Lecture 19: Clustering

Study Chapter 10.1 – 10.3

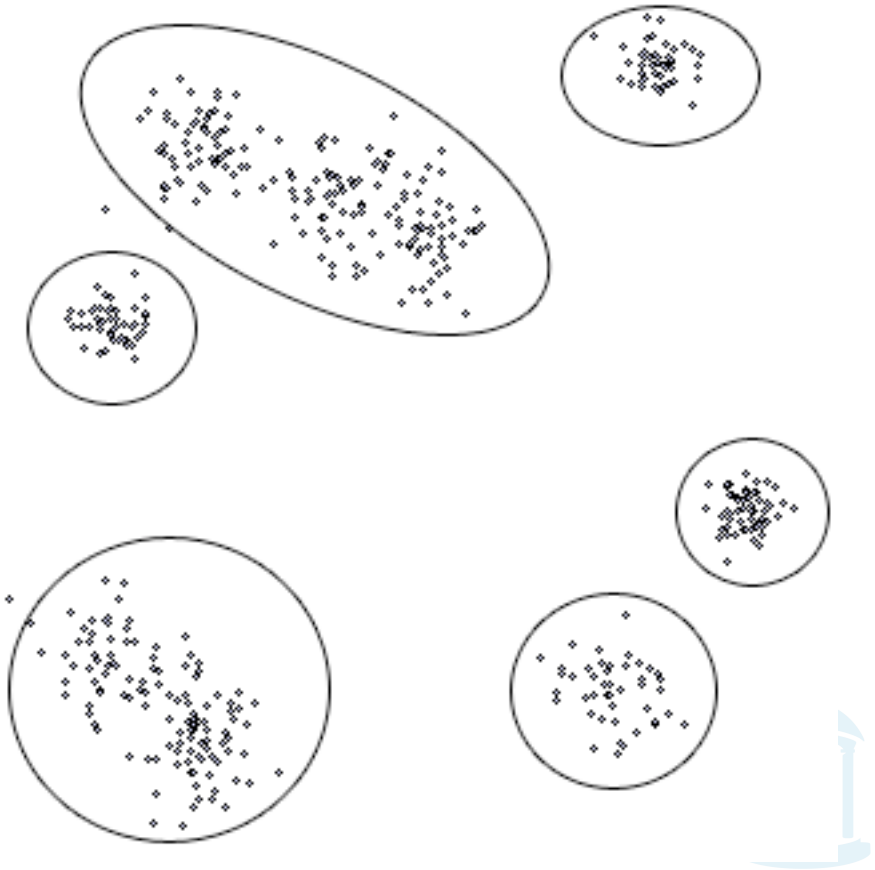
Problem Set 3 is due today

Applications of Clustering

- Viewing and analyzing vast amounts of biological data as a whole set can be perplexing

- It is often easier to interpret data if they are partitioned into *similar* subgroups.

- Such similar groups are "clusters"



Inferring Gene Functionality

- Researchers often want to know the functions of newly sequenced genes
- Comparing the new gene sequences to known DNA sequences often does not give away the function of gene
- For 40% of sequenced genes, functionality cannot be ascertained using only comparisons to sequences of other known genes
- Microarrays allow biologists to infer gene function even when sequence similarity alone is insufficient to infer function.

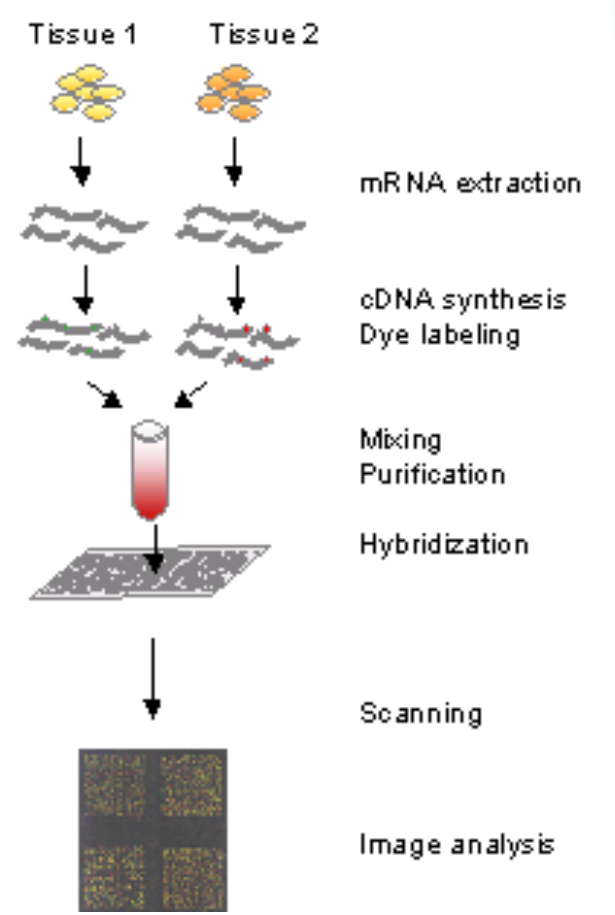
Microarrays and Expression Analysis

- Microarrays compare the activity (expression level) of the genes
  - Under varying conditions (e.g.. with and w/o disease)
  - At different time points
  - In different tissues
- Expression level is estimated by measuring the amount of mRNA for that particular gene
  - A gene is active if it is being transcribed
  - More mRNA usually indicates more gene activity

Microarray Experiments

- Produce cDNA from mRNA (DNA is more stable)
- Attach phosphor to cDNA to see when a particular gene is expressed
- Different color phosphors are available to compare many samples at once
- Hybridize cDNA over the microarray
- Scan the microarray with a phosphor-illuminating laser
- Illumination reveals transcribed genes
- Scan microarray multiple times for the different color phosphor's

Microarray


Tissue 1
Tissue 2
mRNA extraction
cDNA synthesis
Dye labeling
Mixing
Purification
Hybridization
Scanning
Image analysis

