Department of Applied Mathematics

Faculty of Mathematical Sciences

University of Khartoum

Group Testing Via Binary Matrices

A dissertation submitted to the Faculty of Mathematical Sciences, University of Khartoum in partial fulfillment of the requirements for the degree of Msc in pure mathematics (Mathematics)

By: **Khalid Omer Elamin Hassan**

Supervisor:

Dr. Abdoelnaser Mahmoud Degoot Kharief

July, 2022

ACKNOWLEDGEMENTS

I would like to thank Dr. Abdoelnaser Degoot, who has not save any effort in helping me finishing this work. His valuable feedbacks and comments were a source of knowledge and encouragement.

Also, I would like to thank my family and friends for their continuous support and encouragement.

Abstract

Group testing is a fast way to find a few defective items among a large number of non-defective items, and it has many practical applications in fields as diverse as biology, engineering, information technology, and data science. The group testing problem has the same complexity as the compressing sensing problem—finding solutions to undetermined linear systems.

In essence, group testing generally requires two things: an efficient design matrix and a robust decoding algorithm. This study presents the group testing problem, sets up its relationship to compressed sensing, and implements the Binary Iterative Hard Threshold (BIHT) algorithm for reconstructing the original input signal using random binary matrices as design schemes. Using several in-silico simulations of hypothetical viral prevalence situations, we show that the BIHT is efficient and robust in recovering missing information, especially when disease incidents are rare.

Contents

List of Figures

Chapter 1

Introduction

1.1 General introduction

Late 2019 marked the beginning of the spread of SARS-Covid19 where it first appeared in Wuhan China and quickly spread to the rest of the world. Because of the infectious nature of the disease its spread was rapid. Many countries suffered from it, the health systems and facilities have been overwhelmed and failed to meet the demand. To control its transmission governments imposed many measures but the rapid detection of the infectious cases is crucial to contain the disease. Therefore, governments and health organizations conducted massive testing on the population. With this comes another problem that is, large scale testing is very expensive and needs big capacity.

Group testing or pool testing has been used in many countries. This was first studied by Robert Dorfman in the United states during the second world war, where a large number of new recruits of the army were tested for Syphilis [2].

The samples to be tested are taken from a group (pool) rather than individuals. The population is divided into groups and then a sample representing a mixture of each individual sample in the group is tested. If the test result of a particular mixture is negative then all the individuals in the group are free of the disease. If the test comes out positive, then at least one of individual the group is infected. Several tests of such kind can be used to determine the infected individuals of the population in contrast to the traditional way where each individual is tested alone. With arranging the procedure carefully as precise tests, good selection of groups, this would reduce the number of tests dramatically.

In general, suppose that we have n number of individuals to be tested. The traditional way is to conduct n number of tests, that is one test per individual. If the number of infected is large then this would be reasonable. By the time of Dorfman's work, the U.S army was testing the new recruits for Syphilis, and it was rare among the population and he started studying pool testing [3].

The problem can be described as, given a number of items (persons) n and number of

defectives (infected) d, then how many tests t are required to discover the number of defected (infected) among the n items (persons) and what is feasible way to achieve this [3].

This problem can be classified depending on the point of view that is taken into consideration.

1.2 Mathematical background

The problem is to reconstruct the missing information (detection) from the measured data. The basic assumption is that the information is obtained and processed following a linear model, that is the problem is reduced to solving a system of linear equations. Suppose that the results from the laboratory are referred to as $y \in \mathbb{R}^n$, which corresponds to $x \in \mathbb{R}^n$. x represents the information about the detection of the infected.

The problem is formalized as follows:

$$Ax = y \tag{1.1}$$

The matrix $A \in \mathbb{R}^{m \times n}$ is called the design matrix, which represents the testing process, m is the number of tests and n is the number of people tested [1].

The aim is to recover the vector $x \in \mathbb{R}^n$ by solving Eq.(1.1) for x. Ideally, $m = n$ hence the system is easily solved but it is common to have $m < n$ since the problem states that. Linear algebra tells us that this system is undetermined and it has infinitely many solutions. This renders the problem impossible to solve, *i.e.* recover the exact vector x. However, under certain assumptions it is possible to have a solution of the problem, and there are efficient algorithms to recover x exactly. The assumption is the referred to as the **Sparsity**.

Sparsity: a vector is said to be sparse if most of its components are zeros. Formally, a vector $x \in \mathbb{R}^n$ is called s-sparse if at most s of components are non-zeros. Linking this to our work is, since the vector x in eq. 1.1 represents the individuals to be test among which some of them are infected, with knowledge of the infectious rate and it assumed to be small, then with zero represents being free of the disease, then it is appealing to consider that our vector is sparse [1].

In general, looking at the problem of reconstructing x, one would address two main questions [1]:

- What is the best matrix $A^{m \times n}$ that one uses to obtain the best results?
- What is the efficient algorithm that one uses to reconstruct x from $Ax = y$?

1.3 Adaptive and non-adaptive group testing

1.3.1 Adaptive (sequential) tests

In Adaptive (sequential) algorithms, the tests are conducted sequentially one by one. The results from one test are recorded and used in the next test [2]. For instance, imagine a situation of testing 100 individuals for Covid19. At the beginning two groups are formed and one assumes one of the groups is free of the virus. thus for the second round of testing the knowledge from the first test is used as that, all the individuals in the first group are discarded and the second group is divided into smaller distinct groups.

1.3.2 Non-adaptive tests

In the non-adaptive algorithms, the groups to be tested are determined in advance, the tests are conducted in parallel and an individual can be involved in more than one test. It is intuitive that non-adaptive tests are time saving, and in our case, the covid19, where the tests must be conducted simultaneously, this is advantageous [3].

