4313
	GT: sparse rand	240	0.0308	0.9955	5.584	0.4241
	CS: Gaussian	240	0.7136	0.9238	6.206	0.2522
	CS: Hadamard	240	0.7136	0.9238	8.140	0.264
	CS: Expander	240	0.7136	0.9238	6.790	0.2666
Delicious, $d = 983, k_{\max} = 12, n = 1580, nt = 393, p = 500$.	GT: expander	200	6.3670	0.3438	8.666	0.4144
	GT: sparse rand	200	**6.2960**	0.3494	**8.503**	**0.4199**
	CS: Gaussian	200	13.590	0.6360	17.190	0.3014
	CS: Hadamard	200	13.408	**0.6572**	17.842	0.3297
	CS: Expander	200	13.417	0.6036	18.318	0.2827
AmazonCat-13K, $d = 13330, \bar{k} = 4.3, n = 3000, nt = 1200, p = 10000$.	GT: RS code q=16	240	0.0385	0.9986	4.260	0.414
	GT: expander	240	**0.0160**	0.9995	**4.234**	**0.4263**
	GT: sparse rand	200	0.0165	0.9994	4.238	0.4127
	CS: Gaussian	200	5.7135	**1.0**	11.962	0.1349
	CS: Hadamard	200	5.7115	**1.0**	12.318	0.1539
	CS: Expander	200	5.7115	**1.0**	12.698	0.1605
Wiki10-31K,$d = 30938$,$\bar{k} = 5.6$, $n = 3500$,$nt = 1500, p = 10000$.	GT: expander	300	**0.018**	**0.98**	**4.94**	**0.32**
	CS: Gaussian	300	4.313	0.89	11.22	0.09

Table 2. MLGT v/s OvsA

		MLGT			OvsA		
Dataset	d	Err	P@k	time	Err	P@k	time
Medmill	101	**0.68**	**0.24**	5.9s	2.07	0.17	13.1s
Bibtex	159	**1.17**	0.11	14.1s	2.55	**0.16**	36.6s

nal Matching Pursuit (OMP) (Tropp & Gilbert, 2007) was used for sparse recovery in MLCS. Additional details are given in supplementary.

Table 1 compares the performances of MLCS and MLGT for different CS and GT matrices on different datasets. The average training and the test errors (Hamming losses) are reported along with the average $Precison@k$ obtained for training and test data. The methods and the number of classifiers/regressors m used are also listed. For example, GT:RS code q=16, implies MLGT was the method with the RS code construction with q=16. The number of training points n, test points nt and the number of features p used in the experiments are reported next to the datasets. $k_{\max}$ means the maximum sparsity in the data (only those data points below this sparsity were used), anf $bar k$ is the average sparsity. For the latter three datasets, the feature space was reduced to select only dominant features (data are very sparse and only few features are prominent).

We observe that, the MLGT method with all the three GT constructions outperforms the MLCS method. In most cases, the training errors (almost zero) and $Precison@k$ (almost one) for MLGT methods are extremely good. This is because, the binary classifiers are optimally trained on the reduced binary vectors and since the matrices used were k-disjunct, we had zero recovery error in most cases. Hence, the predicted labels for training data were extremely accurate. The results on test data are also better for MLGT in almost all cases. The results we obtained for the dataset Delicious were consistently poor, see supplementary. We also observed that MLGT is significantly faster than MLCS (as expected) because, MLCS uses an optimization algorithm (OMP) for recovery of labels, see Table 3 for runtimes.

In the prediction algorithm of MLGT, we have a parameter e, the number of errors the algorithm should try to correct. The ideal value for e will depend on the GT matrix used, the values of m, k and d. However, note that we can test for different values of e at no additional cost. That is, once we compute the Boolean AND between the predicted reduced vector and the GT matrix (the dominant operation), we can get different prediction vectors for a range of e and choose an e that gives the highest training P@k.

