Introduction to Project II

Group Testing

Tutorial: Aldridge & al.

n people
k infected

Setup

- We have n people
- Infection model $\rightarrow$ combinatorial k infected
  $\hookrightarrow$ probabilistic infected with prob. p

$$\bar{k} = n \cdot p$$

$$u_i = \begin{cases} 1 & \text{if person } i \text{ infected} \\ 0 & \text{otherwise} \end{cases} \quad i = 1, \dots, n \qquad u = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}$$

We perform $T(n)$ tests

- for test t

$$x_{ti} = \begin{cases} 1 & \text{if person } i \text{ tested} \\ 0 & \text{otherwise} \end{cases}$$

- test output

$$y_t = \begin{cases} 1 & \text{if any tested person infected} \\ 0 & \text{otherwise} \end{cases}$$

$$y_t = \bigvee_{\substack{\text{OR} \\ i}} x_{ti} \, u_i$$

x_{ti}: person i takes part in test t

u_i: state of person

Adaptive & non-adaptive testing.

test matrix

one column corresponding to each person

one row corresponding to each test

								Outcome
1	1	1	1	0	0	0	0	Positive
0	0	0	0	1	1	1	1	Positive
1	1	0	0	0	0	0	0	Negative
0	0	1	0	0	0	0	0	Positive
0	0	1	0	1	1	0	0	Positive
0	0	0	0	1	0	0	0	Positive

$X = \{ x_{ti} \}$

→ Create a matrix

→ Observe test outcomes

→ Infer who is infection

?	?	?	?	?	?	?	
1	0	1	0	0	1	0	0
1	1	0	1	0	0	1	1
1	0	0	0	1	0	0	0
0	1	1	0	1	1	0	1
1	0	1	1	0	1	0	1

→ Hamming error rate: $\dfrac{\#\text{ of positions } \hat{K} \text{ \& } K \text{ differ}}{n}$

$\hat{K}$ → "decoded" set of infected people

K → true set of infected people

→ $Pr(\text{false positives}) = Pr(\hat{u}_i = 1 \mid u_i = 0)$

→ $Pr(\text{false negative}) = Pr(\hat{u}_i = 0 \mid u_i = 1)$

Claim: if we want $P(\text{error}) = 0$, then $T \geq \log \binom{n}{k}$

lower bound $\Rightarrow$ necessary condition.

T tests $\rightsquigarrow$ 2^T possible outcomes

$\binom{n}{k}$ possible sets of infected people.

$$2^T \geq \binom{n}{k} \rightsquigarrow T \geq \log_2 \binom{n}{k}$$

$$\binom{n}{k} \sim \left(\frac{n}{k}\right)^k \rightsquigarrow T \geq k \log \frac{n}{k} + O(k)$$

Adaptive Algorithms

① Dorfman's procedure

$$\underbrace{1 \quad 2 \quad 3}_{y_1} \quad \underbrace{4 \quad 5}_{y_2} \quad \cdots\cdots \quad \underbrace{\quad n}_{y_m}$$

divide in m groups, each of size $\frac{n}{m}$

if $y_i = 1$, individually test everyone in the group

if $y_i = 0 \rightarrow$ not infected.