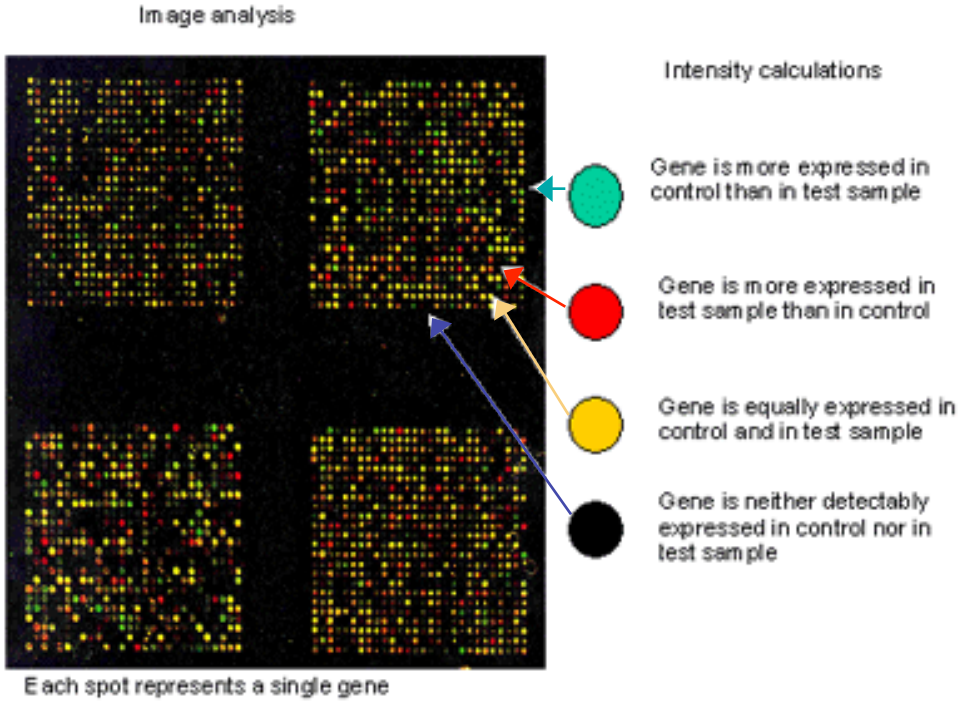
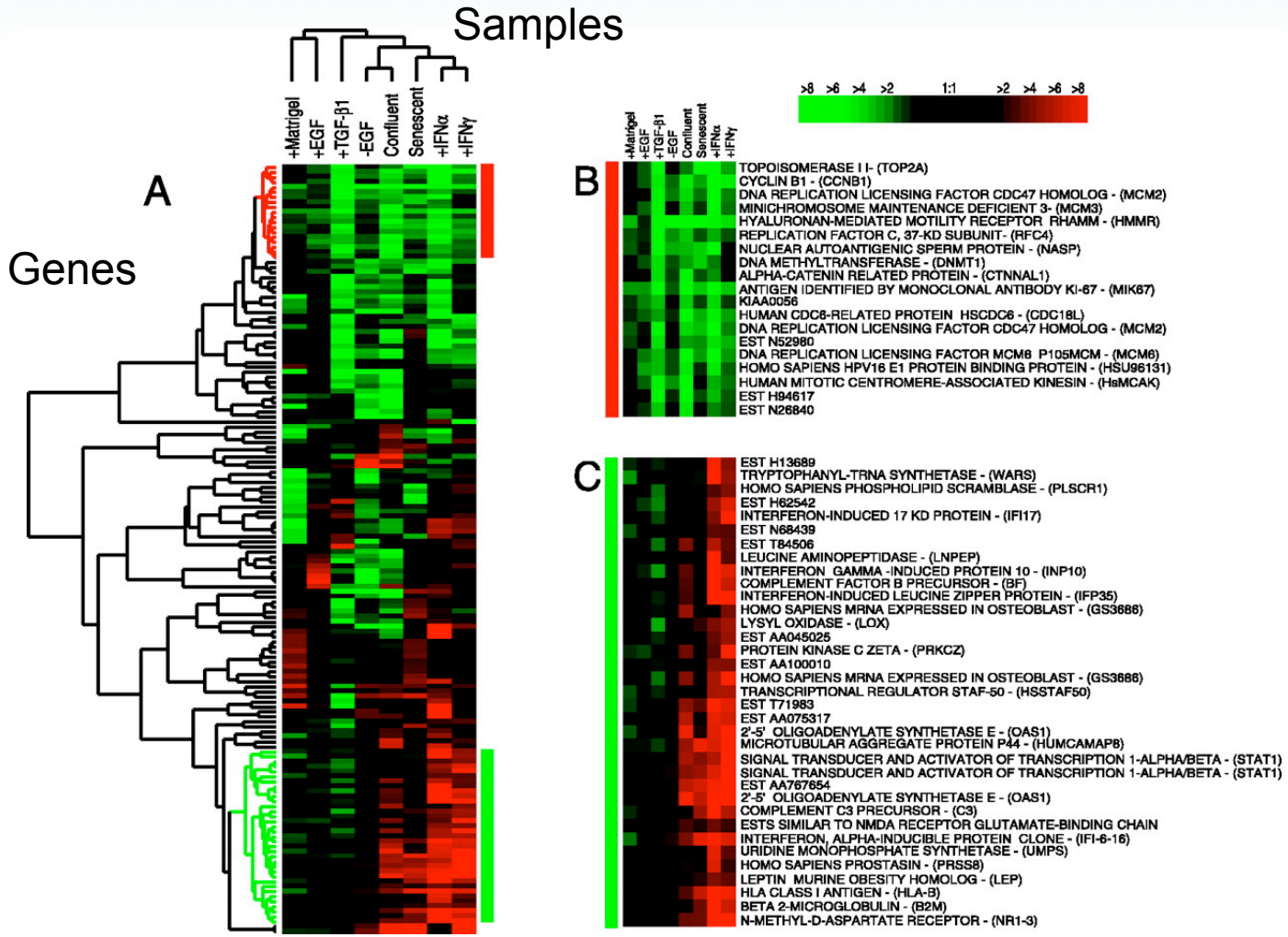


Image analysis
Intensity calculations
Gene is more expressed in control than in test sample
Gene is more expressed in test sample than in control
Gene is equally expressed in control and in test sample
Gene is neither detectably expressed in control nor in test sample
Each spot represents a single gene

Clustering Dimension



Clustering Microarray Data

- Easier to interpret if partitioned into "gene" or "sample" clusters
- Conceptually we could treat each gene in N arrays as a point in N-dimensional space
- Make a distance matrix for the distance between every two gene points in the N-dimensional space
- Genes with a small distance share the same expression characteristics and might be functionally related or similar.
- Clustering reveal groups of functionally related genes

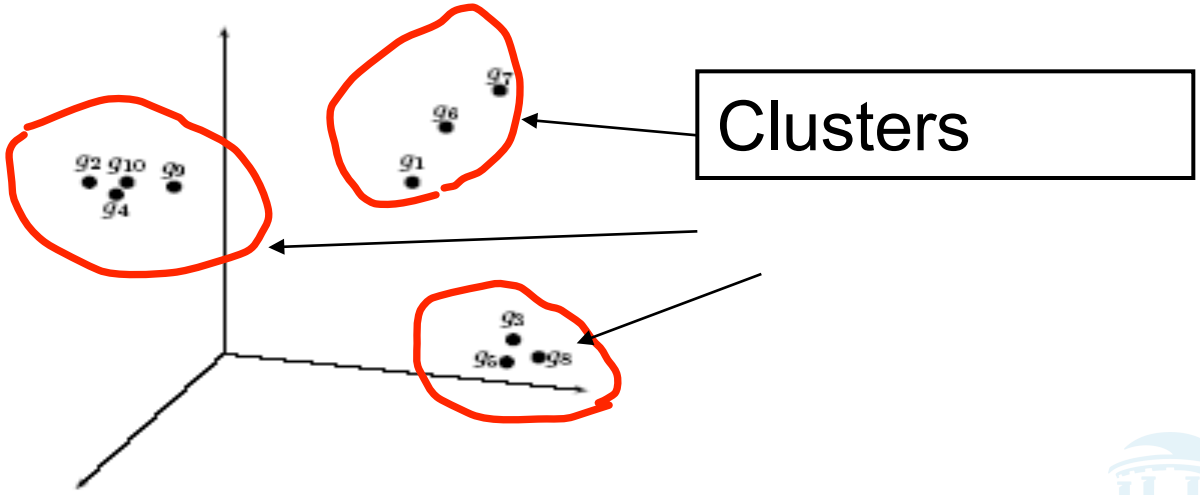
Clustering of Microarray Data (cont'd)

