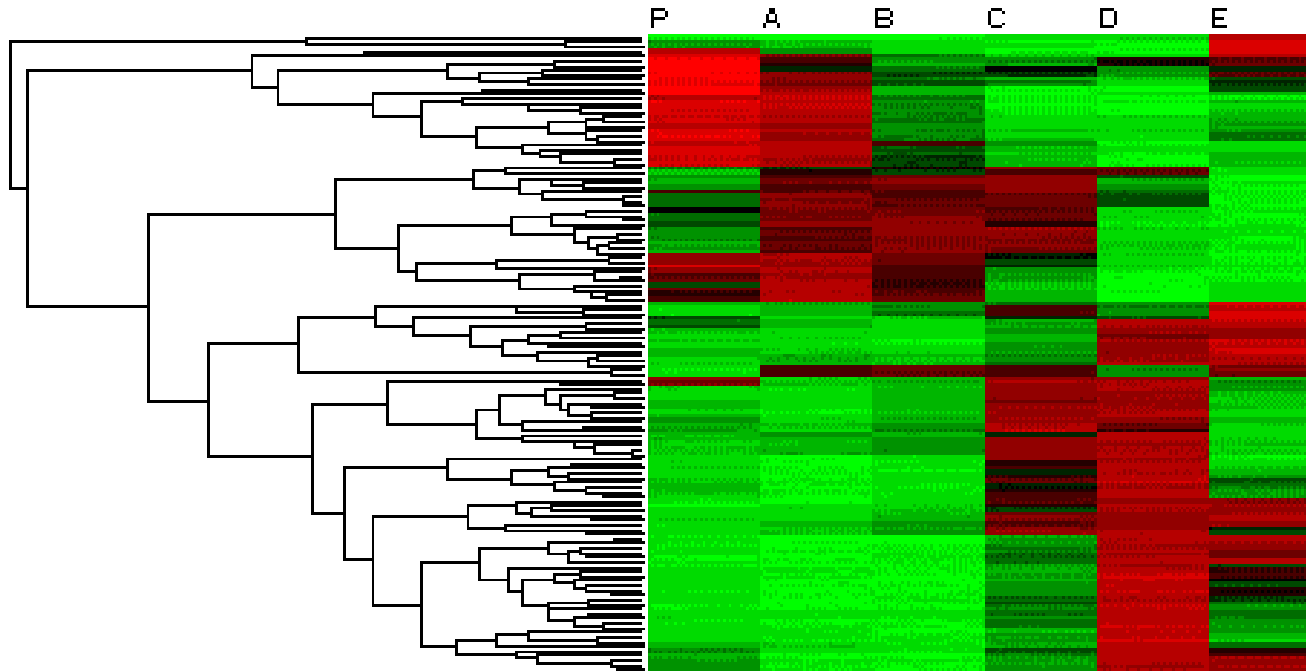


EECS730: Introduction to Bioinformatics

Lecture 15: Microarray clustering

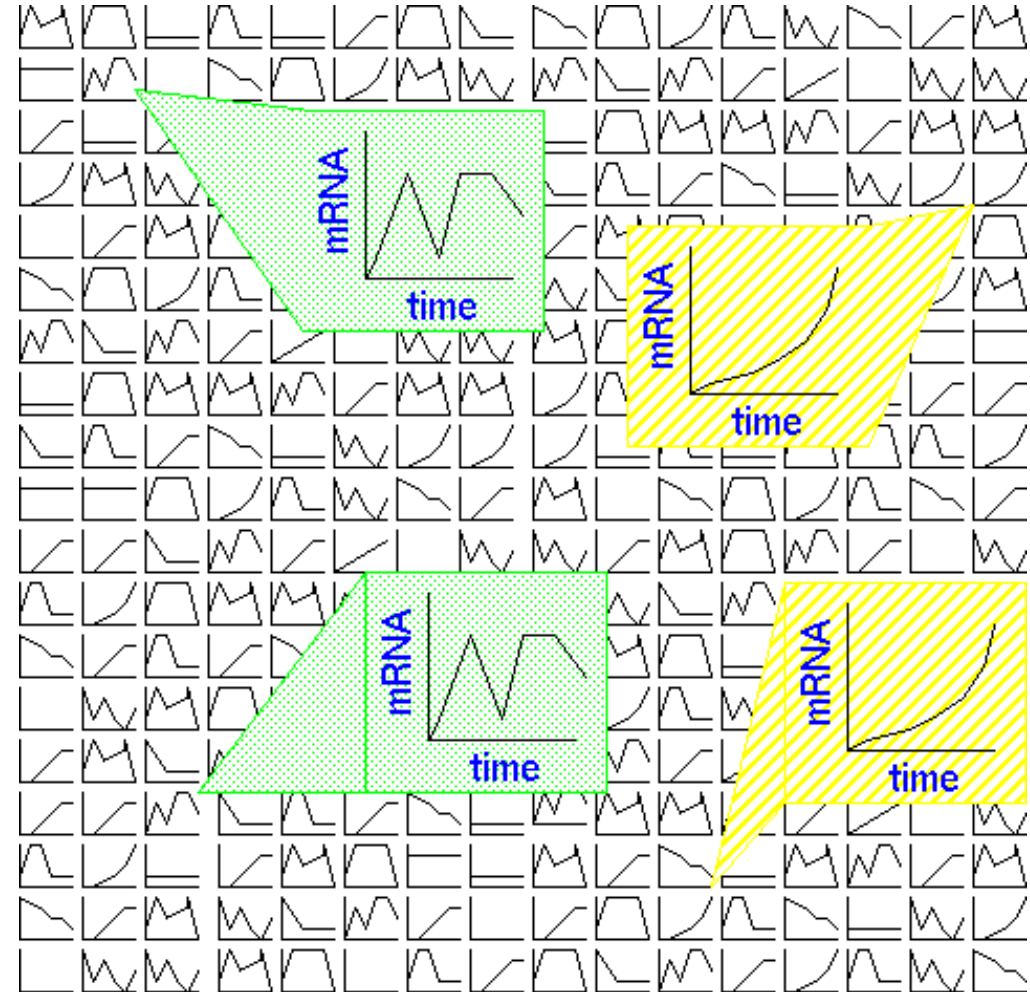


<http://compbio.pbworks.com/f/wood2.gif>

Some slides were adapted from Dr. Shaojie Zhang (University of Central Florida)

Microarray data clustering

- Finding genes that share similar expression pattern across different conditions
- The distance between gene expression profiles can be calculated as the Euclidean distance between the two genes in the N dimensional space, where N is the number of samples
- Or you can come up with your own similarity measure

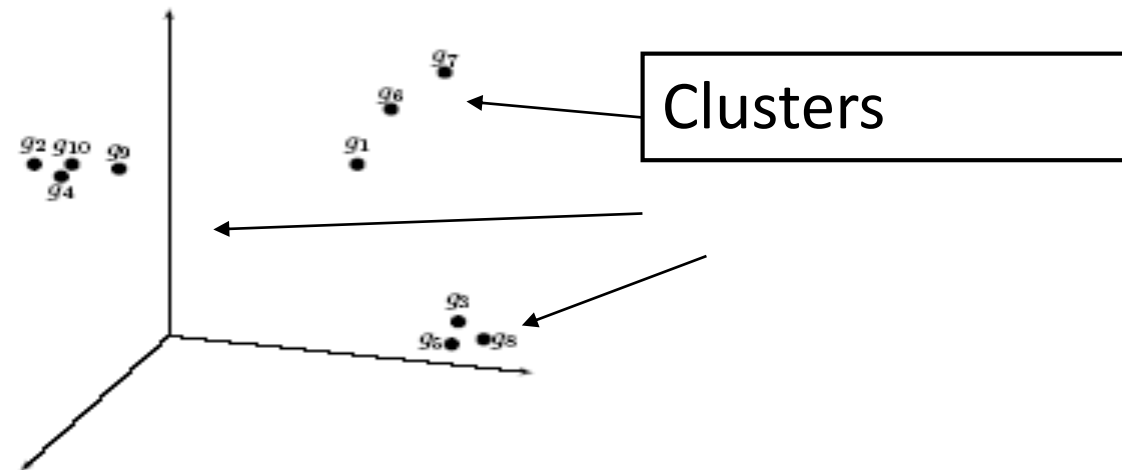


Cluster of microarray data

Time	1 hr	2 hr	3 hr		g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	10.0	8.0	10.0	g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	10.0	0.0	9.0	g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	4.0	8.5	3.0	g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	9.5	0.5	8.5	g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	4.5	8.5	2.5	g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	10.5	9.0	12.0	g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.0	8.5	11.0	g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	2.7	8.7	2.0	g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	9.7	2.0	9.0	g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	10.2	1.0	9.2	g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

(a) Intensity matrix, I

(b) Distance matrix, d

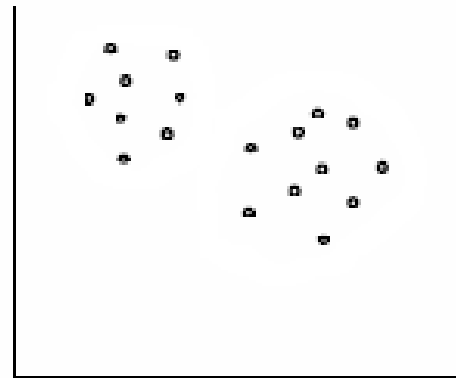


(c) Expression patterns as points in three-dimensional space.

