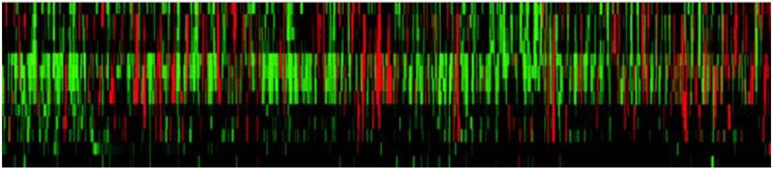
CS 6824: Basic Clustering Algorithms for Gene Expression Analysis

T. M. Murali

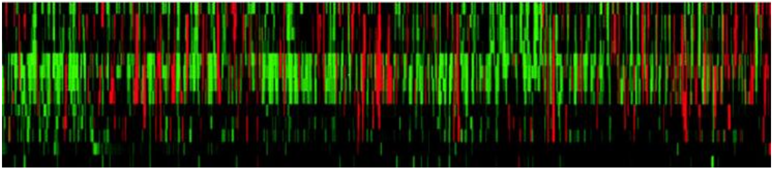
February 14, 2011

Gene Expression Analysis

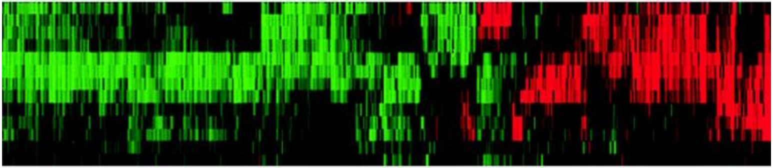


- How do we automatically extract meaning from so much microarray data?

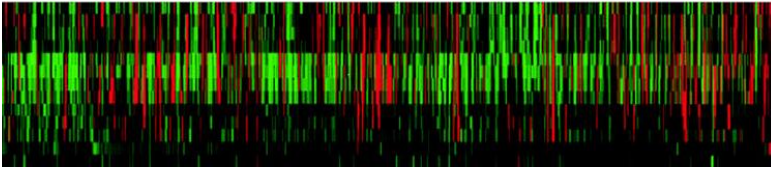
Gene Expression Analysis



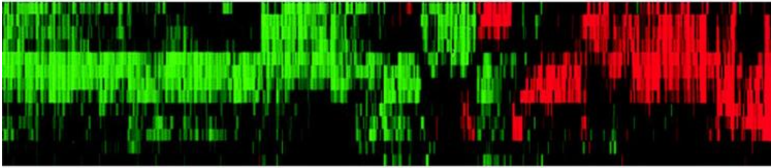
- How do we automatically extract meaning from so much microarray data?



Gene Expression Analysis



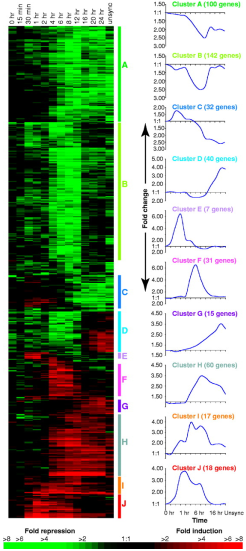
- How do we automatically extract meaning from so much microarray data?



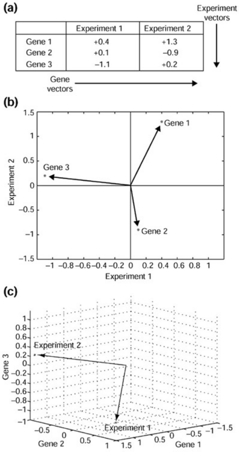
Describe data in terms of clusters of samples and genes that have strong internal similarities.

Example: Iyer and co-authors (Science 1999)

- Measure temporal expression profiles of 8600 human genes in fibroblasts in response to serum addition.
- Over 200 previously unknown genes with specific temporal expression profiles.
- Based on known genes in cluster, authors assign putative functions to these genes.