However, in general the adaptive tests are faster than non-adaptive because it transfers information from the previous tests to next tests. Considering the advantages from both types it is appealing to try to treat one test like the other. Non-adaptive tests can be treated as adaptive by staging the tests. Here the tests are divided into stages and information is transferred from one stage to the next, but within the stage information is not shared [2].

Although the two questions raised above are connected since the design matrix is part of the algorithm to be used, it is often helpful to try to answer them separately. Next we introduce common terminologies of the groups testing models that determine our definition of the model being successful i.e. achieving the best results with a minimum number of tests.

1. Binary and non-binary: In the standard group testing model, the design matrix, the defective vector and the outcomes vector are all in binary form. Where 0 in the design matrix represents not being tested and 1 otherwise, while in the defectives and the outcomes vectors, 0 for being free of the virus and 1 otherwise. Other models consider non-binary representation [3]. In this work we will consider the binary representation. The use of the binary setting is beneficial whereas it allows to use the literature of compressive sensing problem [2].

2. Noise: Noiseless are tests where there is no error of any kind, that is the recovery is done complete, i.e. the test procedures are done perfect. While in noisy tests we expect errors to happen either from the model or other reasons for example human errors. Two common model errors are the false positive (specificity) or false negative (sensitivity), where the former the negative outcome is labeled positive by the algorithm and later a positive outcome is labeled negative. In the realistic models small error probability is allowed, where the set of defective individuals is taken to be the defectives with high probability [2].

3. In our work here we consider a situation where the number of infected is known in advance through the knowledge of the infectious rate. The problem is combinatorial as the set of defectives (infected) is uniformly random among the sets of its size.

Our work will focus on the combinatorial non-adaptive tests and its implementation by building in-silico simulations of a non-adaptive test.

1.4 Useful terms and definitions

1. **Defective set**: We refer to the number of elements to be tested as n. We write $D = \{1, 2, ..., n\}$ for the set of defectives where each number item in the list is a label given to each individual and $d = |D|$ is the number of defectives.

   We write $x_i = 1$ for the defective (infected) item $i \in D$ and $x_i = 0$ for non defective (infected) item $i \notin D$.

2. **Tests**: we refer to the number of tests performed or to be performed by m, and label the tests $\{1, 2, 3..., m\}$. The tests represent the rows of the matrix A where the element a_{ij} is element j of n and tested in the i^{th} test. An important consideration is that each item is included in each test independently with fixed probability. The matrices we will use are sparse matrices. Although there is no particular criterion to judge exactly if a matrix is sparse or not it is common to set the number of non zero elements to be equal to the number of rows or columns of the matrix [3]. Note that throughout this text we will refer to this matrix by A or if else it will be stated.

3. **Standard noiseless group testing model** for known m, n and a test matrix $A^{m\times n}$ this model is described by its outcomes as follows:

$$y_j = \begin{cases} 1 & if \quad \exists i \in D \quad \text{with} \quad x_{ij} = 1 \\ 0 & if \quad \exists i \in D \quad with \quad x_{ij} = 0 \end{cases}$$

   where $y_j \in \{0, 1\}^m$ is the test outcome [3].

4. **Detection algorithm**: It is a function (map) that maps an element of the set $\{0, 1\}^{m\times n} \times \{0, 1\}^m$ to the subsets of $\{1, 2, ..., n\}$. Under an ideal recovery the output of the function above is the set of defectives (infected) [3].

5. **Norms**

   A norm of a vector space is a mapping that associates with each vector v a real number $||v||$ such that the following properties are satisfied for all vectors u, v and all scalars $c \in \mathbb{R}$;

   (a) $||v|| \geq 0$ and $||v|| = 0$ if and only if $v = 0$.

   (b) $||cv|| = |c|||v||$.

(c) $||u + v|| \leq ||u|| + ||v||$

The general norm in $\mathbb{R}$ is defined as follows:

$$||v||_p = (|v_1|^p + ... + |v_n|^p)^{\frac{1}{p}} \tag{1.2}$$

The first norm $||v||_1$ also known as the taxicab is then obtained by setting $p = 1$

$$||v||_1 = |v_1| + |v_2| + ... + |v_n| \tag{1.3}$$

The second norm of the usual euclidean norm is obtained by setting $p = 2$

$$||v||_2 = (|v_1|^2 + ... + |v_n|^2)^{\frac{1}{2}} \tag{1.4}$$

While as for the $||v||_0$ or the pseudo norm is actually a comparison tool rather being the literal norm defined above, the $||v||_0$ or l_0 as refereed to in the text between counts the number of nonzero components in a vector.

6. **Matrix norms induced by p-norms**

A matrix norm on $M_{nn} : V^{m\times n} \longrightarrow \mathbf{R}$ is a mapping that associates with $m \times n$ matrix A a real number $||A||$ called the norm of A, such that the following properties are satisfied for all $m \times n$ matrices and scalar $\alpha \in V$

(a) $||A|| \geq 0$.

(b) $||A|| = 0 \Longleftrightarrow A = 0_{m\times n}$.

(c) $||\alpha A|| = |\alpha|||A||$.

(d) $||A + B|| \leq ||A|| + ||B||$.

For $1 \geq p \leq \infty$ the corresponding operator norm is

$$||A||_p = sup_{x\neq 0}\frac{||Ax||_p}{||x||_p} \tag{1.5}$$

For the special cases $p = 1, 2$ the matrix norm is defined as follows:

$$||A||_1 = max_{1\leq j\leq n}\sum_{i=1}^{m}|a_{ij}| \tag{1.6}$$

$$||A||_2 = \sigma_{max}(A) \tag{1.7}$$

where $\sigma_{max}(A)$ is the square root of the largest eigen value of AA^T.

