Clustering

CMPUT 466/551

Nilanjan Ray

What is Clustering?

- Attach label to each observation or data points in a set
- You can say this “unsupervised classification”
- Clustering is alternatively called as “grouping”
- Intuitively, if you would want to assign same label to a data points that are “close” to each other
- Thus, clustering algorithms rely on a distance metric between data points
- Sometimes, it is said that the for clustering, the distance metric is more important than the clustering algorithm

Distances: Quantitative Variables

Some examples

Identity (absolute) error

$$d_j(x_{ij}, x_{i'j}) = I(x_{ij} \neq x_{i'j})$$

Squared distance

$$d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$$

L_q norms

$$L_{qii'} = \left[\sum_j |x_{ij} - x_{i'j}|^q\right]^{1/q}$$

Canberra distance

$$d_{ii'} = \sum_j \frac{|x_{ij} - x_{i'j}|}{|x_{ij} + x_{i'j}|}$$

Correlation

$$\rho(x_i, x_{i'}) = \frac{\sum_j (x_{ij} - \bar{x}_i)(x_{i'j} - \bar{x}_{i'})}{\sqrt{\sum_j (x_{ij} - \bar{x}_i)^2 \sum_j (x_{i'j} - \bar{x}_{i'})^2}}$$

Data point:

$x_i = [x_{i1} \ldots x_{ip}]^T$

Distances: Ordinal and Categorical Variables

- Ordinal variables can be forced to lie within (0, 1) and then a quantitative metric can be applied:

$$\frac{k-1/2}{M},\ k=1,2,\ldots,M$$

- For categorical variables, distances must be specified by user between each pair of categories.

Combining Distances

- Often weighted sum is used:

$$D(x_i, x_j) = \sum_{l=1}^{p} w_l d(x_{il}, x_{jl}), \quad \sum_{l=1}^{p} w_l = 1,\ w_l > 0.$$

Combinatorial Approach

- In how many ways can we assign *K* labels to *N* observations?

- For each such possibility, we can compute a cost. Pick up the assignment with best cost.

- Formidable number of possible assignments:

$$S(N,K) = \frac{1}{K!}\sum_{k=1}^{K}(-1)^{K-k}\binom{K}{k}k^N$$

$$S(10,4) = 34105, \text{ but } S(19,4) \simeq 10^{10}$$

(I'll post a page about the origin of this formula)

K-means Overview

- An unsupervised clustering algorithm
- "*K*" stands for number of clusters, it is typically a user input to the algorithm; some criteria can be used to automatically estimate *K*
- It is an approximation to an NP-hard combinatorial optimization problem
- *K*-means algorithm is iterative in nature
- It converges, however only a local minimum is obtained
- Works only for numerical data
- Easy to implement

K-means: Setup

- $x_1, \ldots, x_N$ are data points or vectors of observations
- Each observation (vector x_i) will be assigned to one and only one cluster
- $C(i)$ denotes cluster number for the i^{th} observation
- Dissimilarity measure: Euclidean distance metric
- *K*-means minimizes within-cluster point scatter:

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(j)=k}\left\|x_i - x_j\right\|^2 = \sum_{k=1}^{K} N_k \sum_{C(i)=k}\left\|x_i - m_k\right\|^2$$

(Exercise)

where

m_k is the mean vector of the k^{th} cluster

N_k is the number of observations in k^{th} cluster

Within and Between Cluster Criteria

Let's consider total point scatter for a set of *N* data points:

$$T = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} d(x_i, x_j)$$

Distance between two points

T can be re-written as:

$$T = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\left(\sum_{C(j)=k} d(x_i, x_j) + \sum_{C(j)\neq k} d(x_i, x_j)\right)$$
$$= W(C) + B(C)$$

Where,

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(j)=k} d(x_i, x_j)$$

Within cluster scatter

$$B(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(j)\neq k} d(x_i, x_j)$$

Between cluster scatter

If *d* is square Euclidean distance, then

$$W(C) = \sum_{k=1}^{K} N_k \sum_{C(i)=k} \|x_i - m_k\|^2$$

and $$B(C) = \sum_{k=1}^{K} N_k \|m_k - m\|^2$$

Ex.