Time	1 hr	2 hr	3 hr
g_1	10.0	8.0	10.0
g_2	10.0	0.0	9.0
g_3	4.0	8.5	3.0
g_4	9.5	0.5	8.5
g_5	4.5	8.5	2.5
g_6	10.5	9.0	12.0
g_7	5.0	8.5	11.0
g_8	2.7	8.7	2.0
g_9	9.7	2.0	9.0
g_{10}	10.2	1.0	9.2

(a) Intensity matrix, I

	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

(b) Distance matrix, d

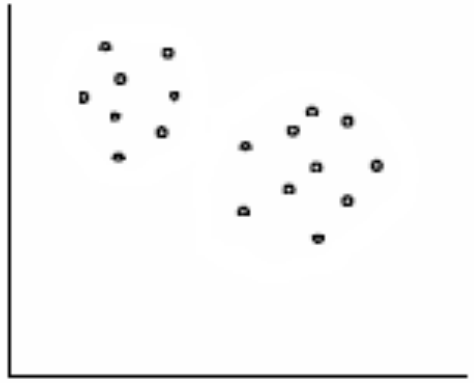


(c) Expression patterns as points in three-dimentsional space.

Homogeneity and Separation Principles

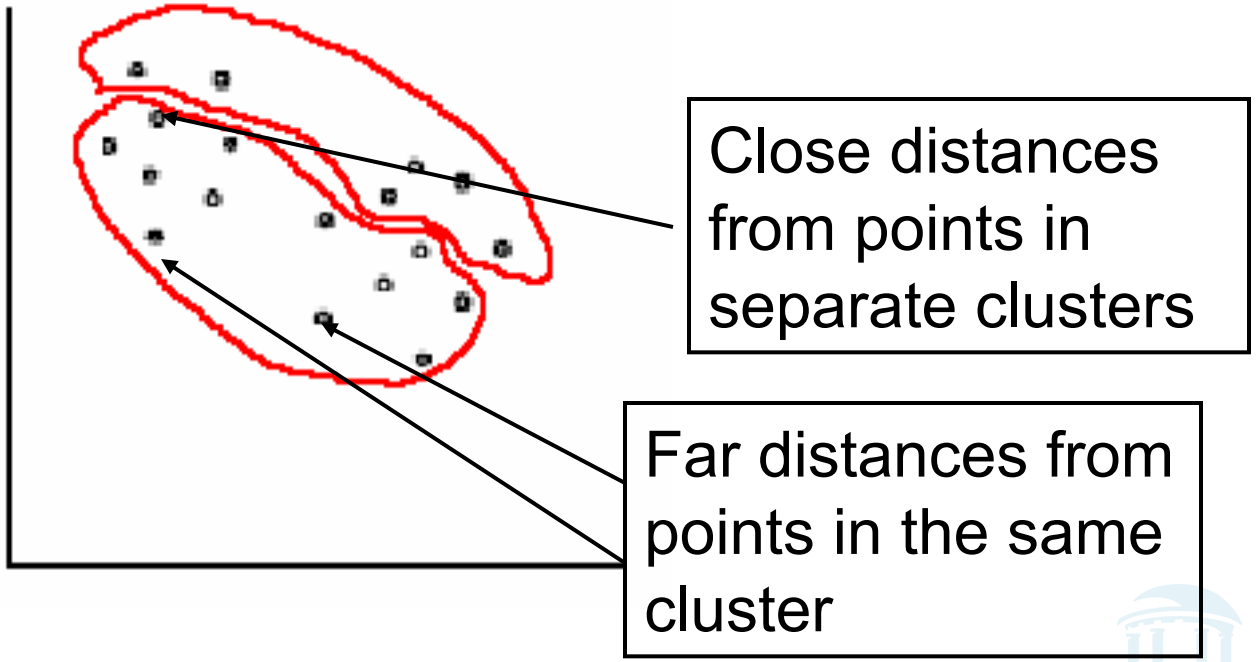
- **Homogeneity:** Elements within a cluster are close to each other
- **Separation:** Elements in different clusters tend to be further apart from each other
- …clustering is not an easy task!