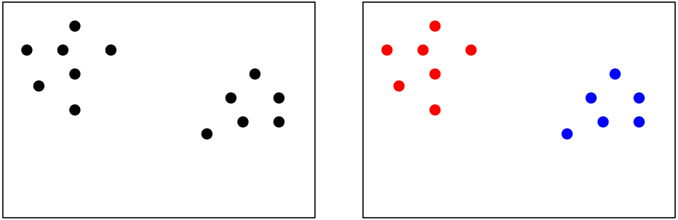


Viewing DNA Microarray Data as Multi-Dimensional Points



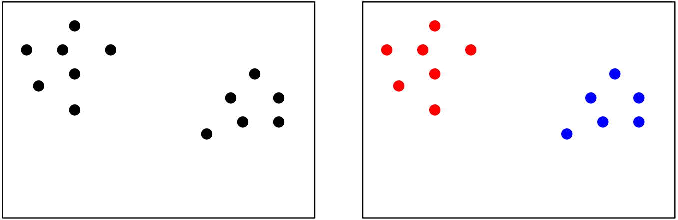
- m genes and n samples.
- Figure (b)
  - Gene $\equiv$ point: m points
  - Condition $\equiv$ dimension: n-dimensional space
  - Expression level $\equiv$ coordinate.
- Figure (c)
  - Sample $\equiv$ point: n points.
  - Condition $\equiv$ dimension: m-dimensional space.
  - Expression level $\equiv$ coordinate.
- For a point p, p_i is its ith coordinate.

Definition of Clustering



Given a set of m genes whose expression levels are measured across n conditions, find the best partition of the genes into subsets such that each subset contains genes whose expression profiles are similar to each other.

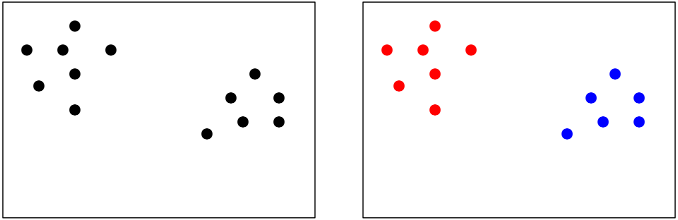
Definition of Clustering



Given a set of m genes whose expression levels are measured across n conditions, find the best partition of the genes into subsets such that each subset contains genes whose expression profiles are similar to each other.

- How many subsets?

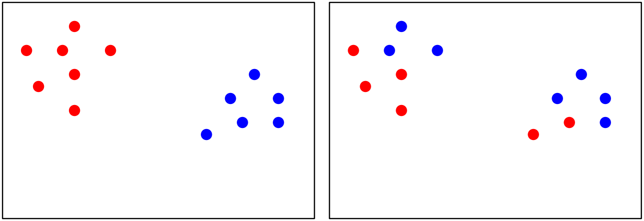
Definition of Clustering



Given a set of m genes whose expression levels are measured across n conditions, find the best partition of the genes into subsets such that each subset contains genes whose expression profiles are similar to each other.

- How many subsets?
- How do we measure how similar the expression profiles of two genes are?

Definition of Clustering



Given a set of m genes whose expression levels are measured across n conditions, find the best partition of the genes into subsets such that each subset contains genes whose expression profiles are similar to each other.

- How many subsets?
- How do we measure how similar the expression profiles of two genes are?
- How do we compare two different partitions?

Measuring Similarity of Points

Measuring Similarity of Points

- Distance between two points p and q is $d(p, q)$.
- Euclidean metric: $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.

Measuring Similarity of Points

- Distance between two points p and q is $d(p, q)$.
- Euclidean metric: $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.
- Manhattan metric: $d(p, q) = \sum_i |p_i - q_i|$.