Grand mean

Minimizing *W*(*C*) is equivalent to maximizing *B*(*C*)

K-means Algorithm

- For a given cluster assignment *C* of the data points, compute the cluster means m_k:

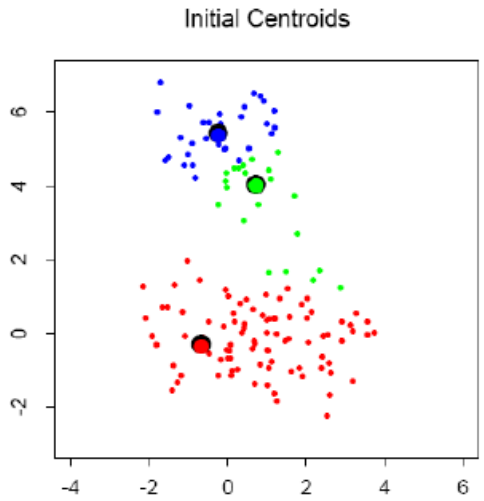
$$m_k = \frac{\sum_{i:C(i)=k} x_i}{N_k},\ k = 1, \ldots, K.$$

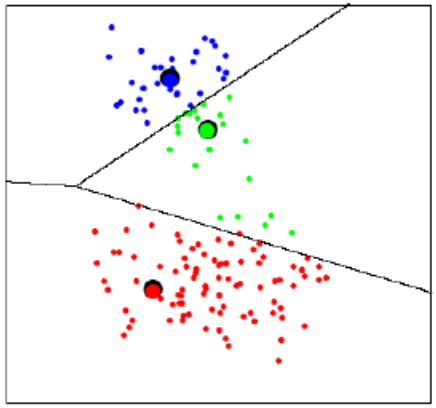
- For a current set of cluster means, assign each observation as:

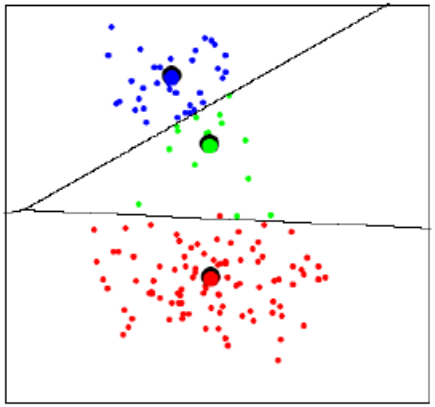
$$C(i) = \arg\min_{1 \le k \le K} \left\| x_i - m_k \right\|^2,\ i = 1, \ldots, N$$

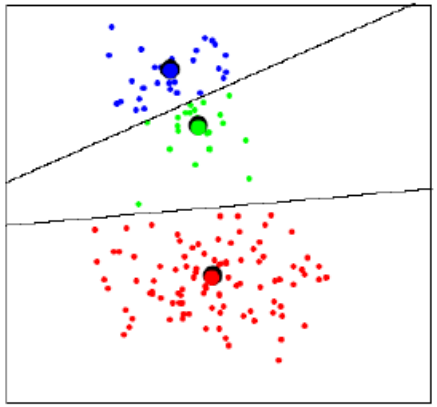
- Iterate above two steps until convergence