Given these points a clustering algorithm ⟶ might make two distinct clusters



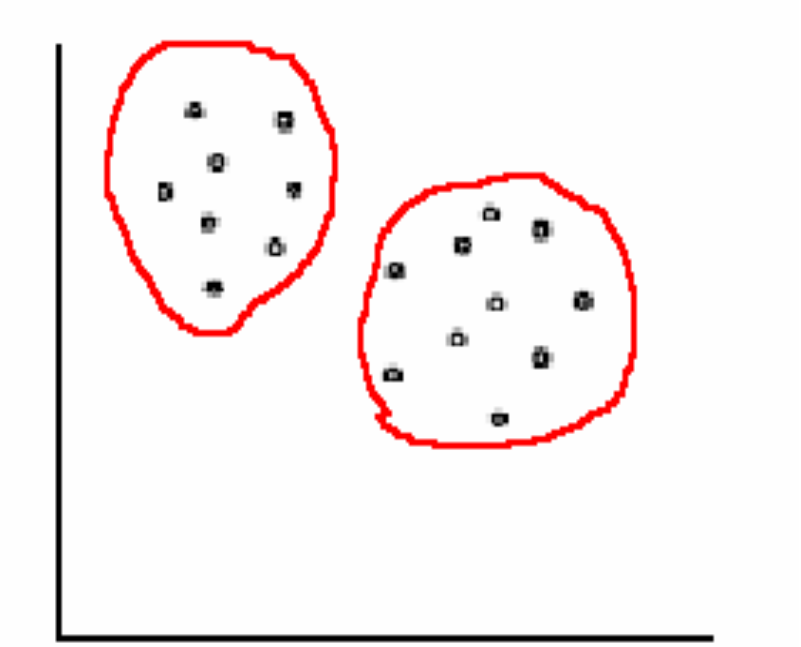
Bad Clustering

This clustering violates both Homogeneity and Separation principles



Good Clustering

This clustering satisfies both Homogeneity and Separation principles

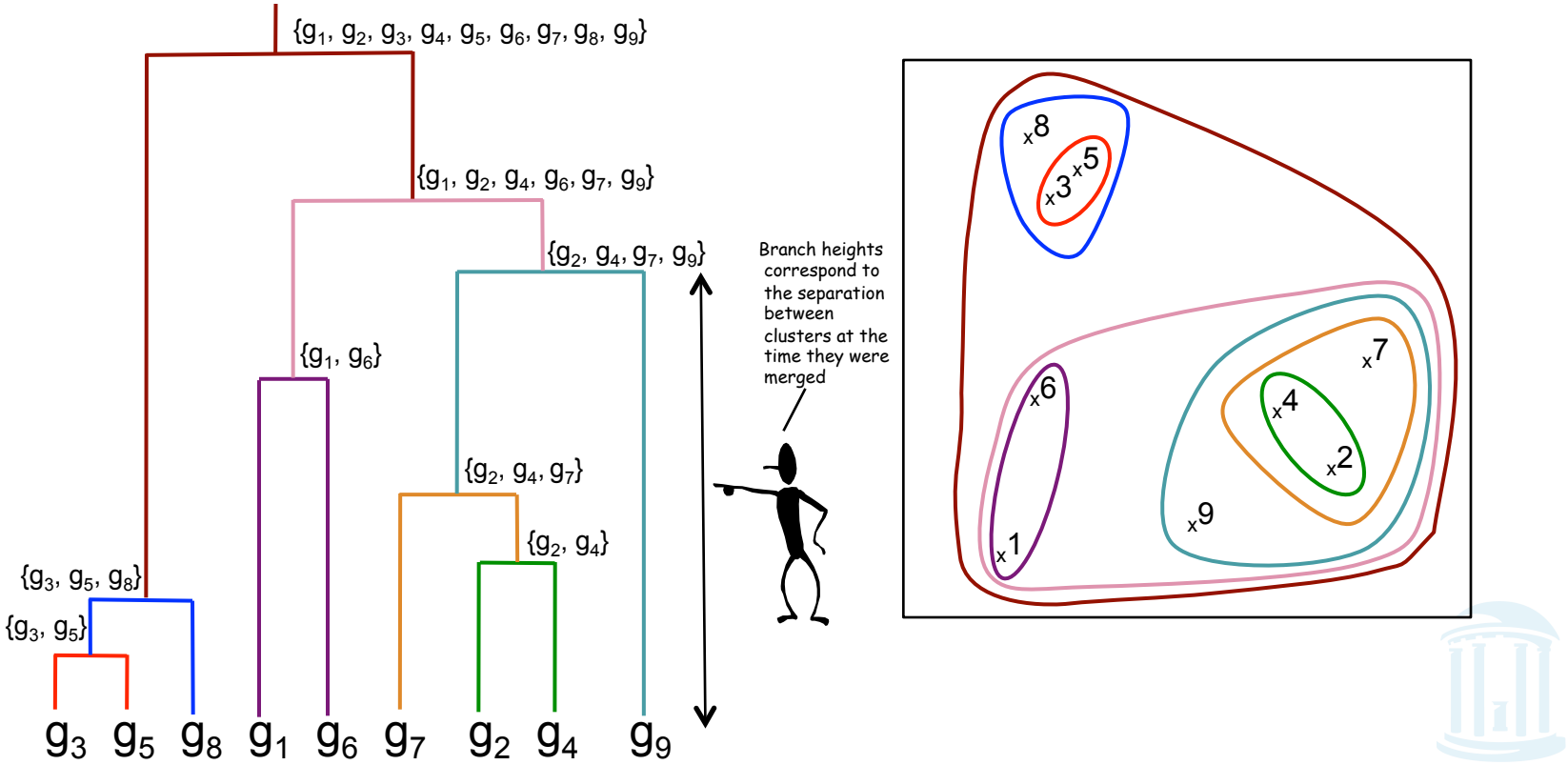


Clustering Techniques and Terms

- **Hierarchical:** Organize elements into a tree, leaves represent genes and the length of the paths between leaves represents the distances between genes. Similar genes lie within the same subtrees.
  - **Agglomerative:** Start with every element in its own cluster, and iteratively join clusters together
  - **Divisive:** Start with one cluster and iteratively divide it into smaller clusters
- **Optimization based:** Determine point sets that attempt to minimize distances within clusters (homogeneity) or maximize distances between clusters (separation)
  - K-means, K-mediods, Vector Quantization (VQ)
- **Dendrogram**: A tree representation of clustering, where one dimension is metric and others are some meaningful ordering of the points being clustered

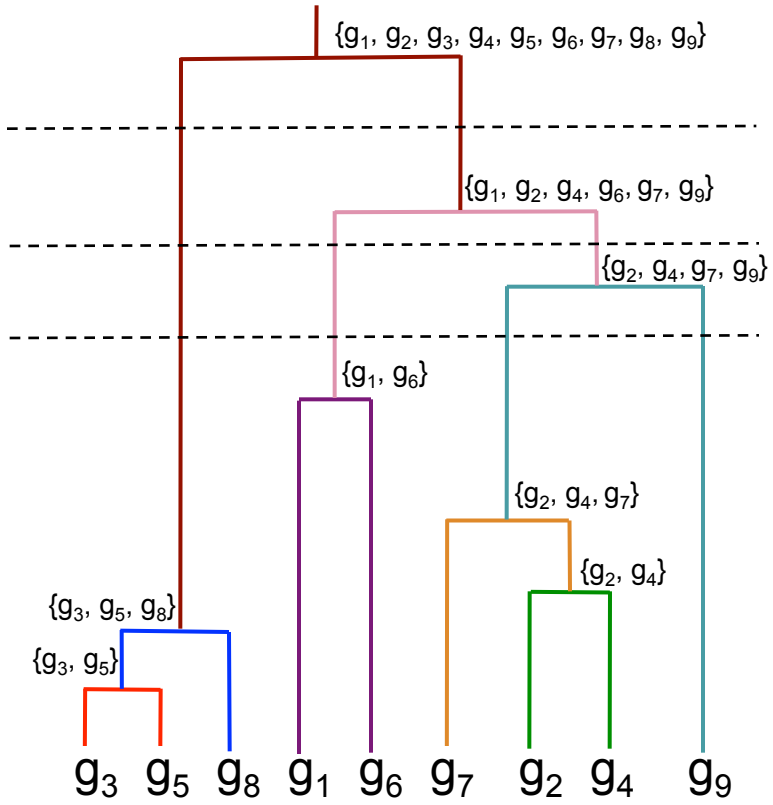
Hierarchical Clustering: Example

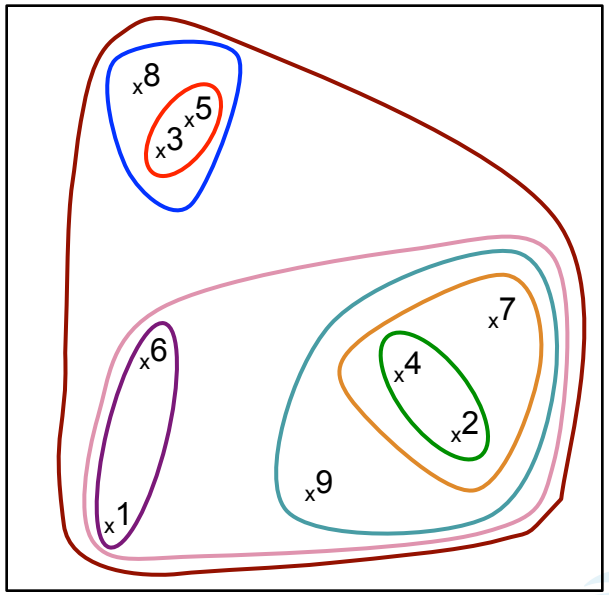
Agglomerative: Start with each point as a cluster, Join closest two clusters, Form a new cluster using the joined clusters, Repeat.



