Group testing with features
A project for the course "Mathématiques des données" (G. Peyré, ENS Ulm)

Linus Bleistein - linus.bleistein@ens.fr

Abstract

Our project for this class aims at introducing prior information on tested individuals in the classical framework of nonadaptative group testing. We use group testing to improve the performance of a classifier by testing the individuals lying close to the classification threshold, i.e. for which the classifier predicts a class with little confidence. We first give a rapid tour of group testing, presenting its fundamental notions and ideas. We then test our method on synthetic datasets. Code is available on Github.

1 INTRODUCTION

Group testing is a method of testing large populations through binary tests conducted on subsamples of the population. In its most classical form, one considers an unknown vector

$$\mathbf{x} = (x_1, \dots, x_N) \in \{0,1\}^N , \tag{1}$$

one wishes to recover through binary tests, which consist in multiplications of $\mathbf{x}$ with vectors $\mathbf{b}$ in $\{0,1\}^N$. The result of a test is 1 if their product is greater or equal than 1, and 0 otherwise, i.e.

$$\text{Result}(\mathbf{x}, \mathbf{b}) := \mathbb{1}_{\mathbf{x}\cdot\mathbf{b}\geq 1}.$$

This testing procedure gives rise to the name of "group testing", since it can be interpreted as conducting a single binary test on a subgroup of the population (corresponding to the individuals i for which $\mathbf{b}_i$ is equal to 1), which is positive only if there is at least one positive individual within the subgroup. The central motivation of group testing lies in reducing the number of tests need to identify the "defective set", i.e. the subgroup of positive individuals. Indeed, if the vector $\mathbf{x}$ is very sparse, one can drastically reduce the number of tests needed to recover $\mathbf{x}$: instead of conducting n individual tests, one could, by carefully designing tests, form subgroups and with a few tests determine the status of numerous individuals. Group testing is thus usually studied in contexts where $\mathbf{x}$ is sparse; some hypothesis on the Hamming weight of $\mathbf{x}$ are often made, usually of the form $K \ll N$, where K is the Hamming weight of $\mathbf{x}$.
Group testing was originally introduced by [4] during World War 2 for syphilis detection. As numerous soldiers had to be tested for *treponema pallidum* (see figure 1) and the prevalence among soldiers was quite low, Dorfman designed this idea to reduce the number of tests need and thus the testing costs for the American Army: one could simply pool blood samples for a selected subrgroup of soldiers, mix those samples and conduct one single, that would in best cases yield a negative result for the whole group [1]. Group testing has also recently drawn huge attention in the mist of the COVID-19 pandemic, some researchers and scientific experts arguing that it might simultaneously tackle the problem of testing costs and testing capacities[2].

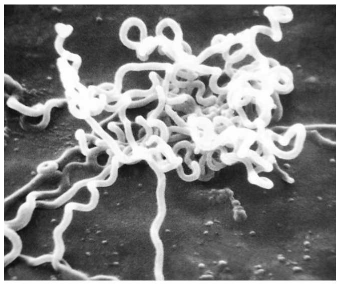

Figure 1. The bacteria *Treponema pallidum*, which causes the STD syphillis, and for whose detection group testing was originally introduced.

Our project is made of two sections. First, we implement the main reconstruction algorithms described in [2] and test them in various contexts. In a second time,

[1]Interestingly, Robert Dorfman (1916-2002) was a failed poet; he turned to mathematics after realising he had no future in poetry. After several major contributions to statistics - he is one of the pioneers of the famous *delta method* - he changed course once again and became an economics professor at Harvard.

[2]See for instance this project by INRIA Lille Nord-Europe.

we try to combine group testing with classifiers, trained on features that characterise the tested individuals. Relying on features, classifiers - one can think of a basic logistic regression - usually compute a score (a real number) and classify an individual as 0 if this score is bellow a given threshold and as 1 otherwise. However, depending of the structure of the features (for instance, their linear separability), classifiers may classify certain samples with high confidence and make more hesitant predictions for others whose score lies close to the decision threshold.
Our idea is to use a two stage procedure. First, we train a classifier on a badly separable dataset. We then incorporate the predictions of the classifier in the construction of the test matrix.
This pipeline can be interpreted in the context of the COVID-19 pandemic as a two stage testing procedure. Individuals arriving in a testing facilities are first asked a set of questions about their symptoms, their recent social contacts and their respect of social distancing measures. In some cases - think for instance of somebody who has clear COVID-19 symptoms, has frequent and intimate contact with a sick person, uses public transportation and does not wash hands regularly - the medical staff, relying on labelled data, may decide that it is not necessary to test this individual and declare it positive. They may take the same decision in opposite cases: somebody who lives in complete isolation and has no symptoms can be confidently declared negative. Individuals whose status is unclear may then be grouped into subgroups for group testing. Put otherwise, our method is aimed at improving classifier performance on badly separable datasets in which some polar cases can be easily classified, but most of the data lies close to the decision boundary (see figure 2 for illustration).

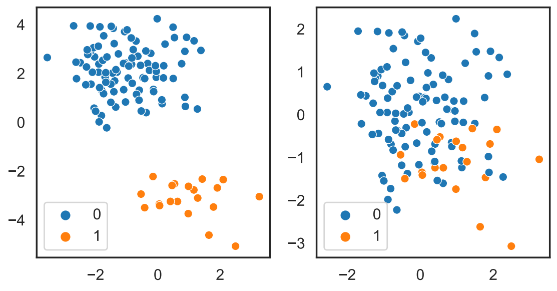

Figure 2. Easily separable data (left), for which our method is not very useful, and confused data (right), for which it was designed.

Our project report is structured as follows.

1. Section 2 gives a short presentation of the main ideas and frameworks of group testing.
2. Section 3 deals with the so called decoding or reconstruction algorithms, which reconstruct the vector $\mathbf{x}$ given the performed tests and the result vector.
3. In section 4, we describe our classification pipeline and test it on synthetic using a simple classifier.

The appendix provides some additional information on classification metrics, supplementary plots and pseudo-code for the algorithms. The code used in this project is available on Github.