Chapter 2

The Recovery Process

In this chapter we describe the path which we took to recover the missing vector of infection and building the in-silico simulation. First we start by re mentioning the questions that need to be answered to build a good group testing model. First, what is the best matrix that one uses to obtain best results. this question is important as the rows of the matrix represent the number of tests to be ran, so fewer rows are better, but also this should not come at a cost of the recovery. The second question which concerns the best algorithm to be used in order to get the best recovery results. Here in this work we focus on an iterative algorithms called the Hard thresholding algorithms.

2.1 The measurement matrix

Our goal is to solve the minimization problem which is sometimes referred to as the basis pursuit,

$$\min||x||_1, \quad \text{subject to} \quad y = Ax \tag{2.1}$$

Here we introduce some theoretical guarantees for finding the best measurement matrix.

2.1.1 Mutual coherence

The mutual coherence of a matrix A is the largest absolute correlation between the columns of the matrix A. It is important to note that if two columns of A are strongly correlated then it will hard to know their contribution to the measurement result y. In and extreme case where the the mutual coherence is very high, then it would nearly impossible to recover a sparse signal [11].

Theorem 2.1.1. *Assume that* $||x||_0 \leq d$ *for the true vector* x *and* $\mu < \frac{1}{2d-1}$. *Then,* x *is a solution for the* l_0 *and* l_1 *problems [7].*

where μ is the mutual coherence metric, and for a matrix A it is defined by [11]

$$\mu(A) = max_{i \neq j} \frac{| < a_i, a_j > |}{||a_i||_2 ||a_j||_2} \tag{2.2}$$

2.1.2 Restricted isometry property (RIP)

The restricted isometry constant $\delta > 0$ of a matrix A, $\delta(A)$ is the smallest number such that the following inequality holds:

$$1 - \delta_{3d}(A)||x||_2^2 \leq ||Ax||_2^2 \leq (1 + \delta_{3d}(A))||x||_2^2$$

A matrix A is said to satisfy the restricted isometry property with constant $\delta_{2d}(A)$ if $\delta_{2d}(A) < 1$ [10].

Next we present a very important theorem which relates the RIP to our minimization problem.

Theorem 2.1.2. *Assume that $||x||_0 \leq d$ for the true vector x and $\delta_{2d} < 1$ then x is the unique solution of the l_0 and l_1 problems [11].*

where δ is the restricted isometry constant (RIC). Several improvements has been done the RIC, other types also available for instance $\delta < 0.1$ [5]. It is also known that stronger results can be obtained by using RIP as compared to the mutual coherence. But it can be seen that unlike the mutual coherence the RIP has a higher computation complexity [10].

2.2 Practical considerations

2.2.1 Probabilistic and Deterministic matrices

The measurement matrix is called deterministic if every test is obtained or determined with a probability of 1. While in contrast, the matrix is deterministic if the tests are arranged to some probabilistic distribution, or in simple words part of the matrix or the whole matrix is obtained by chance. In this work the matrix is generated randomly, particularly, the elements the matrix are generated using the **uniform random distribution**. It is known from the literature that a randomly generated matrix actually preserves the restricted isometry property with high probability [8].

2.2.2 Combinatorial and Probabilistic distribution of defectives

Another consideration is the distribution of positive, i.e. the location of positives and the number of positives in the vector x, that is the vector to be recovered. There are two

common distribution in the literature, the *probabilistic*, in which the locations and number of positives in the tested set is determined according to some probability. The other type is combinatorial scheme, where here the any set of up to d defectives can be positive. Here the number of defective is known in advance, where as their locations are set randomly. In our work here we considered the second case. In combinatorial case several schemes have proposed to attain a low number of test namely of $O(d^{1+o(1)}log^{1+o(1)}n)$ [9]. It is important to note that we are interested in the case where the number of individuals to be tested is very big i.e. $n \to \infty$, whereas the number of defectives d is constant.

2.3 The decoding algorithm

In this part we focus on recovering of the defectives vector, it will be referred to by x through out this text or else mentioned. The problem here is to recover x from the knowledge of the measurement matrix and the results vector or symbolically, the recovery of x from A and y.

In this work we look at two versions of the sparse approximation problem, the $C_{l_0}(y)$ optimization problem,

$$C_{l_0}(y) = ||x - Ay||_2^2 + \tau||y||_0, \tag{2.3}$$

where C_{l_0} is cost function and the objective is to optimize this function.

The Other problem is the sparse constrained problem,

$$min_y||x - Ay||_2^2 \quad \text{subject to} \quad ||Ax||_1 = c \tag{2.4}$$

Solving 2.4 is known to be NP-Hard in general [5].

2.4 The Hard Thresholding algorithms

Here we examine a class of algorithms called iterative thresholding algorithms. This class of algorithms is built basically on the idea of fixed point iteration.

2.4.1 Basic and normalized hard thresholding algorithms

The iterative Hard Thresholding (IHT) algorithms was first introduced for recovery problem by Blumensath and Davies in their work [5] and [6]. Elementary analysis, particularly proven in [12], shows that there are good theoretical guarantees of the algorithm. It is built on the simple intuition that, given $y = Ax$ where $y \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$, $m << n$, estimating x amounts to solving the square system $A * Ax = y$, and the classical iterative methods suggests defining a sequence $x^n = (I - A * A)x^n + A * y$. The proposed algorithm is given in [1]

Algorithm 1 The Basic Iterative Hard Thresholding algorithm

Input: measurement matrix A, measurement vector y, and the sparsity level d.
Initialization: d-sparse vector x^0, typically $x^0 = 0$.
Iteration: repeat until a stopping criteria is met

$$x^{n+1} = H_d(x^n + A * (y - Ax^n))$$

Output: The d-sparse vector x^*

Note that: the sparsity level is d is approximated by $p \times N$, where p is prevalence rate of the disease, and N is length of the vector x.