K-means clustering example

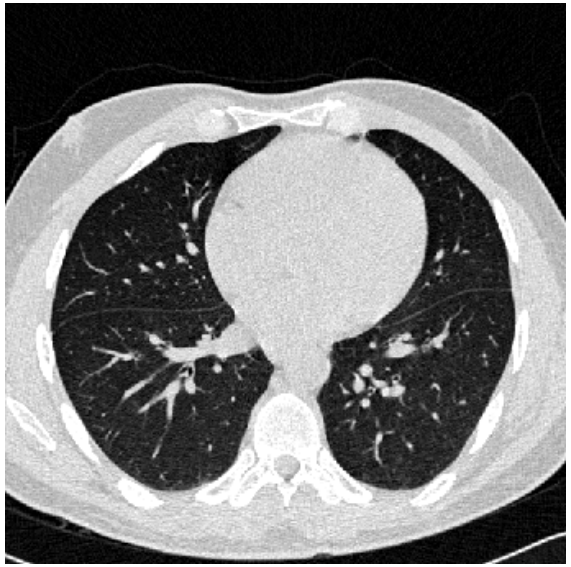








K-means Image Segmentation



An image (*I*)

Three-cluster image (*J*) on gray values of *I*

Matlab code:

```
I = double(imread('…'));

J = reshape(kmeans(I(:),3),size(I));
```

Note that *K*-means result is "noisy"

K-means: summary

- Algorithmically, very simple to implement
- *K*-means converges, but it finds a local minimum of the cost function
- Works only for numerical observations
- *K* is a user input; alternatively BIC (Bayesian information criterion) or MDL (minimum description length) can be used to estimate *K*
- Outliers can considerable trouble to *K*-means

K-medoids Clustering

- *K*-means is appropriate when we can work with Euclidean distances
- Thus, *K*-means can work only with numerical, quantitative variable types
- Euclidean distances do not work well in at least two situations
  - Some variables are categorical
  - Outliers can be potential threats
- A general version of *K*-means algorithm called *K*-medoids can work with any distance measure
- *K*-medoids clustering is computationally more intensive

K-medoids Algorithm

- Step 1: For a given cluster assignment *C*, find the observation in the cluster minimizing the total distance to other points in that cluster:

$$i_k^* = \arg\min_{\{i:C(i)=k\}} \sum_{C(j)=k} d(x_i, x_j).$$

- Step 2: Assign $m_k = x_{i_k^*},\ k = 1,2,\ldots,K$

- Step 3: Given a set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest (current) cluster center:

$$C(i) = \arg\min_{1\le k\le K} d(x_i, m_k),\ i = 1,\ldots,N$$

- Iterate steps 1 to 3

K-medoids Summary

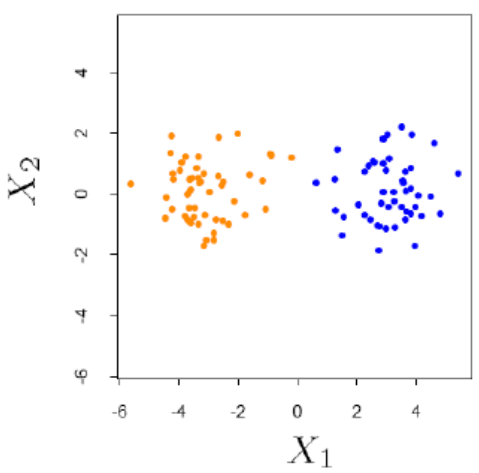
- Generalized *K*-means
- Computationally much costlier that *K*-means
- Apply when dealing with categorical data
- Apply when data points are not available, but only pair-wise distances are available
- Converges to local minimum

Choice of K?

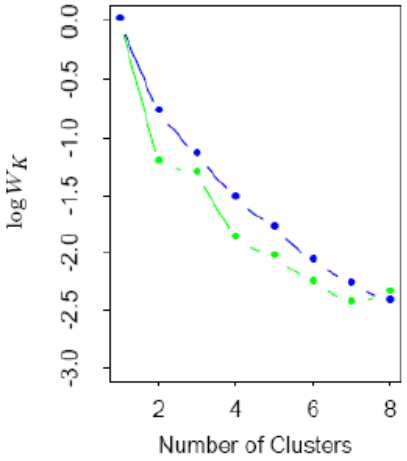
- Can $W_K(C)$, *i.e.*, the within cluster distance as a function of K serve as any indicator?
- Note that $W_K(C)$ decreases monotonically with increasing K. That is the within cluster scatter decreases with increasing centroids.
- Instead look for gap statistics (successive difference between $W_K(C)$):