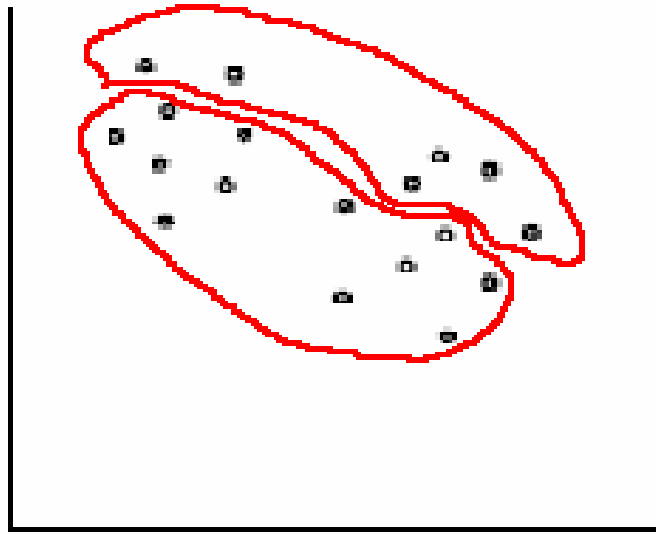
Homogeneity and Separation Principles

- **Homogeneity:** Elements within a cluster are close to each other (maximize in-cluster similarity)
- **Separation:** Elements in different clusters are further apart from each other (minimize between-cluster similarity)
- ...clustering is not an easy task! (no known polynomial solution that produces optimal answer)

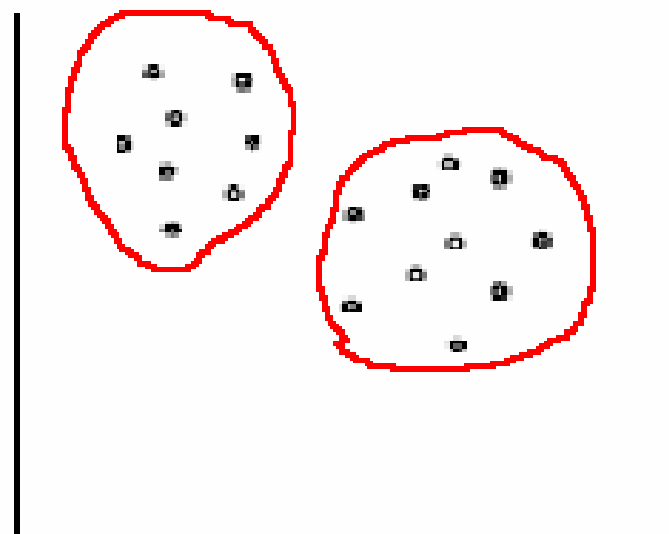
Given these points a clustering algorithm might make two distinct clusters as follows