Hierarchical Clustering: Example

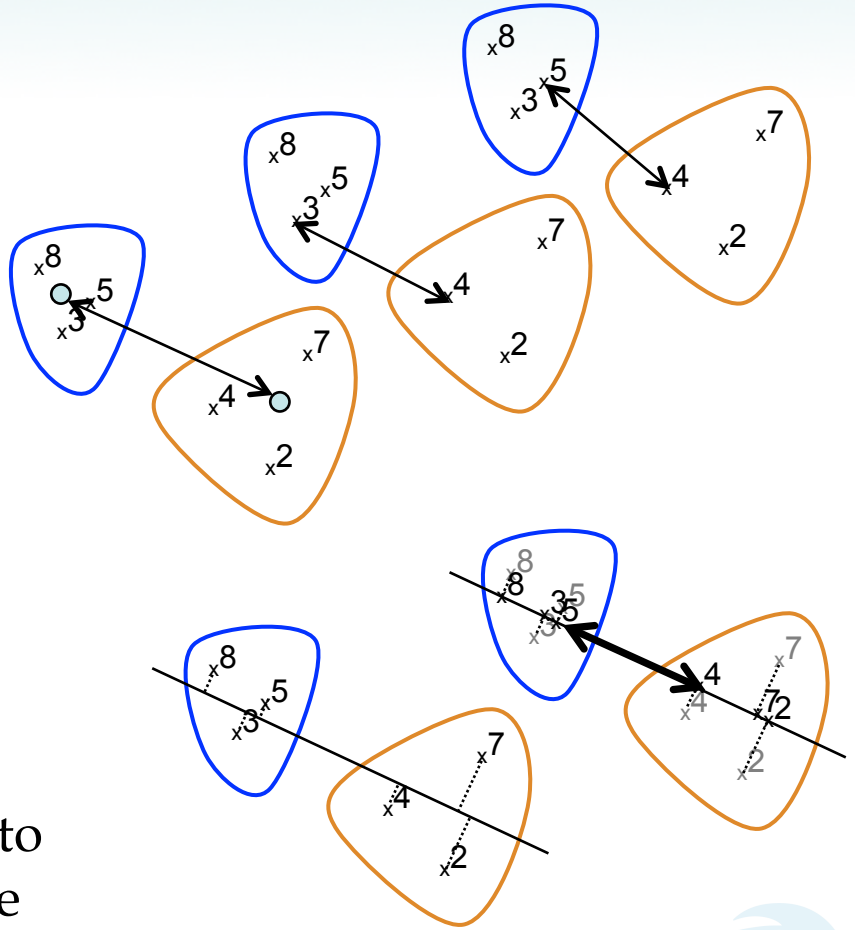
Assigning Clusters: Establish a threshold of joining distance. Remove all clusters above it.





Agglomerative Issues

- Which clusters to join?
  - Distance based
    - Cluster means
    - Closest pair
    - Closest to mediod (most centrally located point in cluster)
  - Variance based
    - Minimize residuals of a model fit
    - Closest after projection onto axis with greatest variance



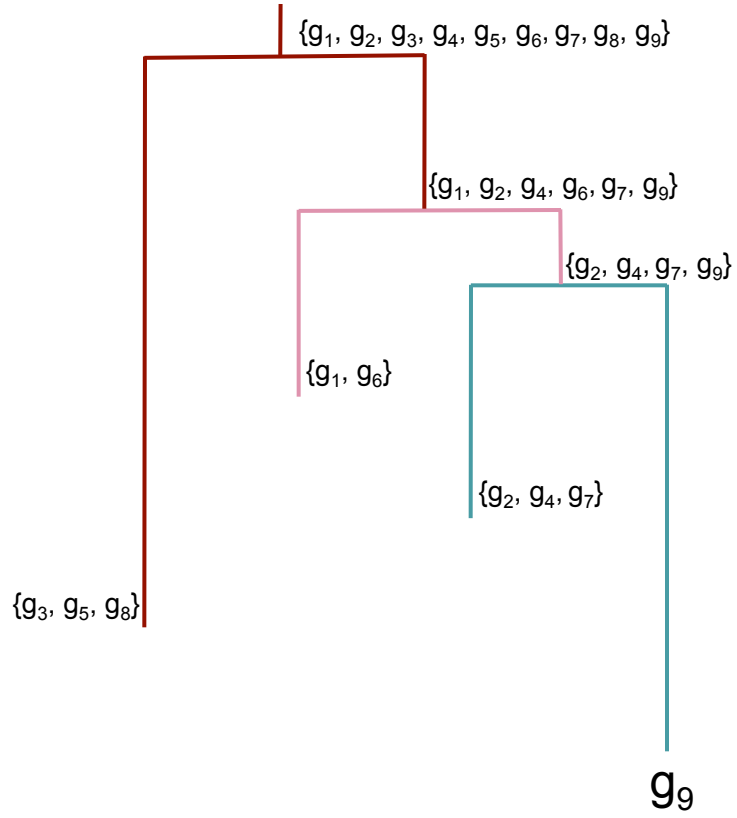
Hierarchical Clustering Algorithm

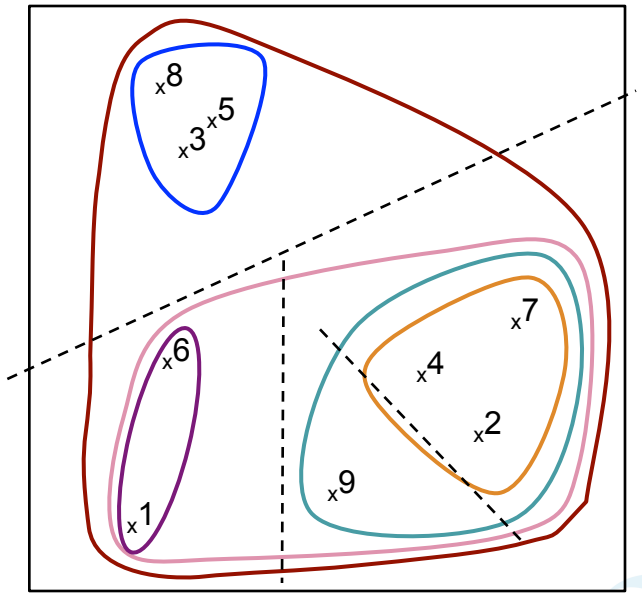
1. Hierarchical Clustering (***d*** , *n*)
2. Form *n* clusters each with one element
3. Construct a graph *T* by assigning one vertex to each cluster
4. **while** there is more than one cluster
5. Find the two "closest" clusters C_1 and C_2
6. Merge C_1 and C_2 into new cluster *C* with $|C_1| + |C_2|$ elements
7. **Compute distance from *C* to all other clusters**
8. Add a new vertex ***C*** to *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 *T*

The algorithm takes an *n*x*n* distance matrix ***d*** of pairwise distances between points as an input.

Hierarchical Clustering: Example

Divisive: Start with a single cluster composed of all points, Choose largest cluster, Split or partition it based on any metric.



