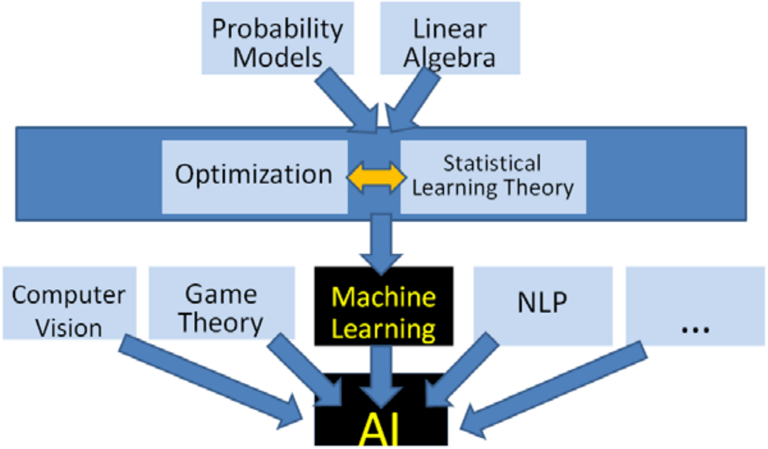
MSDS 422: Practical Machine Learning

Unsupervised Learning



NORTHWESTERN
UNIVERSITY

Dr. Nathan Bastian

References

- *An Introduction to Statistical Learning, with Applications in R* (2013), by G. James, D. Witten, T. Hastie, and R. Tibshirani.

- *The Elements of Statistical Learning* (2009), by T. Hastie, R. Tibshirani, and J. Friedman.

- *Machine Learning: A Probabilistic Perspective* (2012), by K. Murphy

Unsupervised Learning

- Most of this course has focused on *supervised learning* methods such as regression and classification.

- Remember that in supervised learning we observe both a set of features as well as a response (or outcome variable).

- In *unsupervised learning*, however, we only observe features. We are not interested in prediction because we do not have an associated response variable.

- The goal is to discover interesting things about the measurements.

Unsupervised Learning (cont'd)

- Here, we discuss two methods:
  - **Principal Components Analysis**: a tool used for data visualization or data pre-processing before supervised techniques are applied.
  - **Clustering:** a broad class of methods for discovering unknown subgroups in data.
- Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis (such as prediction of a response).
- Applications:
  - Subgroups of breast cancer patients groups by their gene expression measurements.
  - Groups of shoppers characterized by their browsing and purchase histories.
  - Movies grouped by the ratings assigned by movie viewers.

Principal Components Analysis

- The input matrix **X** of dimension *N* x *p*:

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,p} \\ x_{2,1} & x_{2,2} & \ldots & x_{2,p} \\ \ldots & \ldots & \ldots & \ldots \\ x_{N,1} & x_{N,2} & \ldots & x_{N,p} \end{pmatrix}$$

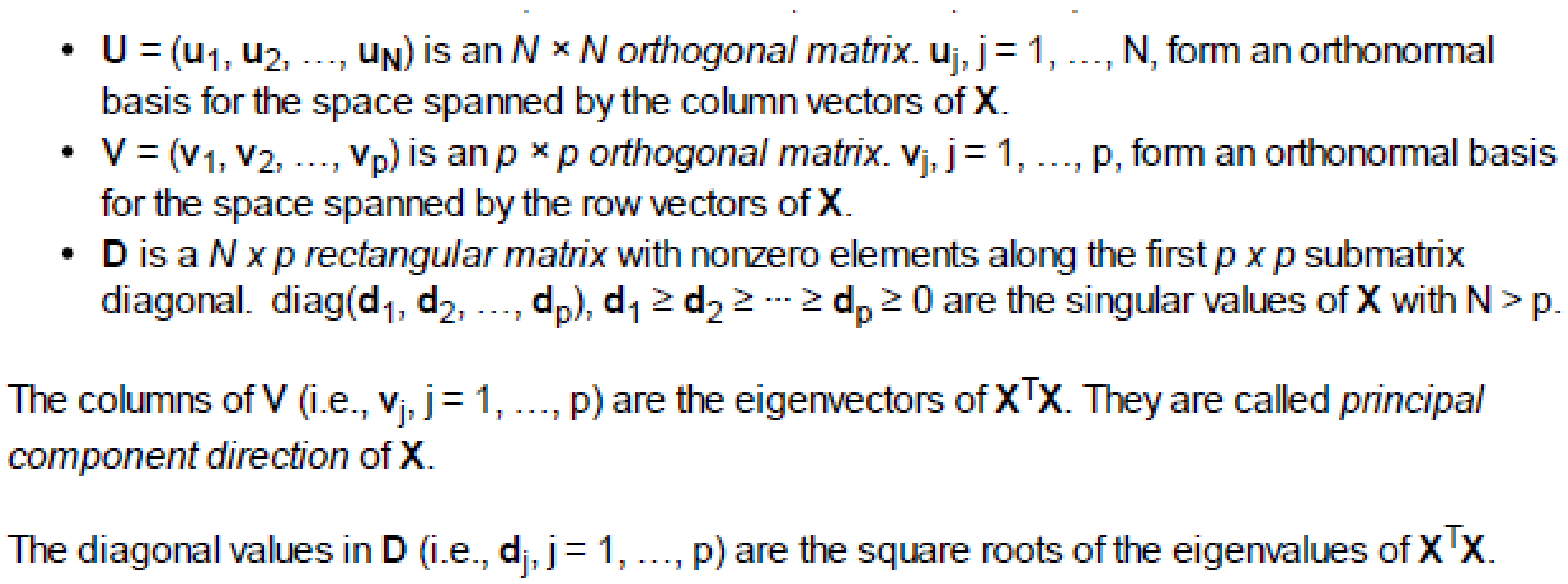
- The rows represent cases and the columns represents variables.

- Assume that **X** is mean-centered (i.e., the estimated mean is subtracted from each column).

- Singular value decomposition (SVD) is a critical part of principal components analysis (PCA).

Principal Components Analysis

- The SVD of the *N* x *p* matrix **X** has the form **X = UDV**^{T}



- $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N)$ is an $N \times N$ *orthogonal matrix*. $\mathbf{u}_j$, $j = 1, \ldots, N$, form an orthonormal basis for the space spanned by the column vectors of **X**.
- $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p)$ is an $p \times p$ *orthogonal matrix*. $\mathbf{v}_j$, $j = 1, \ldots, p$, form an orthonormal basis for the space spanned by the row vectors of **X**.
- **D** is a *N x p rectangular matrix* with nonzero elements along the first *p x p* submatrix diagonal. $\text{diag}(\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_p)$, $\mathbf{d}_1 \geq \mathbf{d}_2 \geq \cdots \geq \mathbf{d}_p \geq 0$ are the singular values of **X** with $N > p$.

The columns of V (i.e., $\mathbf{v}_j$, $j = 1, \ldots, p$) are the eigenvectors of $\mathbf{X}^T\mathbf{X}$. They are called *principal component direction* of **X**.

The diagonal values in **D** (i.e., $\mathbf{d}_j$, $j = 1, \ldots, p$) are the square roots of the eigenvalues of $\mathbf{X}^T\mathbf{X}$.

Principal Components Analysis

- The **two-dimensional plane** can be shown to be spanned by
  - The linear combination of the variables that has maximum sample variance,
  - The linear combination that has maximum variance subject to being uncorrelated with the first linear combination.

- It can be extended to the **k-dimensional projection**.

- We can take the process further, seeking additional linear combinations that maximize the variance subject to being uncorrelated with all those already selected.

- PCA is the main method used for linear dimension reduction.

Principal Components Analysis

- Let’s say you start with 10 dimensions and you want to reduce to two dimensions.

- The idea of PCA is to use two directions that capture the *variation* in the data as much as possible.

- The sample covariance matrix of **X** is given as: $S = \mathbf{X}^T\mathbf{X}/N$

- If you do the Eigen decomposition of $\mathbf{X}^T\mathbf{X}$:

$$\begin{aligned}\mathbf{X}^T\mathbf{X} &= (\mathbf{U}\mathbf{D}\mathbf{V}^T)^T(\mathbf{U}\mathbf{D}\mathbf{V}^T)\\ &= \mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{U}\mathbf{D}\mathbf{V}^T\\ &= \mathbf{V}\mathbf{D}^2\mathbf{V}^T\end{aligned}$$

Principal Components Analysis

- It turns out that if you have the SVD then you already have the Eigen value decomposition for $\mathbf{X}^T\mathbf{X}$.

- The $\mathbf{D}^2$ is the diagonal matrix. This simply means to square every element on the diagonal.

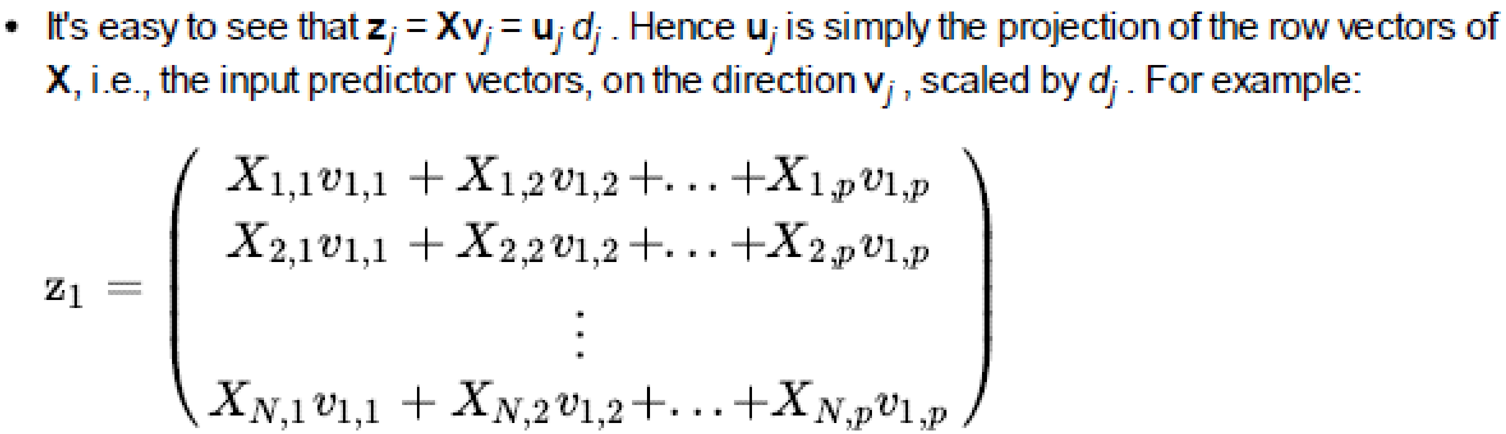
- Also, we should point out that we can show using linear algebra that $\mathbf{X}^T\mathbf{X}$ is a semi-positive definite matrix. This means that all of the eigenvalues are guaranteed to be nonnegative.