Good and bad clustering

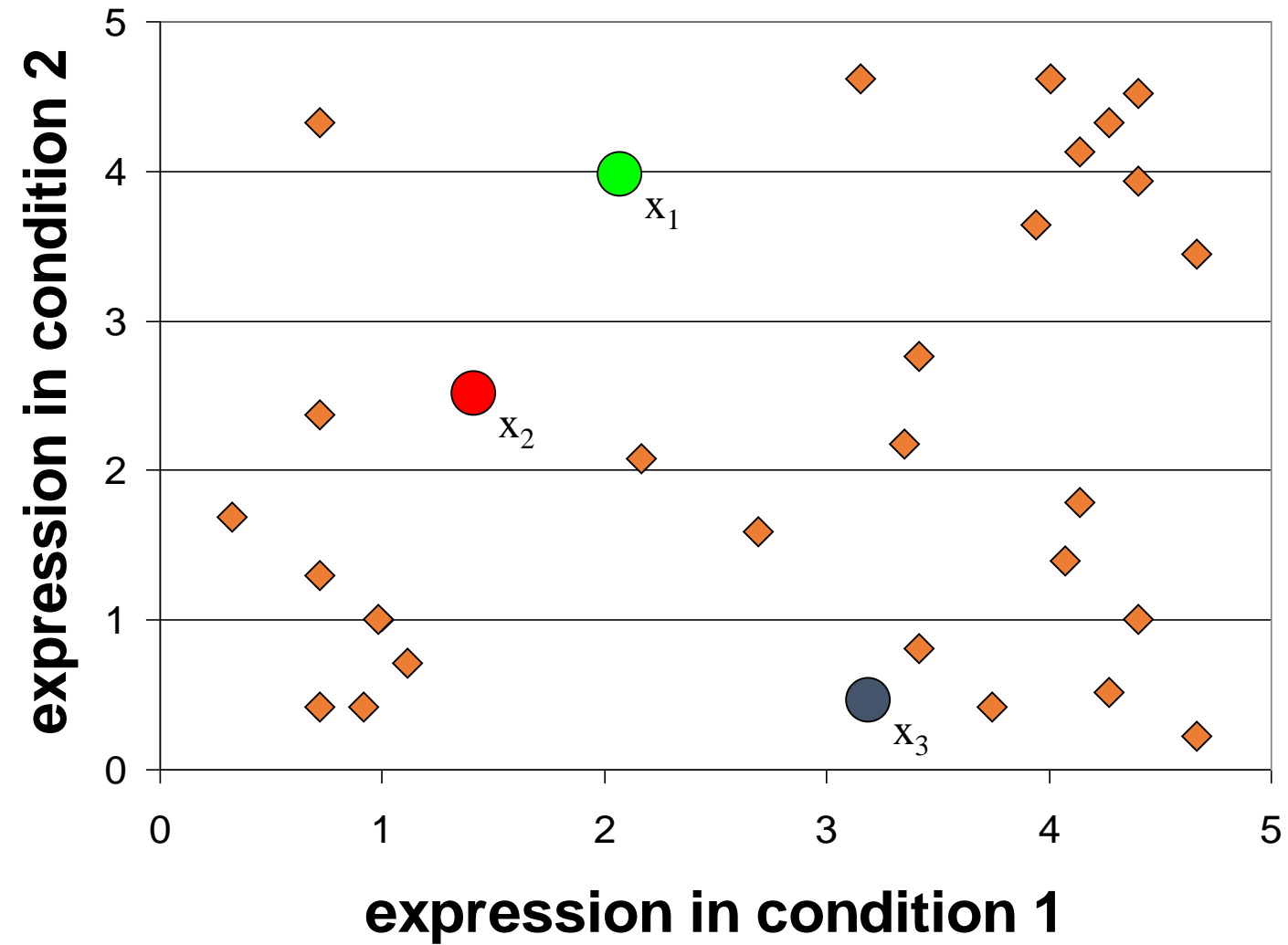


Bad

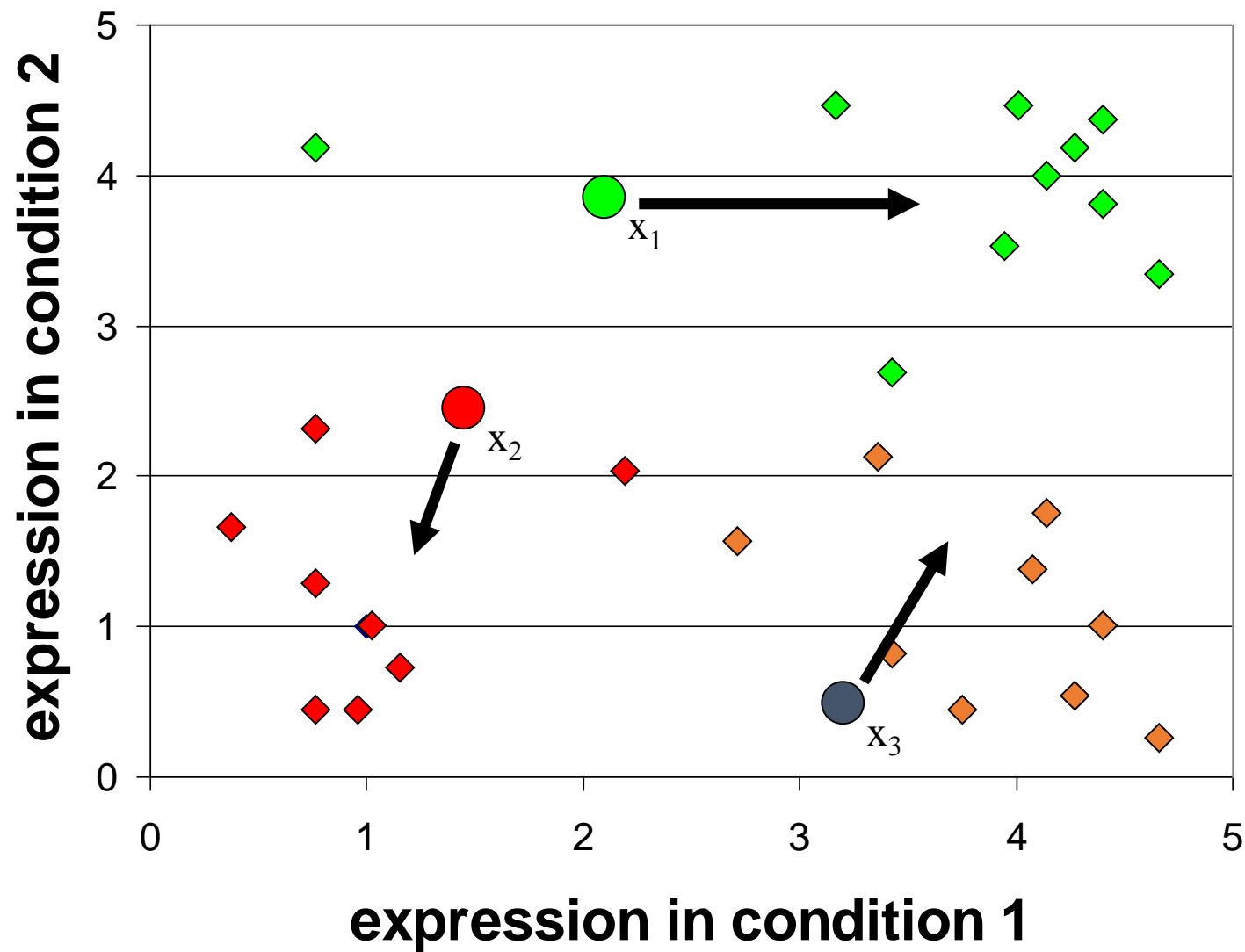


Good

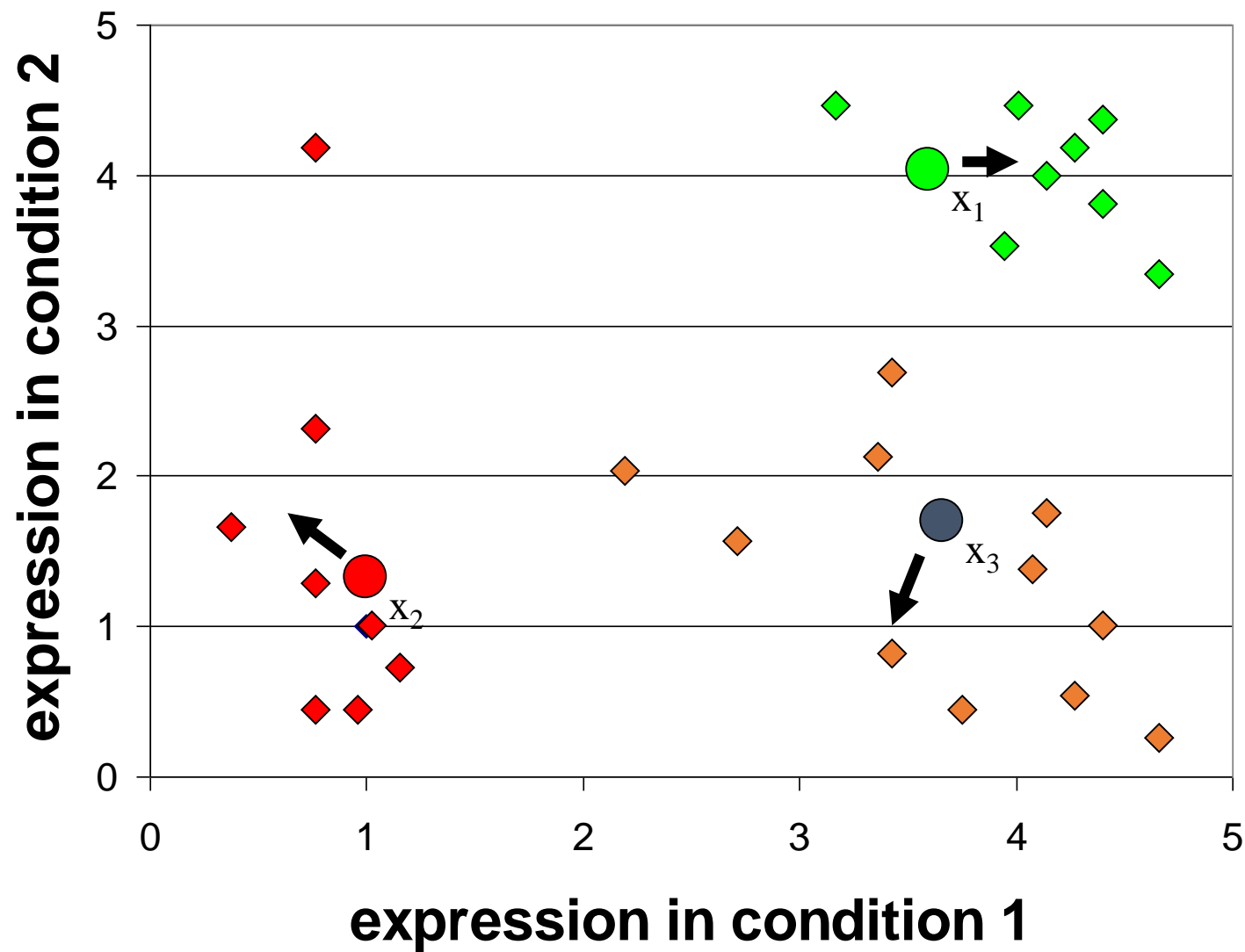
K-mean



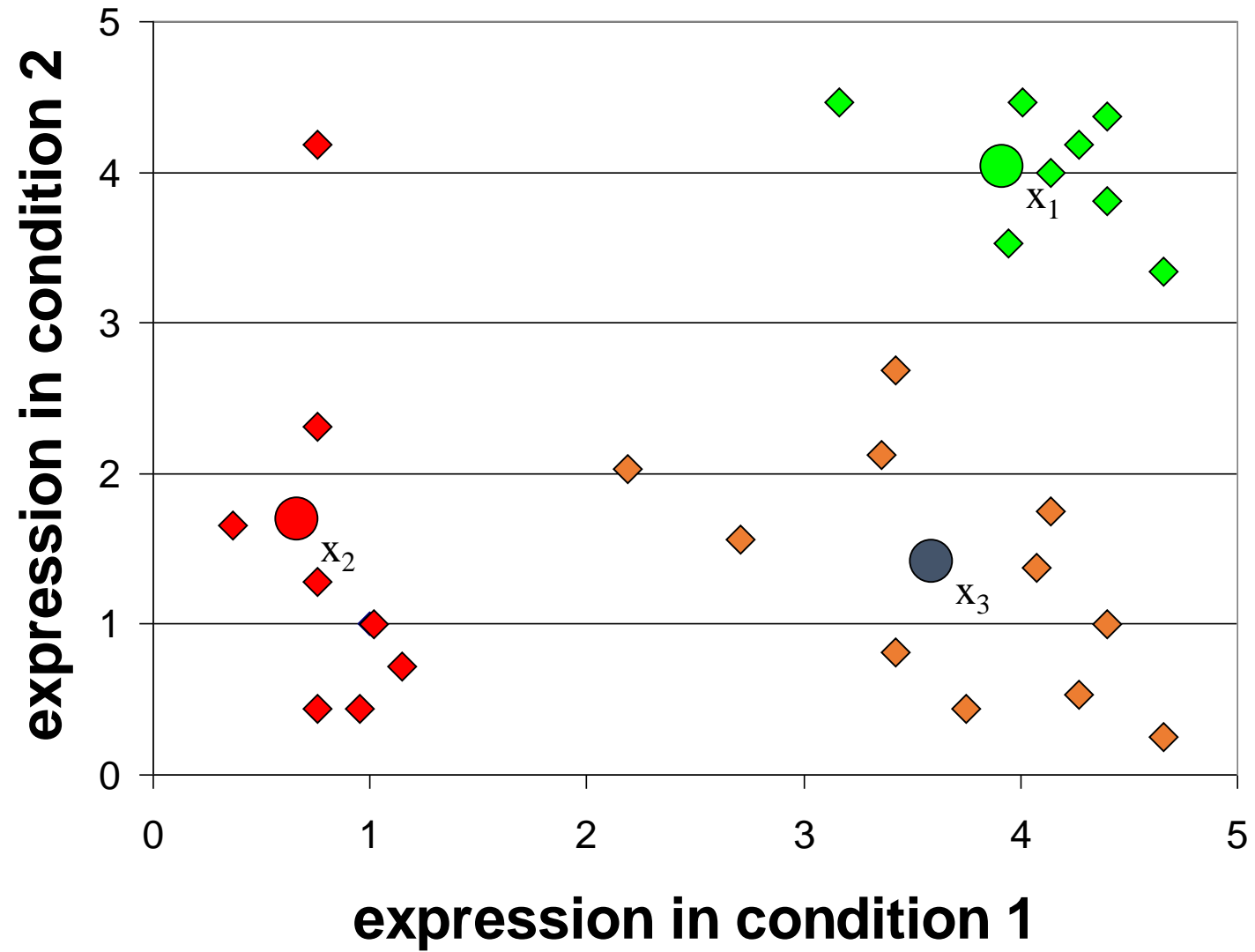
K-mean



K-mean



K-mean



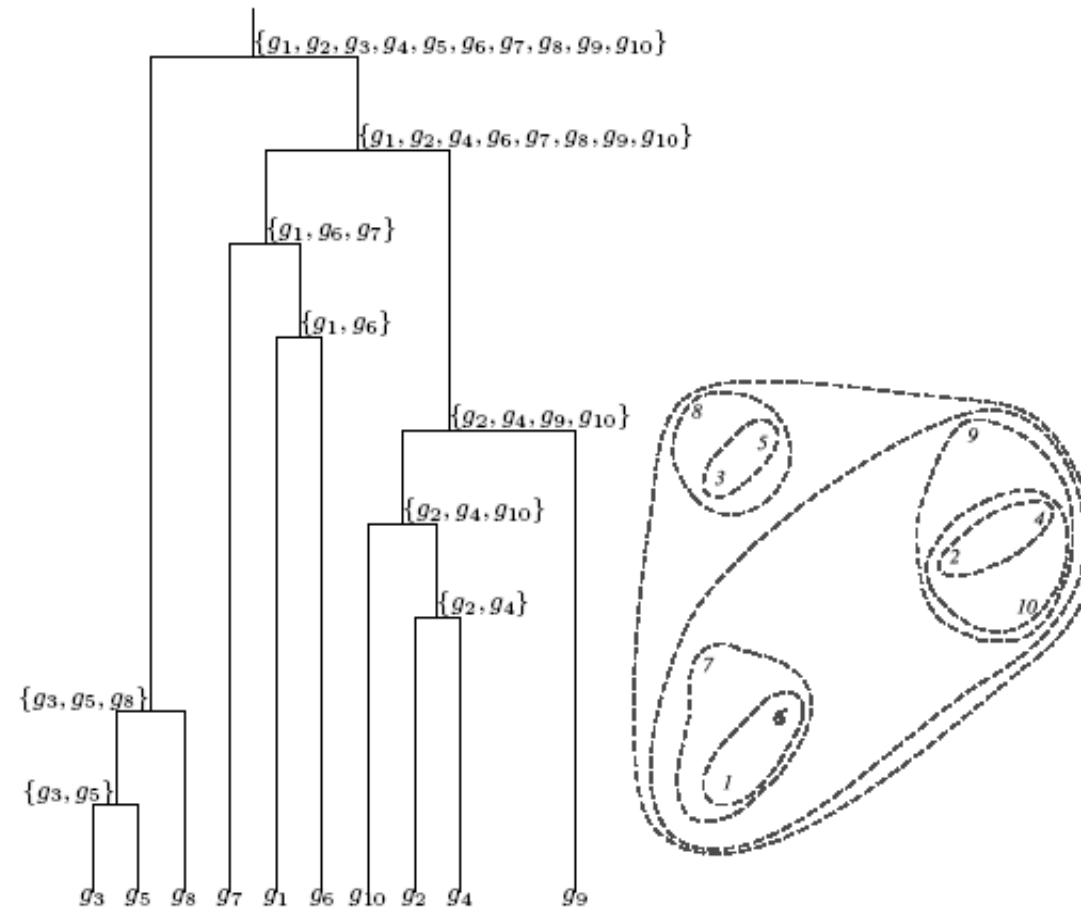
K-mean clustering

1. Lloyd Algorithm
2. Arbitrarily assign the k cluster centers
3. **while** the cluster centers keep changing
4. Assign each data point to the cluster C_i corresponding to the closest cluster representative (center) ($1 \leq i \leq k$)
5. After the assignment of all data points, compute new cluster representatives according to the center of gravity of each cluster, that is, the new cluster representative is
 $\frac{\sum v \in C}{|C|}$ for all v in C for every cluster C

K -mean clustering