$$\{W_K - W_{K+1} : K < K^*\} >> \{W_K - W_{K+1} : K \geq K^*\}$$

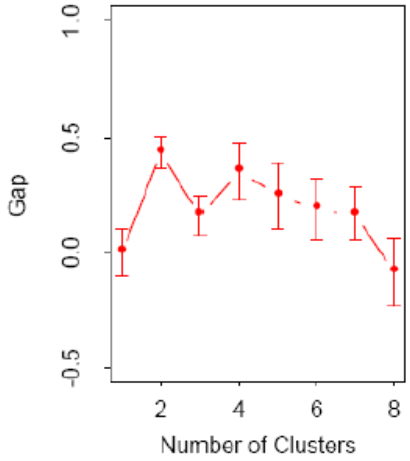
Choice of *K*…



Data points simulated from two pdfs



Log(W_K) curve



Gap curve

This is essentially a visual heuristic

Vector Quantization

- A codebook (a set of centroids/codewords):

$$\{m_1, m_2, \ldots, m_K\}$$

- A quantization function:

$$q(x_i) = m_k$$

Often, the nearest-neighbor function

- *K*-means can be used to construct the codebook

Image Compression by VQ

8 bits/pixel

1.9 bits/pixel,
using 200 codewords

0.5 bits/pixel,
using 4 codewords

Otsu's Image Thresholding Method

- Based on the clustering idea: Find the threshold that *minimizes the weighted within-cluster point scatter.*
- This turns out to be the same as *maximizing the between-class scatter.*
- Operates directly on the gray level histogram [*e.g.* 256 numbers, P(i)], so it's fast (once the histogram is computed).

Otsu's Method…

- Histogram (and the image) are *bimodal.*
- No use of *spatial coherence,* nor any other notion of object structure.
- Assumes uniform illumination (implicitly), so the bimodal brightness behavior arises from object appearance differences only.

The *weighted within-class variance* is:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

Where the class probabilities are estimated as:

$$q_1(t) = \sum_{i=1}^{t} P(i) \qquad q_2(t) = \sum_{i=t+1}^{I} P(i)$$

And the class means are given by:

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \qquad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

Finally, the individual class variances are:

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

Now, we could actually stop here. All we need to do is just run through the full range of *t* values [1, 256] and pick the value that minimizes $\sigma_w^2(t)$.

But the relationship between the within-class and between-class variances can be exploited to generate a recursion relation that permits a much faster calculation.

Finally...

Initialization... $q_1(1) = P(1)\,;\ \mu_1(0) = 0$

Recursion...

$$q_1(t+1) = q_1(t) + P(t+1)$$

$$\mu_1(t+1) = \frac{q_1(t)\,\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)}$$

$$\mu_2(t+1) = \frac{\mu - q_1(t+1)\,\mu_1(t+1)}{1 - q_1(t+1)}$$

After some algebra, we can express the total variance as...

$$\sigma^2 = \underbrace{\sigma_w^2(t)}_{\text{Within-class, from before}} + \underbrace{q_1(t)[1-q_1(t)][\mu_1(t)-\mu_2(t)]^2}_{\text{Between-class, } \sigma_B^2(t)}$$

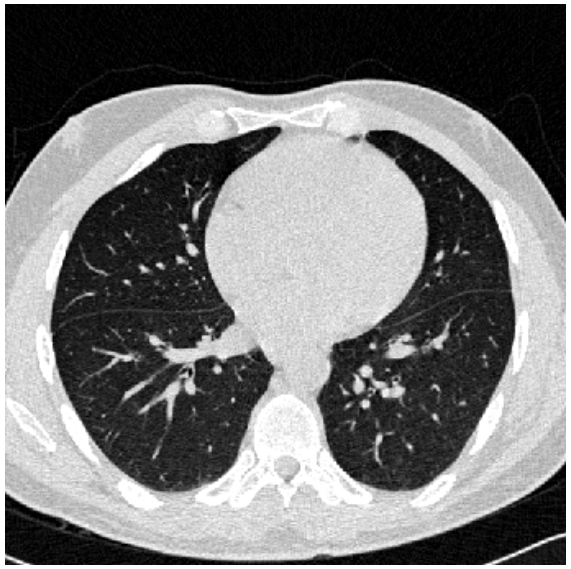
Since the total is constant and independent of *t*, the effect of changing the threshold is merely to move the contributions of the two terms back and forth.