Figure 1 plots the average training and test errors and average $Precison@k$ against the sparsity k of the label vectors (data with label sparsity k used) obtained for MLGT and MLCS methods with the three different matrices respectively, seen in Table 1. The dataset used was RCV1-2K. This dataset has at least 2000 training points and 500 testing points for each label sparsity ranging from 1 to 10. We observe that the training error for MLGT methods are almost zero and training $Precison@k$ almost one. (This behavior was seen in Table 1 as well). Results with test data

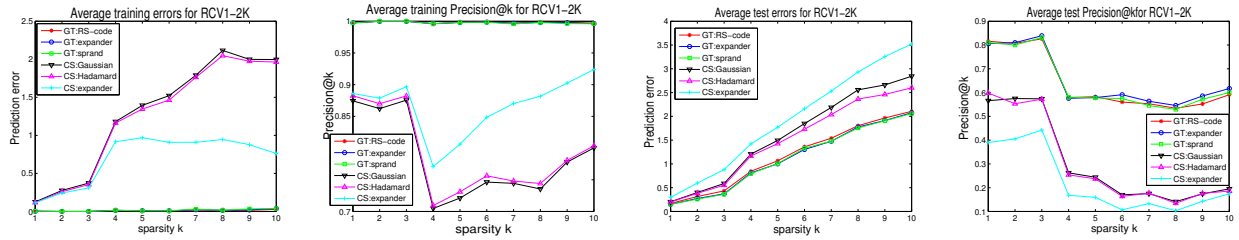

Figure 1. Average training and test errors and $Precison@k$ versus sparsity k for RCV1-2K for different MLGT and MLCS methods.

Table 3. Comparisons with embedding methods. Average Test errors.

		MLGT		MLCS		ML-CSSP		PLST		SLEEC	
Dataset	m	Err	time	Err	time	Err	time	Err	time	Err	time
Mediamill($d = 101, \bar{k} = 4.3$)	40	**2.267**	2.35s	4.096	4.04s	4.44	13.4s	10.10	**2.13s**	4.374	4.55s
Bibtex ($d = 159, \bar{k} = 2.4$)	50	**1.571**	**7.81s**	2.926	14.7s	5.63	23.9s	7.39	12.1s	4.851	38.3s
Delicious ($d = 983, \bar{k} = 12$)	150	4.995	18.1s	9.432	29.9s	5.66	47.7s	15.66	**17.3s**	**4.790**	18.7s
RCV2K($d = 2456, \bar{k} = 5$)	200	**1.141**	**154.3s**	4.370	387.8s	24.60	339.5s	20.98	290.9s	4.944	210.0s
EurLex($d = 3993, \bar{k} = 5.3$)	200	**3.033**	**160.6s**	8.554	367.1s	9.30	337.3s	15.40	297.2s	4.838	455.1s
AmznC($d = 13330, \bar{k} = 4.3$)	250	**4.496**	**12min**	11.07	22min	13.66	19min	7.502	18min	6.923	1.2hrs
Wiki10($d = 30938, \bar{k} = 5.6$)	300	**5.586**	**283s**	14.224	18min	8.30	17min	15.15	17min	6.621	54min

for MLGT are also impressive, achieving $Precison@k$ of almost 0.8 for small k.

One vs all: We next compare MLGT against the one versus all (OvsA) method on two small datasets. Note that OvsA required d classifiers to be trained, hence is impractical for larger datasets, and we will need a distributed implementation such as DiSMEC (Babbar & Schölkopf, 2017). Table 2 gives the results for MLGT and OvsA methods for two small datasets. $n = 5000$,$nt = 1000$, and for MLGT $m = 50$. The table lists the Hamming test errors and P@k for the two methods. The table also gives the overall runtimes for the two methods. We note that wrt. to both metrics, MLGT performs better than OvsA. This is due to two reasons. First, MLGT groups the labels hence has more training samples per group, yielding better classifiers. Second, the error correction by the prediction algorithm corrects few classification errors. Clearly, MLGT is faster than OvsA. However, OvsA gives better training errors.

Embedding methods: In the next set of experiments, we compare the performance of MLGT against the popular embedding based methods. We compare with the following methods. ML-CSSP, is an embedding method based on column subset selection (Bi & Kwok, 2013). PLST, is Principal Label Space Transformation (Tai & Lin, 2012), an embedding method based on SVD (code is made available online by the authors). SLEEC, Sparse Local Embeddings for Extreme Classification (Bhatia et al., 2015), is the state of the art embedding method based on clustering using nearest neighbors and then embedding in the cluster space (code is made available online by the authors). For MLGT, we use the random expander graph constructions. For MLCS, we use random Gaussian matrices. Same least squares regressor was used in all the latter four methods.

