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 $|\mathtt{supp}(A^{(i)}) \setminus \mathtt{supp}(A^{(j)})| > e$ for any two distinct columns $A^{(i)}$ and $A^{(j)}$ (Corollary 8.3.2 in (Du & Hwang, 2000)). For any two fixed columns, $|\mathtt{supp}(A^{(i)}) \setminus \mathtt{supp}(A^{(j)})|$ is a binomial random variable $\mathrm{Bin}(m, \frac{k}{(k+1)^2})$. 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 $m = q m_q$.

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[2] (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.

Details of the experiments:

[2]https://manikvarma.github.io/downloads/XC/XMLRepository.html

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

- 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

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.