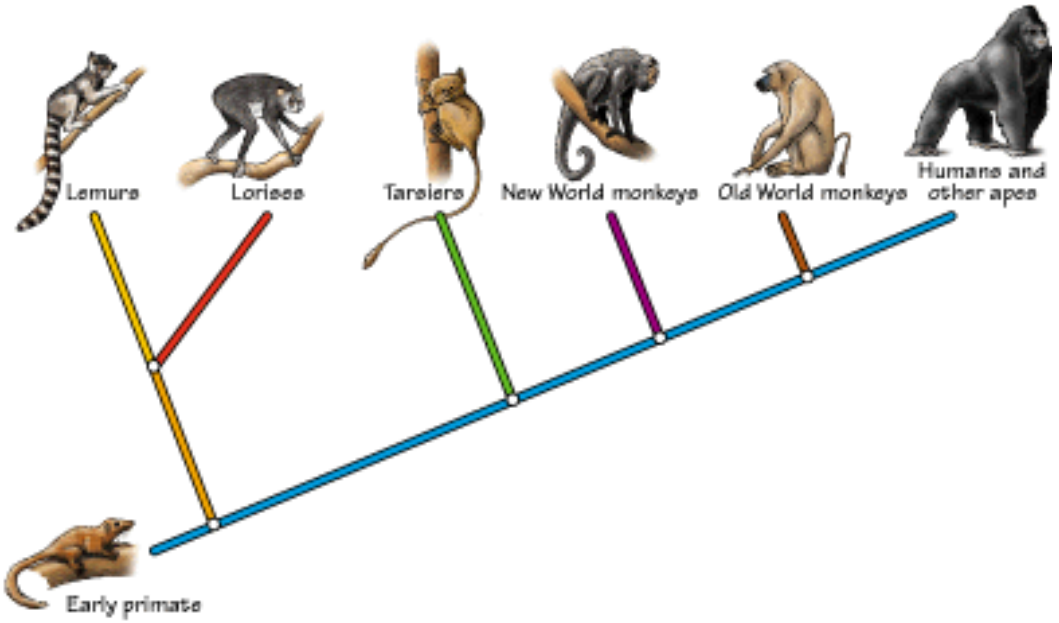


Divisive Issues

- Advantage: Terminates when objective is met
  - A target number of clusters
  - Minimize size/variance of largest cluster
  - Achieves a desired separation metric between clusters
- Division Criteria
  - Minimize distance between the separating hyperplane and the closest point to each cluster
  - Minimize residual variance

Hierarchical Clustering (cont'd)

- Hierarchical Clustering is often used to construct trees for explaining evolutionary history (Phylogeny Trees)



Hierarchical Clustering Algorithm

1. Hierarchical Clustering (*d* , *n*)
2. Form *n* clusters each with one element
3. Construct a graph *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 *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 *T*

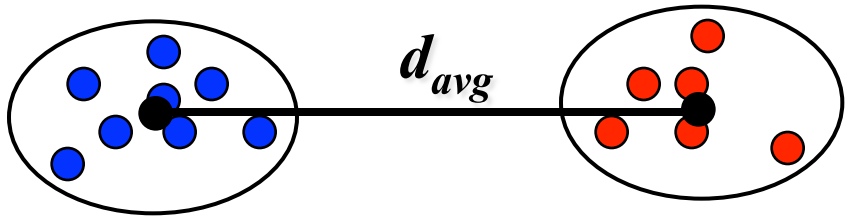
Different definitions of "distances between clusters" may lead to different clusterings

Hierarchical Clustering: Recomputing Distances

- $d_{min}(C, C^*) = \min\ d(x,y)$
  for all elements x in C and y in C^*

  - Distance between two clusters is the **smallest** distance between any pair of their elements

- $d_{avg}(C, C^*) = (1 / |C^*||C|) \sum d(x,y)$
  for all elements x in C and y in C^*



  - Distance between two clusters is the **average** distance between all pairs of their elements

Optimization-based Approaches

- Need a function to optimize– "*Squared-Error Distortion*"
- Given a data point ***v*** and a set of points ***X***,
  define the **distance** from ***v*** to ***X***

$$d(v, \boldsymbol{X})$$

as the (Euclidean) distance from ***v*** to the ***closest*** point from ***X***.

- Given a set of n data points $\boldsymbol{V}=\{v_1 \ldots v_n\}$ and a set of k points ***X***,
  define the **Squared Error Distortion**

$$d(\boldsymbol{V},\boldsymbol{X}) = \sum d(v_i, \boldsymbol{X})^2 \ / \ n \qquad 1 \leq i \leq n$$

K-Means Clustering Problem: Formulation

- **Input**: A set, ***V***, consisting of *n* points and a parameter *k*
- **Output**: A set ***X*** consisting of *k* points (*cluster centers*) that minimizes the squared error distortion *d(**V**,**X**)* over all possible choices of ***X***

1-Mean Clustering Problem: an Easy Case

- **Input**: A set, ***V***, consisting of *n* points

- **Output**: A single point ***x*** (*cluster center*) that minimizes the squared error distortion *d(**V**,**x**)* over all possible choices of ***x***

  x is just the centroid (mean) of all points

  1-Mean Clustering problem is easy. However, it becomes very difficult (NP-complete) for more than one center.

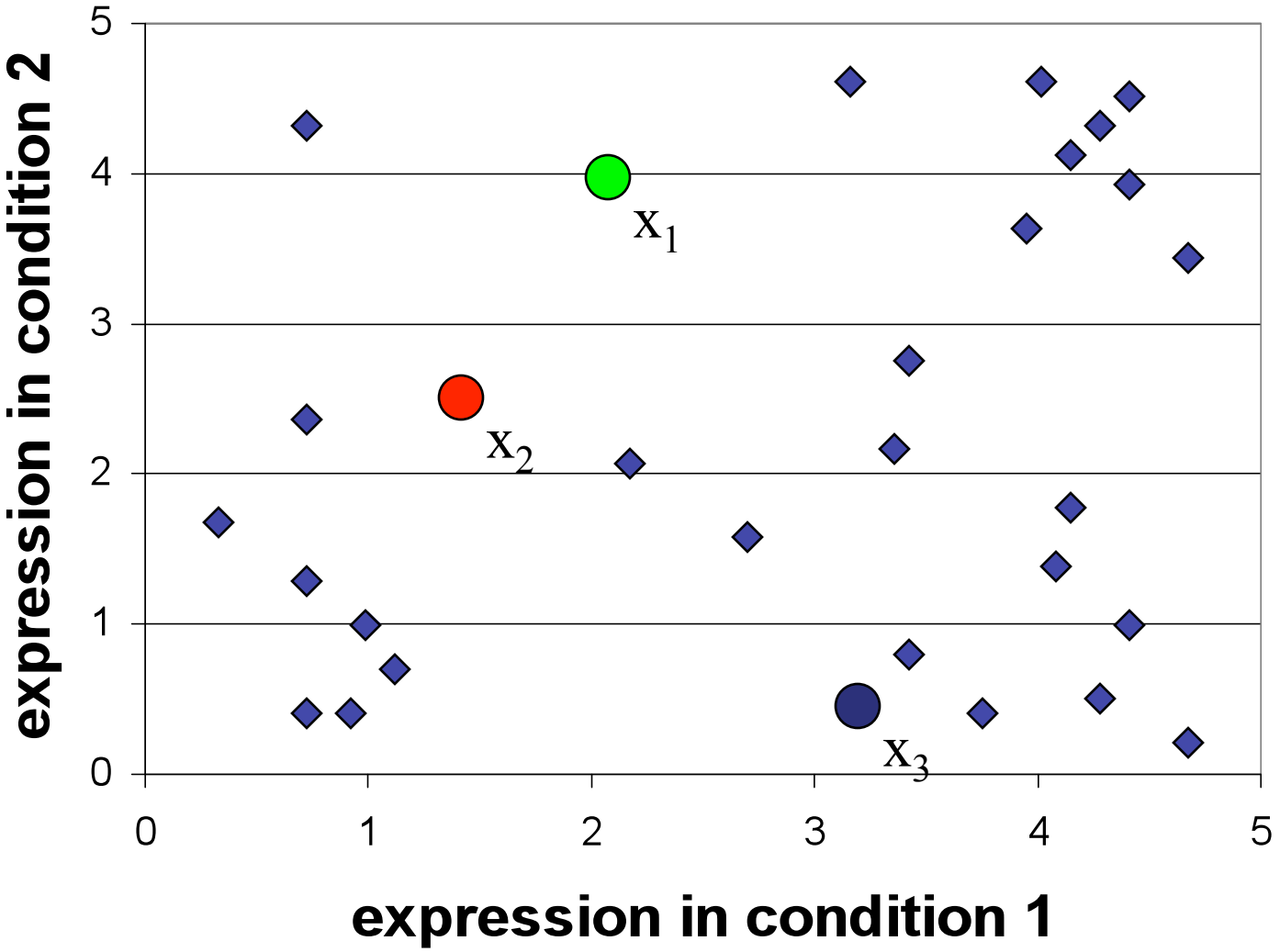
  An efficient ***heuristic*** method for K-Means clustering is the Lloyd algorithm