- The eigen values are in matrix $\mathbf{D}^2$. Since these values are squared, every diagonal element is nonnegative.

Principal Components Analysis

- The eigenvectors of $\mathbf{X}^T\mathbf{X}$, v_j, can be obtained either by doing an Eigen decomposition of $\mathbf{X}^T\mathbf{X}$, or by doing a SVD from $\mathbf{X}$.

- These v_j's are called principal component directions of $\mathbf{X}$. If you project $\mathbf{X}$ onto the principal components directions you get the principal components.



- It's easy to see that $\mathbf{z}_j = \mathbf{X}\mathbf{v}_j = \mathbf{u}_j\, d_j$. Hence $\mathbf{u}_j$ is simply the projection of the row vectors of $\mathbf{X}$, i.e., the input predictor vectors, on the direction $\mathbf{v}_j$, scaled by d_j . For example:

$$\mathbf{z}_1 = \begin{pmatrix} X_{1,1}v_{1,1} + X_{1,2}v_{1,2} + \ldots + X_{1,p}v_{1,p} \\ X_{2,1}v_{1,1} + X_{2,2}v_{1,2} + \ldots + X_{2,p}v_{1,p} \\ \vdots \\ X_{N,1}v_{1,1} + X_{N,2}v_{1,2} + \ldots + X_{N,p}v_{1,p} \end{pmatrix}$$

Principal Components Analysis

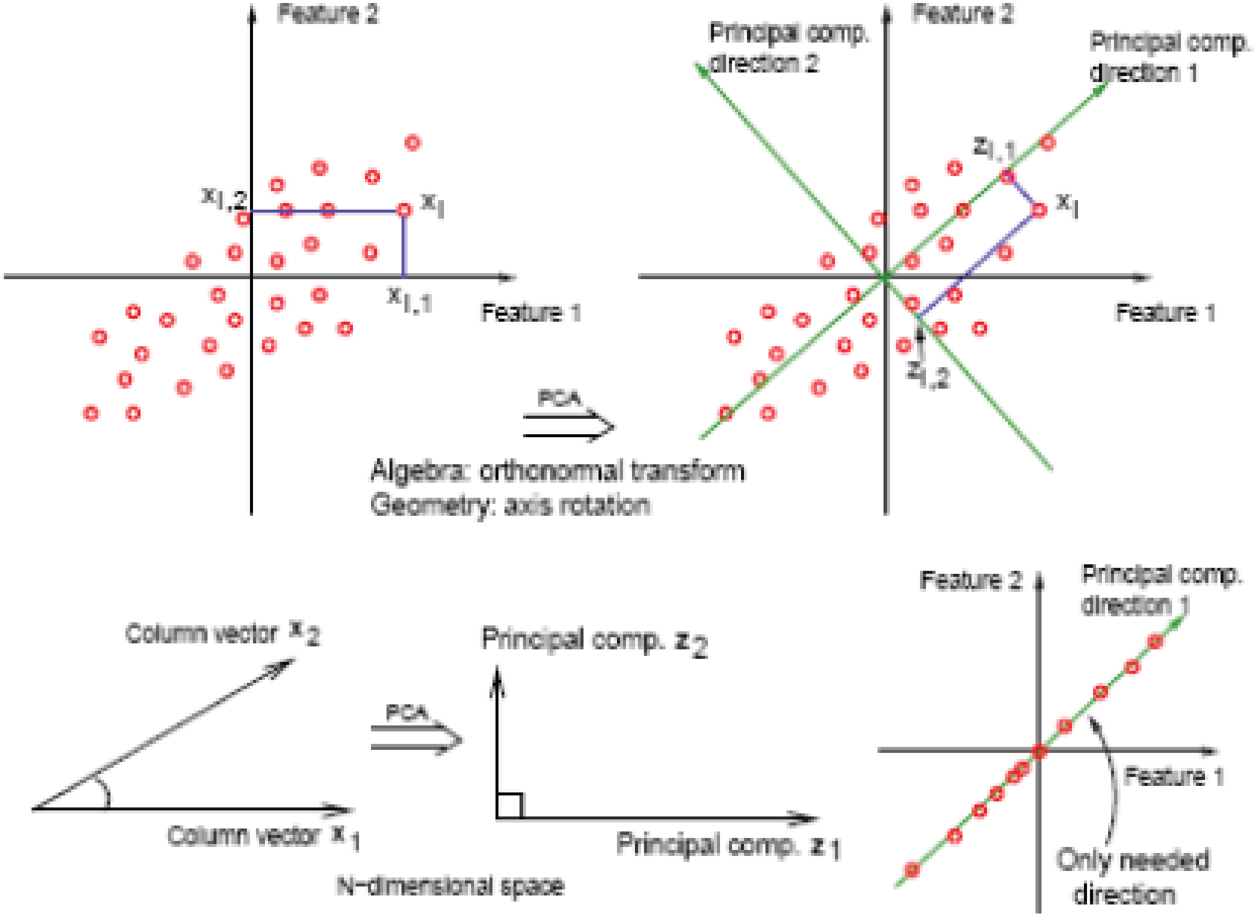
- The principal components of **X** are $\mathbf{z}_j = d_j\mathbf{u}_j$, $j = 1,\ldots,p$

- The first principal component of **X**, $\mathbf{z}_1$, has the largest sample variance amongst all normalized linear combinations of the columns of **X**.

$$Var(\mathbf{z}_1) = d_1^2/N$$

- Subsequent principal components $\mathbf{z}_j$ have maximum variance d_j^2/N, subject to being orthogonal to the earlier ones.

Principal Components Analysis



Principal Components Analysis

- Why are we interested in this? Consider an extreme case, where your data all lie in one direction.

- Although two features represent the data, we can reduce the dimension of the dataset to one using a single linear combination of the features (as given by the first principal component).

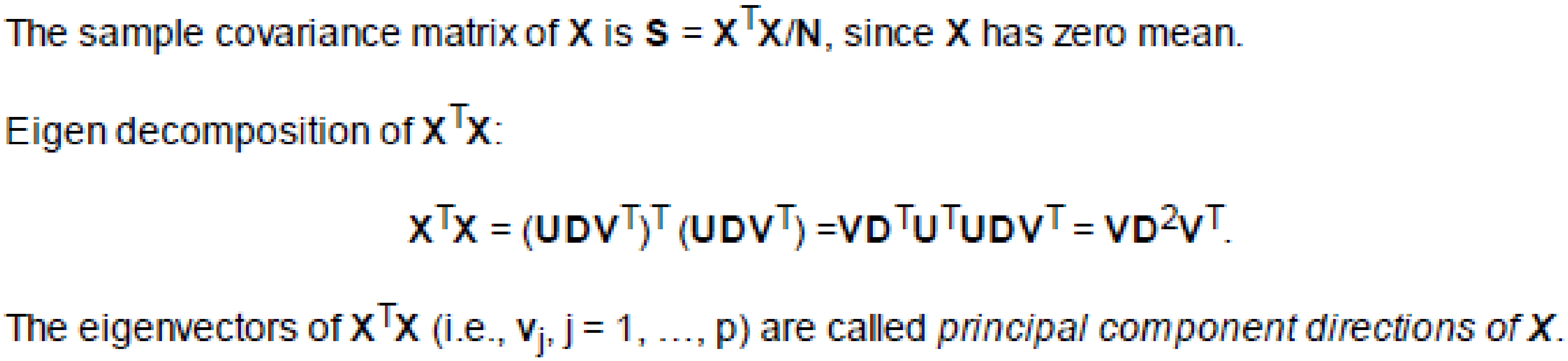
- Just to summarize the way you do dimension reduction. First you have **X**, and you remove the means. On **X** you do a singular value decomposition and obtain matrices **U**, **D**, and **V**.

- Then you call every column vector in **V**, v_j, where $j = 1, \ldots, p$.

Principal Components Analysis (cont'd)

- The v_j is called the principal components direction.

- PCA is used to reduce the dimensionality of a data set, either to ease interpretation or as a way to avoid overfitting and to prepare for subsequent analysis.



The sample covariance matrix of **X** is **S** = $\mathbf{X}^T\mathbf{X}/N$, since **X** has zero mean.

Eigen decomposition of $\mathbf{X}^T\mathbf{X}$:

$$\mathbf{X}^T\mathbf{X} = (\mathbf{UDV}^T)^T(\mathbf{UDV}^T) = \mathbf{VD}^T\mathbf{U}^T\mathbf{UDV}^T = \mathbf{VD}^2\mathbf{V}^T.$$

The eigenvectors of $\mathbf{X}^T\mathbf{X}$ (i.e., v_j, j = 1, …, p) are called *principal component directions of* ***X***.

Principal Components Analysis (cont'd)

The *first principal component direction* $\mathbf{v}_1$ has the following properties that

- $\mathbf{v}_1$ is the eigenvector associated with the largest eigenvalue, $\mathbf{d}_1^2$, of $\mathbf{X}^T\mathbf{X}$.
- $\mathbf{z}_1 = \mathbf{X}\mathbf{v}_1$ has *the largest sample variance* amongst all normalized linear combinations of the columns of X.
- $\mathbf{z}_1$ is called *the first principal component of* ***X***. And, we have $Var(\mathbf{z}_1) = \mathbf{d}_1^2 / N$.

The *second principal component direction* $\mathbf{v}_2$ (the direction orthogonal to the first component that has the largest projected variance) is the eigenvector corresponding to the second largest eigenvalue, $\mathbf{d}_2^2$, of $\mathbf{X}^T\mathbf{X}$, and so on. (The eigenvector for the kth largest eigenvalue corresponds to the kth principal component direction $\mathbf{v}_k$.)