2 SETUP AND SOME RESULTS ON GROUP TESTING

2.1 Notations and principal distinctions

We consider n individuals and write $\mathcal{K} \subset \{1, \ldots, n\}$ the set of defectives, i.e. sick individuals. The vector $\mathbf{x} \in \{0,1\}^n$ summarises the individuals status: $x_i = 1$ if individual i is sick, and $x_i = 0$ otherwise. We will say that an individual who is sick, i.e. who has status $x_i = 1$, is *positive* for coherence with the testing framework. This is coherent with the classification framework, in which a sample with status 1 is said to be positive (this is, for instance, the convention used by the package `sklearn`).
We conduct T binary group tests, summarized by the test matrix of size $T \times n$ written $\mathbf{B} = (b_{ti})_{t,i}$. The rows of $\mathbf{B}$ represent tests, while its columns stand for individuals: $b_{ti} = 1$ means that individual i is included in test t.
Group testing can thus be written as the non-convex optimisation problem

$$\begin{aligned} &\arg\min_{\mathbf{x}} ||\mathbf{x}||_0 \\ &\text{s.t. } \mathbf{Bx} = \mathbf{y} \text{ and } \mathbf{x} \in \{0,1\}^n , \end{aligned} \tag{2}$$

as explained in [5]. $k := |\mathcal{K}|$ is the number of defective individuals, and maybe included in the optimization problem as an upper bound on the cardinal of $\mathbf{x}$ if it is known as a upper boud, or as an equality constraint depending on the context. Following [2], we distinguish three sparsity regimes that encode the asymptotic relation between k and n:

1. The **very sparse regime** corresponds to the setting in which k is fixed as $n \to +\infty$.
2. In the **sparse regime**, k scales sublinearly with n, i.e. $k = \mathcal{O}(n^\alpha)$, with $\alpha \in]0,1[$.
3. Finally, the **linear regime** encompasses the case where $k = \beta n$, $\beta \in]0,1[$.

When dealing with asymptotic performance of algorithms and test designs, we will always refer to these three regimes.
The literature on group testing typically distinguishes between different settings (see [2] for a review):

1. **Adaptative vs. nonadaptative:** in adaptative testing, tests are performed successively, and the $T+1$-th test is design with knowledge of the T-th

test. In nonadaptative testing, all tests are performed in parallel, leading to potential inefficiency due to multiple tests yielding redundant information. However, this setting can be considered as being more realistic since all tests can be performed at the same time, thus reducing logistic complications.

2. **Noiseless vs. noisy testing:** in noiseless testing, tests perform perfectly: if a single individual is tested positive, the result is 1 with probability 1. Noisy group testing allows for some corruption of test results, i.e. false positives and false negatives in the testing procedure.
3. **Binary vs. non-binary outcomes:** in the binary setting, the test result is 0 or 1, while the non-binary setting allows for outcomes that quantify the amount of positive individuals in the tested subgroup.
4. **Combinatorial vs. i.i.d. prior on k:** in the combinatorial setting, there are k sick individuals amongst the n individuals. $\mathcal{K}$ is thus chosen uniformly amongst the $\binom{n}{k}$ subsets of size k. In the i.i.d. prior setting, every item is set to be defective with small probability q, independently of other items.
5. **Known vs. unknown number of defectives:** this distinguishes between cases where k (or q if we are in an i.i.d. prior setting) is known, and where it is unknown (or simply bounded from above).

In this project, we will focus on the noiseless, nonadaptative setting with binary outcome. We will consider the number of defectives (or the probability q) to be known (thanks, for instance, to a public health survey), and work with both i.i.d. and combinatorial priors.
Another question concerns the evaluation of a group testing procedure. One can either aim for perfect reconstruction or partial reconstruction ; this can happen with probability 1 (i.e. for all vectors $\mathbf{x}$), or one can allow for errors (i.e. with a small probability, one has an imperfect reconstruction).
This section will give some results and concepts associated to two central questions in group testing:

1. How should one design the test matrix $\mathbf{B}$?
2. How can one reconstruct the vector $\mathbf{x}$ when given a test matrix $\mathbf{B}$ and a result vector $\mathbf{y} \in \{0,1\}^n$?

2.2 Test design: from K-disjunct matrices to random tests

Intuitively, one wants to construct a test matrix $\mathbf{B}$ such that tests are non redundant, and isolate individuals in such a way that reconstruction is possible. The literature of group testing usually considers three different kind of matrices.
We define $S(i) := \{t \,|\, b_{ti} = 1\} \subset \{1, \dots, T\}$ - these are the tests in which individual i is included - and also

$$S(\mathcal{L}) := \bigcup_{i \in \mathcal{L}} S(i)$$

the tests that include at least one member of the subgroup $\mathcal{L}$. $S(\mathcal{K})$ thus are the positive tests when the set of sick individuals is $\mathcal{K}$.

Definition 1. *A test matrix $\mathbf{B} \in \{0,1\}^{T \times n}$ is said to be k-disjunct if for every subset $\mathcal{L}$ of $\{1, \dots, n\}$ of size $|\mathcal{L}| = k$, and every $i \notin \mathcal{L}$, one never has $S(i) \subset S(\mathcal{L})$.*

In other words, if a test matrix is k-disjunct, no healthy individual will appear only in positive tests. If this condition is met, we can safely identify the defective set of individuals by simply declaring healthy all individuals that are in negative tests, and sick all others.
Less requiring conditions are k-separability and $\bar{k}$-separability.

Definition 2. *A test matrix $\mathbf{B} \in \{0,1\}^{T \times n}$ is said to be k-separable all sets $S(\mathcal{L})$, for subset $\mathcal{L} \subset \{1, \dots, n\}$ of size $|\mathcal{L}| = k$, are distinct.*

Definition 3. *A test matrix $\mathbf{B} \in \{0,1\}^{T \times n}$ is said to be $\bar{k}$-separable all sets $S(\mathcal{L})$, for subset $\mathcal{L} \subset \{1, \dots, n\}$ of size $|\mathcal{L}| \leq k$, are distinct.*

If the test matrix is not k-separable and there are exactly k defectives, there exist two subsets of size k, $\mathcal{L}_1$ and $\mathcal{L}_2$, such that $S(\mathcal{L}_1) = S(\mathcal{L}_2)$. If one of those two subsets is equal to $\mathcal{K}$ and the test matrix is design in order to test $\mathcal{L}_1$ and $\mathcal{L}_2$, the two subgroups will not be distinguishable.
It immediatly follows from these definitions that

$$k\text{-disjunct} \Rightarrow \bar{k}\text{-separable} \Rightarrow k\text{-separable}.$$

A question that follows from these definitions is: how many tests does one need in order to form a k-disjunct matrix ? According to [2], the literature on this subject is vaste ; we only give the following bound, due to [9].

Theorem 2.1. *Suppose there exists a k-disjunct matrix of size $T \times n$. Then*

$$T \geq \min\left\{\frac{15+\sqrt{33}}{24}(k+1)^2, n\right\}.$$

However, in practice, when dealing with large samples, it is often more efficient and easier to construct random test matrices. Furthermore, taking random test matrices eases the theoretical analysis of reconstruction algorithms. These matrices are usually constructed in three different ways.

1. In **Bernouilli design**, each item is included at random in every test with probability p, that is, $\mathbb{P}(b_{ti} = 1) = p$ for all t, i.

