CS612 Algorithm Design and Analysis

Lecture 19. Bi-Clustering problem: random sampling and random rounding [1]

Presented by Dongbo Bu

Institute of Computing Technology
Chinese Academy of Sciences, Beijing, China

[1] The slides are made based on *Approximation algorithms for Bi-clustering problems* by L. Wang, Y. Lin, and X. Liu.

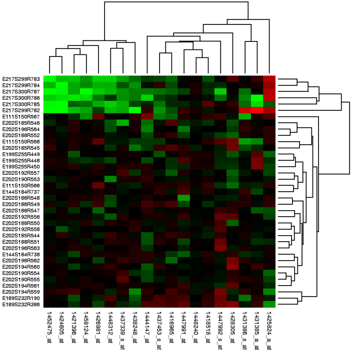
Outline I

- Introduction to BI-CLUSTERING problems;
- CONSENSUSSUBMATRIX problem: random sampling algo;
- BOTTLENECKSUBMATRIX problem: random rounding algo;

Background: What is DNA array?

DNA microarrays can be used to measure changes in expression levels of genes, to detect single nucleotide polymorphisms (SNPs) , to genotype or resequence mutant genomes.

- Row denotes a gene, and a column denotes a condition;
- Color: represent the expression levels of genes. Red: high, green: low.

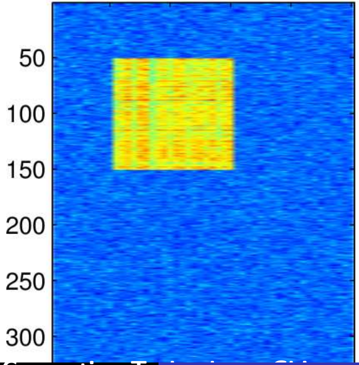


General Bi-clustering Problem

- Input: a $n \times m$ matrix A.
- Output: a sub-matrix $A_{P,Q}$ of A such that the rows of $A_{P,Q}$ are *similar*. That is, all the rows are identical.

  Why sub-matrix?

  A subset of *genes* are co-regulated and co-expressed under specific *conditions*. It is interesting to find the subsets of genes and conditions.

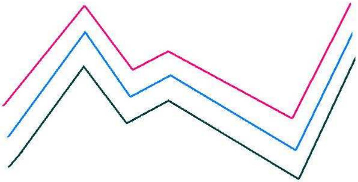


Similarity of Rows (1-5)

- 1. All rows are identical
  $1\ 1\ 2\ 3\ 2\ 3\ 3\ 2$
  $1\ 1\ 2\ 3\ 2\ 3\ 3\ 2$
  $1\ 1\ 2\ 3\ 2\ 3\ 3\ 2$
- 2. All the elements in a row are identical
  $1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$
  $2\ 2\ 2\ 2\ 2\ 2\ 2\ 2$
  $5\ 5\ 5\ 5\ 5\ 5\ 5\ 5$
  (the same as 1 if we treat columns as rows)

Similarity of Rows (1-5)

- 3. The curves for all rows are similar (additive) $a_{i,j} - a_{i,k} = c(j,k)$ for $i = 1, 2, \ldots, m$. Case 3 is equivalent to case 2 (thus also case 1) if we construct a new matrix $a^*_{i,j} = a_{i,j} - a_{i,p}$ for a fixed p indicate a row.



Similarity of Rows (1-5)

- 4. The curves for all rows are similar (multiplicative)

$$\begin{array}{ccccc} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,m} \\ c_1 a_{1,1} & c_1 a_{1,2} & c_1 a_{1,3} & \dots & c_1 a_{1,m} \\ c_2 a_{1,1} & c_2 a_{1,2} & c_2 a_{1,3} & \dots & c_2 a_{1,m} \\ \dots & & & & \\ c_n a_{1,1} & c_n a_{1,2} & c_n a_{1,3} & \dots & c_n a_{1,m} \end{array}$$

Transfer to case 2 (thus case 1) by taking log and subtraction.
Case 3 and Case 4 are called bi-clusters with coherent values.