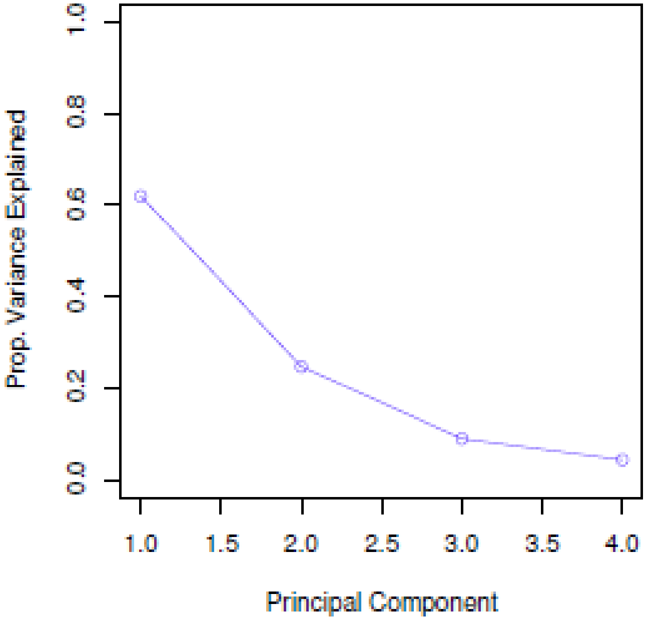
The kth principal component of **X**, $\mathbf{z}_k$, has maximum variance $\mathbf{d}_k^2$ / N, subject to being orthogonal to the earlier ones.

Principal Components Analysis (cont'd)

- The **scree plot** is a useful visual aid that shows the amount of variance explained by each consecutive eigenvalue.

- The choice of how many components to extract is fairly arbitrary.

- When conducting PCA prior to further analysis, it is risky to choose too small a number of components, which may fail to explain enough of the variability in the data.



Clustering

- Clustering refers to a very broad set of techniques for finding subgroups, or clusters, in a data set.

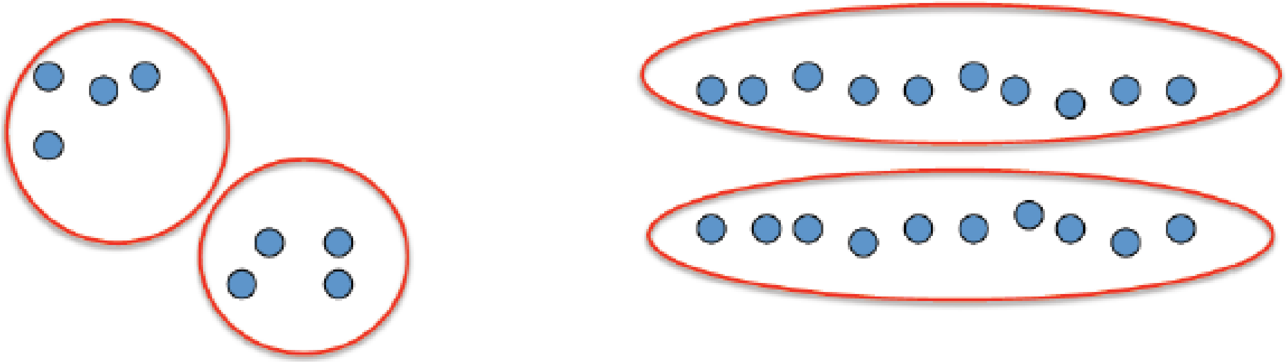
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other.

- *Clustering* looks for homogeneous subgroups among the observations.

- In *K-means clustering*, we seek to partition the observations into a pre-specified number of clusters. In *hierarchical clustering*, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations.

Clustering (cont'd)

- Basic idea: group together similar instances
- Example: 2D point patterns



- What could "similar" mean?
  - One option: small Euclidean distance (squared)

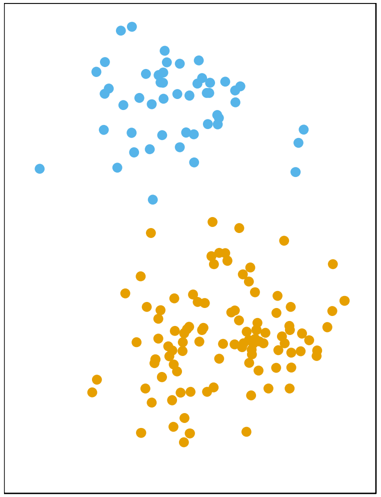
$$\mathrm{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$$

  - Clustering results are crucially dependent on the measure of similarity (or distance) between "points" to be clustered

K-Means Clustering

- To perform K-means clustering, one must first specify the desired number of clusters K.

- Then, the K-means algorithm will assign each observation to exactly one of the K clusters.



K-Means Clustering (cont’d)

- We would like to partition the data set into K clusters, where each observation belongs to at least one of the K clusters.

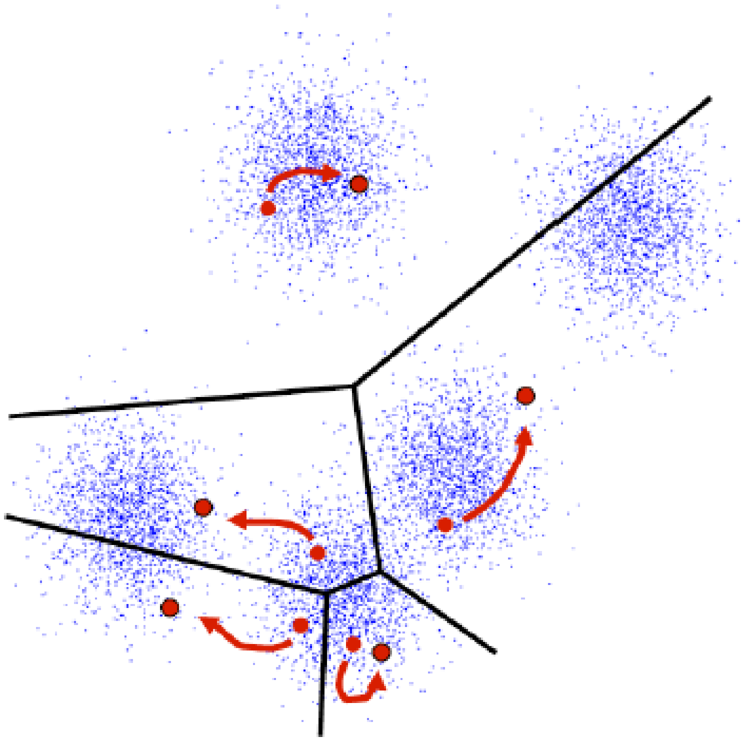
- The clusters are non-overlapping, i.e. no observation belongs to more than one cluster.

- The **objective** is to have a minimal “within-cluster-variation”, i.e. the elements within a cluster should be as similar as possible.

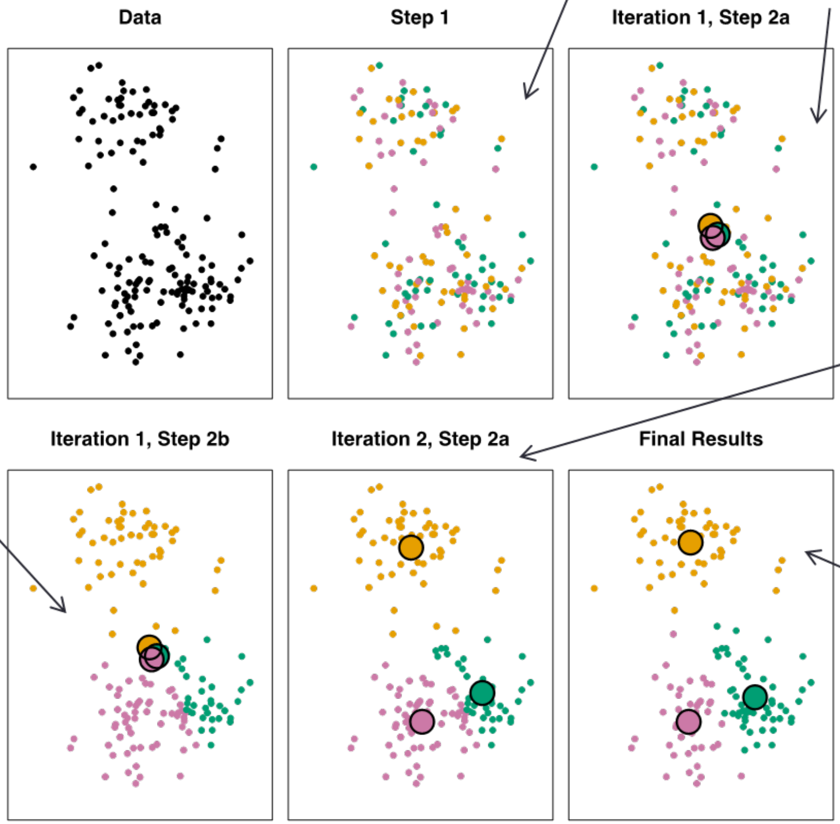
- One way of achieving this is to minimize the sum of all the pair-wise squared Euclidean distances between the observations in each cluster.