2. In **constant tests-per-item** design, every item is included in a fixed number L of tests, independently from other items.
3. In **doubly regular design**, every item is included in L tests, and every test includes exactly M items.

In our project, we mainly focus on the first two designs. A natural problem that arises is the choice of p and L. In Bernouilli testing, p is chosen as $p = \mathcal{O}(\frac{1}{k})$; in constant tests-per-item design, one takes $L = \mathcal{O}(\frac{T}{k})$.

2.3 Metrics and information theory bounds

Under the combinatorial prior, one can define an absolute error probability.

Definition 4. *Under the exact recovery criterion, the average error probability for a reconstruction algorithm $\mathcal{A}$ under a combinatorial prior is defined as*

$$err(\mathcal{A}) := \frac{1}{\binom{n}{k}} \sum_{\mathcal{K}:|\mathcal{K}|=k} \mathbb{P}\left(\mathcal{A}(\mathbf{y}, \mathbf{B}) \neq \mathcal{K}\right),$$

where the randomness stems from $\mathbf{B}$ and $\mathcal{A}$ if it is randomized. The average sucess probability is defined as

$$suc(\mathcal{A}) := 1 - err(\mathcal{A}).$$

We are also interested in quantifying the probability of partial recovery.

Definition 5. *The average d-error probability, for $d \in \{0, \ldots, k\}$, of a reconstruction algorithm $\mathcal{A}$ under the combinatorial prior is defined as*

$$err_d(\mathcal{A}) = \frac{1}{\binom{n}{k}} \sum_{\mathcal{K}:|\mathcal{K}|=k} \mathbb{P}\left(|\mathcal{A} \cap \mathcal{K}^c| > d,\, |\mathcal{A}^c \cap \mathcal{K}| > d\right).$$

In words, the average d-error probability is the probability, averaged over all possible subsets of k sick individuals, that a reconstruction algorithm, given a test design procedure, makes more that d false negatives and false positives.
Since group testing is a classification task, we also introduce classical metrics used for classification (as precision or recall; see appendix A).
Another important theoretical tool is the notion of *rate*. From an information theory point of view, one could want to know how many bits of information are gained, on average, by each test. The rate of a group testing procedure (i.e. a defective prior, a test design and a reconstruction algorithm) is defined as follows.

Definition 6. *Given a group testing strategy under the combinatorial prior with k defectives among n items and T tests, the rate of the strategy is defined as*

$$\frac{\log_2 \binom{n}{k}}{T}.$$

This definition can be understood by seeing that the entropy of the random process generating the defective set by choosing a set uniformly among the $\binom{n}{k}$ possible sets is equal to

$$\log_2 \binom{n}{k}.$$

With this definition in mind, one can provide a metric based on the rate for group testing strategies.

Definition 7. *Consider a group testing setting, where the number of defectives is $k(n)$, and the number of tests is $T(n)$.*

- *A rate is said to be* ***achievable*** *by the testing strategy if for any $\delta, \varepsilon > 0$, there is a n large enough such that*
$$\frac{\log_2 \binom{n}{k}}{T} > R - \delta$$
and the error probability is at most ε.
- *A rate is said to be* ***zero-error achievable*** *if for any $\delta > 0$, there is a n large enough such that*
$$\frac{\log_2 \binom{n}{k}}{T} > R - \delta$$
and the error probability is zero.
- *The* ***maximum achievable rate****, resp.* ***maximum zero-error achievable rate*** *of a deterministic or random test design is the supremum of all achievable rates, resp. achievable zero-error rates, over all possible reconstruction algorithms.*
- *The* ***capacity****, written C, is defined as the maximum achievable rate when taking the supremum over both the decoding algorithm and the test design. Conversely, we also define the* ***zero-error capacity*** *C_0.*

The achievable rate gives an information about how good a given testing strategy (test design *and* reconstruction algorithm) is: it gives a sens of the number of bits one can learn on average by test. The maximum achievable rate quantifies how good a test design strategy is. Finally, the capacity gives a sens of how hard a group testing problem is, and what one could hope for with a perfect (but eventually purely theoretical and highly unpractical) algorithm.
Let us now state a few facts about group testing.

Theorem 2.2. *Any algorithm for recovering the defective set with T tests has success probability bounded by*

$$\frac{2^T}{\binom{n}{k}}.$$

This bound is sometimes called the *counting bound*. Its proof is straightforward.

Proof. Suppose one wants to find the defective set with probability 1. Then one needs at least T tests with T satisfying

$$2^T \geq \binom{n}{k}.$$

Indeed, since every test has a binary answer (0 or 1), one can get 2^T possible answers with T tests. With k defectives, there are $\binom{n}{k}$ possible defective sets, and T thus has to verify this inequality to ensure that one has enough possible tests results to identify the defective set.
It follows from the definition of the success probability that proving the theorem is equivalent to proving

$$\mathbb{P}(\text{err}) \geq 1 - \frac{2^T}{\binom{n}{k}}.$$

By definition, one has

$$\begin{aligned}\mathbb{P}(\text{err}) &= \frac{1}{\binom{n}{k}} \sum_{\mathcal{K}:|\mathcal{K}|=k} (1 - \mathbb{P}(\mathcal{A} = \mathcal{K})) \\ &= 1 - \frac{1}{\binom{n}{k}} \sum_{\mathcal{K}:|\mathcal{K}|=k} \mathbb{P}(\mathcal{A} = \mathcal{K}) \\ &\geq 1 - \frac{2^T}{\binom{n}{k}}.\end{aligned}$$

□

This bound may seem quite trivial, but for instance allows us to state that $\mathbb{P}(\text{sucess}) \to 0$ as n gets large whenever $T \leq (1-\varepsilon)\log_2\binom{n}{k}$ for arbitary small ε.
Let us finally give some results about the capacity of non adaptative group testing.

Theorem 2.3. *Consider the sparseness regime $k = \mathcal{O}(n^\alpha)$, $\alpha \in]0,1[$.*

- *Zero-error non adaptative testing has zero-error capacity 0 for all α.*
- *Small error non adaptative testing has capacity 1 for $\alpha \in [0, 0.409]$. For $\alpha > 0.409$, the capacity is unknown.*
- *With Bernouilli test design, small error non adaptative testing has capacity 1 for $\alpha \in [0, 3]$.*

This result essentially tells that zero-error nonadaptative testing is hopeless: one cannot hope to design tests such that one gains in average a strictly positive amount of information while making no error in identifying the defective set. When allowing for small errors, one can asymptotically guarantee an average gain of information of one bit provided that the set of individuals is sparse enough.