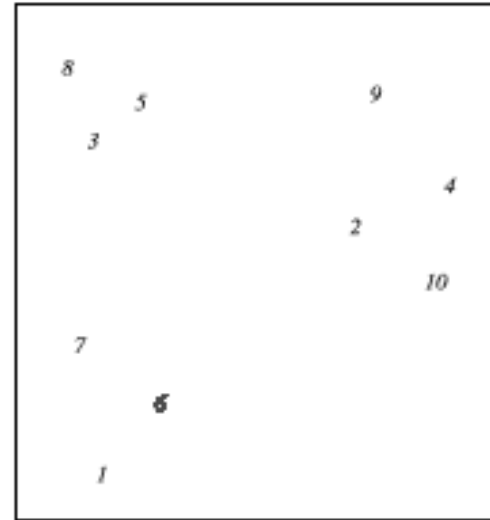
- You need to know k before clustering
- Local optimality: the clustering result is dependent of the initial choice of the locations of the k centroid; the process is random. Simple solution: run K -mean clustering multiple times and select the one with the best result
- K -means assumptions
 - the variance of the distribution of each attribute (variable) is spherical
 - all variables have the same variance
 - the prior probability for all k clusters is the same, i.e., each cluster has roughly equal number of observations

Hierarchical clustering

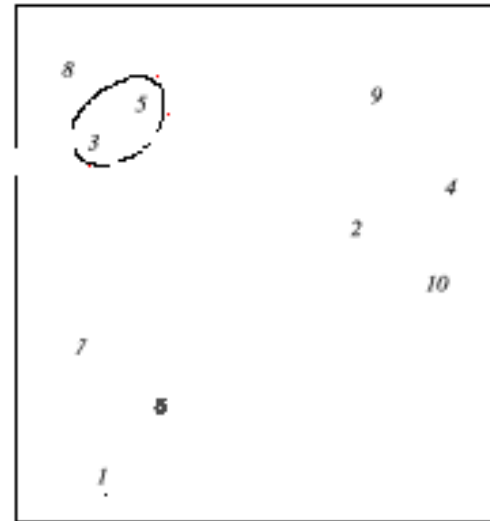
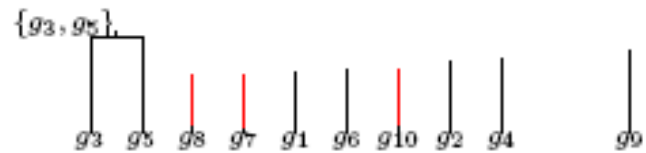


Hierarchical clustering

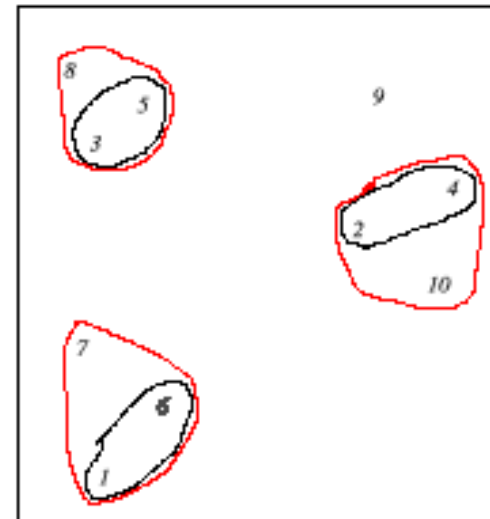
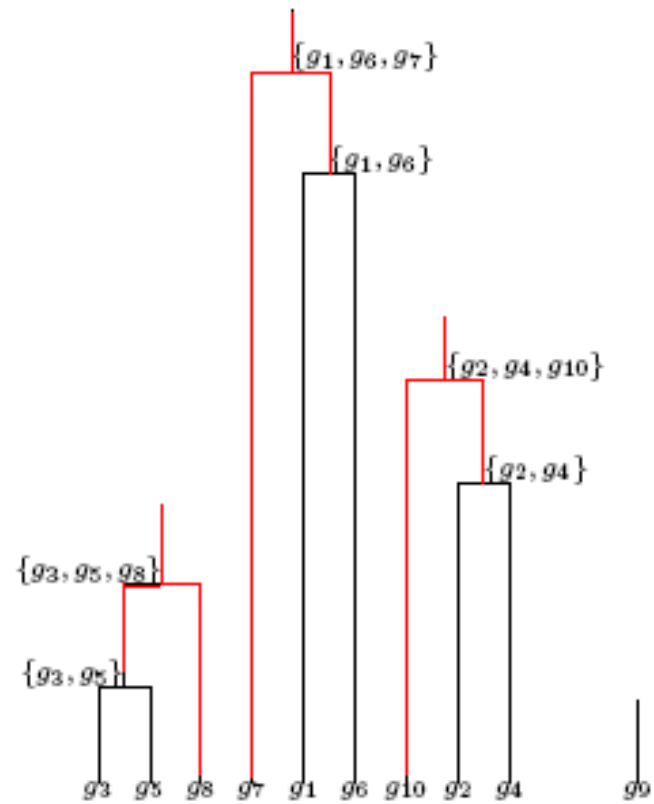
g3 g5 g8 g7 g1 g6 g10 g2 g4 g9