Measuring Similarity of Points

- Distance between two points p and q is $d(p, q)$.
- Euclidean metric: $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.
- Manhattan metric: $d(p, q) = \sum_i |p_i - q_i|$.
- Pearson correlation coefficient:

$$\frac{1}{n}\sum_i \left(\frac{p_i \qquad}{\qquad}\right)\left(\frac{q_i \qquad}{\qquad}\right)$$

Measuring Similarity of Points

- Distance between two points p and q is $d(p, q)$.
- Euclidean metric: $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.
- Manhattan metric: $d(p, q) = \sum_i |p_i - q_i|$.
- Pearson correlation coefficient:

$$\frac{1}{n} \sum_i \left(\frac{p_i - \mu(p)}{} \right) \left(\frac{q_i }{} \right)$$

  - $\mu(p)$: average of p's coordinates,

Measuring Similarity of Points

- Distance between two points p and q is $d(p, q)$.
- Euclidean metric: $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.
- Manhattan metric: $d(p, q) = \sum_i |p_i - q_i|$.
- Pearson correlation coefficient:

$$\frac{1}{n} \sum_i \left(\frac{p_i - \mu(p)}{\sigma(p)} \right) \left(\frac{q_i}{} \right)$$

  - $\mu(p)$: average of p's coordinates, $\sigma(p)$: standard deviation of p's coordinates.

Measuring Similarity of Points

- Distance between two points p and q is $d(p, q)$.
- Euclidean metric: $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.
- Manhattan metric: $d(p, q) = \sum_i |p_i - q_i|$.
- Pearson correlation coefficient:

$$\frac{1}{n} \sum_i \left(\frac{p_i - \mu(p)}{\sigma(p)} \right) \left(\frac{q_i - \mu(q)}{} \right)$$

  - $\mu(p)$: average of p's coordinates, $\sigma(p)$: standard deviation of p's coordinates.

Measuring Similarity of Points

- Distance between two points p and q is $d(p, q)$.
- Euclidean metric: $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.
- Manhattan metric: $d(p, q) = \sum_i |p_i - q_i|$.
- Pearson correlation coefficient:

$$\frac{1}{n} \sum_i \left(\frac{p_i - \mu(p)}{\sigma(p)} \right) \left(\frac{q_i - \mu(q)}{\sigma(q)} \right)$$

  - $\mu(p)$: average of p's coordinates, $\sigma(p)$: standard deviation of p's coordinates.

Measuring Similarity of Points

- Distance between two points p and q is $d(p, q)$.
- Euclidean metric: $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.
- Manhattan metric: $d(p, q) = \sum_i |p_i - q_i|$.
- Pearson correlation coefficient:

$$\frac{1}{n} \sum_i \left(\frac{p_i - \mu(p)}{\sigma(p)} \right) \left(\frac{q_i - \mu(q)}{\sigma(q)} \right)$$

  - $\mu(p)$: average of p's coordinates, $\sigma(p)$: standard deviation of p's coordinates.
- Other distances: normalised dot product, K-L divergence, relative entropy.

Measuring Similarity of Points

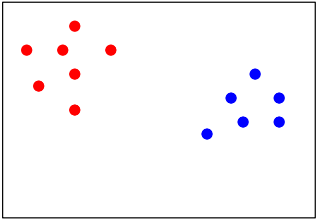
- Distance between two points p and q is $d(p, q)$.
- Euclidean metric: $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.
- Manhattan metric: $d(p, q) = \sum_i |p_i - q_i|$.
- Pearson correlation coefficient:

$$\frac{1}{n} \sum_i \left(\frac{p_i - \mu(p)}{\sigma(p)} \right) \left(\frac{q_i - \mu(q)}{\sigma(q)} \right)$$

  - $\mu(p)$: average of p's coordinates, $\sigma(p)$: standard deviation of p's coordinates.
- Other distances: normalised dot product, K-L divergence, relative entropy.
- Metrics obey triangle inequality: $d(p, q) + d(q, r) \geq d(p, r)$.
  - Euclidean, Manhattan distances are metrics.
  - Correlation, dot product are not metrics.

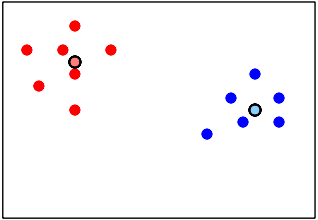
Quality of a Partition

- Partition points into k clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.
- Define quality q_i of a cluster C_i and define quality $q(\mathcal{C})$ in terms of q_is.