Similarity of Rows (1-5)

- 5. The curves for all rows are similar (multiplicative and additive)

  $$a_{i,j} = c_i a_{k,j} + d_i$$

  Transfer to case 2 (thus case 1) by subtraction of a fixed row (row i), taking log and subtraction of row i again.
  The basic model: All the rows in the sub-matrix are identical.

Cheng and Church's model

The model introduced a similarity score called the mean squared residue score H to measure the coherence of the rows and columns in the submatrix.

$$H(P,Q) = \frac{1}{|P||Q|} \sum_{i \in P, j \in Q} (a_{i,j} - a_{i,Q} - a_{P,j} + a_{P,Q})^2$$

where

$$a_{i,Q} = \frac{1}{|Q|} \sum_{j \in Q} a_{i,j}, \quad a_{P,j} = \frac{1}{|P|} \sum_{i \in P} a_{i,j}, a_{P,Q} = \frac{1}{|P||Q|} \sum_{i \in P, j \in Q} a_{i,j}.$$

If there is no error, H(P, Q)=0 for case 1, 2 and 3. A lot of heuristics (programs) have been produced.

J. Liu's statistical model

Consider a microarray dataset with N genes and P conditions (or samples), in which the expression value of the i^{th} gene and j^{th} condition is denoted as y_{ij}, $i = 1, 2, \bullet\bullet\bullet, N$, $j = 1, 2, \bullet\bullet\bullet, P$. We assume that

$$Y_{ij} = \sum_{k=1}^{K}((\mu_k + \alpha_{ik} + \beta_{jk} + \epsilon_{ijk})\delta_{ik}\kappa_{jk}) + e_{ij}(1 - \sum_{k=1}^{K}\delta_{ik}\kappa_{jk}),$$

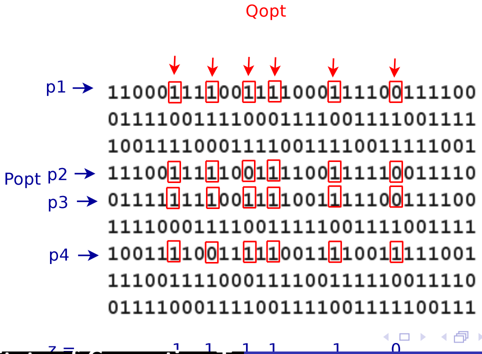
where K is the total number of clusters (unknown), μ_k is the main effect of cluster k, and α_{ik} and β_{jk} are the effects of gene i and condition j, respectively, in cluster k, ε_{ijk} is the noise term for cluster k, and e_{ij} models the data points that do not belong to any cluster. Here δ_{ik} and κ_{jk} are

Consensus Sub-matrix Problem

- Input: a $n \times m$ matrix A, integers l and k.
- Output: a sub-matrix $A_{P,Q}$ of A with l rows and k columns and a consensus row z (of k elements) such that

$\sum_{r_i \in P} d(r_i|^Q, z)$ is minimized.

Here $d(\ ,\ )$ is the Hamming distance.



Bottleneck Sub-matrix Problem

- Input: a $n \times m$ matrix A, integers l and k.
- Output: a sub-matrix $A_{P,Q}$ of A with l rows and k columns and a consensus row z (of k elements) such that for any r_i in P

  $d(r_i|^Q, z) \leq d$ and d is minimized

  Here $d(\, , \,)$ is the Hamming distance.

NP-Hardness Results

- Theorem 1: Both consensus sub-matrix and bottleneck sub-matrix problems are NP-hard.

  Proof: We use a reduction from maximum edge bipartite problem.

Approximation Algorithm for Consensus Sub-matrix Problem

- Input: a $n \times m$ matrix A, integers l and k.
- Output: a sub-matrix $A_{P,Q}$ of A with l rows and k columns and a consensus row z (of k elements) such that

  $\sum_{r_i \in P} d(r_i|^Q, z)$ is minimized.

  Here $d(\ , \ )$ is the Hamming distance.

Trial: brute-force

A brute-force method:

- By enumerating all size k subset of columns, and all length k vector, we could know Q_{opt} and z at some moment;
- Then we can find P_{opt} in poly-time to minimize the consensus score.
- However, the first step will take $\binom{n}{k} \times 2^k$ time.