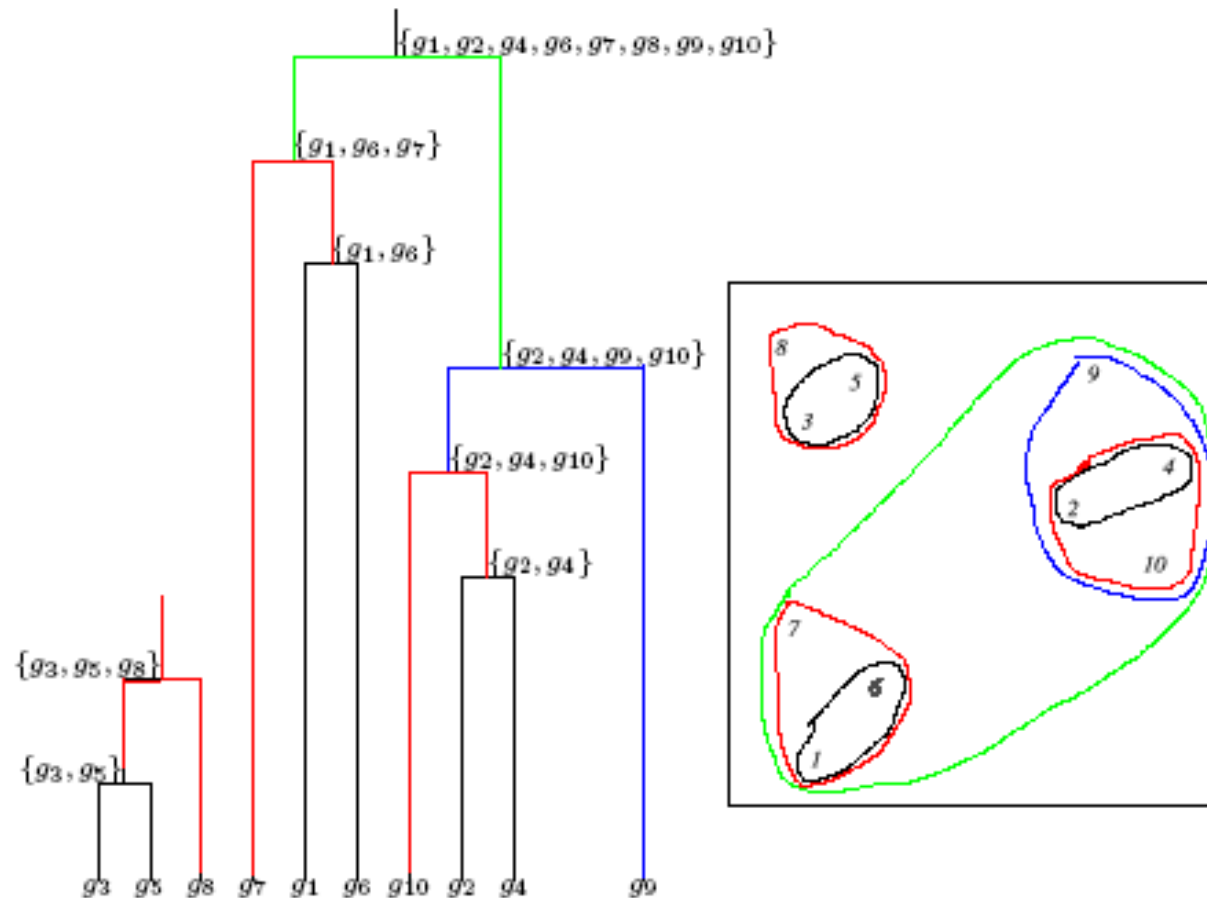
Hierarchical clustering



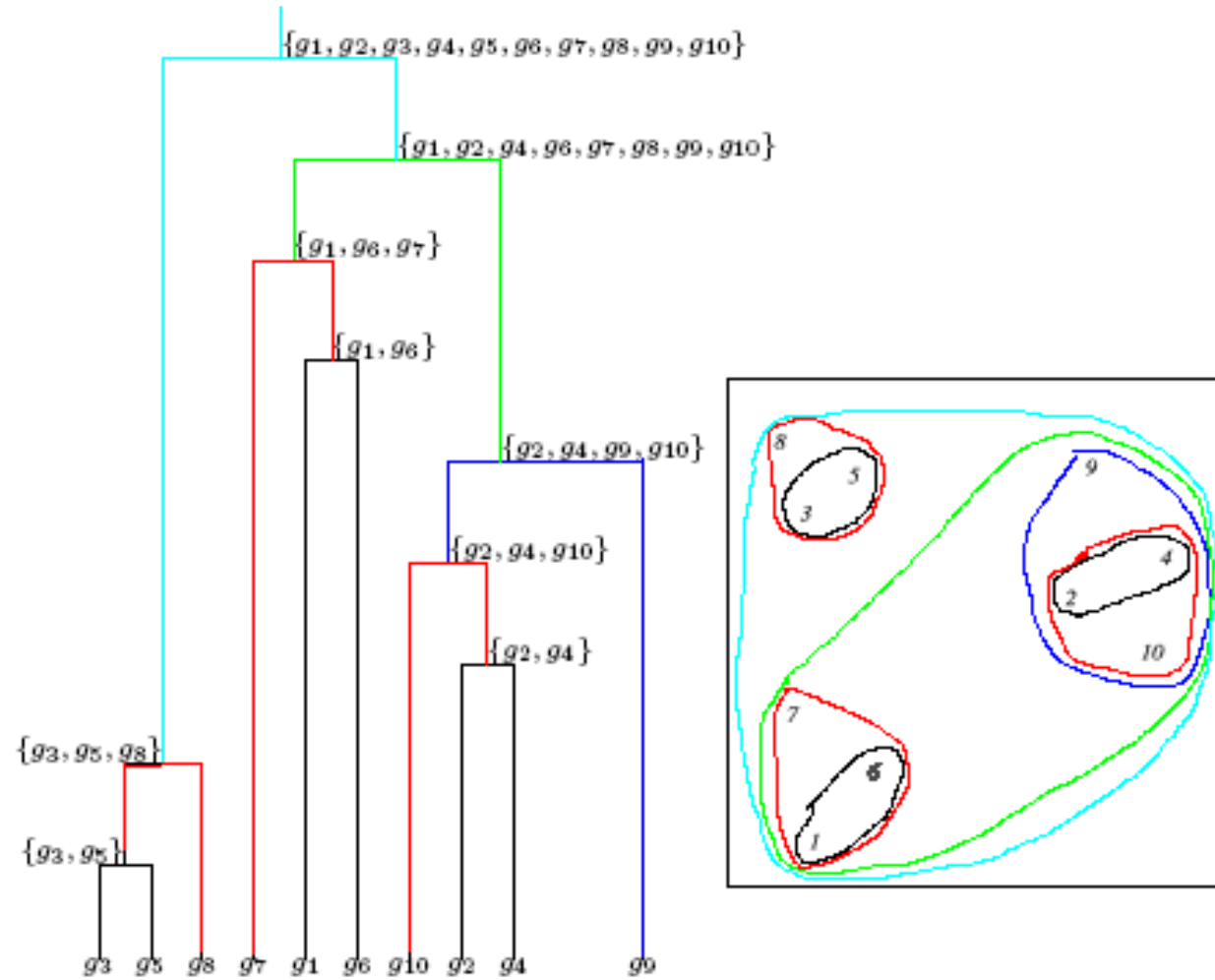
Hierarchical clustering



Hierarchical clustering

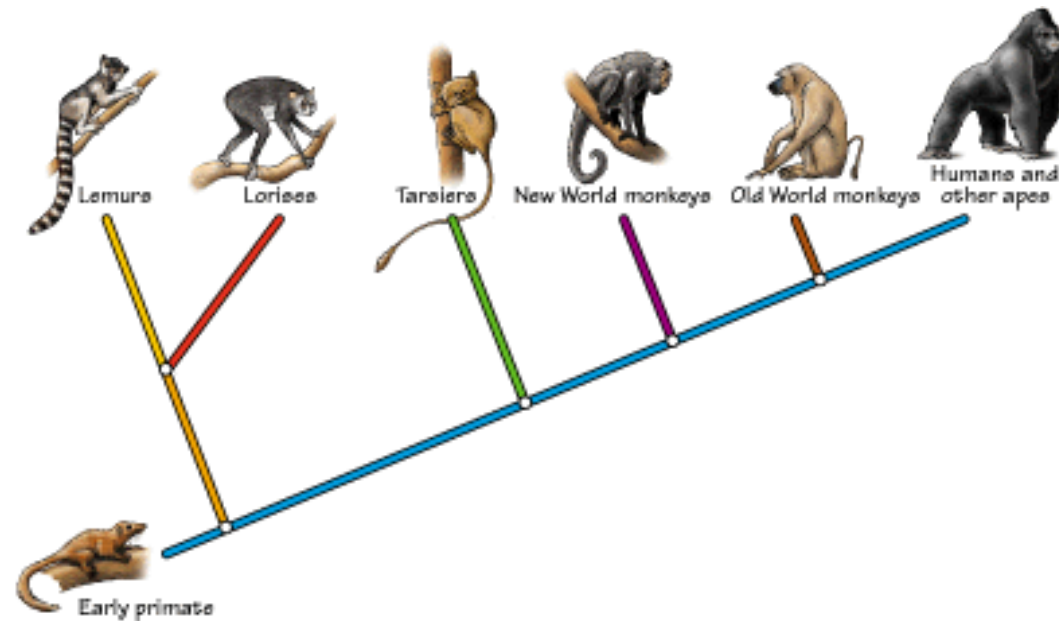


Hierarchical clustering



Hierarchical clustering

- Hierarchical Clustering is often used to reveal evolutionary history



Hierarchical clustering

1. Hierarchical Clustering (d, n)
2. Form n clusters each with one element
3. Construct a graph \mathcal{T} by assigning one vertex to each cluster
4. **while** there is more than one cluster
5. Find the two closest clusters C_1 and C_2
6. Merge C_1 and C_2 into new cluster C with $|C_1| + |C_2|$ elements
7. **Compute distance from C to all other clusters**
8. Add a new vertex C to \mathcal{T} and connect to vertices C_1 and C_2
9. Remove rows and columns of d corresponding to C_1 and C_2
10. Add a row and column to d corresponding to the new cluster C
11. return \mathcal{T}

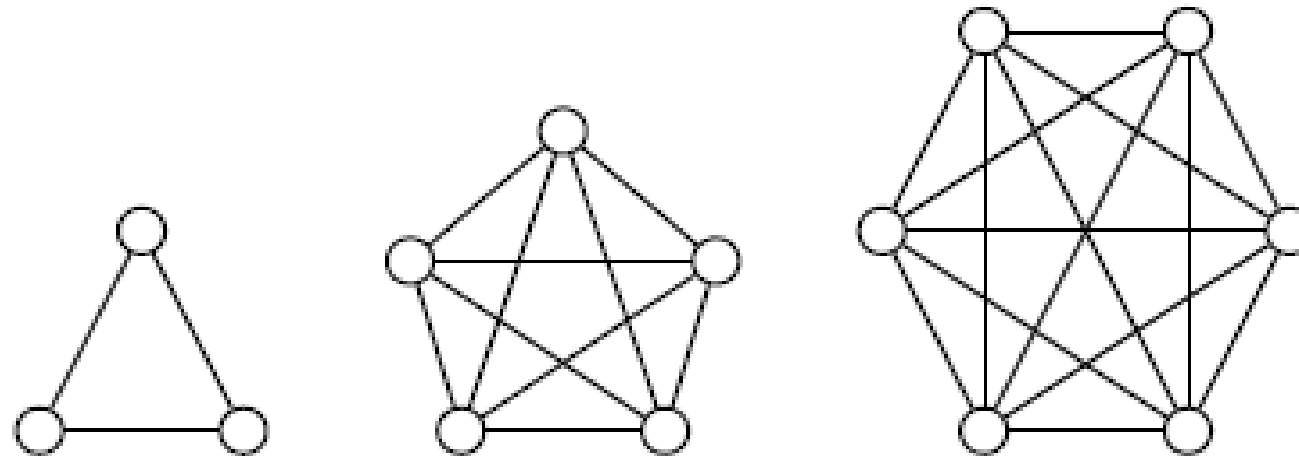
Different ways to define distances between clusters may lead to different clustering

Hierarchical clustering

- Hierarchical clustering is a greedy algorithm and it may be stuck in the local optima; but the clustering process is deterministic and there is no simple solution like running it multiple times as in *K*-mean. Careful interpretation is required.
- Errors propagate; the problem gets more severe if it happens at the early stage of the clustering
- The clustering result is not disjoint, a proper way is required to cut the hierarchical tree to produce disjoint clusters
- Need a careful choice of the distance measure

Clique clustering

- A **clique** is a graph with every vertex connected to every other vertex
- A **clique graph** is a graph where each connected component is a clique



Distance matrix to clique graph

- Turn the distance matrix into a distance graph
 - Genes are represented as vertices in the graph
 - Choose a distance threshold ϑ
 - If the distance between two vertices is below ϑ , draw an edge between them
 - The resulting graph may contain cliques
 - These cliques represent clusters of closely located data points!