Quality of a Partition

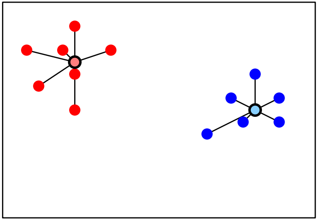
- Partition points into k clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.
- Define quality q_i of a cluster C_i and define quality $q(\mathcal{C})$ in terms of q_is.



- Sum of squared errors.
  - $\mu_i =$ average of points in C_i.

Quality of a Partition

- Partition points into k clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.
- Define quality q_i of a cluster C_i and define quality $q(\mathcal{C})$ in terms of q_is.



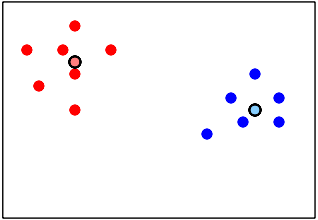
- Sum of squared errors.
  - μ_i = average of points in C_i.
  - $q_i = \frac{1}{n_i} \sum_{p \in C_i} d(p, \mu_i)^2$ = average of squared distance from every point in C_i to q_i.
  - $q(\mathcal{C}) = \sum_i q_i$.

Algorithms

- k-means algorithm.
- Hierarchical clustering.

Algorithms

- *k*-means: find *k* cluster "centres" and form clusters by assigning a point to the closest cluster centre.



k-means algorithm

Partition S into k clusters that minimise the sum of squared errors $q(\mathcal{C}) = \sum_i \sum_{p \in C_i} \|p - \mu_i\|^2$ over all possible partitions of S into k clusters.

k-means algorithm

Partition S into k clusters that minimise the sum of squared errors $q(\mathcal{C}) = \sum_i \sum_{p \in C_i} \|p - \mu_i\|^2$ over all possible partitions of S into k clusters.

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

Details of k-means algorithm

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

Details of k-means algorithm

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Initialisation:

Details of *k*-means algorithm

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Initialisation: random μ_i's or "well-separated" μ_i's.

Details of k-means algorithm

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Initialisation: random μ_i's or "well-separated" μ_i's.
- Checking for termination :

Details of k-means algorithm

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Initialisation: random μ_i's or "well-separated" μ_i's.
- Checking for termination :
  - use thresholds to avoid numerical errors.
  - check if sets in the partition do not change.

Properties of k-means

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Each iteration takes time.

Properties of k-means

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Each iteration takes $O(kmn)$ time.

Properties of k-means

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Each iteration takes $O(kmn)$ time.
- $q(\mathcal{C})$

Properties of k-means

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Each iteration takes $O(kmn)$ time.
- $q(\mathcal{C})$ does not increase.

Properties of k-means

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Each iteration takes $O(kmn)$ time.
- $q(\mathcal{C})$ does not increase.
- Algorithm can get stuck in a local minimum.

Properties of k-means

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

- Each iteration takes $O(kmn)$ time.
- $q(\mathcal{C})$ does not increase.
- Algorithm can get stuck in a local minimum.

Properties of k-means

1. Initialise centres $\mu_1, \mu_2, \ldots \mu_k$.
2. Repeat
   - For each point p, put p in cluster C_i if μ_i is the centre closest to p.
   - Recalculate μ_i's (average of points in C_i).
3. until μ_i's don't change.

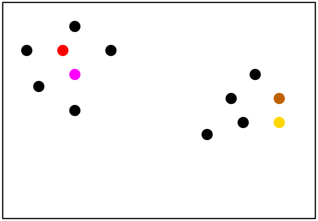
- Each iteration takes $O(kmn)$ time.
- $q(\mathcal{C})$ does not increase.
- Algorithm can get stuck in a local minimum.
- Does not work particularly well in very high (≥ 40) dimensions.

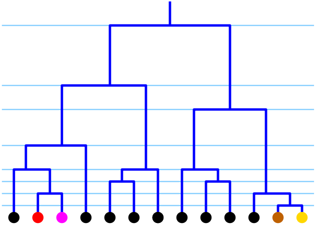
Algorithms

- k-means algorithm.
- Hierarchical clustering.