K-Means Clustering (cont’d)

- An iterative clustering algorithm
  - Initialize: Pick *K* random points as cluster centers
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points’ assignments change



K-Means Clustering (cont’d)



Random Assignment of points

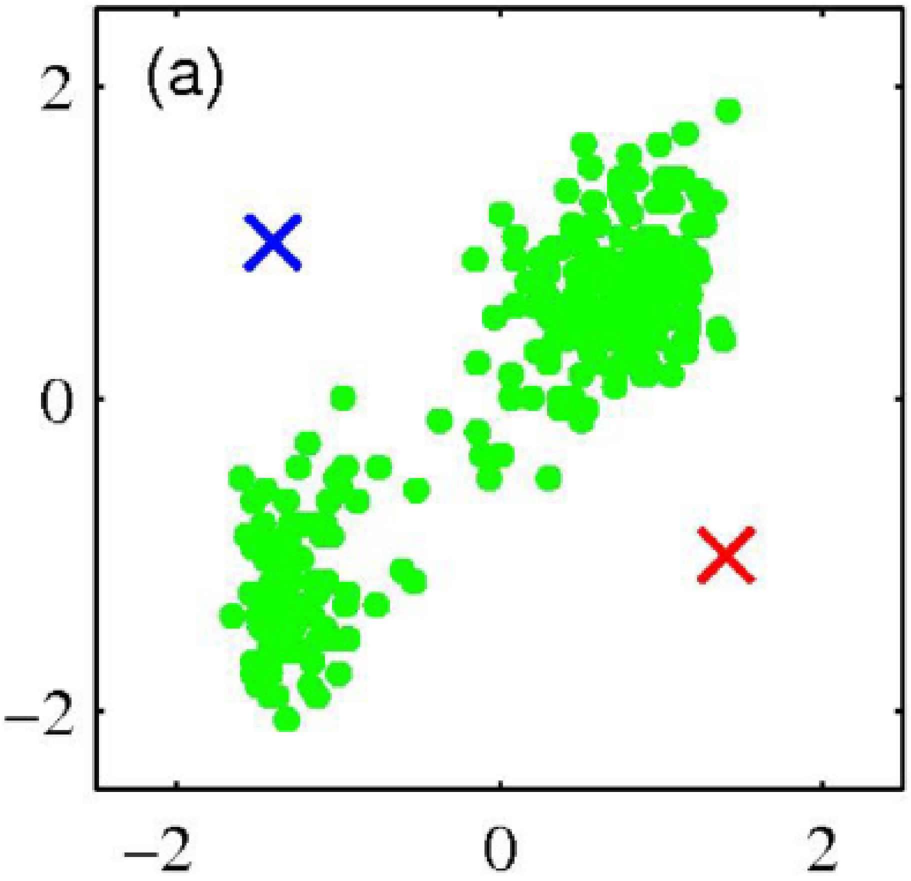
Compute cluster centers from initial assignments

Computer new cluster centers

Assign points to closest cluster center

Now there is no further change so stop

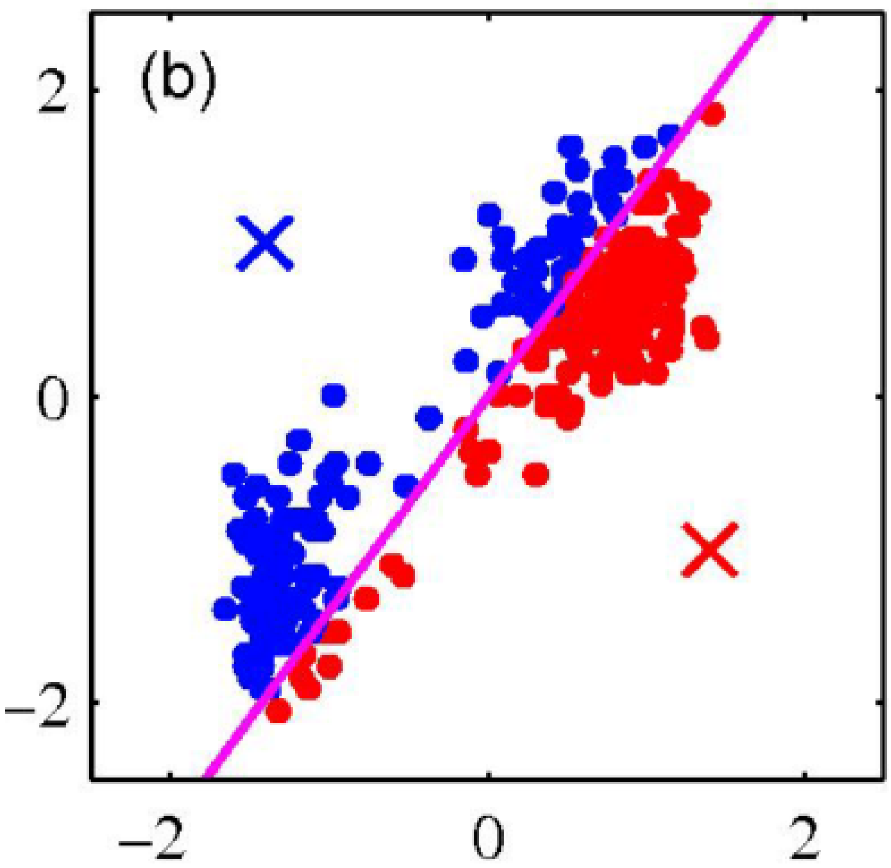
K-Means Clustering: Example



- Pick *K* random points as cluster centers (means)