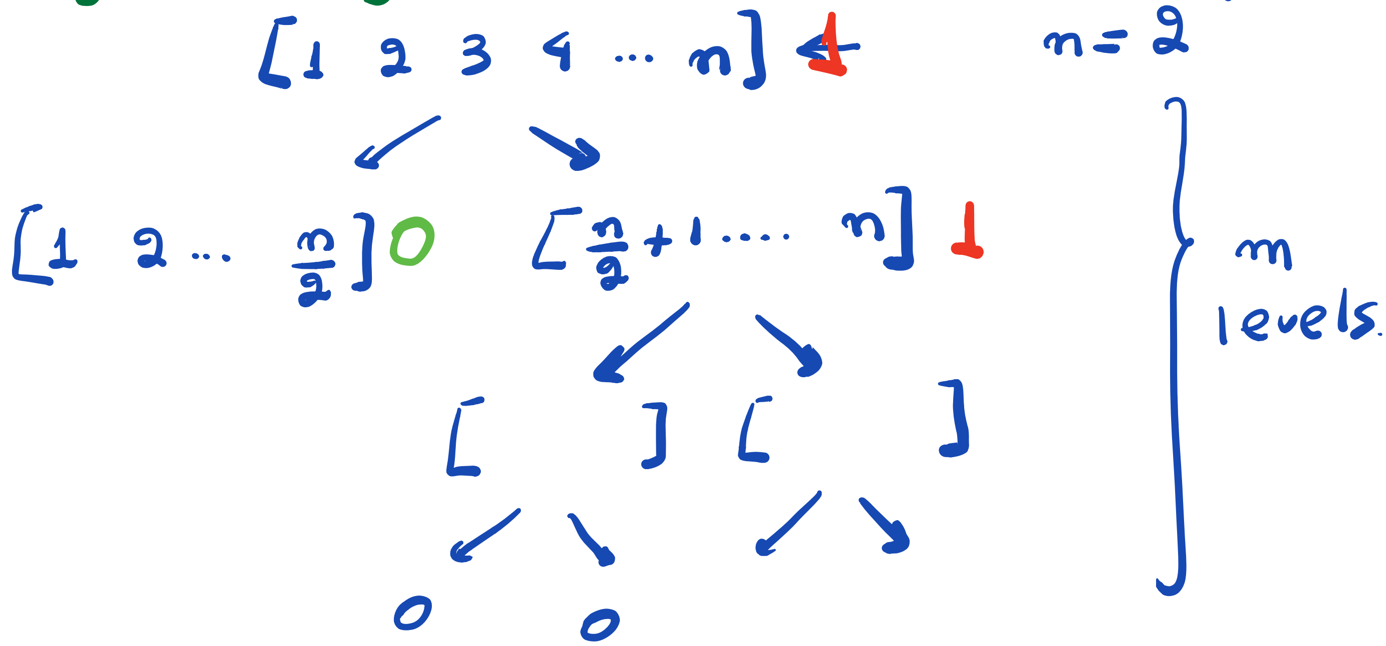
$$T(m) = m + k \cdot \frac{n}{m}$$

minimize T wrt m

$$m = \sqrt{kn}$$

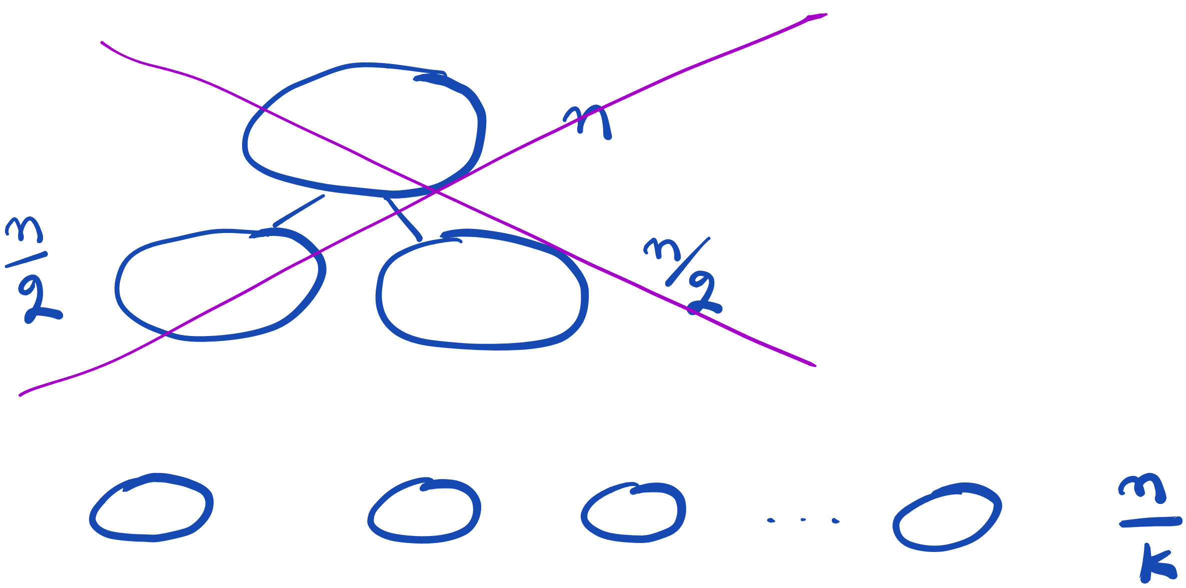
$$T = 2\sqrt{kn} \ll n$$

② Binary Splitting



Algorithm 1.1 (Binary splitting)**.** Given a set A:

1. Initialize the algorithm with set A. Perform a single test containing every item in A.
2. If the preceding test is negative, A contains no defective items, and we halt. If the test is positive, continue.
3. If A consists of a single item, then that item is defective, and we halt. Otherwise, pick half of the items in A, and call this set B. Perform a single test of the pool B.
4. If the test is positive, set $A := B$. If the test is negative, set $A := A \setminus B$. Return to Step 3.



Improvement

Algorithm 1.2. Divide the n items into k subsets of size n/k (rounding if necessary), and apply Algorithm 1.1 to each subset in turn.

Non-adaptive

$$\begin{array}{cc} & n \text{ columns} \\ \begin{array}{c} T \\ \text{rows} \end{array} & \begin{bmatrix} & 1 & \\ & 0 & \\ & 1 & \\ & 1 & \\ & 0 & \\ & 0 & \end{bmatrix} \end{array}$$