Our method

- Basic idea: instead of the whole Q_{opt} and z_{opt}, knowing a small part is enough. In other words, the whole Q_{opt} can be approximated based on the small part.
- Key questions:
  1. What is the size of the small part?
  2. How to approximate the whole Q_{opt} based on the small part?
  3. How to obtain such a small part?

Algorithm 1 for The Consensus Submatrix Problem

Basic Ideas:

- We use a random sampling technique to randomly select O(log$m$) columns in Q_{opt}, enumerate all possible vectors of length O(log$m$) for those columns.
- At some moment, we know O(log$m$) bits of r_{opt} and we can use the partial z_{opt} to select the l rows which are closest to z_{opt} in those O(log$m$) bits.
- After that we can construct a consensus vector r as follows: for each column, choose the (majority) letter that appears the most in each of the l letters in the l selected rows.
- Then for each of the n columns, we can calculate the number of mismatches between the majority letter and the l letters in the l selected rows. By selecting the best k columns, we can get a good solution.

Input: one $m \times n$ matrix A, integers l and k, and $\epsilon > 0$
Output: a size l subset P of rows, a size k subset Q of columns and a length k consensus vector z
Step 1: randomly select a set B of $\lceil (c+1)(\frac{4 \log m}{\epsilon^2} + 1) \rceil$ columns from A.
(1.1) **for** every size $\lceil \frac{4 \log m}{\epsilon^2} \rceil$ subset R of B **do**
(1.2) **for** every $z|^R \in \Sigma^{|R|}$ **do**
(a) Select the best l rows $P = \{p_1, ..., p_l\}$ that minimize $d(z|^R, x_i|^R)$.
(b) **for** each column j **do**
Compute $f(j) = \sum_{i=1}^{l} d(s_j, a_{p_i,j})$, where s_j is the majority element of the l rows in P in column j. Select the best k columns $Q = \{q_1, ..., q_k\}$ with minimum value $f(j)$ and let $z(Q) = s_{q_1} s_{q_2} \ldots s_{q_k}$.
(c) Calculate $H = \sum_{i=1}^{l} d(x_{p_i}|^Q, z)$ of this solution.
Step 2: Output P, Q and z with minimum H.

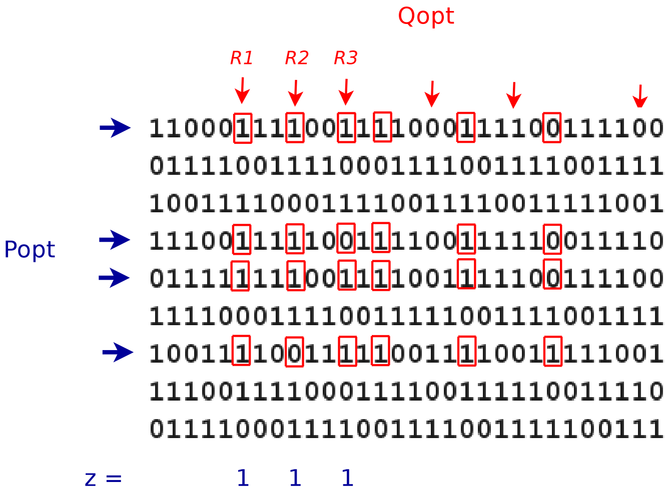
Qopt

R1 *R2* *R3*

Popt

```
110001111001111000111100111100
011110011110001111001111001111
100111100011110011110011111001
111001111100111100111110011110
011111111001111001111100111100
111100011110011111001111001111
100111100111110011110011111001
111001111000111100111110011110
011110001111001111001111100111
```

Step 1: randomly sampling (1+c)logm columns,
and enumerating all log(m) columns,
we will know log(m) bits of Qopt with high prob.
Denote these bits as R.



Step 2: enumerating all possible z|R to know z_opt|R. We can esitmate d(a_i|Q, z|Q) from d(a_i|R, z|R). Choose the best l rows.

Qopt

R1 R2 R3

Popt

```
110001111001111000111100111100
011110011110001111001111001111
100111100011110011110011111001
111001111100111100111110011110
011111111001111001111100111100
111100011110011111001111001111
100111100111110011110011111001
111001111000111100111110011110
011110001111001111001111100111
```

z = 1 1 1 1 1 1

Step 3: Considering the selected rows P.
For each column, caculating the majority,
and use the majority as z,
select the best k columns.