Shown here for *K*=2

K-Means Clustering: Example

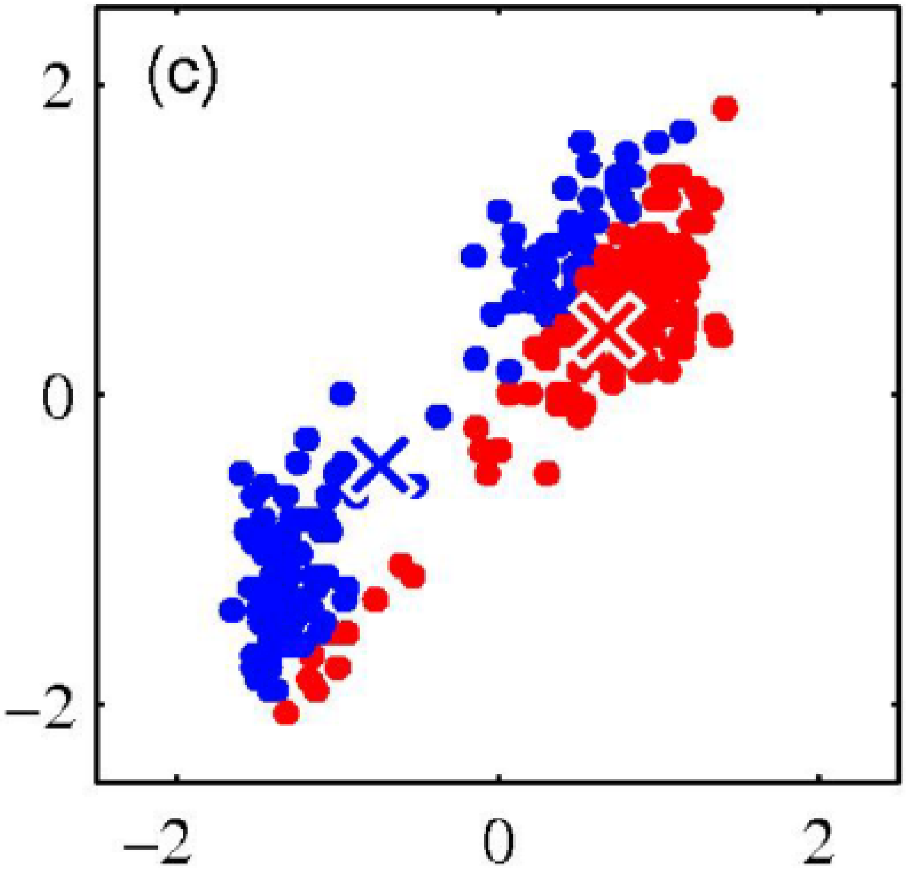


Iterative Step 1

- Assign data points to closest cluster center

Assignment step

K-Means Clustering: Example

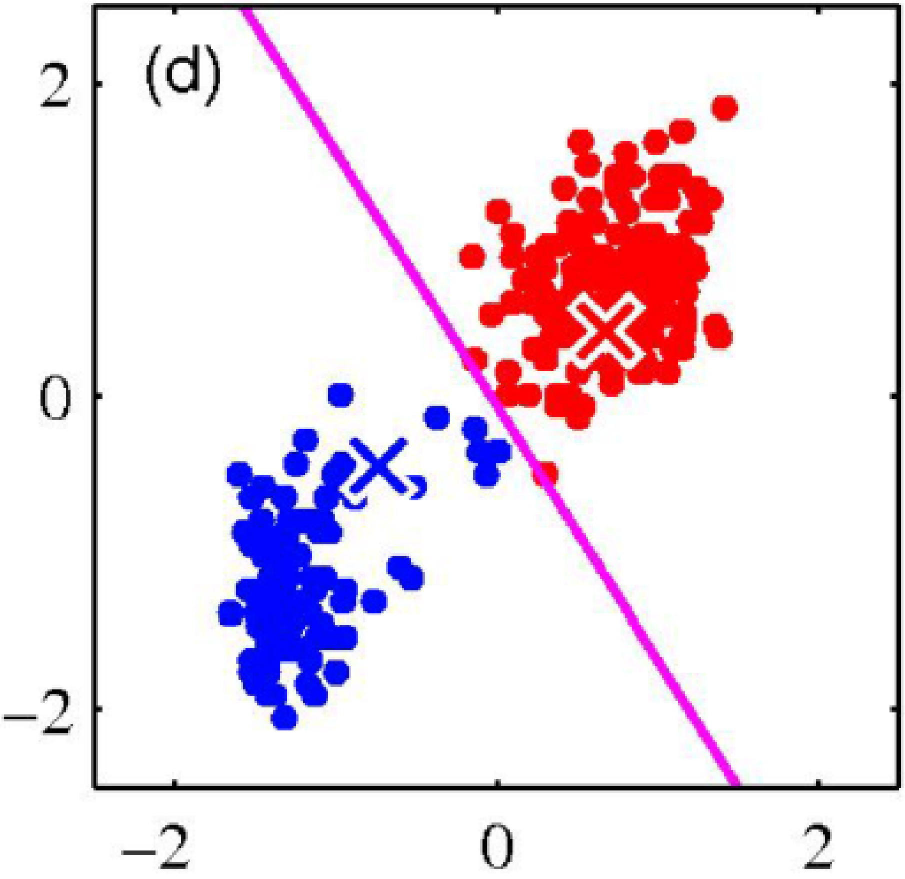


Iterative Step 2

- Change the cluster center to the average of the assigned points

Adjustment step

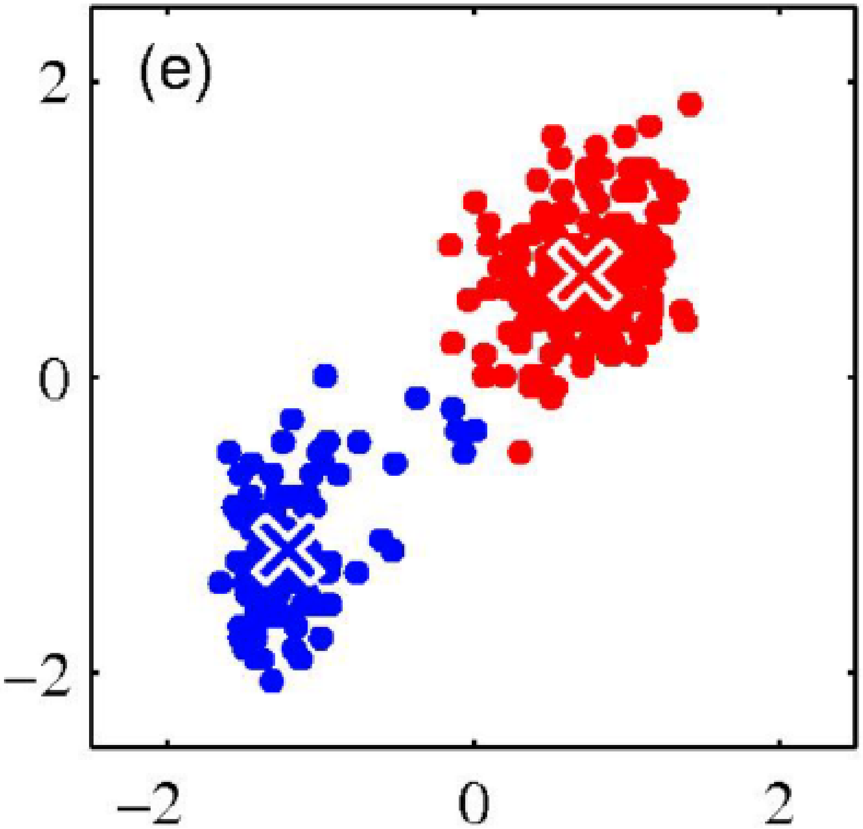
K-Means Clustering: Example



- Repeat until convergence

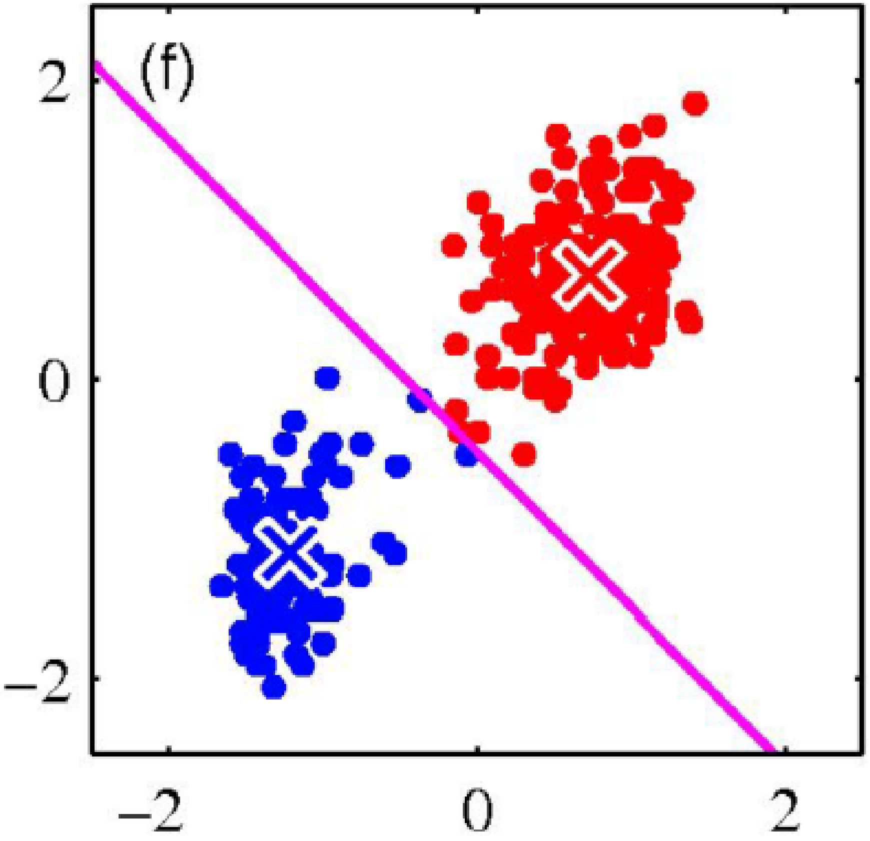
Assignment step

K-Means Clustering: Example



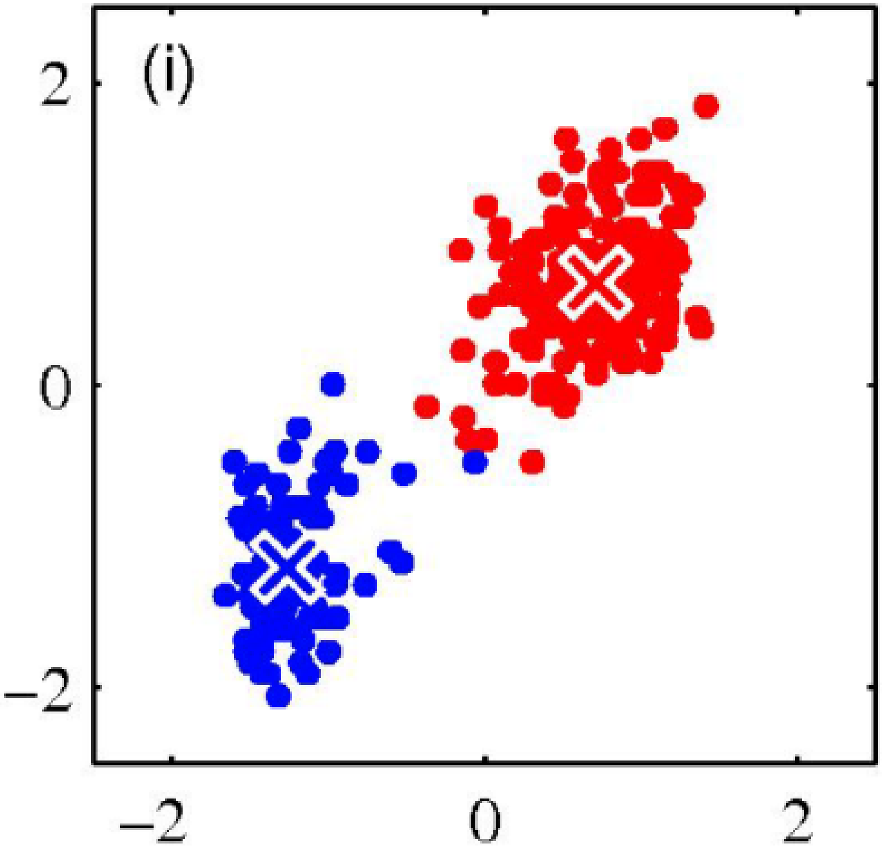
Adjustment step

K-Means Clustering: Example



Assignment step

K-Means Clustering: Example



Stop:
No more adjustment

K-Means Clustering (cont’d)

- An iterative clustering algorithm
  - Initialize: Pick *K* random points as cluster centers

***Randomly** choose k observations from the dataset and uses these as the initial means*

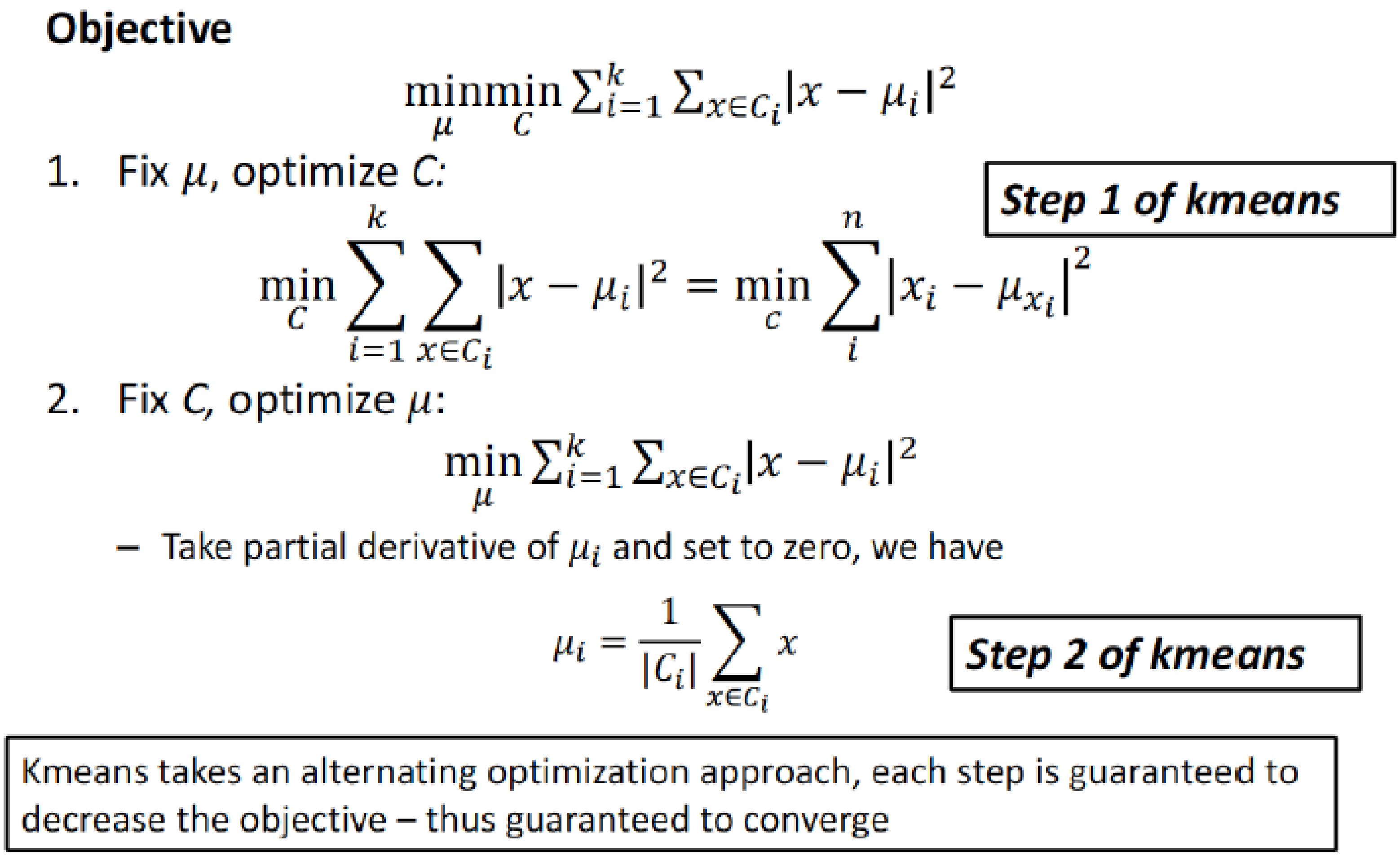
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points