Remove minimum number of edges

The distance graph (threshold $\vartheta=7$) is transformed into a clique graph after removing the two highlighted edges

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

(a) Distance matrix, d (distances shorter than 7 are shown in bold).

After transforming the distance graph into the clique graph, the dataset is partitioned into three clusters

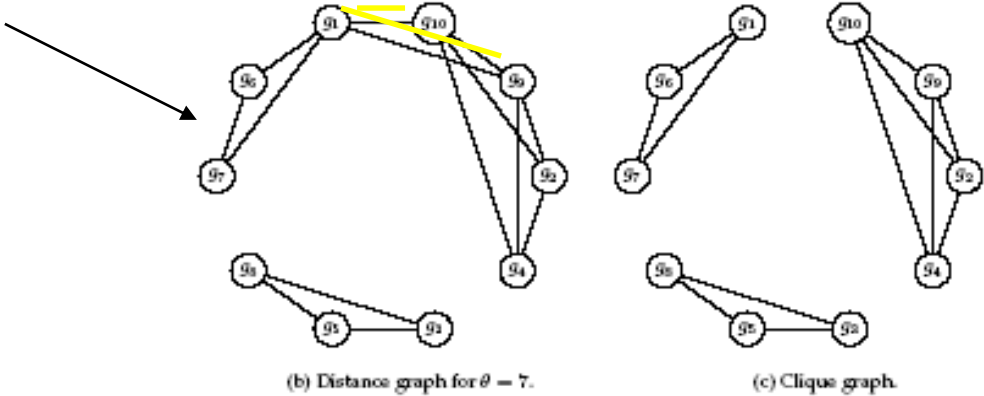


Figure 10.6 The distance graph (b) for $\theta = 7$ is not quite a clique graph. However, it can be transformed into a clique graph (c) by removing edges (g_1, g_{10}) and (g_1, g_9) .

CAST algorithm

- Corrupted Cliques problem is NP-Hard, some heuristics exist to approximately solve it:
- **CAST** (Cluster Affinity Search Technique): a practical and fast algorithm:
 - **CAST** is based on the notion of genes *close* to cluster C or *distant* from cluster C
 - Distance between gene i and cluster C :

$d(i,C)$ = average distance between gene i and all genes in C

Gene i is **close** to cluster C if $d(i,C) < \theta$ and **distant** otherwise

CAST algorithm

Define the distance between a vertex and a set as the average of the distances between the vertex and each element in the set.

1. CAST(S, G, θ)
2. $P \leftarrow \emptyset$
3. **while** $S \neq \emptyset$
4. $V \leftarrow$ vertex of maximal degree in the distance graph G
5. $C \leftarrow \{V\}$
6. **while** a **close** gene i **not in** C or **distant** gene i **in** C exists
7. Find the nearest close gene i not in C and add it to C
8. Remove the farthest distant gene i in C
9. Add cluster C to partition P
10. $S \leftarrow S \setminus C$
11. Remove vertices of cluster C from the distance graph G
12. **return** P

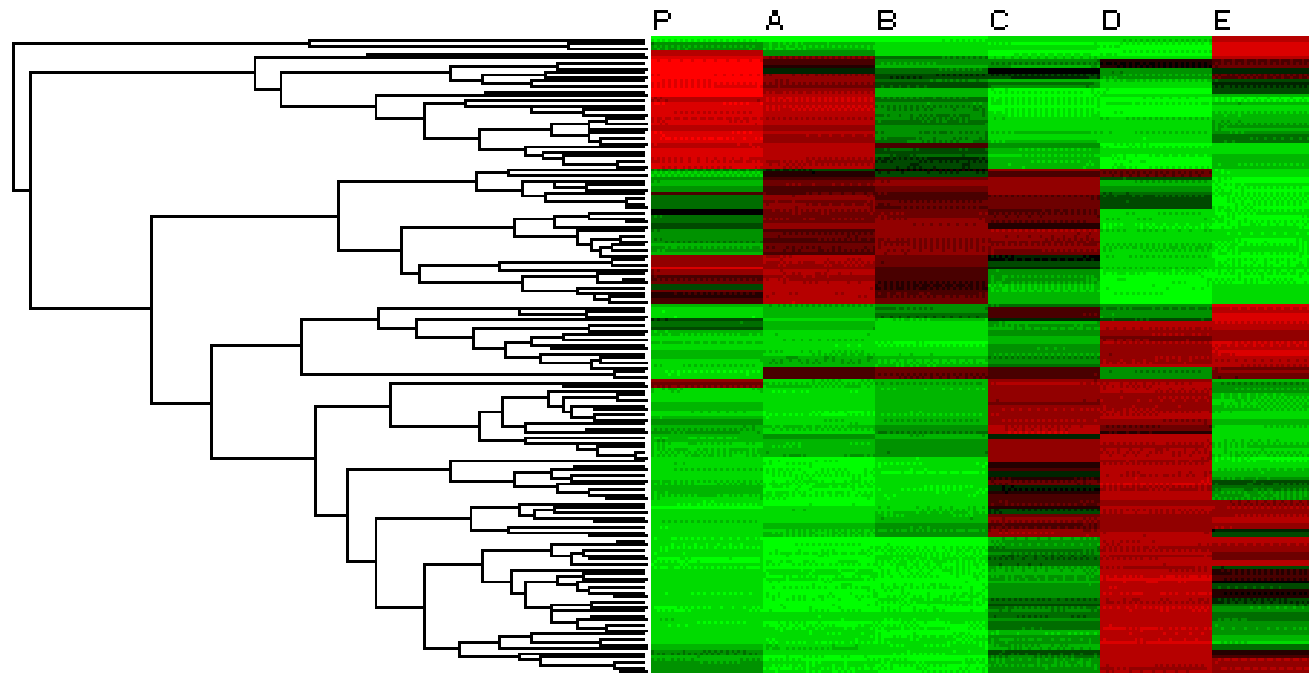
S – set of elements, G – distance graph, θ – distance threshold

Clique-based clustering

- Although it is a heuristic algorithm, the performance of clique-based clustering algorithms is quite stable
 - Imagine that adding or deleting a vertex will not change to graph structure significantly
- Allows an easy way to reduce memory consumption while allowing user-defined distance metric
 - We do not need to store all pairwise distances, we can only store distances that are smaller than a threshold
 - Hierarchical clustering, on the other hand, requires all pairwise distance
 - *K*-mean clustering is difficult to adapt complex distance function (and by default it uses Euclidean distance)

Bi-clustering

- Finding a set of genes that have similar expression profile in a set of conditions

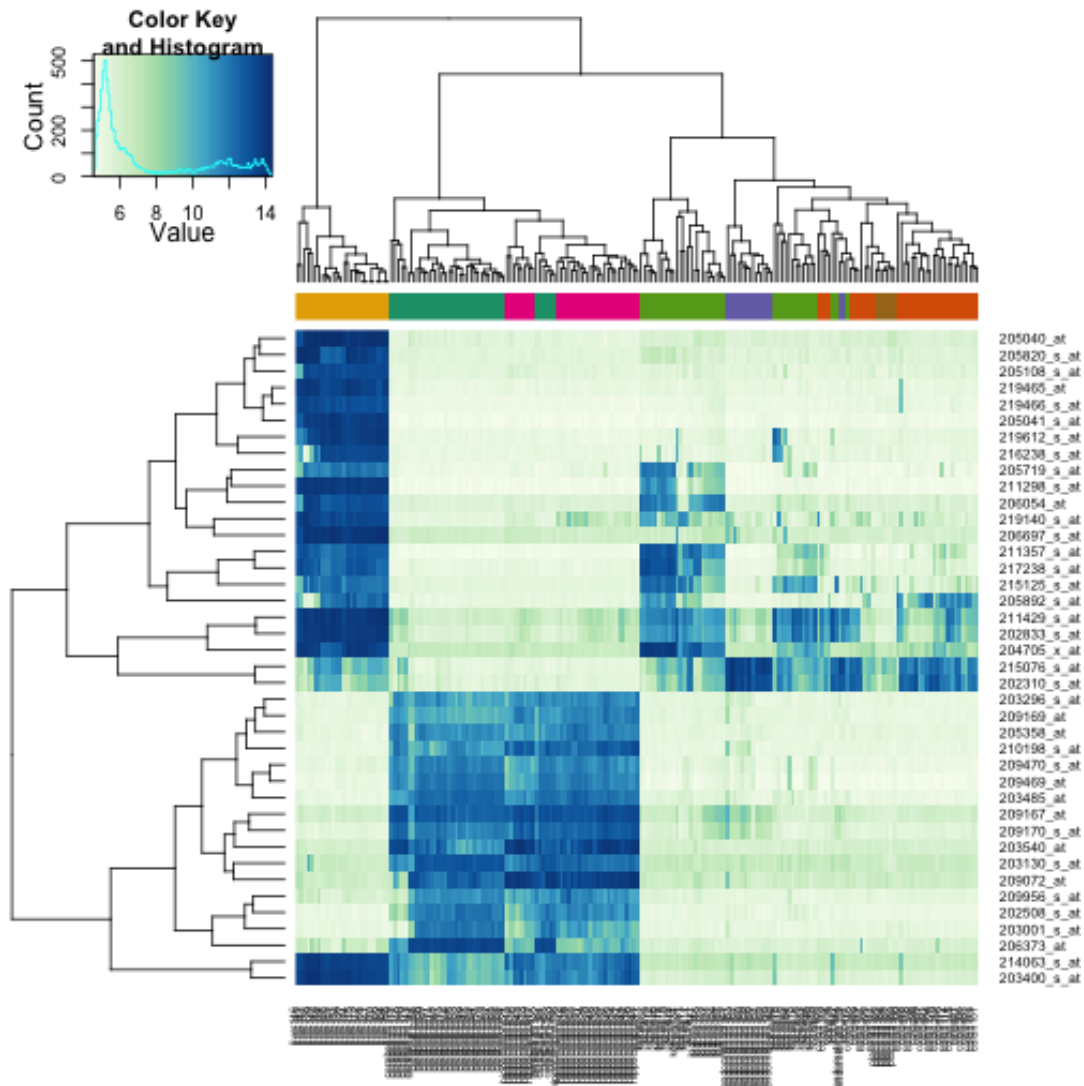


Each entry is the log fold change (relative abundance) of the corresponding gene in the corresponding condition

