CS 5984: Basic Clustering Algorithms for Gene Expression Analysis

T. M. Murali

January 26, 2006
# Project Groups and Meeting Times

<table>
<thead>
<tr>
<th>Annotation of human genes</th>
<th>Andrew, Pallavi</th>
<th>5pm, Tuesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicluster analysis</td>
<td>Fiona, Kim, Revonda</td>
<td>2pm, Wednesday</td>
</tr>
<tr>
<td>PPI transfer</td>
<td>Amrita, Matt, Tim</td>
<td>10am, Friday</td>
</tr>
<tr>
<td>Tandem array genes</td>
<td>Valia</td>
<td>11am, Friday</td>
</tr>
<tr>
<td>VIRGO, DAMI</td>
<td>Naveed</td>
<td>4pm, Friday</td>
</tr>
</tbody>
</table>
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 et al. (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

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

\[
\begin{array}{|c|c|c|}
\hline
\text{Gene} & \text{Experiment 1} & \text{Experiment 2} \\
\hline
\text{Gene 1} & 0.4 & 1.3 \\
\text{Gene 2} & 0.1 & -0.9 \\
\text{Gene 3} & -1.1 & 0.2 \\
\hline
\end{array}
\]
Definition of Clustering

*Given a set of $n$ genes whose expression levels are measured across $d$ 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 $n$ genes whose expression levels are measured across $d$ 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 $n$ genes whose expression levels are measured across $d$ 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 n genes whose expression levels are measured across d 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
Introduction

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: 

- Pearson correlation coefficient: 

- Other distances: normalised dot product, K-L divergence, relative entropy.
- Metrics obey triangle inequality: 

- Euclidean, Manhattan distances are metrics.
- Correlation, dot product are not metrics.
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: $d(p, q) = \sum_i |p_i - q_i|$. 

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.
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:
  \[
  \frac{1}{d} \sum_i \left( \frac{p_i}{\mu(p)} \right) \left( \frac{q_i}{\mu(q)} \right)
  \]
  \( \mu(p) \): 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.
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: 
  \[
  \frac{1}{d} \sum_i \left( \frac{p_i - \mu(p)}{\sigma(p)} \right) \left( \frac{q_i}{\sigma(q)} \right)
  \]
  - $\mu(p)$: average of $p$’s coordinates,
Introduction

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:
  \[
  \frac{1}{d} \sum_i \left( \frac{p_i - \mu(p)}{\sigma(p)} \right) \left( \frac{q_i}{\sigma(p)} \right)
  \]
  - \( \mu(p) \): 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:
  
  $$
  \frac{1}{d} \sum_i \left( \frac{p_i - \mu(p)}{\sigma(p)} \right) \left( q_i - \mu(q) \right)
  $$

  - $\mu(p)$: 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:
  \[
  \frac{1}{d} \sum_i \left( \frac{p_i - \mu(p)}{\sigma(p)} \right) \left( \frac{q_i - \mu(q)}{\sigma(q)} \right)
  \]
  - \( \mu(p) \): 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:
  \[
  \frac{1}{d} \sum_i \left( \frac{p_i - \mu(p)}{\sigma(p)} \right) \left( \frac{q_i - \mu(q)}{\sigma(q)} \right)
  \]
  - $\mu(p)$: 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:
  $$\frac{1}{d} \sum_i \left( \frac{p_i - \mu(p)}{\sigma(p)} \right) \left( \frac{q_i - \mu(q)}{\sigma(q)} \right)$$
  - $\mu(p)$: 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 $q_i$s.
Quality of a Partition

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

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

$\sum_i q_i = q(C)$
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 $q_i$s.

Sum of squared errors.

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

\[
q(C) = \sum_i \sum_{p \in C_i} \| p - \mu_i \|^2
\]

over all possible partitions of $S$ into $k$ clusters.
**k-means algorithm**

*Partition S into k clusters that minimise the sum of squared errors*

\[ q(C) = \sum_i \sum_{p \in C_i} \| p - \mu_i \|^2 \] 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 \( \mu_i \) is the centre closest to \( p \).
   - Recalculate \( \mu_i \)'s (average of points in \( C_i \)).
3. until \( \mu_i \)'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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’s don’t change.

- Initialisation: random $\mu_i$’s or “well-separated” $\mu_i$’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 \( \mu_i \) is the centre closest to \( p \).
   - Recalculate \( \mu_i \)’s (average of points in \( C_i \)).
3. until \( \mu_i \)’s don’t change.

- Initialisation: random \( \mu_i \)’s or “well-separated” \( \mu_i \)’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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’s don’t change.

- Initialisation: random $\mu_i$’s or “well-separated” $\mu_i$’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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’s don’t change.

- Each iteration takes $O(dkn)$ time.
- $q(C)$ does not increase.
- Algorithm can get stuck in a local minimum.
- Does not work particularly well in very high ($\geq 40$) dimensions.
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 \( \mu_i \) is the centre closest to \( p \).
   - Recalculate \( \mu_i \)'s (average of points in \( C_i \)).
3. until \( \mu_i \)'s don’t change.

   - Each iteration takes \( O(dkn) \) 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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’s don’t change.

- Each iteration takes $O(dkn)$ time.
- $q(C)$
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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’s don’t change.

- Each iteration takes $O(dkn)$ time.
- $q(C)$ 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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’s don’t change.

- Each iteration takes $O(dkn)$ time.
- $q(C)$ 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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’s don’t change.

- Each iteration takes $O(dkn)$ time.
- $q(C)$ 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 $\mu_i$ is the centre closest to $p$.
   - Recalculate $\mu_i$’s (average of points in $C_i$).
3. until $\mu_i$’s don’t change.

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

- \textit{k}-means and \textit{k}-median.
- 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.

![Diagram of hierarchical clustering algorithm]
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 $C_j$. 

![Diagram of hierarchical clustering process]
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 Diagram]
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

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

Computing $d_{\text{min}}, d_{\text{max}}, d_{\text{avg}}$ takes $O(n_i n_j)$ time.

Computing $d_{\text{mean}}$ takes $O(n_i + n_j)$ time.
**Measuring Distance between Clusters**

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

Computing $d_{\text{min}}$, $d_{\text{max}}$, $d_{\text{avg}}$ takes $O(n_i n_j)$ time.

Computing $d_{\text{mean}}$ takes $O(n_i + n_j)$ time.
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.
Measuring Distance between Clusters

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

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

\( d_{\text{mean}}(D_i, D_j) = d(\mu_i, \mu_j) \).
Measuring Distance between Clusters

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

Computing $d_{\text{min}}$, $d_{\text{max}}$, $d_{\text{avg}}$ takes $O(n_i n_j)$ time.

Computing $d_{\text{mean}}$ takes $O(n_i + n_j)$ time.
Measuring Distance between Clusters

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

- Computing \(d_{\text{min}}, d_{\text{max}}, d_{\text{avg}}\) takes \(O(n_i n_j)\) time.
- Computing \(d_{\text{mean}}\) 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 $O(n^2)$ inter-point distances.
   - At each iteration, compute distance between every pair of clusters: takes $O(dn^2)$ time in total.
   - There are $n$ iterations, so overall running time is $O(dnn^2) = O(dn^3)$. 
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.