*“closeness” defined by **Euclidean** distance*

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

  - Stop when no points’ assignments change

K-Means Clustering (cont’d)



Objective

$$\min_{\mu}\min_{C}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2$$

1. Fix μ, optimize C:

Step 1 of kmeans

$$\min_{C}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2=\min_{c}\sum_{i}^{n}\left|x_i-\mu_{x_i}\right|^2$$

2. Fix C, optimize μ:

$$\min_{\mu}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2$$

– Take partial derivative of μ_i and set to zero, we have

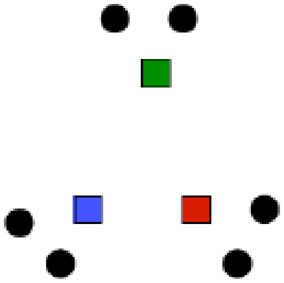
$$\mu_i=\frac{1}{|C_i|}\sum_{x\in C_i}x$$

Step 2 of kmeans

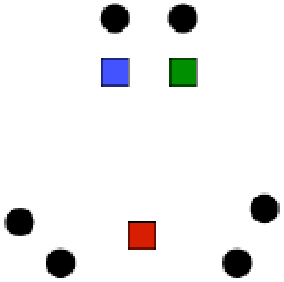
Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

K-Means Clustering (cont'd)

- K-means **algorithm** is a heuristic
  - Requires initial means
  - It does matter what you pick!



  - What can go wrong?
  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics



Run multiple times with different initializations

Hierarchical Clustering

- K-means clustering requires choosing the number of clusters. If we don’t want to do that, an alternative is to use *hierarchical clustering*.

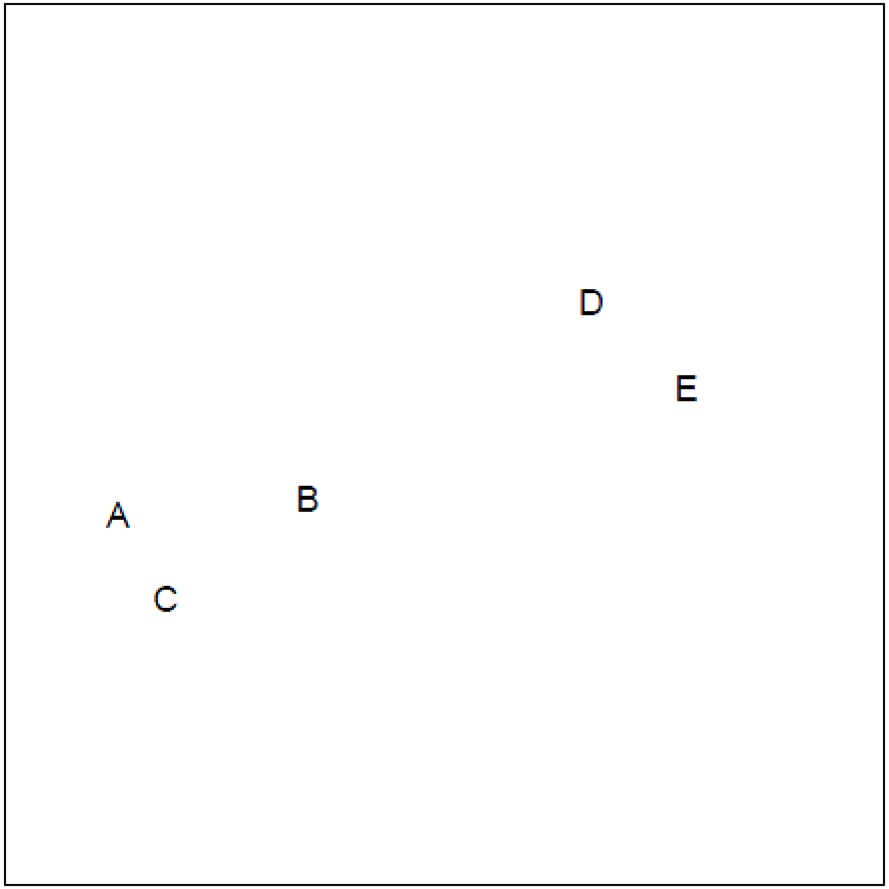
- Hierarchical clustering has an added advantage that it produces a tree-based representation of the observations, called a **dendrogram**.

- The most common type of hierarchical clustering is called *agglomerative clustering* (or bottom-up), which refers to the fact that a dendrogram is built starting from the leaves and combining clusters up to the trunk.

- Agglomerative clustering can be used as long as we have *pairwise distances* between any two objects.

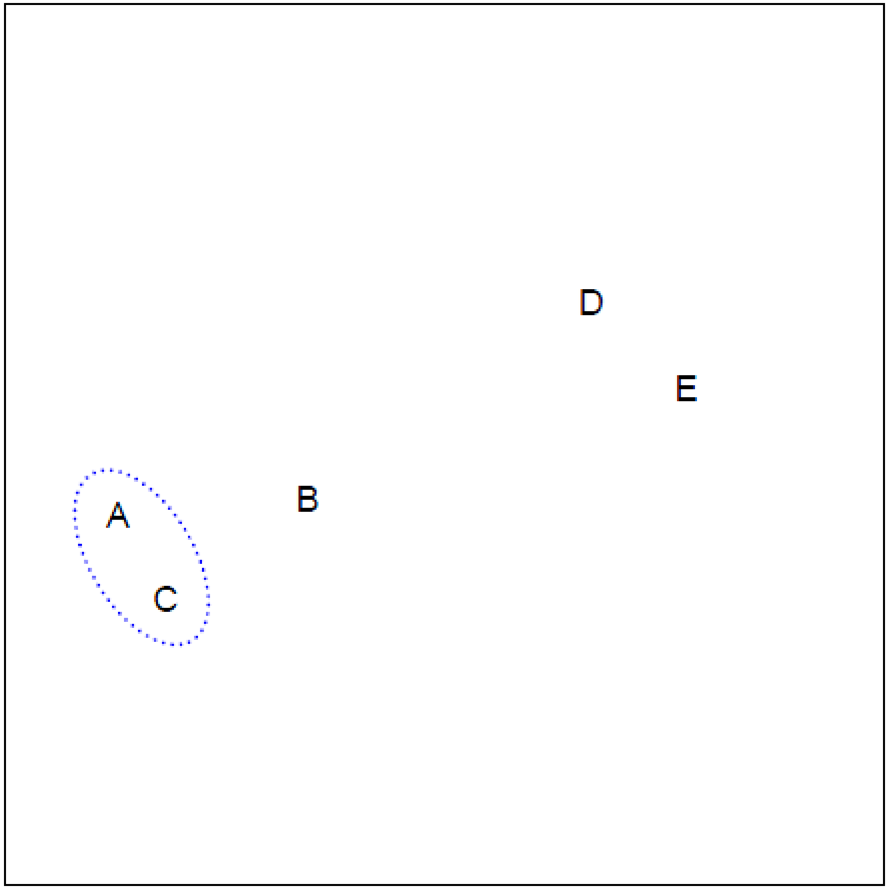
Hierarchical Clustering (cont’d)

- Builds a hierarchy in a bottom-up fashion:



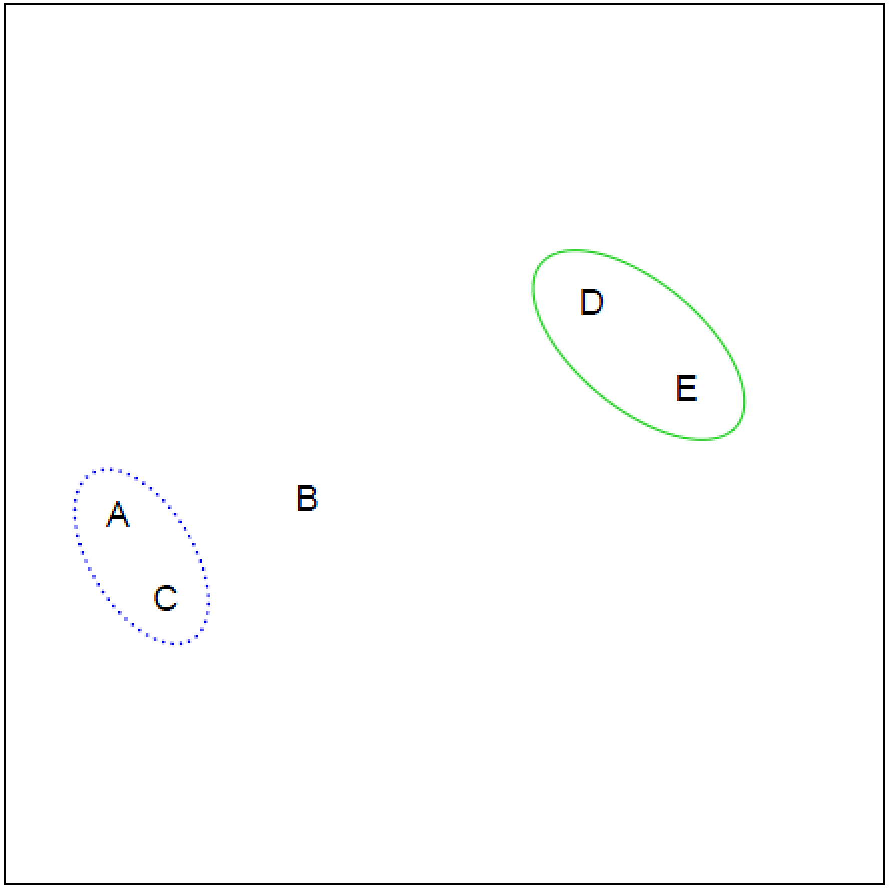
Hierarchical Clustering (cont’d)