3 RECONSTRUCTION ALGORITHMS

Given a test matrix $\mathbf{B}$ and a result vector $\mathbf{y}$, the goal of reconstruction algorithms is to reconstruct the unknown vector $\mathbf{x}$ which satisfies

$$\mathbf{y} = \mathbf{B}\mathbf{x}. \tag{3}$$

Formally, a reconstruction algorithm thus is a mapping

$$\mathcal{A} : \{0,1\}^T \times \{0,1\}^{T\times n} \to \mathcal{P}(\{1,\ldots,n\}).$$

We first give some definitions concerning metrics that quantify reconstruction performance, and then give a short presentation of the main reconstruction algorithms detailed in [1]. All cited algorithms are implemented in the project's repository in the Python script `reconstruction_alg.py`. Pseudo-code is given in the appendix.

3.0.1 The COMP algorithm

The COMP algorithm is the most simple and conservative algorithm used for reconstruction. It works by declaring healthy all individuals that are in negative tested groups, and positive all others, thus only making false positives but no false negatives.
[1] prove the following theorem for the COMP algorithm.

Theorem 3.1. *In the α-sparse regime, the COMP algorithm with $1/k$-Bernoulli design has rate*

$$R^*_{COMP} \geq \frac{1-\alpha}{e\log 2}.$$

3.0.2 The DD algorithm

The DD algorithm proceeds by a first COMP stage: it declares healthy all individuals that are in negative groups. It then singles out individuals whose health status is unclear and that are in test groups where all other individuals have been declared healthy in the COMP stage, and declares those sick. Finally, all other individuals are declared healthy. This last step is motivated by the supposed high sparsity of the vector $\mathbf{x}$: if the test matrix $\mathbf{B}$ is well designed, i.e. in order to single out sick individuals, then one makes few mistakes in average when declaring negative all individuals with unclear status, since they are very rare. Remark that the DD algorithm only makes false negatives, and no false positives.
The following theorem is provided by [1].

Theorem 3.2. *In the α-sparse regime, the COMP algorithm with $1/k$-Bernoulli design has rate*

$$R^*_{COMP} \geq \frac{1}{e\log 2}\min\left[1, \frac{1-\alpha}{\alpha}\right].$$

3.0.3 The SCOMP algorithm

The SCOMP algorithm proceeds by iterating:

1. It makes a first guess $\tilde{\mathbf{x}}$ of sick and healthy individuals by using the DD algorithm without its last step (declaring all unidentified individuals to be healthy).
2. It then iterates the two following steps. If the guess is consistent with the test result, i.e. if $\mathbf{y} = \mathbf{B}\tilde{\mathbf{x}}$, the algorithm terminates. Otherwise, it composes a new guess by finding the individual which is present in the greatest number of tests unexplained by the initial guess, and adds it to this guess.

3.0.4 The LP and Enhanced LP algorithms

These algorithms were first introduced by [5], and essentially proceed by relaxing the combinatorial NP-hard group testing problem to a convex, LP-type program. Assuming that one wants to find the sparsest vector $\mathbf{x}$ which is compatible with the test results, the group testing problem writes

$$\min ||x||_1$$
$$\text{s.t. } \mathbf{y} = \mathbf{B}\mathbf{x} \text{ and } \mathbf{x} \in \{0,1\}^N .$$

Letting $\mathbf{B}_+$ and $\mathbf{B}_-$ denote the tests whose results are resp. positive and negative. The relaxed problem writes

$$\begin{aligned} &\min \sum_{i=1}^{N} x_i \\ \text{s.t. } &\forall i, 0 \leq x_i \leq 1, \\ &\mathbf{B}_+\mathbf{x} \geq \mathbf{1}, \\ &\mathbf{B}_-\mathbf{x} = \mathbf{0}, \end{aligned}$$

which is a LP-program. Once the problem is solved, the algorithm sets all coordinates from the solution vector that are less that 1 to 0, and returns the solution. The algorithm is implemented using the solver `CVXPY` [3].

4 NUMERICAL EXPERIMENTS

We run two series of numerical experiments: the first one comparses reconstruction algorithms in various settings; the second one tests our pipeline of enhanced group testing.

4.1 Comparaison of different reconstruction algorithms

We first run some simulations to compare the performances of reconstruction algorithms.

Sparse vs. dense setting. We first test the algorithms for different levels of sparsity of $\mathbf{x}$. We choose $n = 100$, $k = 2$ for the sparse setting and $k = 10$ for the dense setting. For every number of tests, we run 100 reconstructions for random input vectors and Bernoulli test matrices, and eventually average. Results are displayed in figures 3 and 4.

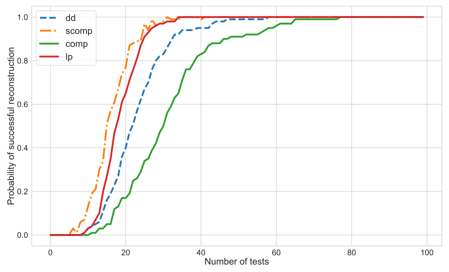

Figure 3. Probability of exact recovery given as a function of the number of group tests, for different reconstruction algorithms. $n = 100$, $k = 2$.

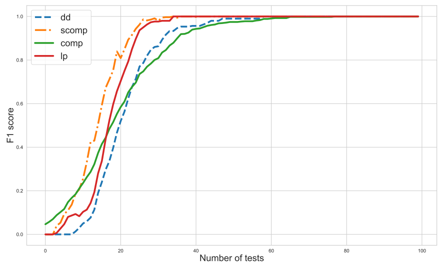

Figure 4. F1 score as a function of the number of group tests, for different reconstruction algorithms. $n = 100$, $k = 2$.

We find the same results as [1]: the SCOMP and LP algorithms outperform the other algorithms in terms of probability of exact recovery. For the F1 score, the result is more contrasted since some algorithms make no false positives or no false negatives. However, we still find that with as little as 35 tests, the SCOMP and LP algorithm fully identify the defective set with probability 1, which is a huge improvement compared to the individual testing setup, where 100 tests would have been required.
Figures 5 and 6 show what happens in the dense case, i.e. when 10% of the population is marked as sick. In this setting, group testing is ineffective: with 100 group tests, the true vector $\mathbf{x}$ is not fully recovered with probability 1 by any algorithm. In this case, it makes more sens to perform individual testing (through 100 individual tests). Another interesting fact is that the COMP algorithm is completely outperformed by all other three algorithms: this is most likely due to the fact that for a dense vector $\mathbf{x}$, little tests yield a negative result, and the algorithm is unable to make good predictions.