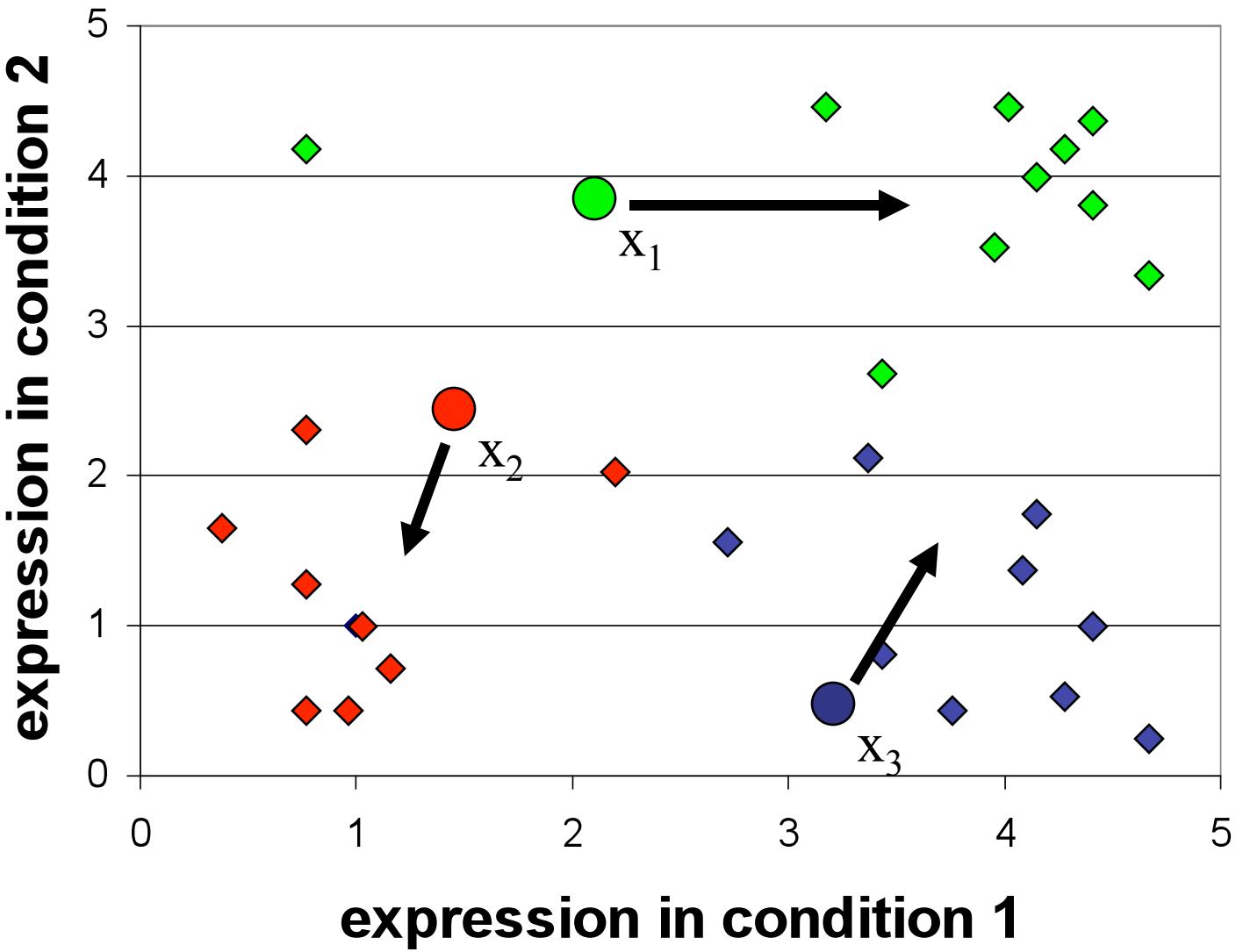
K-Means Clustering: Lloyd Algorithm

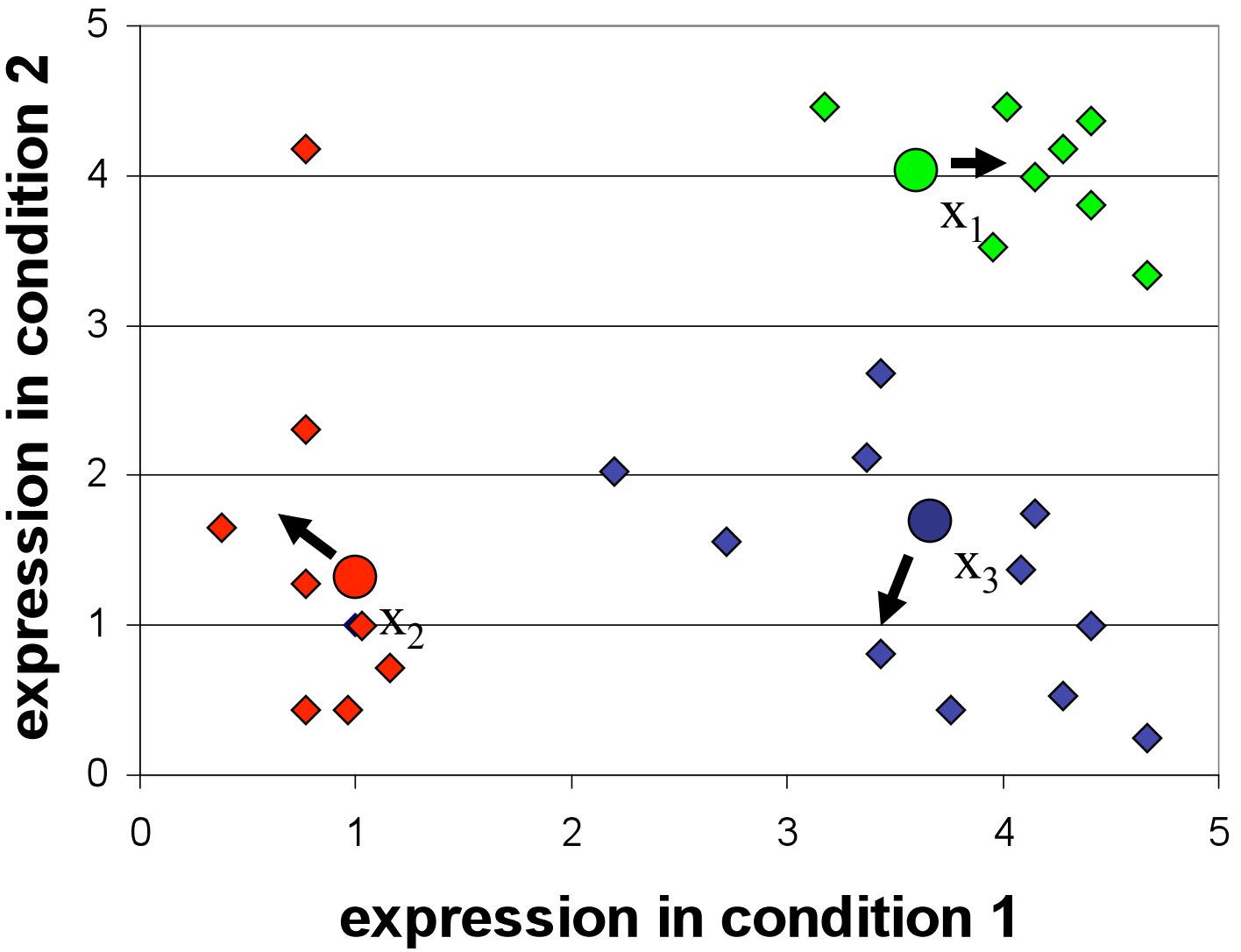
1. Lloyd Algorithm
2. Arbitrarily assign the k cluster centers
3. **while** the cluster centers keep changing
4. Assign each data point to the cluster C_i corresponding to the closest cluster representative (center) $(1 \leq i \leq k)$
5. After the assignment of all data points, compute new cluster representatives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