- Builds a hierarchy in a bottom-up fashion:



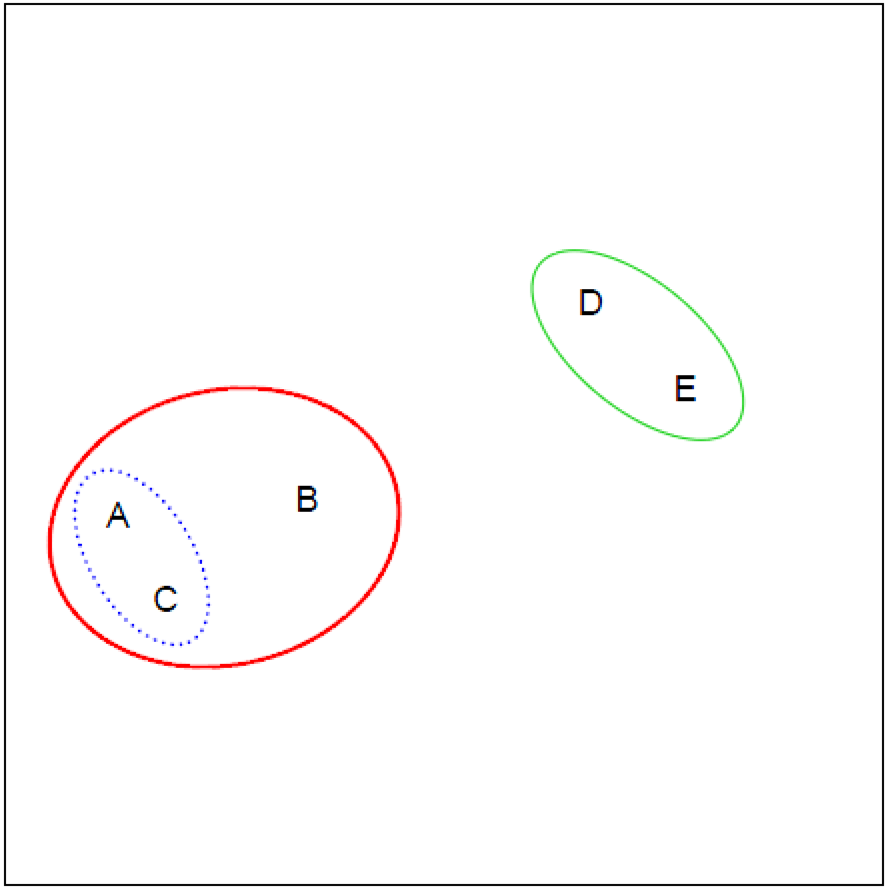
Hierarchical Clustering (cont'd)

- Builds a hierarchy in a bottom-up fashion:



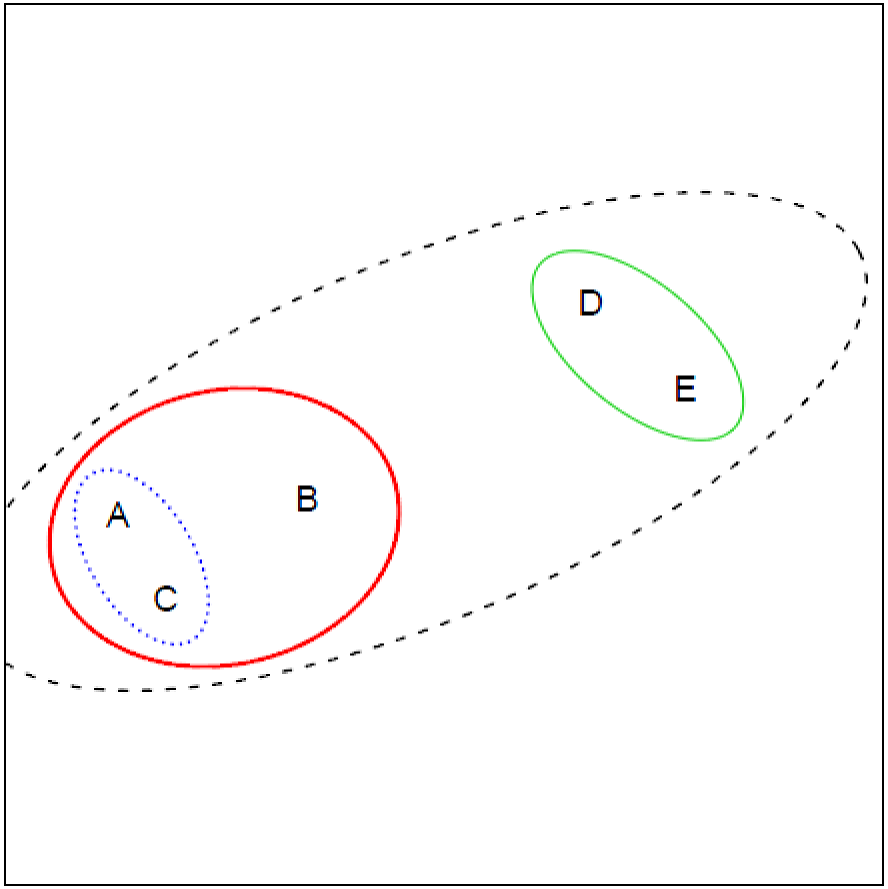
Hierarchical Clustering (cont’d)

- Builds a hierarchy in a bottom-up fashion:



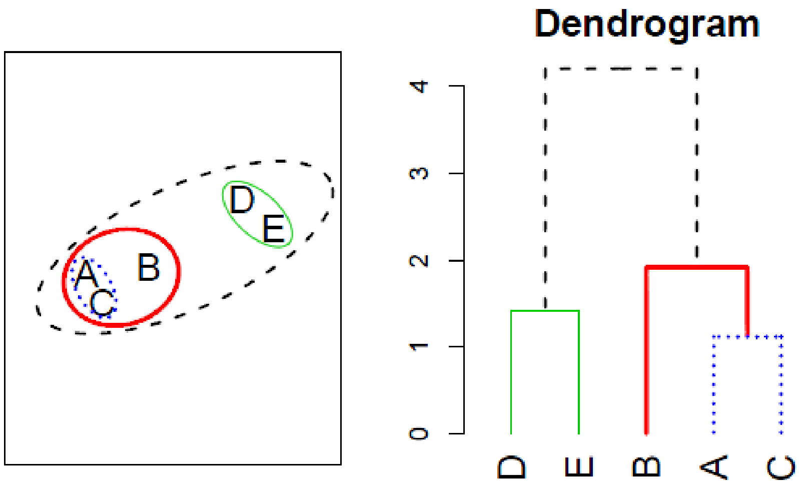
Hierarchical Clustering (cont'd)

- Builds a hierarchy in a bottom-up fashion:



Hierarchical Clustering (cont'd)

- **Algorithm (Agglomerative Approach):**
  - Start with each point as a separate cluster (*n* clusters).
  - Calculate a measure of dissimilarity between all points/clusters.
  - Fuse two clusters that are most similar, so that there are now *n*-1 clusters.
  - Fuse the next two most similar clusters, so there are now *n*-2 clusters.
  - Continue until all points are in a single cluster.



Hierarchical Clustering (cont’d)

- Each “leaf” of the dendrogram represents one of the observations.

- At the bottom of the dendrogram, each observation is a distinct leaf. However, as we move up the tree, some leaves begin to fuse. These correspond to observations that are similar to each other.

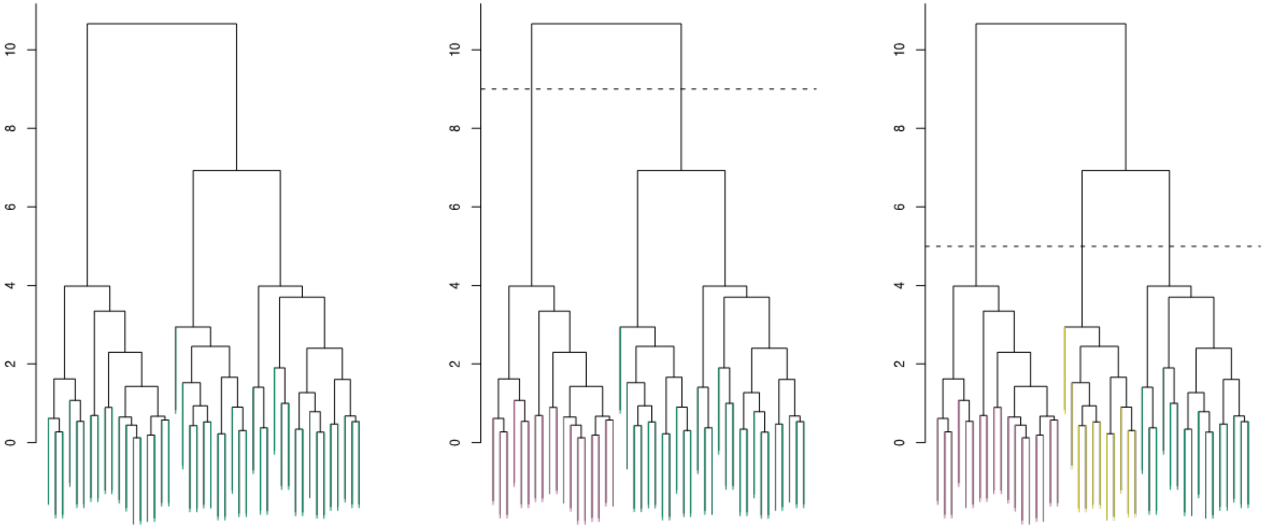
- As we move higher up the tree, an increasing number of observations have fused. The earlier (lower in the tree) two observations fuse, the more similar they are to each other.

- Observations that fuse later are quite different.

Hierarchical Clustering (cont’d)

- To choose clusters we draw lines **across** the dendrogram.

- We can form any number of clusters depending on where we **draw** the break point.