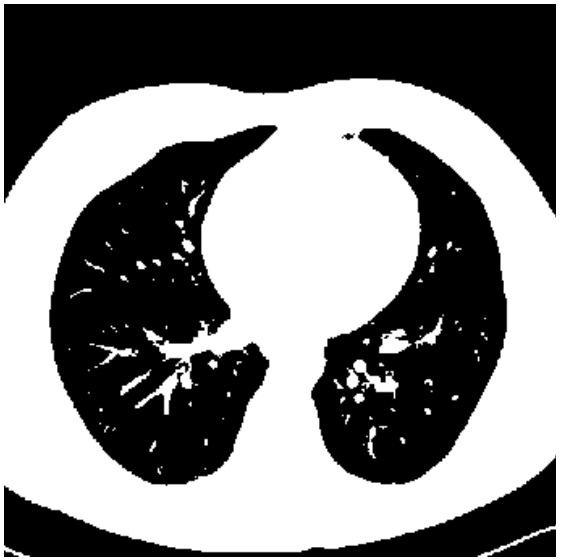
So, *minimizing the within-class variance is the same as maximizing the between-class variance*.

The nice thing about this is that we can compute the quantities in $\sigma_B^2(t)$ *recursively* as we run through the range of *t* values.

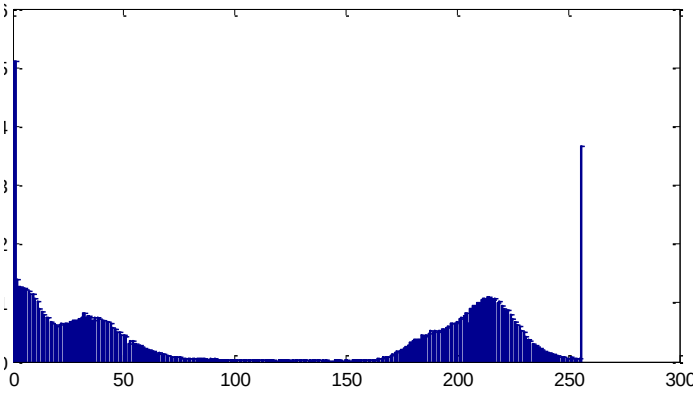
Result of Otsu's Algorithm

An image





Binary image
by Otsu's method



Gray level histogram

Matlab code:

```
I = double(imread('…'));

I = (I-min(I(:)))/(max(I(:))-min(I(:)));

J = I>graythresh(I);
```

Hierarchical Clustering

- Two types: (1) agglomerative (bottom up), (2) divisive (top down)
- Agglomerative: two groups are merged if distance between them is less than a threshold
- Divisive: one group is split into two if intergroup distance more than a threshold
- Can be expressed by an excellent graphical representation called “dendogram”, when the process is monotonic: dissimilarity between merged clusters is increasing. Agglomerative clustering possesses this property. Not all divisive methods possess this monotonicity.
- Heights of nodes in a dendogram are proportional to the threshold value that produced them.