H is a non-linear operator which keeps the d largest values in x^* and sets the rest to zero.

2.4.2 Convergence and Performance

Iterative hard threshold is converged to a local minimum of $||y - Ax||_2^2$ such that $||x||_0 \leq d$ whenever $||A||_2 < 1$.

If $\delta_{3d} \leq \frac{1}{\sqrt{32}}$, then after at most $\left[log_2 \frac{||x_d||_2}{\tilde{\epsilon_d}}\right]$ iterations, the IHT approximation x^* satisfies

$$||x^* - x||_2 \leq 7\tilde{\epsilon_d} \tag{2.5}$$

where $\tilde{\epsilon_d} = ||x - x_d||_2 + \frac{||x - x_d||_1}{\sqrt{d}}$. One may try generalize this algorithm by adding a factor in front the term $A * (y - Ax^n)$ i.e. the iterative part becomes $H_d(x^n + \mu A * (y - Ax^n))$, where μ here is the gradient descent and in the case of the basic algorithm it was taken to be one [5].

One may also tune this factor to make it depend on the step itself, i.e. it is calculated iteration wise rather than being fixed

$$x^{n+1} = H_d(x^n + \mu_n A * (y - Ax^n))$$

This called the binary Iterative Hard threshold algorithm (NIHT) this was proposed in the work of [6]. Here μ_n is considered as a normalization factor and it is calculated by

$$\mu_n = \frac{||A(y - Ax^n)_{d_n}||_2^2}{||A(A(y - Ax^n)_{d_n}||_2^2)}$$

NIHT is guaranteed to converge to a local minimum of $||y - Ax||_2^2$ such that $||x||_0 \leq d$.

If A satisfies the RIP for all $x : ||x||_0 \leq 2d$, then x^* is guaranteed to have d-best term approximation to x, and after at most $\left[log_2 \frac{||x_d||_2}{\tilde{\epsilon_d}}\right]$ iterations estimates x with accuracy

$$||x - x^n||_2 \leq 9\tilde{\epsilon_d}$$

where $\tilde{\epsilon_d} = ||x - x_d||_2 + \frac{||x - x_d||_1}{\sqrt{d}}$. Through justification of these theoretical results is found in [5].

2.5 Binary iterative hard thresholding algorithm

In this section we introduce the 1-bit measurement scheme. Let $A \in \{0,1\}^{m\times n}$ be the measurement matrix, with each entry either 0 or 1. Let $y \in \{0,1\}^m$ be the measurement vector

$$y_j = \begin{cases} 1, & if \quad y_j \geq \tau \\ 0, & if \quad y_j \leq \tau, \end{cases}$$

and τ be a threshold that transfers the values of y_j to the binary category as described before. We wish to recover x such that $Ax = y$, x is n dimensional vector.

The reconstruction of x from A and y amounts to determining the sparsest vector that explains the measurement y. In this case we take l_1 norm as our measurement of sparsity. By minimizing the l_1 norm of the reconstructed vector $||x||_1 = \sum_i |x_i|$. Indeed, the matrix A satisfies the restricted isometry property (RIP), in fact the RIP ensure us among things the consistency of the linear system. As shown before the RIP can be guaranteed with high probability when the matrix A is drawn randomly. And here the number of measurements necessary to guarantee recovery using randomly generated measurement matrix is $O(dlog(N/d))$ [13].

The problem then can formalized as follows

$$x^* = min_x||x||_1 \quad \text{subject to } y = sign(Ax) \tag{2.6}$$

$$||Ax||_1 = c \quad \text{for c} > 0$$

where *sign* is an operator that works on Ax index wise by setting each $(Ax)_i$ to 1 if $(Ax)_i > 0$ and -1 otherwise. In our work we also examined the case where the second constrained is relaxed by setting $||Ax||_1 < c$.

Here is the binary threshold algorithm:

Algorithm 2 The Binary Iterative Hard Thresholding algorithm

Input: A, x and A is chosen such that $||Ax|| \leq c$, y = Ax being the measurement vector written in the binary form.
Initialization: d-sparse vector, taken to be 0, the gradient descent μ^0 calculated at 0.
iteration: repeat until a stopping criteria is met

$$x^{n+1} = \Gamma_d(x^n + \mu^n * A * (y - sign(Ax^n))) \tag{2.7}$$

output: the sparse vector x^*

---

sign is an operator that sets the components of x^* either to 1 or -1, the former if $x_i > 0$ and the latter if $x_i < zero$. Γ is an operator that keeps the d largest values of x^* and sets the rest to zero [14].

Chapter 3

In-silico Simulations

3.1 Implementation

In this work we used Python as an implementation tool. Here we used the libraries Numpy and Matplotlib, whereas the former is a free powerful scientific computing tool. It is equipped with an optimized linear algebra packages as well as module to generate random events. The latter is a presentation library which we used to present our performance tests.

In the beginning we generated a random matrix using the module random from Numpy where it generated an array of zeros and ones uniformally distributed.

We wrote pieces of code, these are, an implementation of IHT and its modification NHIT, also some tests on it. The other is an implementation of BIHT and also accompanied with some tests.

We used various functions in Numpy we conducted the calculations and the iteration in the algorithm. In some occasions we produced our own functions as to help in the calculations.

We used the computing infrastructure **Colab** offered by Google as a running server for our code and experiments.

Next we present some segments of the code and its output. A full and documented code is available in the appendix.