Table 3 lists the test (Hamming) errors obtained for the different methods on various datasets. We use smaller datasets since the embedding based methods do not scale well for large datasets. We also used only 2000 training points and 500 test points in each cases. We observe that MLGT outperforms the other methods in most cases. The datasets have very sparse label (avg. sparsity of around $\bar{k} \approx 4$), but the outputs of MLCSSP and PLST are not very sparse. Hence, we see high Hamming error for these two methods, since they yield a lot of false labels. Moreover, these embedding methods are significantly more expensive than MLGT for larger datasets. The runtimes for each method are also listed in the table.

The runtimes reported (using cputime in Matlab) includes generation of compression matrices, multiplying the matrix to the label vectors (boolean OR/SVD computation), training the m classifiers, and prediction of n training and nt test points. SLEEC performs reasonably well on all datasets (the ideal parameters to be set in this algorithm for each of these datasets were provided by the authors online), and gives better P@k than MLGT for some datasets. For Delicious dataset, the value of k is high and SLEEC beats MLGT. However, SLEEC algorithm has many parameters to set, and for larger datasets, the algorithm is very expensive compared to all other methods.

These experiments illustrate that MLGT performs exceptionally well in practice. The concatenated RS codes and the bipartite expander graphs constructions proposed are simple to generate and they exist for large sizes. Hence, these constructions can be easy applied to extreme classification problems.

Acknowledgements

Authors would like to thank Dr. Manik Varma and his team for making many MLC datasets and codes available online. This work was supported by NSF under grant NSF/CCF-1318597, NSF/CCF-1318093, NSF/CCF 1642550.

References

Agrawal, Rahul, Gupta, Archit, Prabhu, Yashoteja, and Varma, Manik. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 13–24. ACM, 2013.

Babbar, Rohit and Schölkopf, Bernhard. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 721–729. ACM, 2017.

Barutcuoglu, Zafer, Schapire, Robert E, and Troyanskaya, Olga G. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

Bhatia, Kush, Jain, Himanshu, Kar, Purushottam, Varma, Manik, and Jain, Prateek. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pp. 730–738, 2015.

Bi, Wei and Kwok, James Tin Yau. Efficient multi-label classification with many labels. In *30th International Conference on Machine Learning, ICML 2013*, pp. 405–413, 2013.

Capalbo, Michael, Reingold, Omer, Vadhan, Salil, and Wigderson, Avi. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 659–668. ACM, 2002.

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

Cheraghchi, Mahdi. Derandomization and group testing. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pp. 991–997. IEEE, 2010.

Chzhen, Evgenii, Denis, Christophe, Hebiri, Mohamed, and Salmon, Joseph. On the benefits of output sparsity for multi-label classification. *arXiv preprint arXiv:1703.04697*, 2017.

Cisse, Moustapha M, Usunier, Nicolas, Artieres, Thierry, and Gallinari, Patrick. Robust bloom filters for large multilabel classification tasks. In *Advances in Neural Information Processing Systems*, pp. 1851–1859, 2013.

Dietterich, Thomas G. and Bakiri, Ghulum. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1995.

Dorfman, Robert. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14(4):436–440, 1943.

Du, D. Z. and Hwang, F.K. *Combinatorial group testing and its applications*. World Scientific, 2nd edition, 2000.

Dyachkov, Arkadii G, Macula, Anthony J, and Rykov, Vyacheslav V. New applications and results of superimposed code theory arising from the potentialities of molecular biology. In *Numbers, Information and Complexity*, pp. 265–282. Springer, 2000.

Hsu, Daniel, Kakade, Sham M, Langford, John, and Zhang, Tong. Multi-label prediction via compressed sensing. *NIPS*, 22:772–780, 2009.

Jafarpour, Sina, Xu, Weiyu, Hassibi, Babak, and Calderbank, Robert. Efficient and robust compressed sensing using optimized expander graphs. *IEEE Transactions on Information Theory*, 55(9):4299–4308, 2009.

Jain, Himanshu, Prabhu, Yashoteja, and Varma, Manik. 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*, pp. 935–944. ACM, 2016.