Why analyzing genes and conditions together

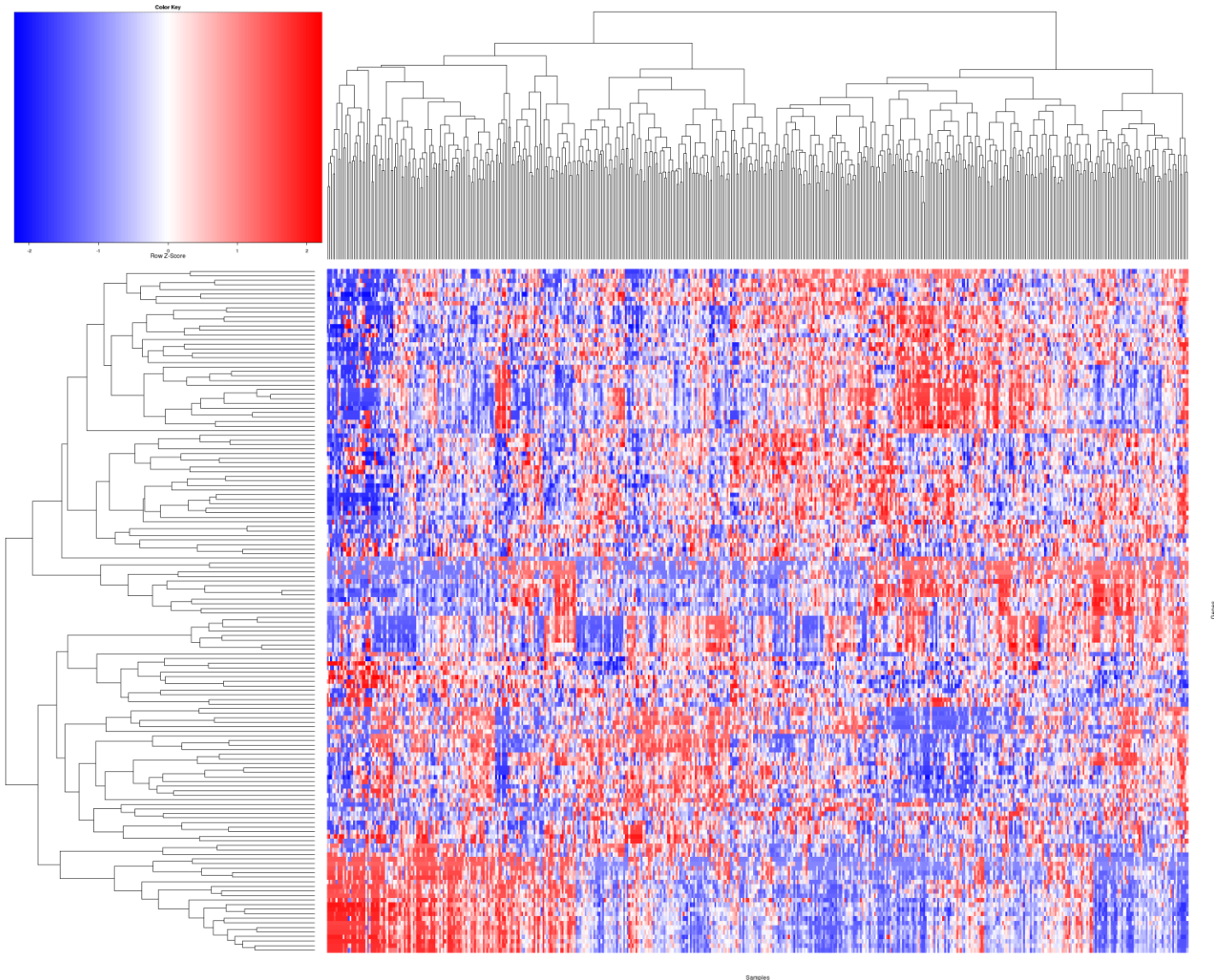
- Group genes according to their expression under different conditions
 - Which genes are collaborating together for a given biological process of interest
- Group conditions according to their gene expression patterns
 - Disease subtyping: i.e. given the expression data of a set of patient with the same symptom, predict whether the molecular mechanisms that causing the symptom are different and find the clustering
 - given a patient sample with gene expression profile, predict whether the individual has disease and/or suggest best treatment plan

Heatmap functions in R or MATLAB



- Two-dimensional hierarchical clustering
- Reorder the rows and columns based on the dendrogram
- Note that the orders of the rows and columns will not affect distance calculation between the columns and the rows in most distance function (such as Euclidean distance, Pearson correlation, Spearman correlation etc.)
- Clusters need to be identified manually
- Very useful but is potentially different from bi-clustering

Heatmap functions in R or MATLAB



- Limitation of one-dimensional clustering
 - A set of genes may only collaborate under some given conditions but not all
 - Not all genes have to share the same pattern to define biological conditions
 - We wish to recognize clustered pattern both row-wise and column-wise simultaneously

Bi-clustering

- It is a computationally hard problem
- Majority solutions are heuristic methods
- A review for bi-clustering algorithms by Eran et al. “A comparative analysis of biclustering algorithms for gene expression data”, Briefings in Bioinformatics 2012.
- We will focus on Cheng and Church algorithm.

Cheng-Church Algorithm

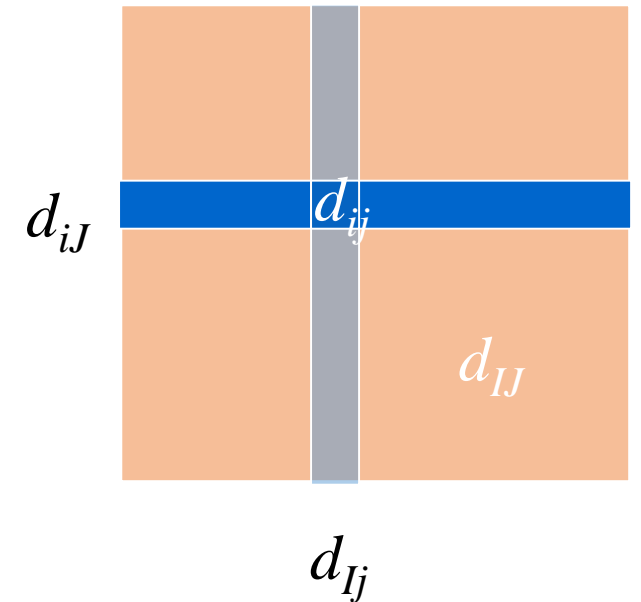
- Key idea: Define uniformity for a bi-cluster (a block of data), and partition the entire data set into a set of non-overlapping bi-clusters to maximize the overall uniformity for all bi-clusters.
- Start with the entire data set
- Delete columns and rows to increase uniformity
- Add back columns and rows that are consistent with the current block
- Output the data block as a bi-cluster, delete corresponding rows and columns from the entire data set, and proceed iteratively

Cheng-Church Algorithm

- Object function for uniformity (given a bi-cluster):

$$H(IJ) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (d_{ij} - d_{iJ} - d_{Ij} + d_{IJ})^2$$

$$d_{iJ} = \frac{1}{|J|} \sum_{j \in J} d_{ij} \quad d_{Ij} = \frac{1}{|I|} \sum_{i \in I} d_{ij} \quad d_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} d_{ij}$$



**Note that if the bi-cluster is completely homogeneous than $H(IJ)=0$.
If $H(IJ)$ is less than a threshold than IJ is considered as a valid bi-cluster.**

Cheng-Church Algorithm

Algorithm 1 (Single Node Deletion).

Input: A , a matrix of real numbers, and $\delta \geq 0$, the maximum acceptable mean squared residue score.

Output: A_{IJ} , a δ -bicluster that is a submatrix of A with row set I and column set J , with a score no larger than δ .

Initialization: I and J are initialized to the gene and condition sets in the data and $A_{IJ} = A$.

Iteration:

1. Compute a_{iJ} for all $i \in I$, a_{Ij} for all $j \in J$, a_{IJ} , and $H(I, J)$. If $H(I, J) \leq \delta$, return A_{IJ} .
2. Find the row $i \in I$ with the largest

$$d(i) = \frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2$$

and the column $j \in J$ with the largest

$$d(j) = \frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2$$

remove the row or column whichever with the larger d value by updating either I or J .

Algorithm 2 (Multiple Node Deletion).

Input: A , a matrix of real numbers, $\delta \geq 0$, the maximum acceptable mean squared residue score, and $\alpha > 1$, a threshold for multiple node deletion.

Output: A_{IJ} , a δ -bicluster that is a submatrix of A with row set I and column set J , with a score no larger than δ .

Initialization: I and J are initialized to the gene and condition sets in the data and $A_{IJ} = A$.