Here is an example of a matrix $A^{4\times 9}$ and a sparse vector $x^{9\times 1}$ with sparsity $d = 2$ and also the measurement vector y, where the matrix A is generated randomly as well as x.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, \quad x = [0,1,0,0,0,1,0,0,0]^T \text{ and } y = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

The first column of the matrix A above reads as the first individual is tested in the tests 2 and 3 but not tested in the first test and the last test, whereas the vector y tells us that the first ,second and fourth tests are positive which implies that at least one those participated

in these tests is infected.

Here is an example of the basic iterative threshold algorithm put to action applied on A, y and with 100 iterations

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad , y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \text{ and } x = [0,0,1,0,1,0,0,0,0]^T$$

the result of iteration is the vector $x^* = [0,0,1,1,1,0,1,1,0]^T$

Let us define an operator, σ , which is in simple words the difference between two vectors index by index, i.e. it counts the differences.

Here is a histogram of the differences of 3000 vectors with $n = 14, m = 7, d = 2$ and their recovered vectors using the basic thresholding algorithms with 1000 iterations.

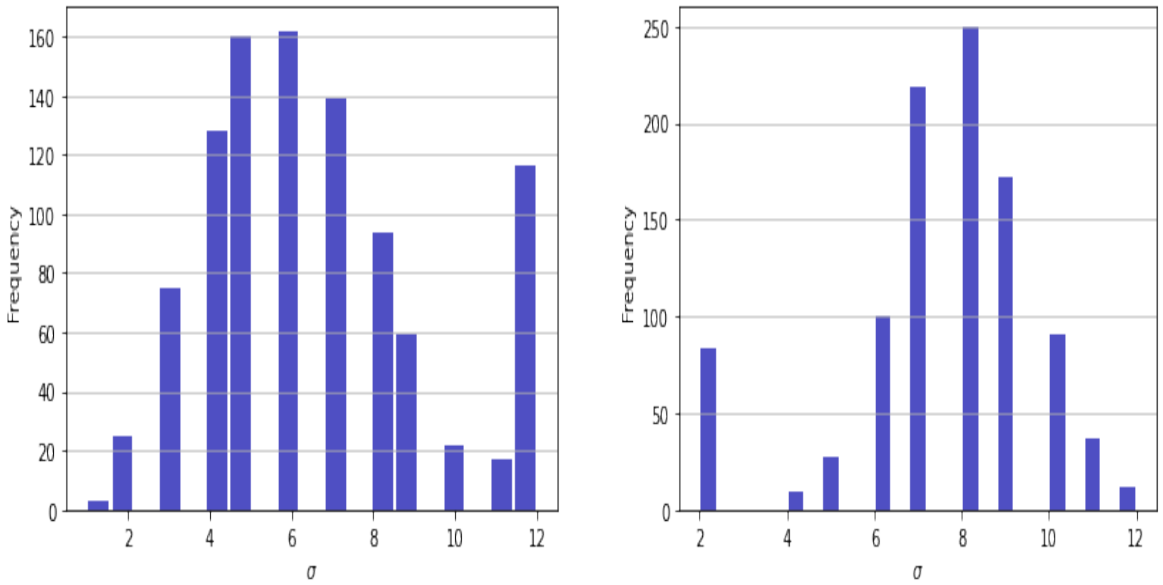

Figure 3.1: σ between the vectors and their recovered using the basic threshold iterative algorithm (left) and the normalized threshold iterative algorithm (right)for 3000 vectors

Now lets test the probability of exact recovery of the algorithm. We ran a 100 simulation of a sample of 100 ($n = 100$) and we took different values of m (tests) each time, that is we increase m be 5 up to n.

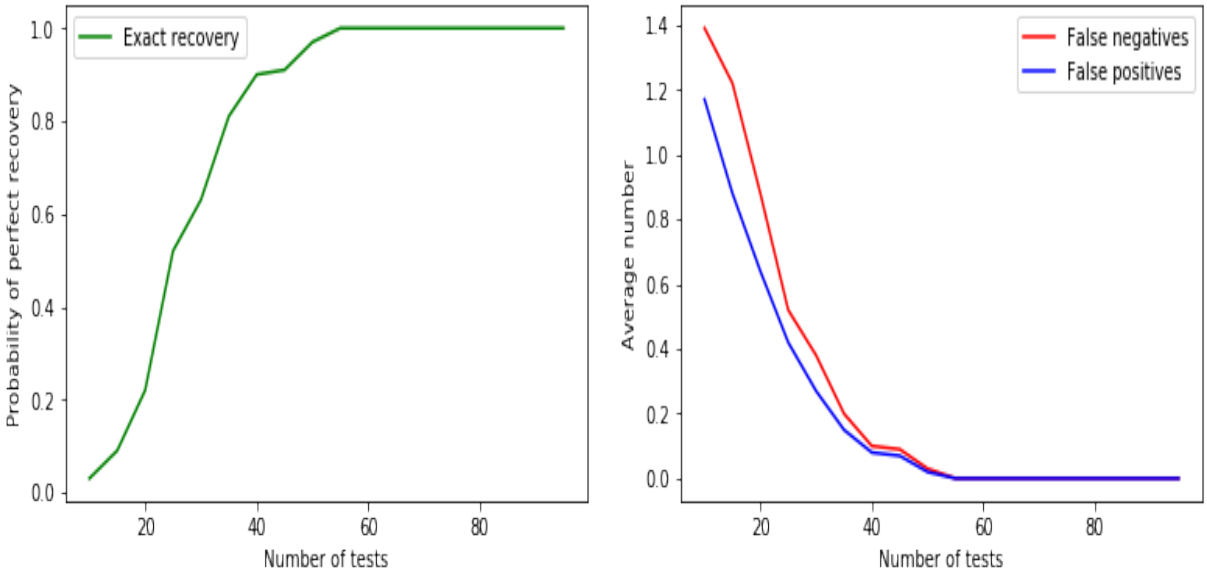

Figure 3.2: This figure shows the probability of exact recovery (left) and the average of false positive and false negatives vs the number of tests m (right), with number of individuals = 100 and the number of defectives 2

3.2 Binary iterative hard thresholding performance

In this section we apply some tests on the binary iterative thresholding algorithm. We chose this algorithms as it is convenient to work with, because of its binary nature, as well as its robustness and stability which is shown in [14].