Bernoulli vs. CTPI test design. We now turn to comparing Bernoulli and constant test-per-item test design frameworks. We test all four algorithms on randomly generated sparse data: we set $n = 100$, and $k = 2$. Our experiments show some improvement between the two settings when the constant number of tests per item

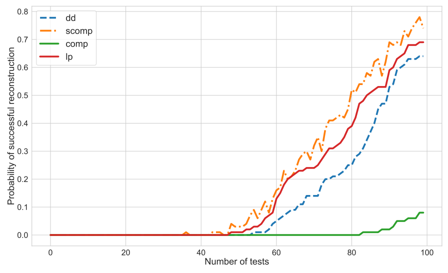

Figure 5. Probability of exact recovery given as a function of the number of group tests, for different reconstruction algorithms. $n = 100$, $k = 10$.

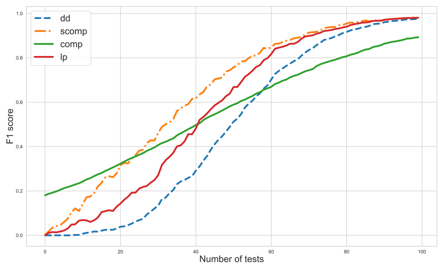

Figure 6. F1 score as a function of the number of group tests, for different reconstruction algorithms. $n = 100$, $k = 2$.

is close to $n/2k$. Figure 7 shows that for $L = n/2k$, we are able to get bellow the threshold of 30 tests needed for perfect reconstruction with the best performing algorithms, which improves on the results of the Bernoulli setting.

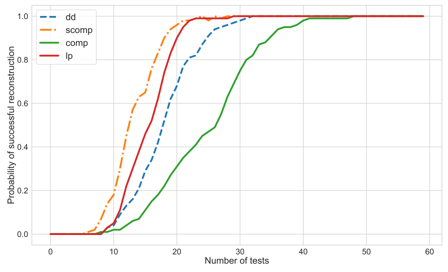

Figure 7. Probability of perfect reconstruction, for different reconstruction algorithms with CTPI test design. $n = 100$, $k = 2$, $L = n/2k$.

Asymptotics. When n, the number of individuals, grows, the sparsity regime greatly determines the efficiency of the different reconstruction algorithms. In this set of simulations, we fix the number of group tests to $T = 40$ and see the attained probability of perfect reconstruction for n ranging from 50 to 300. This is performed for 200 instances, and averaged. Results are plotted in figures 8, 9 and 10.

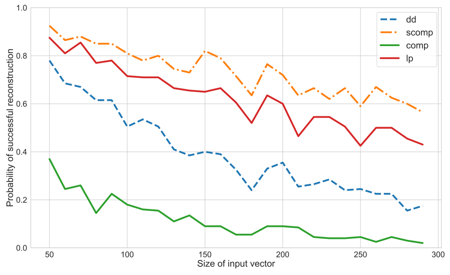

Figure 8. Probability of perfect reconstruction as a function of the size of the input vector in the very sparse regime, with $k = 4$.

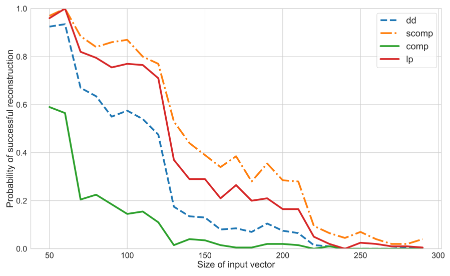

Figure 9. Probability of perfect reconstruction as a function of the size of the input vector in the sparse regime, with $\alpha = 1/3$.

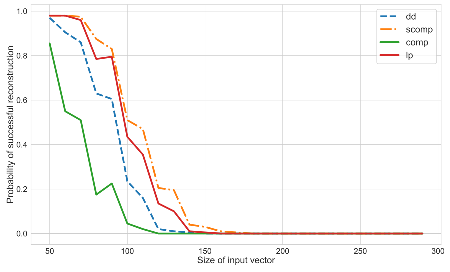

Figure 10. Probability of perfect reconstruction as a function of the size of the input vector in the linear regime, with $\beta = 0.05$.

As anticipated, the sparse and linear regime cause the probability of perfect reconstruction to collapse exponentially fast for all algorithms - in contrast to the decay in the very sparse regime which seems to be linear - because k increases with the size of the input vector.

4.2 Group testing with features

Our central idea is to use group testing to improve classification of badly separable data.
Formally speaking, we consider a model

$$y = f(\mathbf{x})$$

were $y \in \{0, 1\}$ is a class label and $\mathbf{x} \in \mathbb{R}^d$ are features. Our goal is to build a two stage algorithm, trained on a sample of labelled points , that for a sample of unlabelled points $\{(\mathbf{x_1}), \ldots, (\mathbf{x_n})\} \in \mathbb{R}^{nd}$

1. Predicts, for each point $\mathbf{x}_i$ its probability to belong to the class 1, i.e. $\mathbb{P}(y_i = 1|\mathbf{x}_i)$;
2. Performs group testing by taking into account the initial predictions made by the classifier in order to improve its prediction.

We test three types of enhanced group testing:

1. We first test **hard thresholding**: we select all individuals whose probability to be either 1 or 0 lies close to 1/2, discard the predictions of the classifier and group test these individuals.
2. We then turn to **soft thresholding**: we use the predictions of the classifier to construct the testing matrix, including individuals classified with little confidence in more tests.
3. We finally make some simulations for **soft-hard thresholding**, in which items are randomly selected for the group testing stage depending on their predicted probability.

4.2.1 Toy example I: group testing enhanced logistic classification of 2D points with hard thresholding

Logistic classification is a very popular parametric classification method. Given a training set of labeled points $\mathbf{x}_1, \ldots \mathbf{x}_n \in \mathbb{R}^d$, the logistic regression parametrized by $\theta \in \mathbb{R}^d$ predicts the probability of belonging to the class 1 through the sigmoid function ; it is equal to

$$p(y = 1|\mathbf{x}) := \frac{1}{1 + \exp(-\mathbf{x} \cdot \theta)}.$$

Let $\sigma(x) := \frac{1}{1+\exp(-x)}$ be the sigmoid function. The parameter θ is fitted my maximizing the log-likelihood, or equivalently minimizing the negative log-likelihood, of the training sample. It thus solves the optimization problem

$$\min_{\theta \in \mathbb{R}^d} - \sum_{i=1}^{n} \left[y_i \log \sigma(\theta \cdot \mathbf{x}_i) + (1 - y_i) \log 1 - \sigma(\theta \cdot \mathbf{x}_i) \right].$$

This optimisation problem has no closed form solution and has thus to be solved by numerical methods, such as Newton's method or gradient descent (see [7] for more details).

In this first example, we focus on enhancing logistic classification by group testing. We consider point clouds generated through `sklearn.datasets.make_blobs()`, whose centers are very close and with large noise, thus making them badly linearly separable. A sample of this data is plotted in figure 11.