Question 1,2: what size of "small part" is enough? how to approximate? I

Lemma 2: Randomly sample (with replaccement) of $R \subseteq Q_{opt}$, where $|R| = \lceil \frac{4 \log m}{\epsilon^2} \rceil$ and. Let $\rho = \frac{k}{|R|}$. With probability at most m^{-1}, there is a row a_i satisfying

$$\frac{d(z_{opt}, a_i|^{Q_{opt}}) - \epsilon k}{\rho} > d(z_{opt}|^R, a_i|^R).$$

With probability at most $m^{-\frac{1}{3}}$, there is a row a_i satisfying

$$d(z_{opt}|^R, a_i|^R) > \frac{d(z_{opt}, a_i|^{Q_{opt}}) + \epsilon k}{\rho}.$$

Intuition: randomly sample a small subset of Q_{opt} is enough!

```
            |R|= logm
            R1  R2  R3          clogm
Row ai:   1 1 0 0 0 1 1 1 1 0 0 1 1 1 0 0 0 1

z_opt:    1 1 0 1 0 1            means Qopt.

d:        0 0 0 1 0 0
H(z_opt, ai):   0+0+0+1+0+0 = 1
H(z_opt|R,ai|R):  0  +1  +0 = 1
```

Proof:

- Define index variables $x_j = 1$ if j was selected into R, and 0 otherwise.
- $E(d(z_{opt}|^R, a_i|^R)) = \sum_{j=1}^{k} E(x_j) \times d_j = \frac{|R|}{k} d(z_{opt}, a_i|^{Q_{opt}})$.

- For any row a_i,

$$
\begin{aligned}
& \Pr\left(\frac{d(z_{opt}, a_i|^{Q_{opt}}) - \epsilon k}{\rho} > d(z_{opt}|^{R}, a_i|^{R})\right) \\
\leq \quad & exp(-\tfrac{1}{2}|R|\epsilon^2)\text{(by Chernoff bound)} \\
= \quad & m^{-2} \quad (\text{ set } |R| = \frac{4\log m}{\epsilon^2})
\end{aligned}
$$

Lemma 3: When $R \subseteq Q_{opt}$ and $z|^R = z_{opt}|^R$, with probability at most $2m^{-\frac{1}{3}}$, the set of rows $P = \{p_1, \ldots, p_l\}$ selected in Step 1 (a) of Algorithm 1 satisfies $\sum_{i=1}^{l} d(z_{opt}, x_{p_i}|^{Q_{opt}}) > H_{opt} + 2\epsilon kl$.
Intuition: R can be used to approximate Q_{opt}.
Proof:

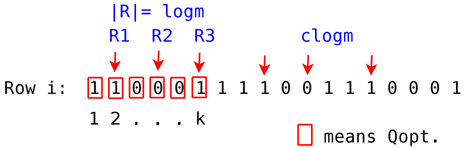
- With probability at most m^{-1}, $\sum_{i=1} ld(z_{opt}, a_i|Q_{opt}) - \epsilon kl \geq \rho \sum_{i=1} ld(z_{opt}|R, a_i|R)$.
- With probability at most $m^{-\frac{1}{3}}$,

$$\begin{aligned} H_{opt} &= \sum_{i=1} ld(z_{opt}, a_i|Q_{opt}) \\ &\leq \rho \sum_{i=1} ld(z_{opt}|R, a_i|R) - \epsilon kl \end{aligned}$$

- Thus the lemma follows by the two facts.

Question 3: how to obtain such a "small part"? I

- Difficulty: How to randomly select O(log$m$) columns in Q_{opt} while Q_{opt} is unknown?
- Our idea: to randomly select a LARGER subset B of $(c+1)$log$m$ columns, and enumerate all size log$m$ subsets of B in poly-time $O(m^{c+1})$.
- Lemma 1: With probability at most $m^{-\frac{2}{\epsilon^2 c^2 (c+1)}}$, no subset R of size $\lceil \frac{4 \log m}{\epsilon^2} \rceil$ used in Step 1 of Algorithm 1 satisfies $R \subseteq Q_{opt}$.
- Intuition: With high probability, we can get a set of log$m$ columns randomly selected from Q_{opt}.