Primarily, we are testing the probability of exact recovery, specificity (false positive), and sensitivity (false negative). We will also include the σ test mentioned above.

We run a number of simulations, in each of which we change the number of tests m according to the number individuals to be tested.

The following graphs represent results from simulating the test 100 times with different number of individuals participating in the test.

It is important that in both of following results the gradient descent (step size) is kept fixed at $\mu = 0.001$

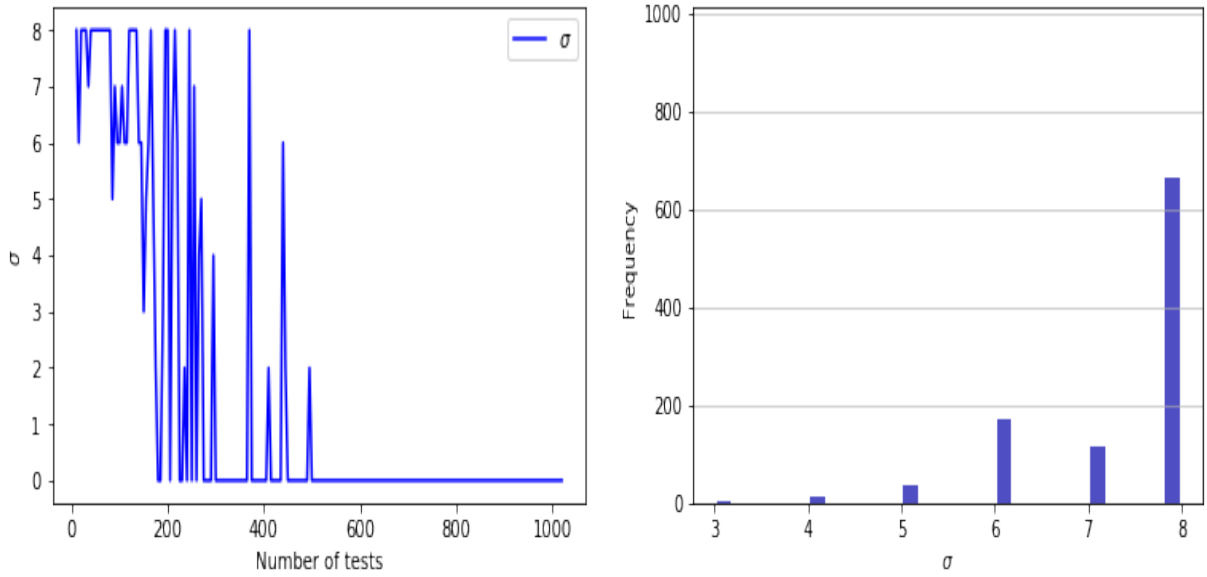

Figure 3.3: σ vs the number of tests m on the right and σ between the vectors and their recovered using the binary threshold iterative algorithm (BIHT) simulated 1000

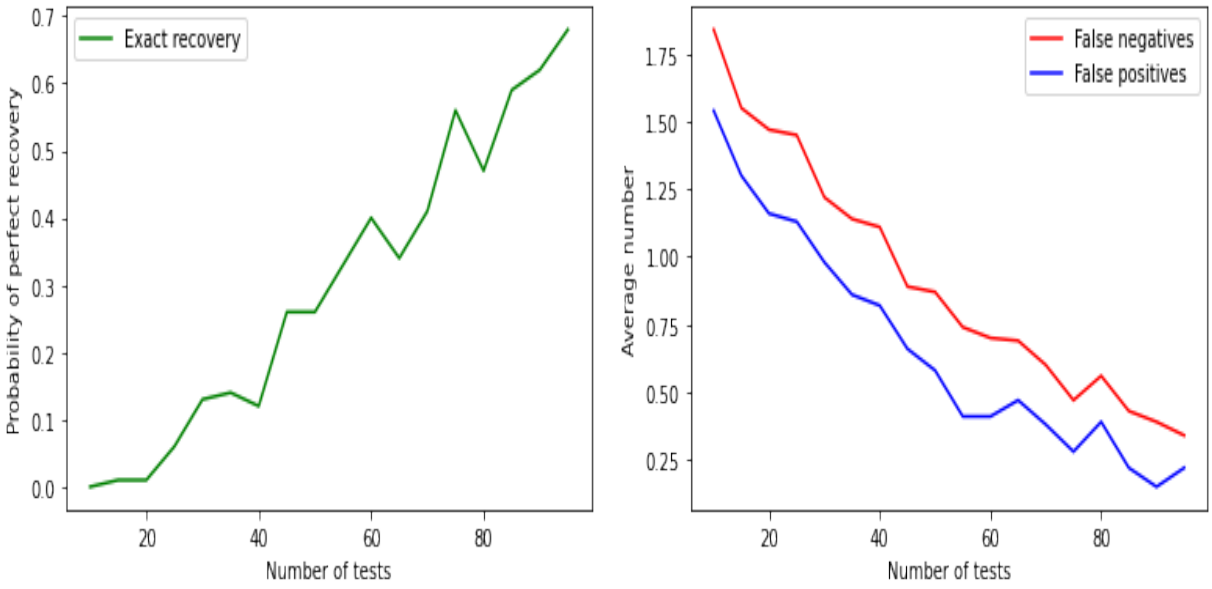

Figure 3.4: This figure shows the probability of exact recovery (left) and the average of false positive and false negatives vs the number of tests m (right), with number of individuals = 512 and the number of defectives 2, with $\mu = 0.001$

Next we increase the number of individuals and watch the three indicators mentioned above, taking the same test for 512, and 1024 individuals also keeping the number defectives low.