The original dataset is first split in training and testing dataset (which are both sparse, in the sens that they include little sick individuals). A logistic classifier is

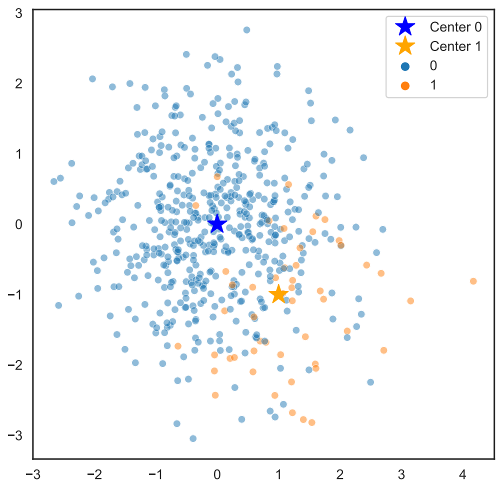

Figure 11. Badly linearly separable mixture of gaussians used in toy example 1. In this plot, $n = 500$, and $k = 50$.

then trained on the training set using `sklearn` [8]. The classification rule of this classifier is to classify all samples for wich $p(y = 1|\mathbf{x}) \geq 1/2$ as 1. We ask the classifier for its prediction on the test set, and select all samples for which

$$\frac{1}{2} - \varepsilon \leq p(y_i = 1|\mathbf{x}_i) \leq \frac{1}{2} + \varepsilon,$$

where ε is a thresholding hyperparameter. The choice of this parameter is crucial: if it is set to low, to few items are selected for group testing and our pipeline does not improve the classifiers performance ; if it is set to high, the algorithm selects to many samples and thus collapses to the original test-intensive group testing framework. Figure 21 in the appendix displays the percentage of the dataset selected for group testing as a function of the threshold. Figure 12 displays the selection our algorithm operates with a threshold set to $\varepsilon = 0.2$.

To test how effective this method is, we compute the average probability of perfect reconstruction as a function of the number of tests performed. We use parameters $n = 400$, $\varepsilon = 0.39$ and $k = 40$. The average probability of perfect reconstruction is displayed in figure 13. Figure 14 displays the average probability of making less than 5 errors, while figure 15 shows the average number of errors.

These first simulations offer a contrasted picture of enhanced group testing. Our procedure does not reconstruct the vector $\mathbf{x}$ perfectly with higher probability than the vanilla group testing. However, if one is willing to tolerate a few errors, our procedure is clearly more efficient: in average, enhanced group testing makes few errors with few tests.

Another question we want to answer is the tuning of

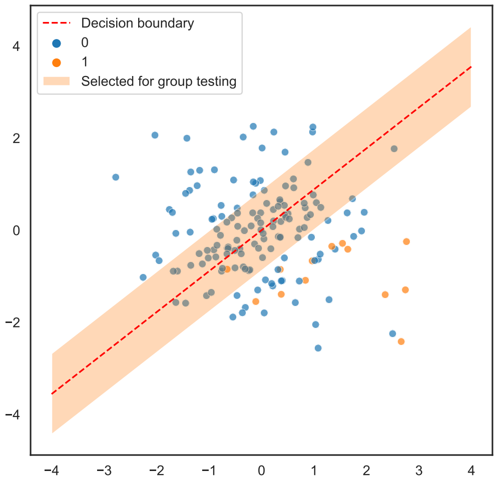

Figure 12. The badly separated data is first classified by a logistic classifier, whose decision boundary (in red) is linear. All samples in the orange area are then selected for group testing. In this plot, $\varepsilon = 0.2$.

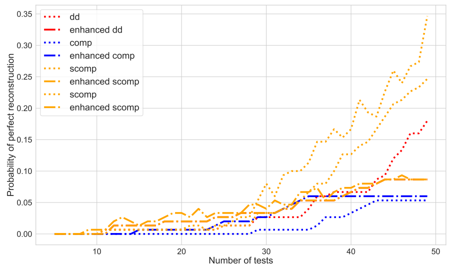

Figure 13. Probability of perfect reconstruction for vanilla and enhanced reconstruction algorithms. $n = 400$, $k = 40$, $\varepsilon = 0.39$.

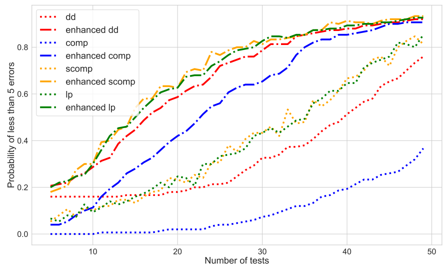

Figure 14. Probability of making less than 5 errors, for vanilla and enhanced group testing algorithms. $n = 400$, $k = 40$, $\varepsilon = 0.39$.

ε: what is the optimal threshold ? 16 shows that the optimal threshold lies around 0.43.

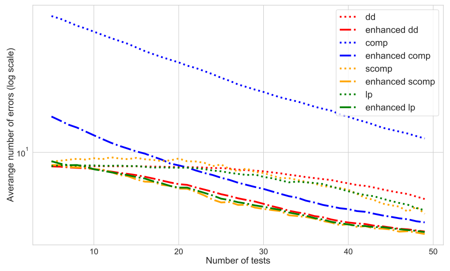

Figure 15. Average number of errors for vanilla and enhanced reconstruction algorithms. $n = 400$, $k = 40$, $\varepsilon = 0.39$.

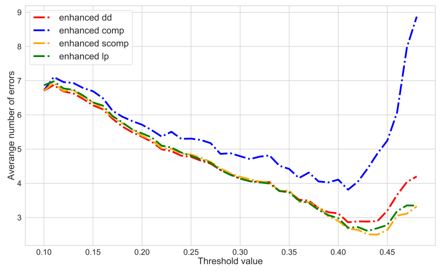

Figure 16. Average number of errors for enhanced group testing, for varying threshold. $n = 400$, $k = 40$.

4.2.2 *Toy example II: group testing enhanced logistic classification of 2D points with soft thresholding*

Results of figure 13 can be explained by the strict way we select samples for group testing based on their predicted class: by completely accepting predictions of the classifier, we also take the risk to accept errors without any verification. A more subtle way to use the classifiers prediction could thus be to modify the number of tests per item depending on their predicted probability of belonging to a given class: we call this procedure *soft thresholding*.