Typical approaches to design test matrix X: (i) Bernoulli design.

select iid entries to be 1 wp p_x

(ii) Constant weight column design.

select uniformly at random L positions in every column to makes

Not-adaptive algorithms

DND PD

?	?	?	?	?	?	?	?	y
1	1	1	1	0	0	0	0	1
0	0	0	0	1	1	1	1	1
1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	0	1	0	1	1	0	1	1
0	0	0	0	1	0	0	0	1

DD

COMP algorithm: identifying DND definetely not defective

PD → possibly defective - all remaining

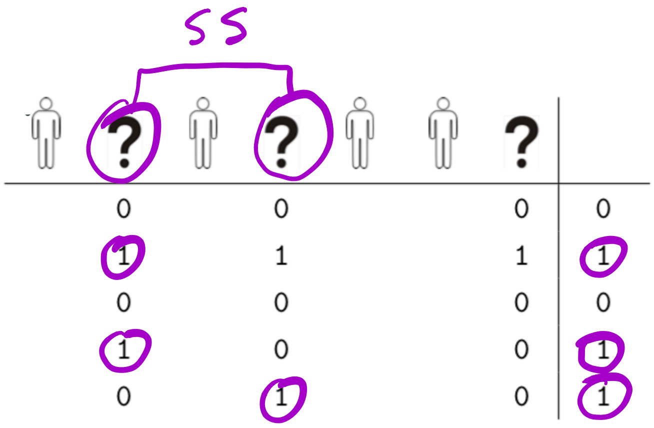
outputs $\hat{k}$ = PD

Algorithm 2.2. The *COMP* algorithm is defined as follows. We call any item in a negative test *definitely nondefective* (*DND*), and call the remaining items *possibly defective* (*PD*). Then the COMP algorithm outputs $\hat{\mathcal{K}}_{\mathrm{COMP}}$ equalling the set of possible defectives.

Algorithm 2.3. The *definite defectives* (*DD*) algorithm is defined as follows.