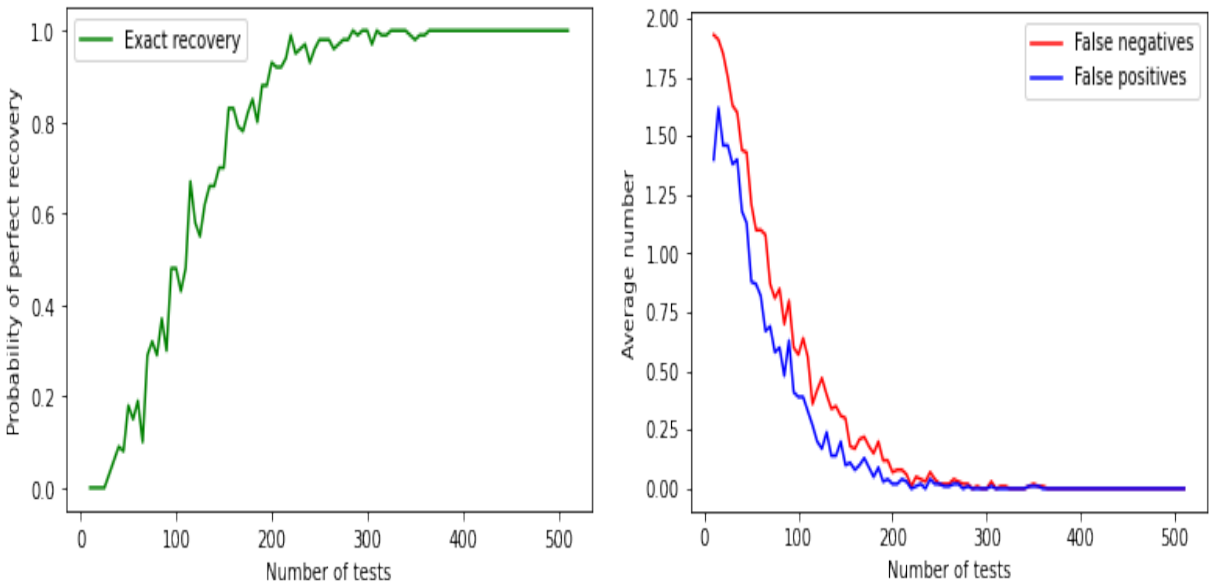

Figure 3.5: This figure shows the probability of exact recovery (left) and the average of false positive and false negatives vs the number of tests m (right), with number of individuals = 512 and the number of defectives 2, with $\mu = 0.001$

Now we change the gradient descent μ, we set to 0.1 which relatively bigger than what chose first 0.001.

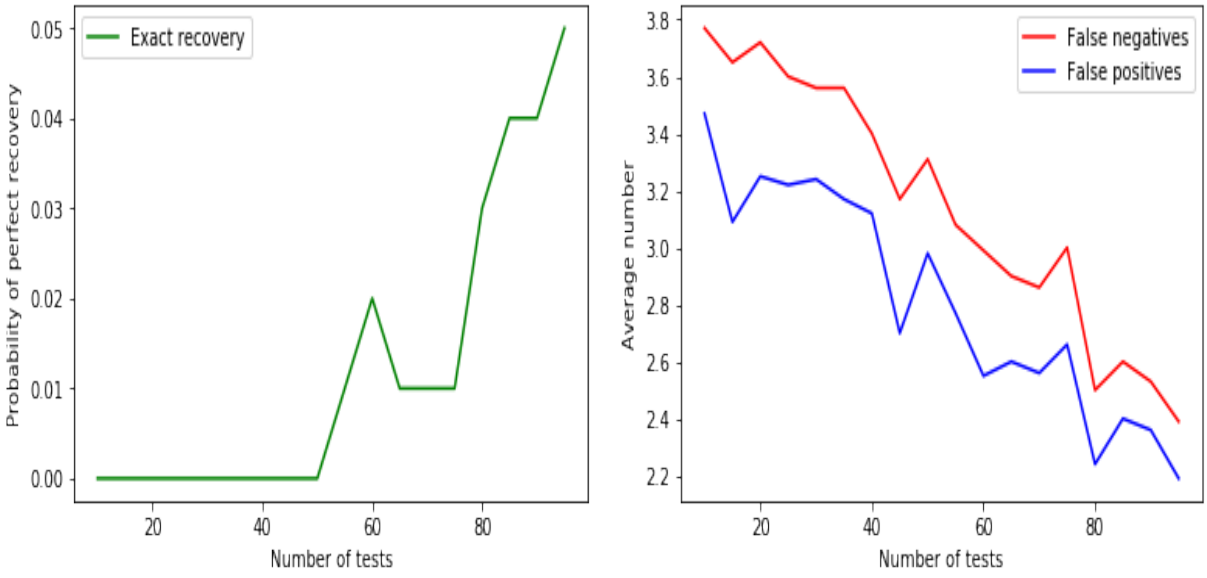

Figure 3.6: This figure shows the probability of exact recovery (left) and the average of false positive and false negatives vs the number of tests m (right), with number of individuals = 512 and the number of defectives 2, here $\mu = 0.1$

3.2.1 Notes on the results of the BIHT simulation

- The σ test that we run on the algorithm roughly tells us the convergence of the algorithm, where as we can see in 3.3 that with simulating the recovery 1000 times, most the σs are around $2d$, being the number of defectives.
- The exact (perfect) recovery probability increases with increasing the number of tests m, but it noticed that from 3.4 on the left, 3.5, with the increase of n, the algorithm starts to be more efficient, and it gives an exact recovery of probability of 1 earlier.
- The false-positives and false-negative starts to fall sharply with increase of but with number of tests approaches certain point they being to stabilize at 0 3.5 on the right.
- As μ being very small the curves starts to be smother 3.5 and 3.6.