The crux of this procedure is to find a mapping that connects the predicted probability of being sick, predicted by the classifier, to the probability to include the item in a given group test (as in the classical Bernoulli design). Formally, this task amounts to finding

$$\sigma : p_i \in [0,1] \mapsto q_i \in [0,1],$$

before constructiong a group testing matrix $\mathbf{B}$ where for each test t,

$$\mathbb{P}(b_{ti} = 1) = q_i.$$

We have no clear intuition for the choice of $\sigma(\cdot)$. Indeed, the link between the probability of successful reconstruction and the inclusion of more or less positive individuals in the testing procedure is, in our eyes, unclear. We conducted numerous numerical experiments with different

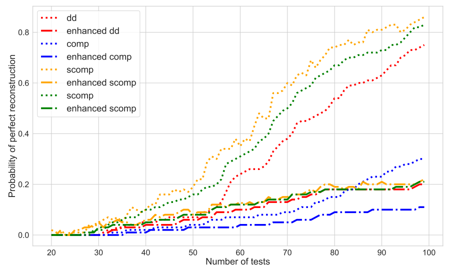

Figure 17. Average probability of perfect reconstruction, for soft-hard thresholding and vanilla algorithms. $n = 600$, $k = 40$.

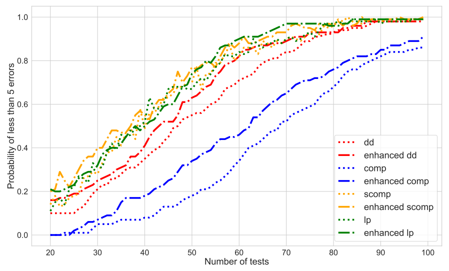

Figure 18. Average probability of making less than 5 errors, for soft-hard thresholding and vanilla algorithms. $n = 600$, $k = 40$.

functions $\sigma(\cdot)$, but none seemed to yield a significant improvement with respect to the vanilla procedure. An example of computations for the function

$$\sigma(p_i) = 1 - p_i$$

is displayed in the appendix in figures 22, 23 and 24.

4.2.3 Toy example III: Soft-hard thresholding for 2D points

Finally, we introduce the idea of *soft-hard thresholding*, a mixed procedure inbetween hard and soft thresholding. Recall that in hard thresholding, we choose items deterministically for group testing based on their predicted probability of belonging to one of the classes; soft thresholding selected all items for group testing but included them in a varying number of group tests, depending once again on their predicted probability.
Hard-soft thresholding procedes by relaxing hard thresholding: we select items for the group testing procedure with probability $f(p_i)$, where p_i is the probability of item i to be healthy and $f(\cdot)$ is decreasing function from $[0, 1]$ to $[0, 1]$. Items that are not selected are declared sick. By doing this, we select little sick individuals for group testing and thus sparsify the tested sample, which should lead to increased performance.
In figures 17, 18 et 19, we plot the results for $k = 40$, $n = 600$. Sample are selected for group testing with probability $1 - p_i^3$, and the original prediction of the classifier is kept if the item is not selected. The method seems to be a little bit more efficient when accepting a few errors (in our plot, less than 5), but does not lead overall to a significant improvement of the vanilla group testing.

5 CONCLUSION AND UNEXPLORED IDEAS

In this project, we have tried to link group testing and classification to improve the group testing procedure by including prior knowledge on tested individuals coming from features. Our results suggest that this pipeline drastically reduces the number of group tests needed

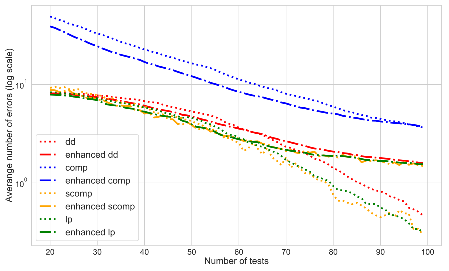

Figure 19. Average number of errors, for soft-hard thresholding and vanilla algorithms. $n = 600$, $k = 40$.

for full identification, even if this improvement of course depends on the structure of the data.
Unfortunately, we could not explore all ideas in this project. The following list gives some ideas which could be further developed:

1. **Code acceleration.** Our code is still quite slow, due to a lack of optimization of some computations. This could be hugely improved.
2. **Other classifiers.** A good improvement would be to test our method on other classifiers than logistic regression (KNN or a simple feed-forward neural network, for instance).
3. **Testing on real data and separability measures.** A good extension of these tests would be to test our pipelines on real datasets while measuring the separability of the data in these datasets, as for instance with the indicators provided in [6].
4. **Noisy group testing.** If the group testing procedure becomes noisy, meaning the group tests can lead to false positives and false negatives, the benefits of group testing an identified subsample of the population might be lessened. The effects of noise could be explored numerically at least.
5. **Theoretical properties of enhanced group testing.** We were not able to provide theoretical guarantees for our pipeline. Many questions remain open, as for instance the optimal design of a test matrix in the non-i.i.d. case, i.e. when every x_i has

a probability p_i of being equal to 1.

6. **Semi-supervised classification using group testing.** Group testing high dimensional data is interesting, since the complexity of the group testing procedure does not depend on number of features (in our framework, at least). Reversing our pipeline, group testing could thus be used to acquire some labels of an unlabelled dataset, before then using semi-supervised classification techniques. This procedure would fit a narrative in which health authorities would first use group testing to gain some knowledge about significant features, before using classification techniques for prediction and prevalence computation.

A METRICS FOR CLASSIFICATION EVALUATION

In binary classification, there are several metrics classically used to evaluate the performance of a classifier. Let us first recall the definitions of false positive and false negatives by a simple chart.

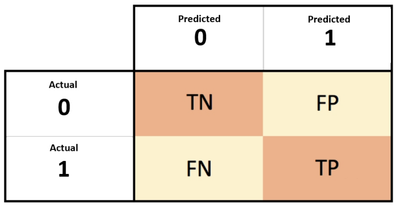

Figure 20. Definitions of true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

Definition 8. *The* ***score*** *of a classifier is the ratio*

$$\frac{TP+TN}{TP+TN+FN+FP}.$$

The score is also sometimes called the $0-1$ accuracy in machine learning, since it is equal to

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{\tilde{x_i}\neq x_i},$$

where $\mathbf{x}=(x_1,\dots,x_n)$ is the ground truth vector and $\tilde{\mathbf{x}}=(\tilde{x_1},\dots,\tilde{x_n})$ is the vector predicted by our classifier. It captures the accuracy of a classifier in a very broad sens, by counting the number of errors our classifier made. The score can be understood as the probability of any prediction being correct. However, in the sparse context of group testing, the score is a misleading metric. Indeed, given a fixed number of sick individuals k, if one classifies at random by choosing uniformly k individuals at random, in the worst case accuracy will be equal to $\frac{n-2k}{n}$. Hence, if $n=100$ and $k=2$ for instance, accuracy will always be greater or equal than 96%. One thus has to rely on other metrics.

Definition 9. *The* ***precision*** *of a classifier is the ratio*

$$\frac{TP}{TP+FP}.$$

Precision captures the share of individuals identified as sick by a test that are truly sick. It can be seen as the probability of a classification as sick being true.