Question 3: how to obtain such a "small part"? II

Proof:

- Define index variables $x_j = 1$ if the j-th trial hits a column in Q_{opt}, and 0 otherwise. Define $X = x_1 + x_2 + ... + x_t$, where $t = (c+1)(\frac{4 \log m}{\epsilon^2} + 1)$.
- $E(X) = t \times k/n = ct$ (assume $k = \Omega(n) = \frac{n}{c}$.)
- $\Pr(X \leq \frac{4log(m)}{\epsilon^2}) \leq exp(-\frac{1}{2}tc^2)$.

Analysis I

- Theorem 2: For any $\delta > 0$, with probability at least $1 - m^{-\frac{8c'^2}{\delta^2 c^2 (c+1)}} - 2m^{-\frac{1}{3}}$, Algorithm 1 will output a solution with consensus score at most $(1+\delta)H_{opt}$ in $O(nm^{O(\frac{1}{\delta^2})})$ time.
- Time-complexity:
  1. Step 1.1 is repeated $O(2^{\frac{4(c+1)\log m}{\epsilon^2}}) = O(m^{O(\frac{1}{\epsilon^2})}) = O(m^{O(\frac{1}{\delta^2})})$.
  2. Step 1.2 is repeated $O(m^{O(\frac{\log|\Sigma|}{\epsilon^2})}) = O(m^{O(\frac{1}{\delta^2})})$.
  3. Total time: $O(nm^{O(\frac{1}{\delta^2})})$.

Approximation Algorithm for Bottleneck Sub-matrix Problem

- Input: a $n \times m$ matrix A, integers l and k.
- Output: a sub-matrix $A_{P,Q}$ of A with l rows and k columns and a consensus row z (of k elements) such that for any r_i in P

  $d(r_i|^Q, z) \leq d$ and d is minimized

  Here $d(\ , \ )$ is the Hamming distance.

Basic Ideas

- Assumptions: $d_{opt} = MAX_{p_i \in P_{opt}} d(x_{p_i}|^{Q_{opt}}, z_{opt}) = O(k)$, $d_{opt} \times c'' = k$ and $|Q_{opt}| = k = O(n)$, $k \times c = n$.
- Basic Ideas:
  (1) Use random sampling technique to know O(log$m$) bits of z_{opt} and select l best rows like Algorithm 1.
  (2) After knowing the l rows, "LP+RR" technique is employed to select k columns in the matrix.

- Linear programming
  Given a set of rows $P = \{p_1, \ldots, p_l\}$, we want to find a set of k columns Q and vector z such that bottleneck score is minimized.

$$\min d;$$

$$\sum_{i=1}^{n} \sum_{j=1}^{|\Sigma|} y_{i,j} = k,$$

$$\sum_{j=1}^{|\Sigma|} y_{i,j} \leq 1, i = 1, 2, \ldots, n,$$

$$\sum_{i=1}^{n} \sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y_{i,j} \leq d, s = 1, 2, \ldots, l.$$

$y_{i,j} = 1$ if and only if column i is in Q and the corresponding bit in z is π_j.
Here, for any $a, b \in \Sigma$, $\chi(a, b) = 0$ if $a = b$ and $\chi(a, b) = 1$ if $a \neq b$.

- Randomized rounding
  To achieve two goals:
  (1) Select k' columns, where $k' \geq k - \delta d_{opt}$.
  (2) Get integers values for $y_{i,j}$ such that the distance (restricted on the k' selected columns) between any row in P and the center vector thus obtained is at most $(1+\gamma)d_{opt}$.
  Here $\delta > 0$ and $\gamma > 0$ are two parameters used to control the errors.