Iteration:

1. Compute a_{iJ} for all $i \in I$, a_{Ij} for all $j \in J$, a_{IJ} , and $H(I, J)$. If $H(I, J) \leq \delta$, return A_{IJ} .
2. Remove the rows $i \in I$ with
$$\frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 > \alpha H(I, J)$$
3. Recompute a_{Ij} , a_{IJ} , and $H(I, J)$.
4. Remove the columns $j \in J$ with
$$\frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 > \alpha H(I, J)$$
5. If nothing has been removed in the iterate, switch to Algorithm 1.

Cheng-Church Algorithm

Algorithm 3 (Node Addition).

Input: A , a matrix of real numbers, I and J signifying a δ -bicluster.

Output: I' and J' such that $I \subset I'$ and $J \subset J'$ with the property that $H(I', J') \leq H(I, J)$.

Iteration:

1. Compute a_{iJ} for all i , a_{Ij} for all j , a_{IJ} , and $H(I, J)$.

2. Add the columns $j \notin J$ with

$$\frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I, J)$$

3. Recompute a_{iJ} , a_{Ij} , and $H(I, J)$.

4. Add the rows $i \notin I$ with

$$\frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I, J)$$

5. For each row i still not in I , add its inverse if

$$\frac{1}{|J|} \sum_{j \in J} (-a_{ij} + a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I, J)$$

6. If nothing is added in the iterate, return the final I and J as I' and J' .

Algorithm 4 (Finding a Given Number of Biclusters).

Input: A , a matrix of real numbers with possible missing elements, $\alpha \geq 1$, a parameter for multiple node deletion, $\delta \geq 0$, the maximum acceptable mean squared residue score, and n , the number of δ -biclusters to be found.

Output: n δ -biclusters in A .

Initialization: Missing elements in A are replaced with random numbers from a range covering the range of non-null values. A' is a copy of A .

Iteration for n times:

1. Apply Algorithm 2 on A' , δ , and α . If the row (column) size is small (less than 100), do not perform multiple node deletion on rows (columns). The matrix after multiple node deletion is B .

2. (Step 5 of Algorithm 2) Apply Algorithm 1 on B and δ and the matrix after single node deletion is C .

3. Apply Algorithm 3 on A and C and the result is the bicluster D .

4. Report D , and replace the elements in A' that are also in D with random numbers.

Gene set enrichment analysis

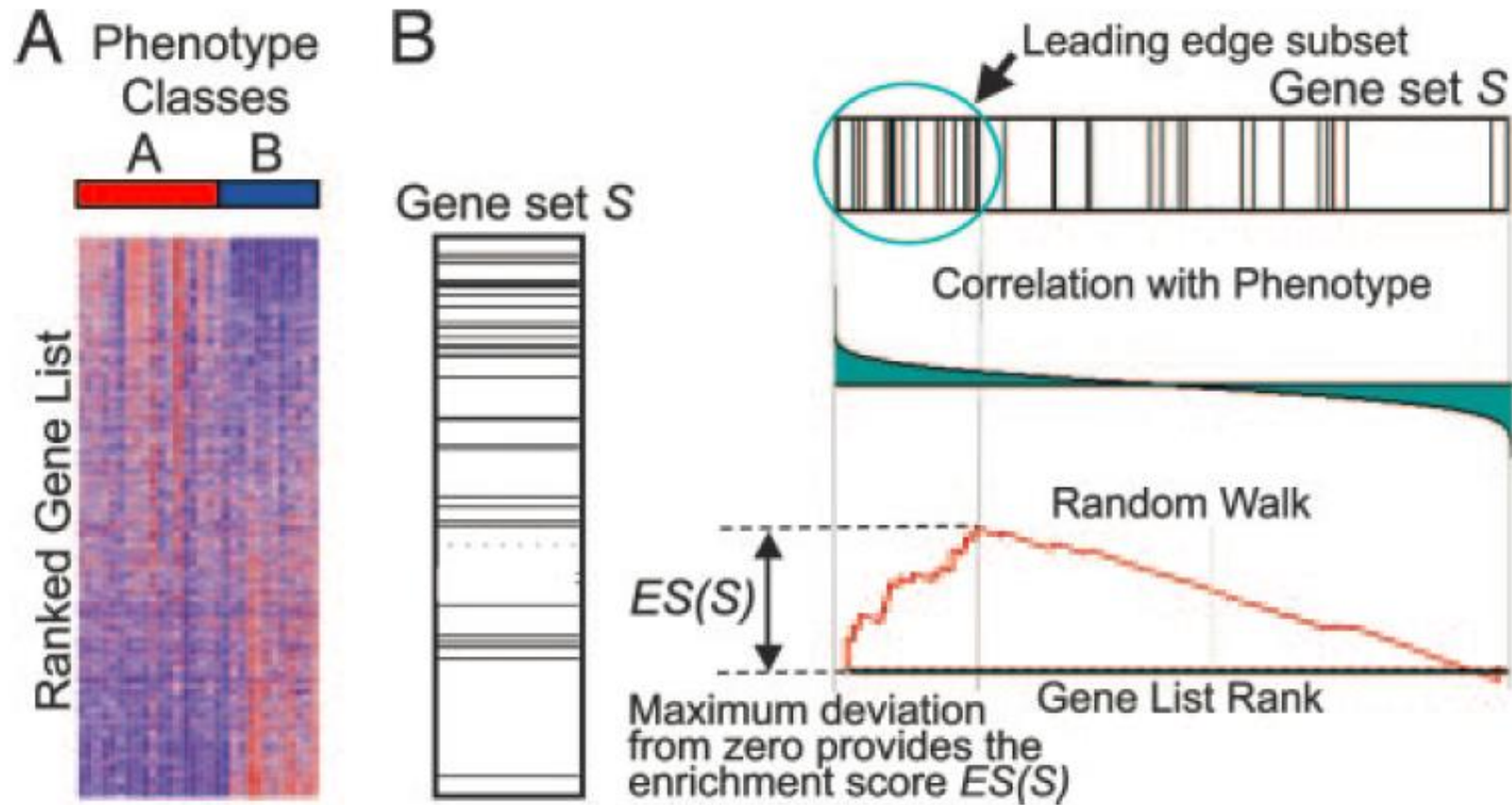
- Is a set of disrupted genes between condition correspond to certain biological pathway
- The method uses statistical approaches to identify significantly enriched or depleted groups of genes
- Depend on a prior knowledge on gene ontology
- Microarray, sequencing, and proteomics results often identify thousands of genes which are used for the analysis

Gene set enrichment analysis

- Given a bag of mixed m red and n blue balls, some one selected m' red balls and n' blue balls from the bag. Is the selection process random or not?
- Red balls: genes related with a given biological process; Blue balls: genes that are not related with a given biological process

- Simple solution: hypergeometric distribution
$$P(X = k) = \frac{\binom{K}{k} \binom{N-K}{n-k}}{\binom{N}{n}}$$

GSEA



GSEA

- Assume two groups of conditions, each condition may contain multiple samples. Also assume a given set of genes of interest.
- Sort genes based on their expression levels and the class distinction.
- Assess whether the members of the gene set are randomly distributed or primarily found at the top or bottom.

GSEA

- Use Kolmogorov–Smirnov-like statistic (a nonparametric test of the equality of continuous, one-dimensional probability distributions) to calculate Enrichment Score (ES) regarding the distribution of members of the gene set in the ranked list.
- Use random walk to speed up the computation
- Compute the statistical significance of ES using simulated data as background
- Apply multiple testing correction


A general schema

- Compute p-value or fold change for differential expression for all genes
- Rank the genes and take the top/bottom ones to form a group, or use suitable clustering or bi-clustering algorithms to identify a set of genes with similar expression pattern
- Perform gene set enrichment test
- Associate the enriched functional categories with the phenotype


Some software

- GSEA
- DAVID
- AmiGO2
- MSigDB
- PlantRegMap
- FunRich
- ConsensusPathDB
- ...

ConsensusPathDB



WVW PLANK-GESellschaft
Max Planck Institute
for Molecular Genetics

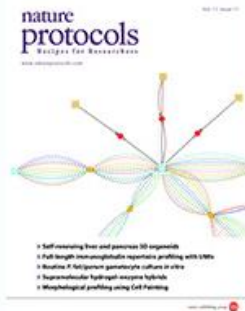


- home
- content information
- search
- interactions of molecules/pathways
- shortest interaction paths
- gene set analysis
- metabolite set analysis
- download / data access
- documentation
- news
- contact



Release 31 (01. Sept. 2015)

ConsensusPathDB-human integrates interaction networks in *Homo sapiens* including **binary and complex protein-protein, genetic, metabolic, signaling, gene regulatory and drug-target** interactions, as well as biochemical pathways. Data originate from currently **32** public resources for interactions (listed below) and interactions that we have curated from the literature. The interaction data are integrated in a complementary manner (avoiding redundancies), resulting in a seamless interaction network containing different types of interactions.



Literature:

A protocol on how to use ConsensusPathDB was recently published in Nature Protocols (also featured on the cover of the corresponding issue).

Kamburov, A. *et al.* (2013) The ConsensusPathDB interaction database: 2013 update. *Nucleic Acids Res.*

Kamburov, A. *et al.* (2011) ConsensusPathDB: toward a more complete picture of cell biology. *Nucleic Acids Res.*

Kamburov, A. *et al.* (2009) ConsensusPathDB--a database for integrating human interaction networks. *Nucleic Acids Res.*

Pentchev, K. *et al.* (2010) Evidence mining and novelty assessment of protein-protein interactions with the ConsensusPathDB plugin for Cytoscape. *Bioinformatics*

A poster about ConsensusPathDB is available here.

Current statistics:

unique physical entities:	158,523
unique interactions:	458,570
gene regulations:	17,098
protein interactions:	261,085
genetic interactions:	443
biochemical reactions:	21,070
drug-target interactions:	158,874
pathways:	4,593

• <http://www.consensuspathdb.org/>