Hierarchical Clustering

- Attempt to recursively find sub-clusters within clusters.
- Natural way to "zoom into" areas of interest.
- Represent using a tree or dendrogram.





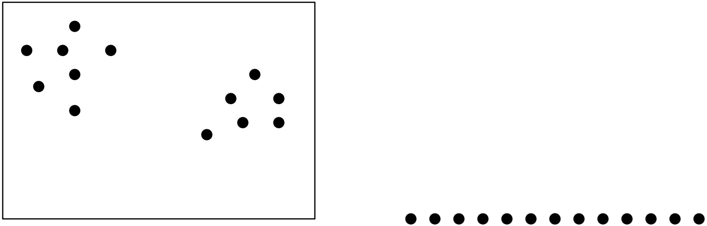
Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

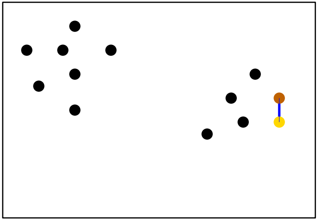
1. Start with every sample (gene) in its own cluster.



Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

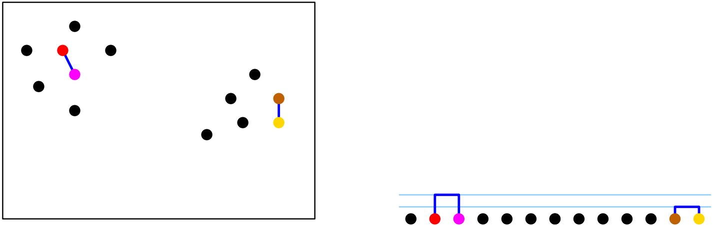
1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let C_i and C_j be the clusters "nearest" each other.
   - Merge C_i and C_j.



Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

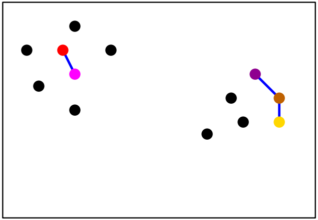
1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let C_i and C_j be the clusters "nearest" each other.
   - Merge C_i and C_j.



Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

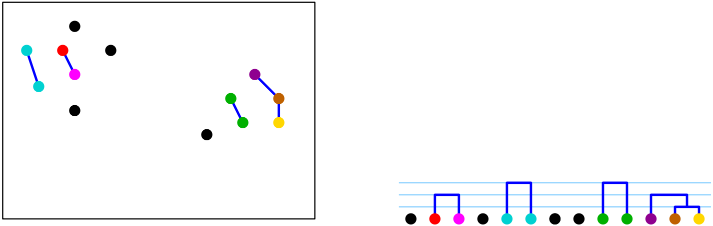
1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let C_i and C_j be the clusters "nearest" each other.
   - Merge C_i and C_j.



Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

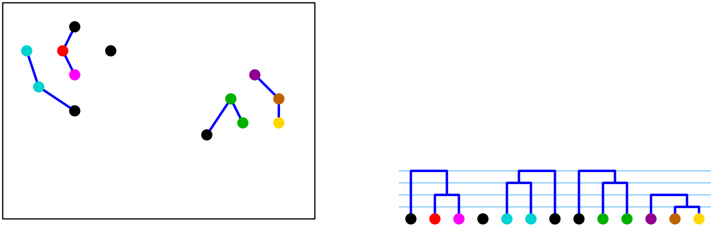
1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let C_i and C_j be the clusters "nearest" each other.
   - Merge C_i and C_j.



Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

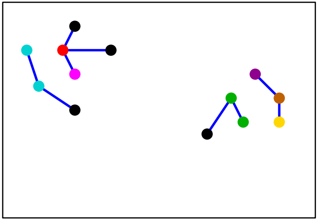
1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let C_i and C_j be the clusters "nearest" each other.
   - Merge C_i and C_j.

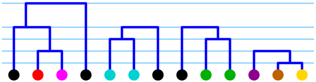


Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let C_i and C_j be the clusters "nearest" each other.
   - Merge C_i and C_j.