Kapoor, Ashish, Viswanathan, Raajay, and Jain, Prateek. Multilabel classification using bayesian compressed sensing. In *Advances in Neural Information Processing Systems*, pp. 2645–2653, 2012.

Kautz, W and Singleton, Roy. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10(4):363–377, 1964.

MacWilliams, Florence Jessie and Sloane, Neil James Alexander. *The theory of error-correcting codes*. Elsevier, 1977.

Mazumdar, Arya. Nonadaptive group testing with random set of defectives. *IEEE Transactions on Information Theory*, 62(12):7522–7531, Dec 2016.

Mazumdar, Arya and Mohajer, Soheil. Group testing with unreliable elements. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pp. 1–3. IEEE, 2014.

Sipser, Michael and Spielman, Daniel A. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996.

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

Trohidis, Konstantinos. Multi-label classification of music into emotions. In *9th International Con- ference on Music Information Retrieval*, pp. 325– 330, 2008.

Tropp, Joel A and Gilbert, Anna C. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666, 2007.

Tsfasman, Michael A, Vlădu, Serge G, and Nogin, Dmitry. *Algebraic geometric codes: basic notions*. Number 139. American Mathematical Soc., 2007.

Tsoumakas, Grigorios, Katakis, Ioannis, and Vlahavas, Ioannis. Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD08)*, 2008.

Ubaru, Shashanka, Mazumdar, Arya, and Barg, Alexander. Group testing schemes from low-weight codewords of BCH codes. In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pp. 2863–2867. IEEE, 2016.

Vadhan, Salil P. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336, 2012.

Wang, Changhu, Yan, Shuicheng, Zhang, Lei, and Zhang, Hong-Jiang. Multi-label sparse coding for automatic image annotation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1643–1650. IEEE, 2009.

Xu, Chang, Tao, Dacheng, and Xu, Chao. Robust extreme multi-label learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1275–1284. ACM, 2016.

Yu, Hsiang-fu, Jain, Prateek, Kar, Purushottam, and Dhillon, Inderjit. Large-scale multi-label learning with missing labels. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 593–601, 2014.

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

Supplementary material: Multilabel Classification with Group Testing and Codes

A. Constructions

We present some additional group testing constructions in this section.

A.1. Random Constructions-Proofs

Proposition. *(Random Construction Prop. 1.) An $m \times d$ random binary $\{0,1\}$ matrix A where each entry is 1 with probability $\rho = \frac{1}{k+1}$, is $(k, 3k\log d)$-disjunct with very high probability, if $m = O(k^2 \log d)$.*

Proof. For the case of $e = 1$, the proof follows from Theorem 8.1.3, Corollary 8.1.4 and 8.1.5 in (Du & Hwang, 2000). The bound on the number of classifiers is $m \leq 3(k+1)^2 \log d$ and the probability is $(k+1)\binom{d}{k+1}[1 - \frac{1}{k+1}(1-\frac{1}{k+1})^k]^m$.

To show that the matrix is also (k,e)-disjunct with high probability, we have to just show that $|\texttt{supp}(A^{(i)}) \setminus \texttt{supp}(A^{(j)})| > e$ for any two distinct columns $A^{(i)}$ and $A^{(j)}$ (Corollary 8.3.2 in (Du & Hwang, 2000)). For any two fixed columns, $|\texttt{supp}(A^{(i)}) \setminus \texttt{supp}(A^{(j)})|$ is a binomial random variable $\text{Bin}(m, \frac{k}{(k+1)^2})$. If we choose, $m = (3+\epsilon)(k+1)^2 \log d, \epsilon > 0$, we find the expectation of this variable to be $3k\log d$. Therefore we can choose $e = 3k\log d$, and the matrix is going to be (k,e)-disjunct with high probability. □

Theorem. *(Restating Theorem 1.) Suppose we wish to recover a k sparse binary vector $y \in \mathbb{R}^d$. A random binary $\{0,1\}$ matrix A where each entry is 1 with probability $\rho = 1/k$ recovers $1-\varepsilon$ proportion of the support of y correctly with high probability, for $\varepsilon > 0$, for $m = O(k\log d)$. This matrix will also detect $e = \Omega(m)$ errors.*

Proof. This is a modification of Theorem 1 in (Mazumdar & Mohajer, 2014). Suppose, $T \subset [d]$ is the set of defectives. The recovery will be successful as long as we return a set T' such that $r \equiv |T \cap T'| \geq (1-\varepsilon)|T|$.