- Lemma 4: When $\frac{n\gamma^2}{3(cc'')^2} \geq 2\log m$, for any $\gamma, \delta > 0$, with probability at most $exp(-\frac{n\delta^2}{2(cc'')^2}) + m^{-1}$, the rounding result $y' = \{y'_{1,1}, \ldots, y'_{1,|\Sigma|}, \ldots, y'_{n,1}, \ldots, y'_{n,|\Sigma|}\}$ does not satisfy at least one of the following inequalities,

$$\sum_{i=1}^{n}(\sum_{j=1}^{|\Sigma|} y'_{i,j}) > k - \delta d_{opt},$$

and for every row $x_{p_s} (s = 1, 2, \ldots, l)$,

$$\sum_{i=1}^{n}(\sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y'_{i,j}) < \overline{d} + \gamma d_{opt}.$$

- Intuition: random rounding can generate a good approximation, i.e., $k' \geq k - \delta d_{opt}$ columns along with an objective value $d \leq d_{opt} + r d_{opt}$

Proof

- Denote $Y = \sum_{i=1}^{n}(\sum_{j=1}^{|\Sigma|} y'_{i,j})$. We have $E(Y) = k$.
-

$$
\begin{aligned}
& \Pr(Y \geq k - \delta d_{opt}) \\
\leq \quad & exp(-\frac{1}{2}n(\frac{\delta d_{opt}}{n})^2) \\
\leq \quad & exp(-\frac{1}{2}n(\frac{\delta}{cc''})^2) \quad (\text{ assume } d_{opt} = \Omega(k) = \frac{k}{c''} = \frac{n}{cc''})
\end{aligned}
$$

Algorithm 2 for The bottleneck Sub-matrix Problem

Input: one matrix $A \in \Sigma^{m \times n}$, integer l, k, a row $z \in \Sigma^n$ and small numbers $\epsilon > 0$, $\gamma > 0$ and $\delta > 0$.
Output: a size l subset P of rows, a size k subset Q of columns and a length k consensus vector z.
if $\frac{n\gamma^2}{3(cc'')^2} \leq 2\log m$ **then** try all size k subset Q of the n columns and all z of length k to solve the problem.
if $\frac{n\gamma^2}{3(cc'')^2} > 2\log m$ **then**

Step 1: randomly select a set B of $\lceil \frac{4(c+1)\log m}{\epsilon^2} \rceil$ columns from A. **for** every $\lceil \frac{4\log m}{\epsilon^2} \rceil$ size subset R of B **do**
for every $z|^R \in \Sigma^{|R|}$ **do**
(a) Select the best l rows $P = \{p_1, ..., p_l\}$ that minimize $d(z|^R, x_i|^R)$.
(b)Solve the optimization problem by linear programming and randomized rounding to get Q and z.

Step 2: Output P,Q and z with minimum bottleneck score d.

Proofs

- Lemma 5: When $R \subseteq Q_{opt}$ and $z|^R = z_{opt}|^R$, with probability at most $2m^{-\frac{1}{3}}$, the set of rows $P = \{p_1, \ldots, p_l\}$ obtained in Step 1(a) of Algorithm 2 satisfies $d(z_{opt}, x_{p_i}|^{Q_{opt}}) > d_{opt} + 2\epsilon k$ for some row $x_{p_i}(1 \leq i \leq l)$.
- Theorem 3: With probability at least $1 - m^{-\frac{2}{\epsilon^2 c^2 (c+1)}} - 2m^{-\frac{1}{3}} - exp(-\frac{n\delta^2}{2(cc'')^2}) - m^{-1}$, Algorithm 2 runs in time $O(n^{O(1)} m^{O(\frac{1}{\epsilon^2} + \frac{1}{\gamma^2})})$ and obtains a solution with bottleneck score at most $(1 + 2c''\epsilon + \gamma + \delta)d_{opt}$ for any fixed $\epsilon, \ \gamma, \ \delta > 0$.

Thanks

- Acknowledgements
  This work is fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 1070/02E].
  This work is collaborated with Dr. Lusheng Wang and Xiaowen Liu in City University of Hong Kong, Hong Kong, China.

Let $X_1, X_2, \ldots, X_n$ be n independent random 0-1 variables, where X_i takes 1 with probability p_i, $0 < p_i < 1$. Let $X = \sum_{i=1}^{n} X_i$, and $\mu = E[X]$. Then for any $0 < \epsilon \leq 1$,

$$\begin{aligned} \mathbf{Pr}(X > \mu + \epsilon\, n) &< e^{-\frac{1}{3}n\epsilon^2}, \\ \mathbf{Pr}(X < \mu - \epsilon\, n) &\leq e^{-\frac{1}{2}n\epsilon^2}. \end{aligned}$$