An Example Hierarchical Clustering

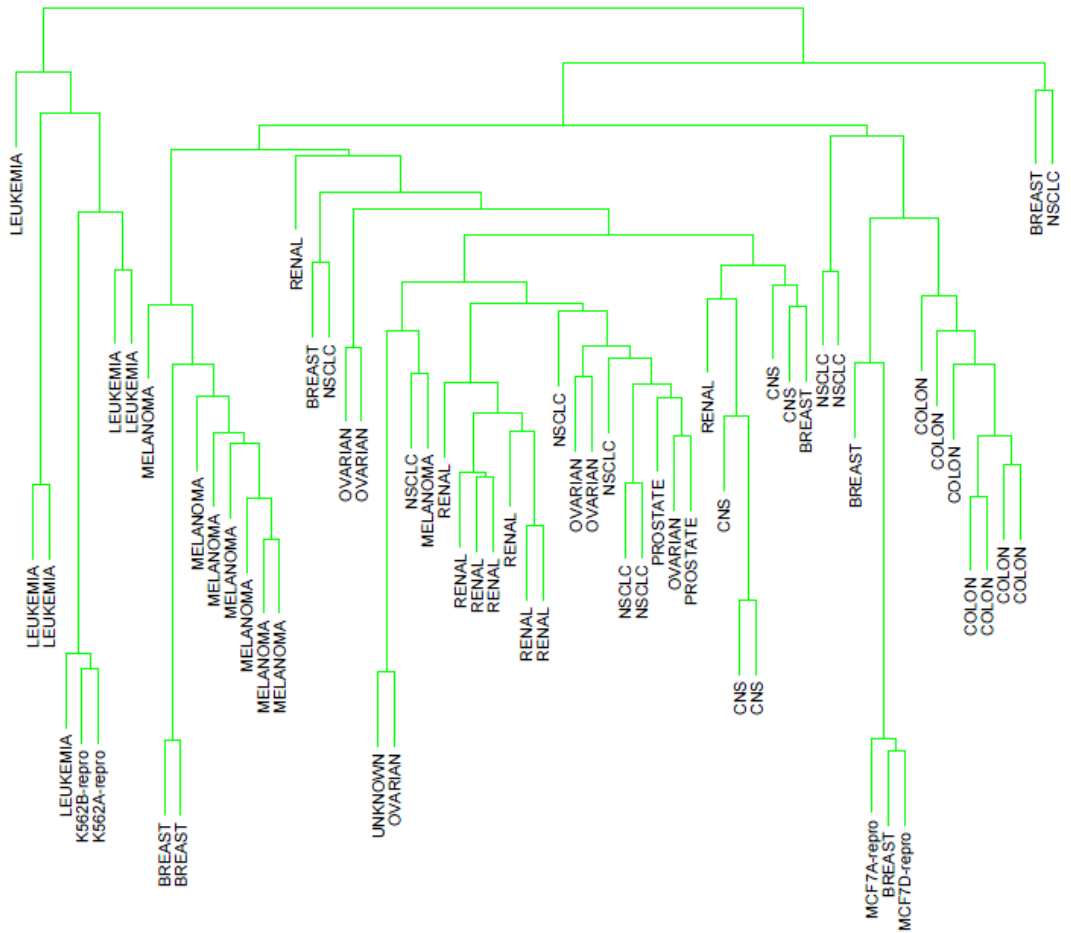


Figure 14.12: *Dendrogram from agglomerative hierarchical clustering with average linkage to the human tumor microarray data.*

Linkage Functions

Linkage functions computes the dissimilarity between two groups of data points:

Single linkage (minimum distance between two groups): $d_{SL}(G,H) = \min_{\substack{i \in G \\ j \in H}} d_{ij}$

Complete linkage (maximum distance between two groups): $d_{CL}(G,H) = \max_{\substack{i \in G \\ j \in H}} d_{ij}$

Group average (average distance between two groups):

$$d_{GA}(G,H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{j \in H} d_{ij}$$

Linkage Functions…

- SL considers only a single pair of data points; if this pair is close enough then action is taken. So, SL can form a “chain” by combining relatively far apart data points.
- SL often violates the compactness property of a cluster. SL can produce clusters with large diameters (D_G).