1. We say that any item in a negative test is *definitely nondefective* (*DND*), and that any remaining item is a *possible defective* (*PD*).
2. If any PD item is the only PD item in a positive test, we call that item *definitely defective* (DD).
3. The DD algorithm outputs $\hat{\mathcal{K}}_{\mathrm{DD}}$, the set of definitely defective items.

$\hat{k}$ = DD



Satisfying set $\mathcal{L}$

Definition 2.1. Consider the noiseless group testing problem with n items, using a test design X and producing outcomes $\mathbf{y}$. A set $\mathcal{L} \subset \{1, 2, \dots, n\}$ is called a *satisfying set* if

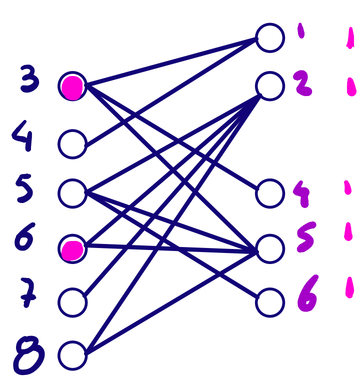
- every positive test contains at least one item from $\mathcal{L}$;
- no negative test contains any item from $\mathcal{L}$.

Smallest Satisfying Set (SSS)

	1	2	3	4	5	6	7	8		
	?	?	?	?	?	?	?	?	y	
	1	1	1	1	0	0	0	0	1	1
	0	0	0	0	1	1	1	1	1	2
	1	1	0	0	0	0	0	0	0	3
	0	0	1	0	0	0	0	0	1	4
	0	0	1	0	1	1	0	1	1	5
	0	0	0	0	1	0	0	0	1	6

one vertex for each person

one vertex for each test



this is the set cover problem

ILP formulation

$$z_i = \begin{cases} 1 & \text{if we include person } i \text{ in SS} \\ 0 & \text{otherwise} \end{cases}$$

↓
variables

$$\min \sum_{i=1}^{n} z_i$$

constant that determines if item i is in test t.

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ti} z_i \geq 1 \quad \text{if } y_t = 1$$

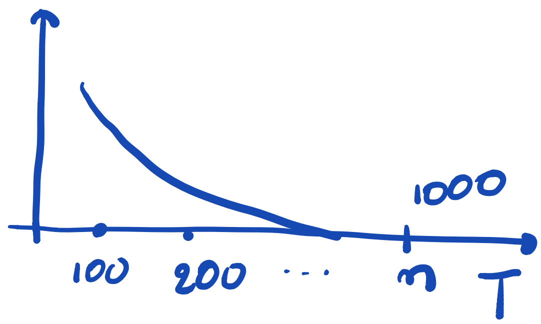
$$\sum_{i=1}^{n} x_{ti} z_i = 0 \quad \text{if } y_t = 0$$

DND

$$z_i \in \{0, 1\}$$

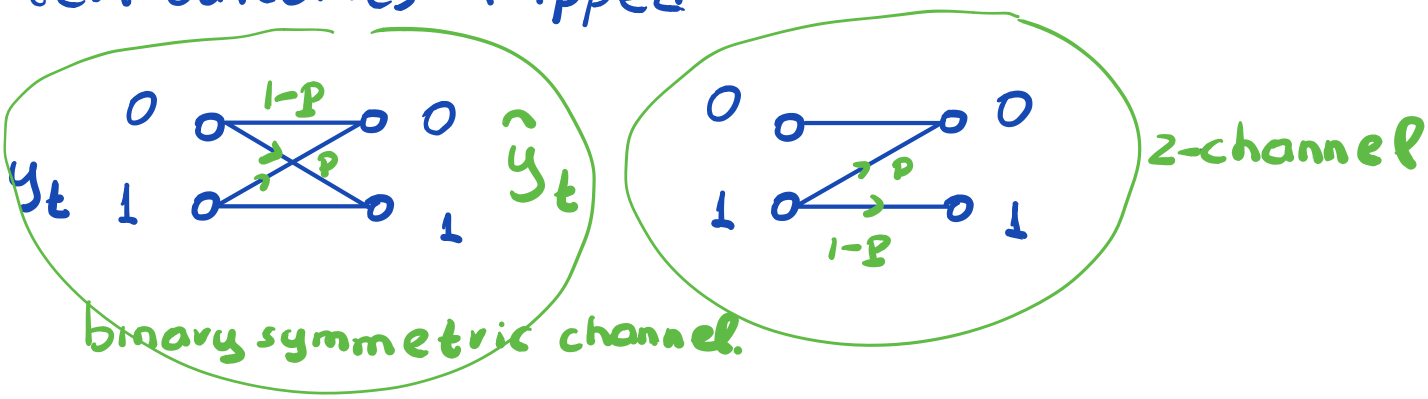
↙
relax this ILP to solve LP.