3.3 Python code

The codes that are used in this thesis are available at: github.com/KhalidOmer

The file README.md contains a brief description of the codes and what they do as well as how to run them.

Conclusion

The idea of group testing is that we divide the population into groups. For each group then mix the samples drawn from each individual in the group and test the mixture. If the result of the test comes out negative then the group is free of the virus and if the result comes out positive then at least one individual of the group is infected.

The goal of group testing research is to minimize the number of tests required to identify infected individuals in a population. The problem is equivalent to solve a system of linear equations, with the number of equations being less the number of the unknowns. Generally, there are two types of group testings, adaptive tests and non-adaptive test. In this work we consider the non-adaptive tests, where the tests are fixed in advance and conducted in parallel.

We present two topics here, first, the construction of efficient test designs. With the perfect design we can reduce the cost of detection, time and storage wise. The second we review an algorithm that works on retrieving the missing information. Here we verify that The Hard thresholding algorithms are robust and accurate in the decoding process.

References

[1] title=An invitation to compressive sensing, author=Foucart, Simon and Rauhut, Holger, booktitle=A mathematical introduction to compressive sensing, pages=1–39, year=2013, publisher=Springer

[2] title=Combinatorial group testing and its applications, author=Du, Dingzhu and Hwang, Frank K and Hwang, Frank, volume=12, year=2000, publisher=World Scientific

[3] title=Group testing: an information theory perspective, author=Aldridge, Matthew and Johnson, Oliver and Scarlett, Jonathan and others, journal=Foundations and Trends® in Communications and Information Theory, volume=15, number=3-4, pages=196–392, year=2019, publisher=Now Publishers, Inc.

[4] title=Hard thresholding pursuit: an algorithm for compressive sensing, author=Foucart, Simon, journal=SIAM Journal on numerical analysis, volume=49, number=6, pages=2543–2563, year=2011, publisher=SIAM

[5] title=Iterative thresholding for sparse approximations, author=Blumensath, Thomas and Davies, Mike E, journal=Journal of Fourier analysis and Applications, volume=14, number=5, pages=629–654, year=2008, publisher=Springer

[6] title=Iterative hard thresholding: Theory and practice, author=Blumensath, Thomas, journal=Inst. Digit. Commun., Signal Image Process., Univ. Edinburgh, Edinburgh, UK, Tech. Rep, year=2009

[7] title = Optimized projections for compressed sensing via direct mutual coherence minimization, journal = Signal Processing, volume = 151, pages = 45-55, year = 2018, issn = 0165-1684, doi = https://doi.org/10.1016/j.sigpro.2018.04.020, url = https://www.sciencedirect.com/science/article/pii/S0165168418301464, author = Canyi Lu and Huan Li and Zhouchen Lin

[8] title=A simple proof of the restricted isometry property for random matrices, author=Baraniuk, Richard and Davenport, Mark and DeVore, Ronald and Wakin, Michael, journal=Constructive Approximation, volume=28, number=3, pages=253–263, year=2008, publisher=Springer

[9] title=Improved algorithms for non-adaptive group testing with consecutive positives, author=Bui, Thach V and Cheraghchi, Mahdi and Nguyen, Thuc D, booktitle=2021

IEEE International Symposium on Information Theory (ISIT), pages=1961–1966, year=2021, organization=IEEE

[10] title = The restricted isometry property and its implications for compressed sensing, journal = Comptes Rendus Mathematique, volume = 346, number = 9, pages = 589-592, year = 2008, issn = 1631-073X, doi = https://doi.org/10.1016/j.crma.2008.03.014, url = https://www.sciencedirect.com/science/article/pii/S1631073X08000964, author = Emmanuel J. Candès

[11] title = Academic press library in signal processing, Volume 7: Array, radar and communications engineering, author = Chellappa, Rama and Theodoridis, S., year = 2017, month = 12, pages = 1-626, journal = Academic Press Library in Signal Processing, Volume 7: Array, Radar and Communications Engineering, doi = 10.1016/C2016-0-00738-6

[12] title="Sparse Recovery Algorithms: Sufficient Conditions in Terms of RestrictedIsometry Constants", author="Foucart, Simon", editor="Neamtu, Marian and Schumaker, Larry", booktitle="Approximation Theory XIII: San Antonio 2010", year="2012", publisher="Springer New York", address="New York, NY", pages="65–77",

[13] title=1-bit compressive sensing, author=Boufounos, Petros T and Baraniuk, Richard G, booktitle=2008 42nd Annual Conference on Information Sciences and Systems, pages=16–21, year=2008, organization=IEEE

[14] author = Laurent Jacques and Jason N. Laska and Petros Boufounos and Richard G. Baraniuk, title = Robust 1-Bit Compressive Sensing via Binary Stable Embeddings of Sparse Vectors, journal = CoRR, volume = abs/1104.3160, year = 2011, url = http://arxiv.org/abs/1104.3160, eprinttype = arXiv, eprint = 1104.3160, timestamp = Mon, 13 Aug 2018 16:46:15 +0200, biburl = https://dblp.org/rec/journals/corr/abs-1104-3160.bib, bibsource = dblp computer science bibliography, https://dblp.org