Definition 10. *The* ***recall*** *of a classifier is the ratio*

$$\frac{TP}{TP+FN}.$$

Recall captures the completeness of our classification procedure: if recall is close to 0, it means that our classifier missed numerous sick people.
Finally, the F-score of the algorithm is the harmonic mean between precision and recall.

Definition 11. *The* ***F-score*** *is the ratio*

$$2 \times \frac{\mathit{Recall} \times \mathit{Precision}}{\mathit{Recall} + \mathit{Precision}}.$$

It might be helpful to determine what to do if either precision, recall or F score are not computable (because their computation involves a division by 0). We choose to follow this convention:

- If $FN = FP = TP = 0$, meaning that there are no sick individuals, and the algorithm perfectly reconstructs this ground truth, we set precision, recall and F-score to 1.
- In all other cases were computation involves division by 0, we set the problematic metric to 0.

B PSEUDO-CODE FOR RECONSTRUCTION ALGORITHMS

Algorithm 1 COMP algorithm

```
Input: $\mathbf{B}$: test matrix of size $T \times N$. $\mathbf{y}$: test result vector of size $T$.
Initialize $\tilde{\mathbf{x}} = \mathbf{1}$.
for i in $1, \dots, T$ do
  if $y_i = 0$ then
    $\tilde{\mathbf{x}}[\mathbf{b_i}] \leftarrow 0$
  end if
end for
return $\tilde{\mathbf{x}}$.
```

Algorithm 2 DD algorithm

```
Input: $\mathbf{B}$: test matrix of size $T \times N$. $\mathbf{y}$: test result vector of size $T$.
Initialize $\tilde{\mathbf{x}} = \mathbf{1}$.
for i in $1, \dots, T$ do
  if $y_i = 0$ then
    $\tilde{\mathbf{x}}[\mathbf{b_i}] \leftarrow 0$
  end if
end for
for i in $1, \dots, T$ do
  Count the number $e_i$ of tested individuals in $\mathbf{b_i}$ that have been declared healthy.
  if $e_i \neq \#\mathbf{b_i} - 1$ then
    $\tilde{\mathbf{x}}[b_i] \leftarrow 0$
  end if
end for
return $\tilde{\mathbf{x}}$.
```

Algorithm 3 SCOMP algorithm

```
Input: $\mathbf{B}$: test matrix of size $T \times N$. $\mathbf{y}$: test result vector of size $T$.
Initialize $\tilde{\mathbf{x}} = \mathbf{0}$.
for i in $1, \dots, T$ do
  if $y_i = 0$ then
    $\tilde{\mathbf{x}}[\mathbf{b_i}] \leftarrow 0$
  end if
end for
for i in $1, \dots, T$ do
  Count the number $e_i$ of tested individuals in $\mathbf{b_i}$ that have been declared healthy.
  if $e_i \neq \#\mathbf{b_i} - 1$ then
    $\tilde{\mathbf{x}}[b_i] \leftarrow 0$
  end if
end for
return $\tilde{\mathbf{x}}$.
```

C SUPPLEMENTARY PLOTS

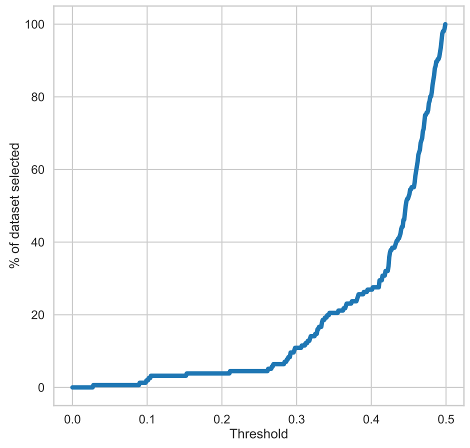

Figure 21. Percentage of the dataset selected for group testing as a function of threshold ε. Data comes from toy example I and is plotted for $n = 700$ datapoints, with $k = 80$.

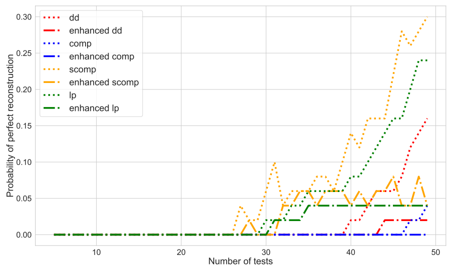

Figure 22. Percentage of the dataset selected for group testing as a function of threshold ε. Data comes from toy example I and is plotted for $n = 700$ datapoints, with $k = 80$.

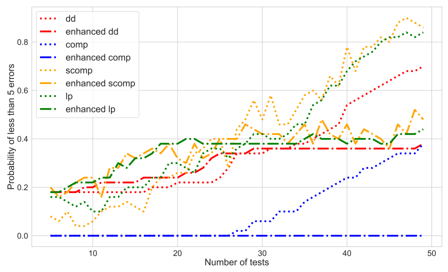

Figure 23. Percentage of the dataset selected for group testing as a function of threshold ε. Data comes from toy example I and is plotted for $n = 700$ datapoints, with $k = 80$.

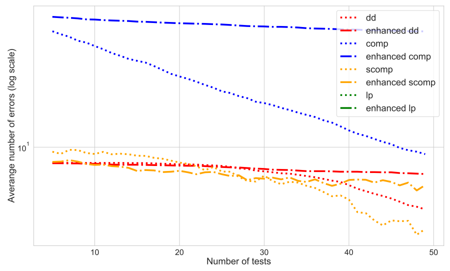

Figure 24. Percentage of the dataset selected for group testing as a function of threshold ε. Data comes from toy example I and is plotted for $n = 700$ datapoints, with $k = 80$.

REFERENCES

[1] Matthew Aldridge, Leonardo Baldassini, and Oliver Johnson. “Group testing algorithms: Bounds and simulations”. In: *IEEE Transactions on Information Theory* 60.6 (2014), pp. 3671–3687.

[2] Matthew Aldridge, Oliver Johnson, and Jonathan Scarlett. “Group testing: an information theory perspective”. In: *arXiv preprint arXiv:1902.06002* (2019).

[3] Steven Diamond and Stephen Boyd. “CVXPY: A Python-embedded modeling language for convex optimization”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2909–2913.

[4] Robert Dorfman. “The detection of defective members of large populations”. In: *The Annals of Mathematical Statistics* 14.4 (1943), pp. 436–440.

[5] Dmitry Malioutov and Mikhail Malyutov. “Boolean compressed sensing: LP relaxation for group testing”. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2012, pp. 3305–3308.

[6] NS Mthembu and JR Greene. “A comparison of three class separability measures”. In: (2004).

[7] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[8] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Chong Shangguan and Gennian Ge. “New bounds on the number of tests for disjunct matrices”. In: *IEEE Transactions on Information Theory* 62.12 (2016), pp. 7518–7521.