n people



Noisy Group Testing

test outcomes flipped



ILP formulation

z_i ~ state of individual

ξ_j ~ whether test outcome was correct or not.

$$\min_{z,\xi} \sum_{i=1}^{n} z_i + a \sum_{j=1}^{T} \xi_j$$

s.t. $\displaystyle\sum_{i=1}^{n} x_{ti} z_i \geq 1 - \xi_t$ when $\hat{y}_t = 1$

0

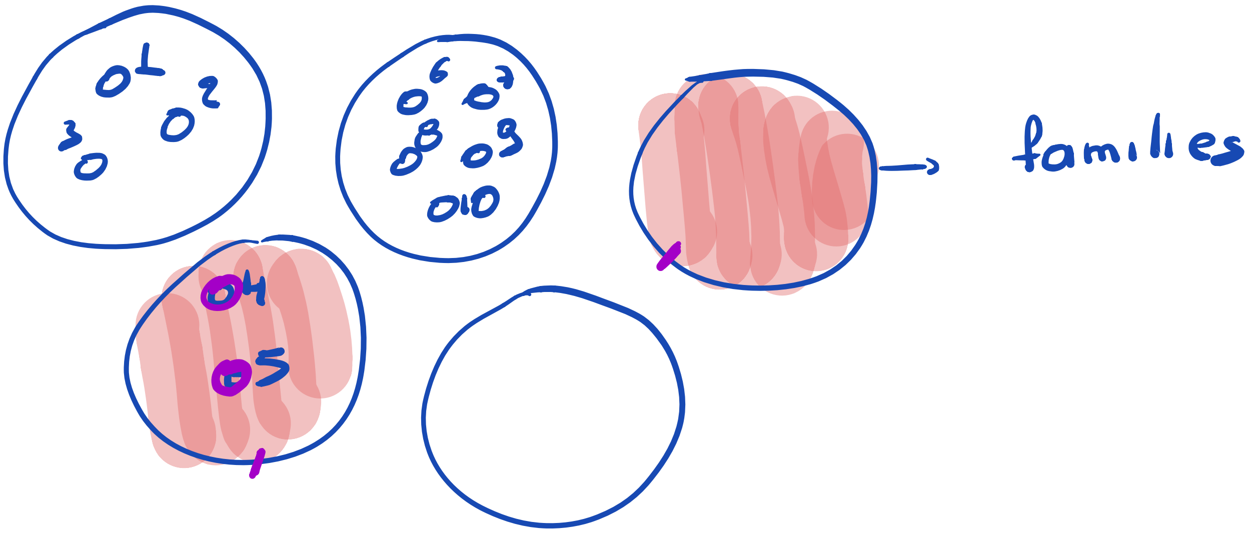
$\displaystyle\sum_{i=1}^{n} x_{ti} z_i \geq \xi_t$ when $\hat{y}_t = 0$

$z_i \in \{0, 1\}$

$\xi_t \in \{0, 1\}$

Project 2: Community Structure

Can we take into account a known community structure, to improve the performance of group testing?



- iid wp q select infected families
- inside each infected family wp p members are infected
- not-infected families have no infected members