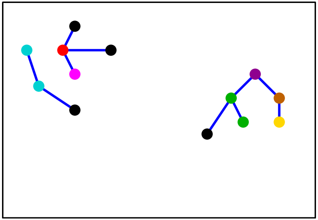


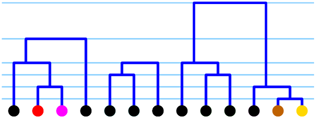


Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let C_i and C_j be the clusters "nearest" each other.
   - Merge C_i and C_j.

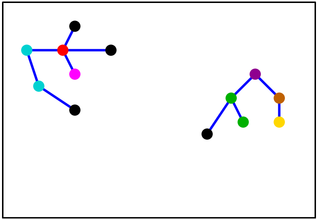


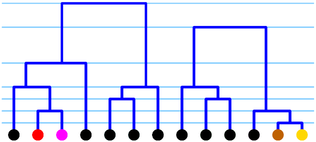


Hierarchical Clustering Algorithm

- Bottom-up clustering algorithm.

1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let C_i and C_j be the clusters "nearest" each other.
   - Merge C_i and C_j.

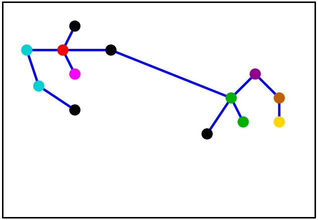


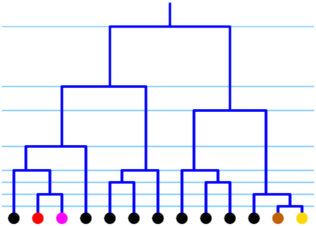


Hierarchical Clustering Algorithm

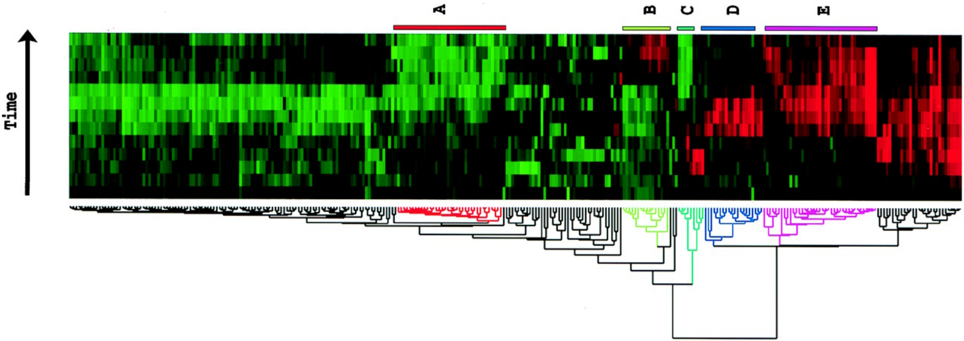
- Bottom-up clustering algorithm.

1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let C_i and C_j be the clusters "nearest" each other.
   - Merge C_i and C_j.
3. until all the samples (genes) are in one cluster.

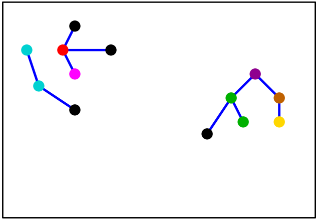


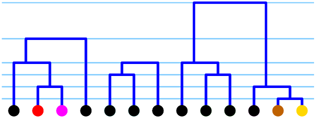


Hierarchical Clustering Result

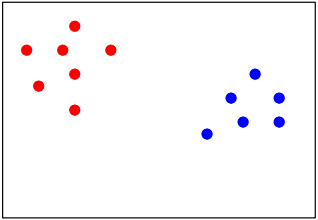


Measuring Distance between Clusters

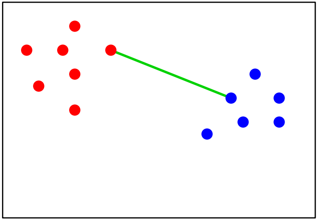




Measuring Distance between Clusters

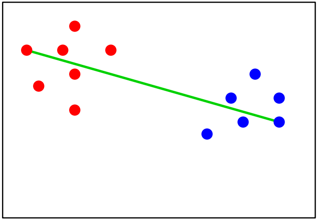


Measuring Distance between Clusters



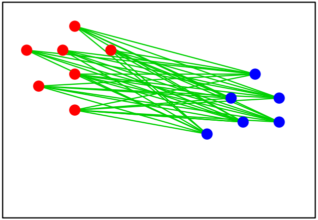
- $d_{min}(D_i, D_j)$ = distance between closest pair of points.