$$D_G = \max_{i \in G,\, j \in G} d_{ij}$$

- CL is just the opposite of SL; it produces many clusters with small diameters.
- CL can violate “closeness” property- two close data points may be assigned to different clusters.
- GA is a compromise between SL and CL

Different Dendograms

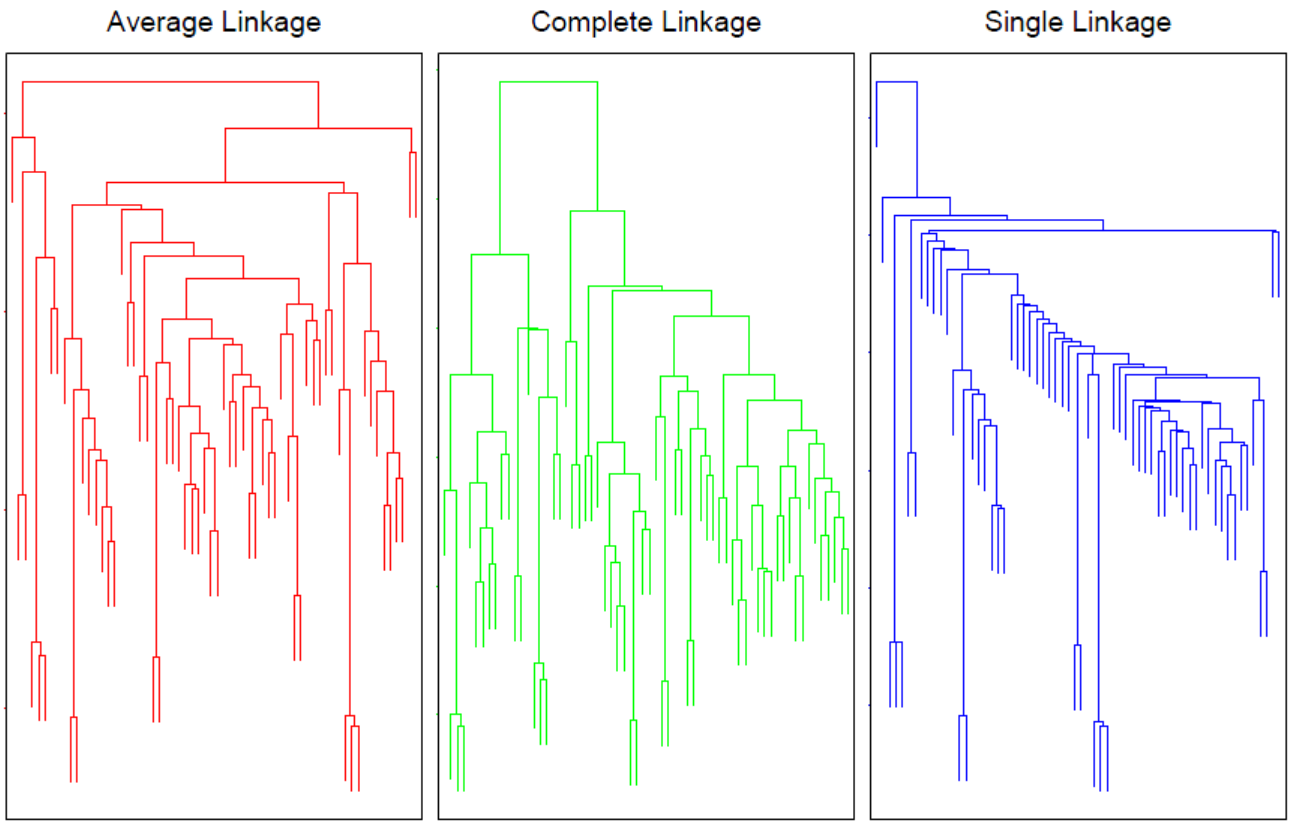
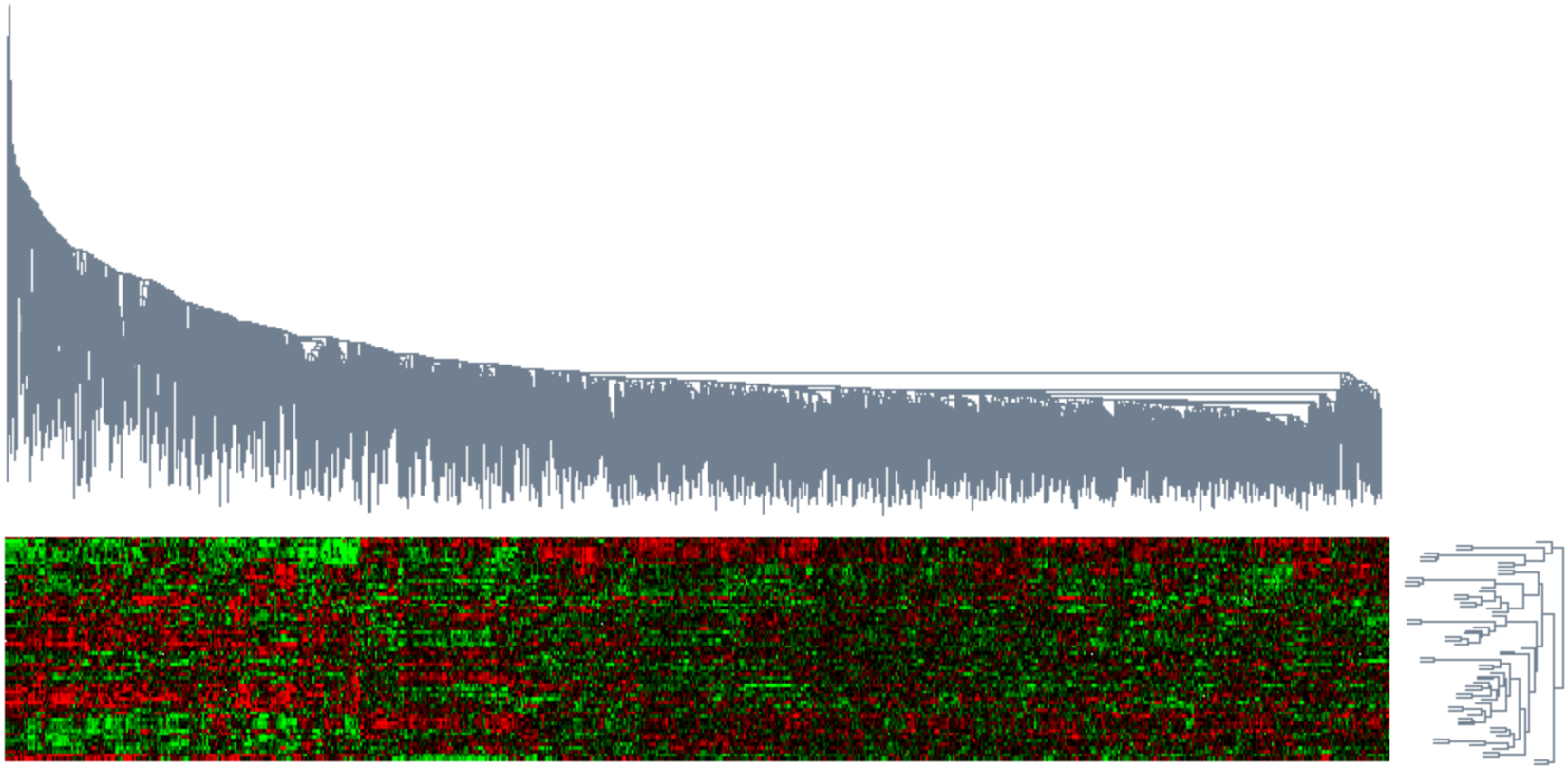


Figure 14.13: *Dendrograms from agglomerative hierarchical clustering of human tumor microarray data.*

Hierarchical Clustering on Microarray Data



Hierarchical Clustering Matlab Demo