Our object of interest is the probability of error P_e, the probability of existence of a pair T and T', $|T|, |T'| \leq k$ such that less than e tests fails to distinguish this pair, where $r < (1-\varepsilon)|T|$.

We assume the testing matrix A is chosen randomly from the ensemble of all $m \times d$ matrix in the following way. Each entry of A is 1 with probability $\rho \equiv \frac{1}{k}$, and it is zero with the remaining probability. In other words, in each test we include an item with probability ρ. We will show that the probability of error P_e in this case is $o(1)$ which will implies existence of a matrix A that achieves P_e of $o(1)$.

The probability that any one test will be successful to distinguish between T and T' is therefore (here we are taking the sizes of T and T' exactly equal to k and not less than equal to, which is permissible without much loss of generality),

$$2(1-\rho)^k(1-(1-\rho)^{k-r}) = 2\Big(1-\frac{1}{k}\Big)^k\Big(1-\Big(1-\frac{1}{k}\Big)^{k-r}\Big)$$
$$\geq 2\cdot 3^{-1}\left(1-\exp\Big(-\frac{k-r}{k}\Big)\right) \geq \frac{2}{3}\Big(1-\exp(-\varepsilon)\Big),$$

where in the second line we have used inequalities $1-x \leq \exp(-x)$ for all x and $1-x \geq 3^{-x}$ for any $x \leq 0.17$ (which is true for any $k \geq 6$). We have also used the fact that $r < (1-\varepsilon)|T| \leq (1-\varepsilon)k$.

Hence the probability that A successfully distinguish between T and T' in less than e tests is,

$$\sum_{i\leq e}\binom{m}{i}\Big(1-\frac{2}{3}\Big(1-\exp(-\varepsilon)\Big)\Big)^{m-i}\Big(\frac{2}{3}\Big(1-\exp(-\varepsilon)\Big)\Big)^i.$$

This probability is going to be upper bounded by $\exp(-\delta m)$ whenever $e < \frac{2}{3}(1-\exp(-\varepsilon))m$ for some $\delta > 0$. Therefore, for this ensemble,

$$P_e \leq \Big(\sum_{i=0}^{k}\binom{d}{i}\Big)^2 \exp(-\delta m) \to 0,$$

for an m such that $m = O(k\log d)$. Therefore $e = O(k\log d)$. □

A.2. Concatenated code based constructions

Many code based constructions have been proposed with the optimal length of $m = \Theta(k^2 \log_k d)$ (Mazumdar, 2016). One such code based construction of interest is the Algebraic-Geometric codes.

Considering $q = r^2$, where r is an integer, using the results in (Tsfasman et al., 2007), we can generate a family of Algebraic-Geometric (AG) codes of length m_q, satisfying $m_q \geq r^{a+1} - r^a + 1$, where a is an even integer. Using the Kautz-Singleton mechanism, we can convert this AG code to a binary code that has constant weight $w = m_q$. The length of the binary code will be $m = qm_q$.

Proposition 7. *We can construct an Algebraic-Geometric code matrix that recovers $1-\varepsilon$ proportion of nonzeros in y with high probability, for $\varepsilon > 0$, with $m \geq 16k\log_{2k} d\log(d/\varepsilon)$. This matrix will also detect $e = \left(8\log(d/\varepsilon) - \frac{8\log(d/\varepsilon)}{\sqrt{2k}-1} - 1\right)\log_{2k} d$ errors.*

Proof. The proof follows from the results developed in (Mazumdar, 2016). For a q-ary Algebraic-Geometric Code with $q \geq 2k$, that is converted to a binary code using Kautz-Singleton mechanism, we have the $1-\varepsilon$ recovery guarantees for $m \geq \frac{16k\log d}{\log 2k}\log(d/\varepsilon)$. We know if the code has a distance h, then $e = h/2$. The q-ary AG code satisfies

$$h \geq 2m/q - 2\log_q d - \frac{2m}{q(\sqrt{q}-1)}.$$

We get the value for e upon substitution. □

For MLGT, we have the following results for different constructions:

- If A is constructed via randomized construction of Prop. 1 with $m = O(k^2\log d)$ rows, then the average error rate is $t/k - \frac{3}{2}\log d$ for $t > 3/2\log d$.
- If A is constructed via randomized construction of Thm. 1 with $m = O(k\log d)$ rows, then the average error rate is $(t/k - O(\log d) + \varepsilon k/d)$ for $t > k\log d$.
- If A is constructed deterministically via Kautz-Singleton Reed-Solomon codes construction of Prop. 2 with $m = O(k^2\log_k d)$ rows, then the average error rate is $\frac{t}{k\log_k d} - O(1)$ for $t > k\log_k d$.
- If A is constructed via expander graph-based construction of Prop. 6 with $m = O(k^2\log(d/k))$ rows, then the average error rate is $t/k - \log(d/k)$ for $t > k/2\log(d/k)$.

The error rate is zero for smaller number of misclassifications t.

B. Experiments

Datasets: We use some popular publicly available multilabel datasets in our experiments. All datasets were obtained from The Extreme Classification Repository (Bhatia et al., 2015). Details about the datasets and the references for their original sources can be found in the repository. Table 4 gives the statistics of these datasets. In the table, d = #labels, $\bar{k}$ =average sparsity per instance, n = #instances and p = #features.

Table 4. Dataset statistics

Dataset	d	$\bar{k}$	n	p
Mediamill	101	4.38	30993	120
Bibtex	159	2.40	4880	1839
Delicious	983	19.03	12920	500
RCV1-2K	2456	4.79	623847	47236
EurLex-4K	3993	5.31	15539	5000
AmazonCat-13K	13330	5.04	1186239	203882
Wiki10-31K	30938	18.64	14146	101938

Details of the experiments:

- We use simple least squares binary classifiers for training and prediction in MLGT. This is because, this classifier is extremely simple and fast. Also, we use least squares regressors for other compared methods (hence, it is a fair comparison). We note that MLGT performs well with this simple classifier. We can improve the performance of MLGT further by using a more advanced classifier.
- In the prediction algorithm of MLGT, we have a parameter e, the number of errors the algorithm should try to correct. The ideal value for e will depend on the GT matrix used, the values of m, k and d. However, note that we can test for different values of e at no additional cost. That is, once we compute the Boolean AND between the predicted reduced vector and the GT matrix (the dominant operation), we can get different prediction vectors for a range of e and choose an e that gives the highest training P@k.
- The Orthogonal Matching Pursuit (OMP) algorithm used for MLCS is as implemented by the SPAMS library http://spams-devel.gforge.inria.fr/.
- Many of the datasets have very sparse feature matrices. In such cases, we reduced the feature dimension by choosing only the prominent features. That is the features that have nonzero values for at least half of the considered training points.
- The results we obtained for the dataset Delicious were consistently poor. This is because, the average sparsity for this dataset is $\bar{k} = 19.03$. We selected data instances with sparsity at most $k_{\max} = 12$. Still, the GT matrices used were not k-disjunct for this case. Also, the feature dimension is small ($p = 500$). Results with CS for this data were poor as well in terms of Hamming loss. However, the precision was better.
- For AmazonCat-13K, the training precision for CS method is perfect. However, the Hamming error is poor because they returned many false classes.
- The runtimes reported in the main paper (using cputime in Matlab) includes generation of compression matrices, multiplying the matrix to the label vectors (boolean OR/SVD computation), training the m

https://manikvarma.github.io/downloads/XC/XMLRepository.html

classifiers, and prediction of n training and nt test points. All runtime experiments were conducted on an Intel Core i7-5557U CPU @ 3.10GHz machine.

- The runtimes were averaged over 3 trials for smaller datasets (first 4). So were the errors. But for larger datasets (last 2), results for just one trial is reported. For larger datasets, our MLGT runtime is almost half of the next best timing and yields the lowest error.

- For SLEEC, there are seven parameters to be tuned. We set these parameters to the values provided by the authors online. The ideal parameters to be set in this algorithm for each of the datasets we used (expect RCV1-2k) were provided by the authors online. In general, we found SLEEC to be very expensive for large datsets. Also, there is no procedure to select these seven parameters and are seem to be selected in a trial and error fashion.