Measuring Distance between Clusters



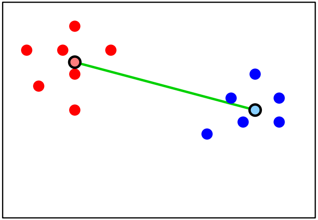
- $d_{min}(D_i, D_j) =$ distance between closest pair of points.
- $d_{max}(D_i, D_j) =$ distance between farthest pair of points.

Measuring Distance between Clusters



- $d_{min}(D_i, D_j) =$ distance between closest pair of points.
- $d_{max}(D_i, D_j) =$ distance between farthest pair of points.
- $d_{avg}(D_i, D_j) =$ average of distances between all pairs of points.

Measuring Distance between Clusters



- $d_{min}(D_i, D_j)$ = distance between closest pair of points.
- $d_{max}(D_i, D_j)$ = distance between farthest pair of points.
- $d_{avg}(D_i, D_j)$ = average of distances between all pairs of points.
- $d_{mean}(D_i, D_j) = d(\mu_i, \mu_j)$.

Measuring Distance between Clusters

- $d_{min}(D_i, D_j) =$ distance between closest pair of points.
- $d_{max}(D_i, D_j) =$ distance between farthest pair of points.
- $d_{avg}(D_i, D_j) =$ average of distances between all pairs of points.
- $d_{mean}(D_i, D_j) = d(\mu_i, \mu_j)$.

- Computing $d_{min}, d_{max}, d_{avg}$ takes $O(n_i n_j)$ time.
- Computing d_{mean} takes $O(n_i + n_j)$ time.

Running Time of Hierarchical Clustering

1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let D_i and D_j be the clusters "nearest" each other.
   - Merge D_i and D_j.
3. until all the samples (genes) are in one cluster.

Running Time of Hierarchical Clustering

1. Start with every sample (gene) in its own cluster.
2. Repeat
   - Let D_i and D_j be the clusters “nearest” each other.
   - Merge D_i and D_j.
3. until all the samples (genes) are in one cluster.

- Store all $O(m^2)$ inter-point distances.
- At each iteration, compute distance between every pair of clusters: takes $O(nm^2)$ time in total.
- There are n iterations, so overall running time is $O(nmm^2) = O(nm^3)$.

Properties of Hierarchical Clustering

- Using d_{min}, tree tends to look like an elongated chain.
- Using d_{max}, clusters may not be well separated.
- Other measures try to alleviate this problem.
- In case of d_{min}, tree produced is the minimum spanning tree.
- In other cases, it is difficult to state what properties the partition satisfies.

Evaluating Cluster Quality

- How do we know a cluster represents "useful" biological knowledge?

Evaluating Cluster Quality

- How do we know a cluster represents "useful" biological knowledge?
- Compute functional enrichment? If there are k genes with a particular function in a cluster with with l genes, is this fact interesting?

Evaluating Cluster Quality

- How do we know a cluster represents "useful" biological knowledge?
- Compute functional enrichment? If there are k genes with a particular function in a cluster with with l genes, is this fact interesting?
- Must look at how many genes overall are annotated with that function.

Evaluating Cluster Quality

- How do we know a cluster represents "useful" biological knowledge?
- Compute functional enrichment? If there are k genes with a particular function in a cluster with with l genes, is this fact interesting?
- Must look at how many genes overall are annotated with that function.
- Use χ^2 test or Fisher's exact test.

Fisher's Exact Test

- Let C be the cluster of interest, $c = \#$genes in the cluster.
- Let u be the number of genes in the gene expression data set.