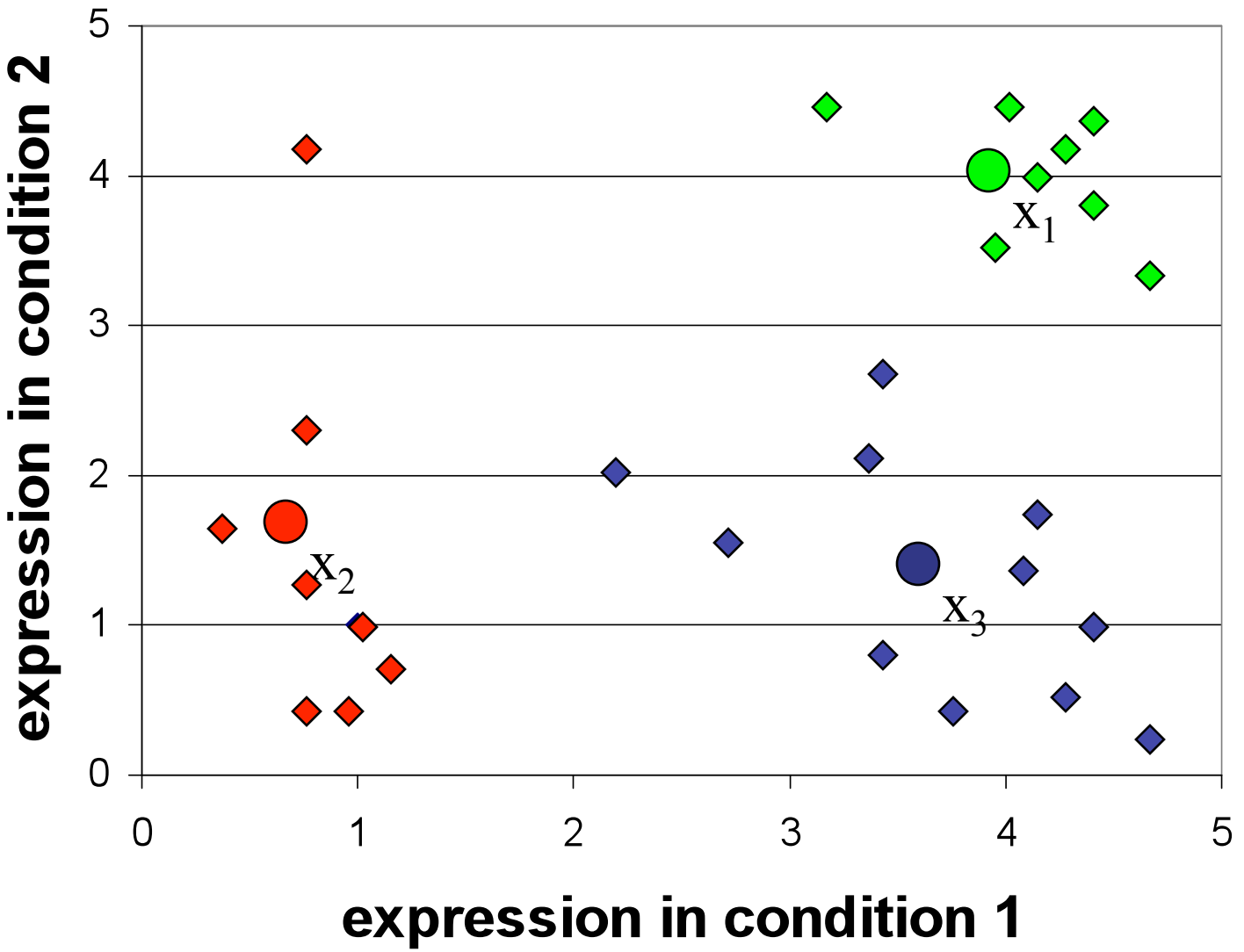
$$\Sigma v \,/\, |C| \quad \textit{for all } v \textit{ in } C \quad \text{for every cluster } C$$

*This may lead to merely a locally optimal clustering.


expression in condition 2
expression in condition 1
0
1
2
3
4
5
x1
x2
x3


expression in condition 2
expression in condition 1
0
1
2
3
4
5
x1
x2
x3


expression in condition 2
expression in condition 1
0
1
2
3
4
5
x1
x2
x3


expression in condition 2
expression in condition 1
0
1
2
3
4
5
x1
x2
x3

Conservative K-Means Algorithm

- Lloyd algorithm is fast but in each iteration it moves many data points, not necessarily converging.
- A more conservative method would be to move one point at a time only if it improves the overall **clustering cost**
  - The smaller the clustering cost of a partition of data points is the better that clustering is
  - Different methods (e.g., the squared error distortion) can be used to measure this clustering cost

K-Means "Greedy" Algorithm

1. ProgressiveGreedyK-Means(*k*)
2. Select an arbitrary partition *P* into *k* clusters
3. **while** forever
4. *bestChange* ← 0
5. **for** every cluster *C*
6. **for** every element *i* not in *C*
7. **if** moving *i* to cluster *C* reduces its clustering cost
8. **if** (cost(*P*) – cost($P_{i \to C}$) > *bestChange*
9. *bestChange* ← cost(*P*) – cost($P_{i \to C}$)
10. i^* ← *I*
11. C^* ← *C*
12. **if** *bestChange* > 0
13. Change partition *P* by moving i^* to C^*
14. **else**
15. **return** *P*

Are there better algorithms?

- Yes!