Fisher's Exact Test

- Let C be the cluster of interest, $c = \#$genes in the cluster.
- Let u be the number of genes in the gene expression data set.
- Let f be the function of interest:

Fisher's Exact Test

- Let C be the cluster of interest, $c = \#$genes in the cluster.
- Let u be the number of genes in the gene expression data set.
- Let f be the function of interest:
  - $u_f = \#$genes in the data set annotated with f.
  - $c_f = \#$genes in the cluster C annotated with f.

Fisher's Exact Test

- Let C be the cluster of interest, $c = \#$genes in the cluster.
- Let u be the number of genes in the gene expression data set.
- Let f be the function of interest:
  - $u_f = \#$genes in the data set annotated with f.
  - $c_f = \#$genes in the cluster C annotated with f.
- Fisher's exact test answers the following question:

If we selected c genes at random from the set of all u genes,

Fisher's Exact Test

- Let C be the cluster of interest, $c = \#$genes in the cluster.
- Let u be the number of genes in the gene expression data set.
- Let f be the function of interest:
  - $u_f = \#$genes in the data set annotated with f.
  - $c_f = \#$genes in the cluster C annotated with f.
- Fisher's exact test answers the following question:

If we selected c genes at random from the set of all u genes, what is the probability that we will select c_f or more genes from the set of u_f genes annotated with f?

Fisher's Exact Test

- Let C be the cluster of interest, $c = \#$genes in the cluster.
- Let u be the number of genes in the gene expression data set.
- Let f be the function of interest:
  - $u_f = \#$genes in the data set annotated with f.
  - $c_f = \#$genes in the cluster C annotated with f.
- Fisher's exact test answers the following question:

If we selected c genes at random from the set of all u genes, what is the probability that we will select c_f or more genes from the set of u_f genes annotated with f?

$$\sum_{i=c_f}^{\min(c,u_f)}$$

Fisher's Exact Test

- Let C be the cluster of interest, $c = \#$genes in the cluster.
- Let u be the number of genes in the gene expression data set.
- Let f be the function of interest:
  - $u_f = \#$genes in the data set annotated with f.
  - $c_f = \#$genes in the cluster C annotated with f.
- Fisher's exact test answers the following question:

If we selected c genes at random from the set of all u genes, what is the probability that we will select c_f or more genes from the set of u_f genes annotated with f?

$$\sum_{i=c_f}^{\min(c,u_f)} \frac{\binom{u_f}{i}\binom{u-u_f}{c-i}}{\binom{u}{c}}$$

Other Issues in Functional Enrichment

- Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists, Huang et al., *Nucleic acids research*, 2008

Other Issues in Functional Enrichment

- Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists, Huang et al., *Nucleic acids research*, 2008
- Compute enrichment of each function, one at a time:

Other Issues in Functional Enrichment

- Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists, Huang et al., *Nucleic acids research*, 2008
- Compute enrichment of each function, one at a time: multiple hypotheses testing.

Other Issues in Functional Enrichment

- Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists, Huang et al., *Nucleic acids research*, 2008
- Compute enrichment of each function, one at a time: multiple hypotheses testing.
- Parent-child relationships between functions:

Other Issues in Functional Enrichment

- Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists, Huang et al., *Nucleic acids research*, 2008
- Compute enrichment of each function, one at a time: multiple hypotheses testing.
- Parent-child relationships between functions: Ontologizer algorithm Bauer et al., *Bioinformatics*, 2008.

Other Issues in Functional Enrichment

- Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists, Huang et al., *Nucleic acids research*, 2008
- Compute enrichment of each function, one at a time: multiple hypotheses testing.
- Parent-child relationships between functions: Ontologizer algorithm Bauer et al., *Bioinformatics*, 2008.
- Enrichment of multiple functions at the same time:

Other Issues in Functional Enrichment

- Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists, Huang et al., *Nucleic acids research*, 2008
- Compute enrichment of each function, one at a time: multiple hypotheses testing.
- Parent-child relationships between functions: Ontologizer algorithm Bauer et al., *Bioinformatics*, 2008.
- Enrichment of multiple functions at the same time: GenGO algorithm Lu et al., *Nucleic Acids Research* 2008