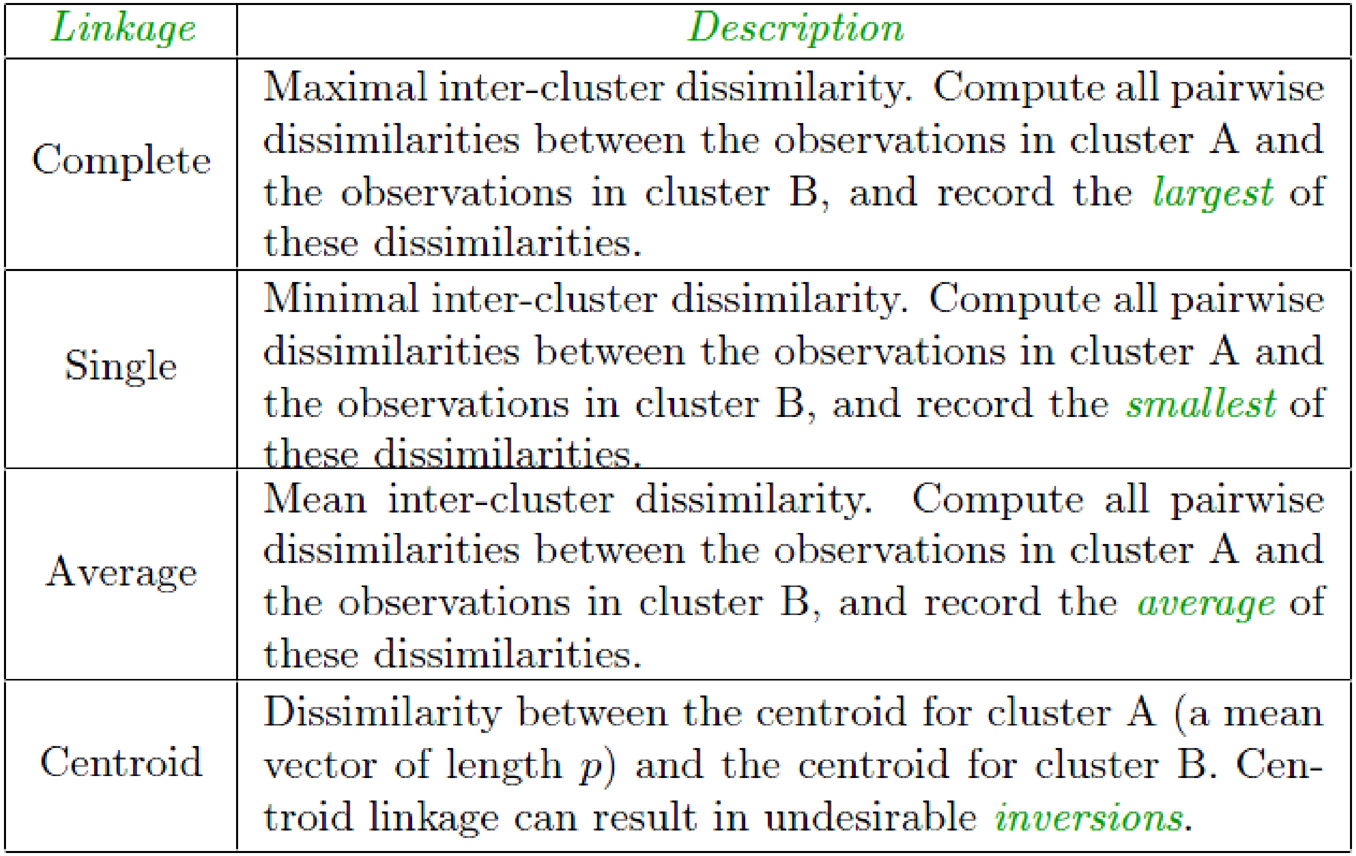
One Cluster

Two Clusters

Three Clusters

Hierarchical Clustering (cont'd)

- **Types of Linkage (Dissimilarity):**

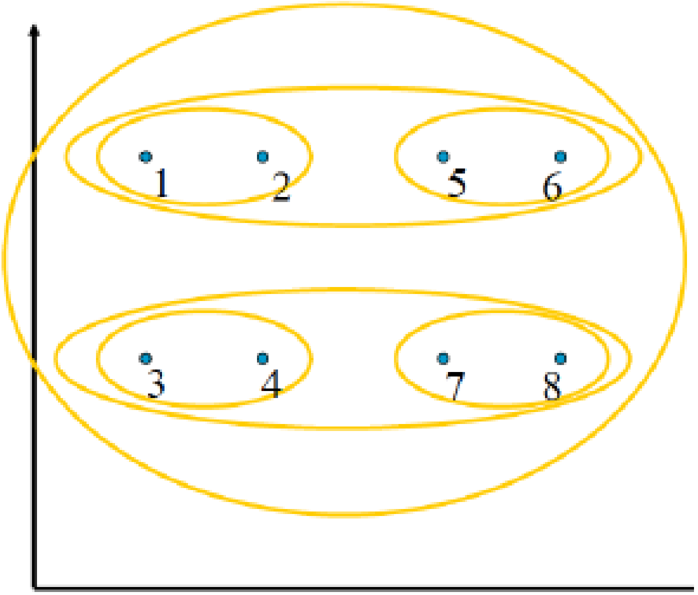


Linkage	*Description*
Complete	Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities.
Average	Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

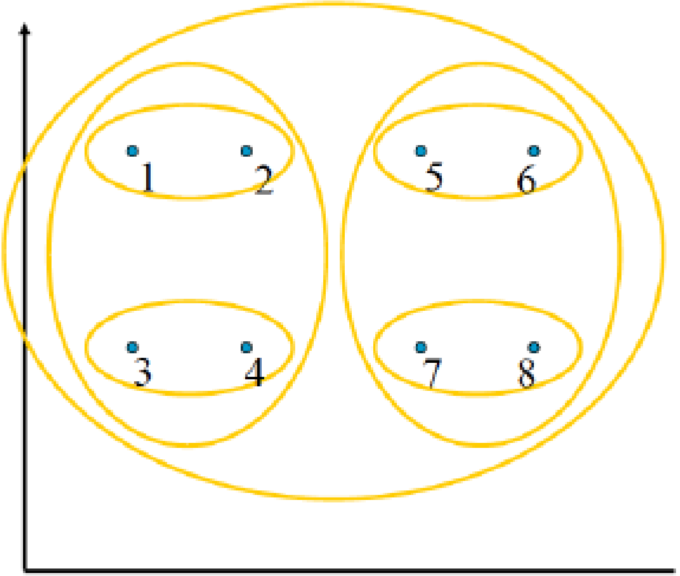
Hierarchical Clustering (cont’d)

- **Types of Linkage (Dissimilarity):**

Closest pair
(single-link clustering)

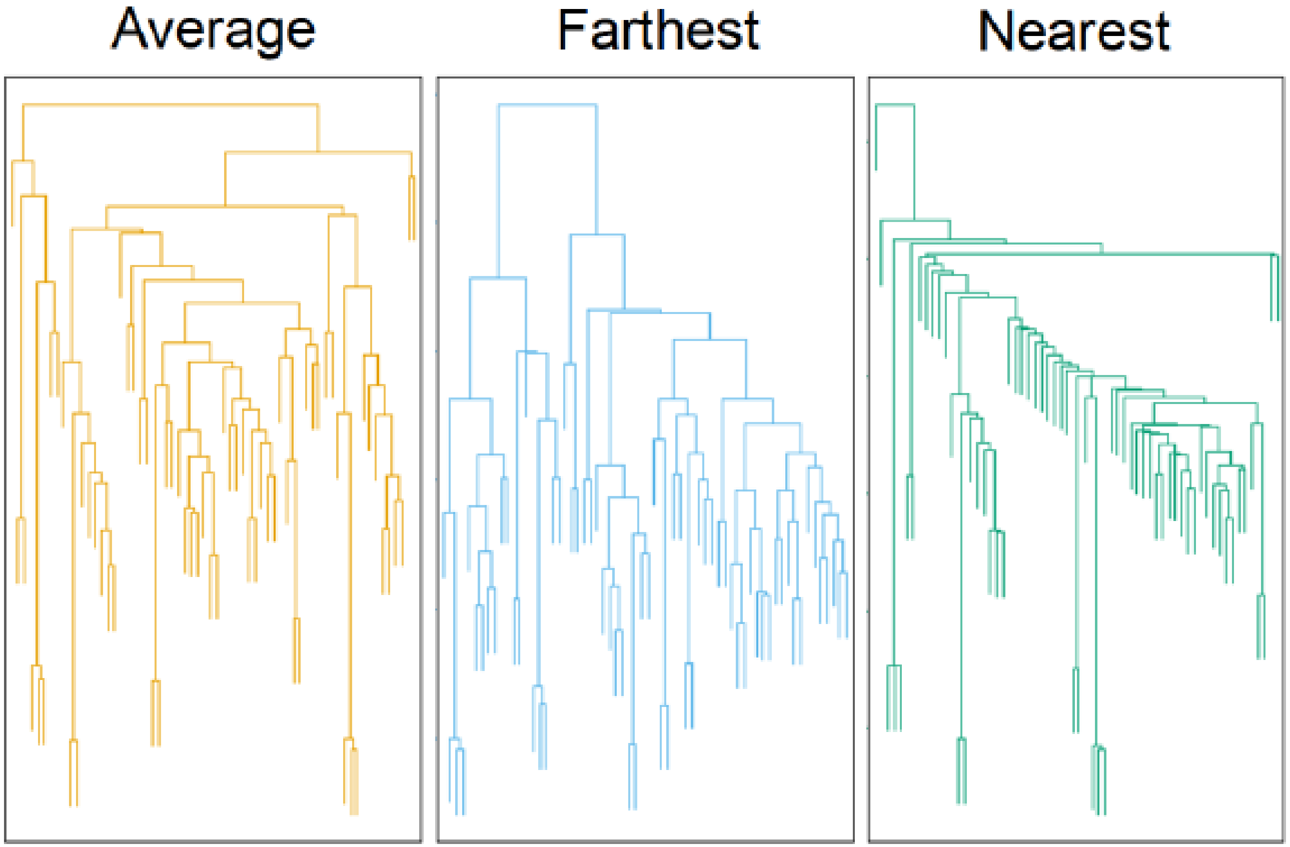


Farthest pair
(complete-link clustering)



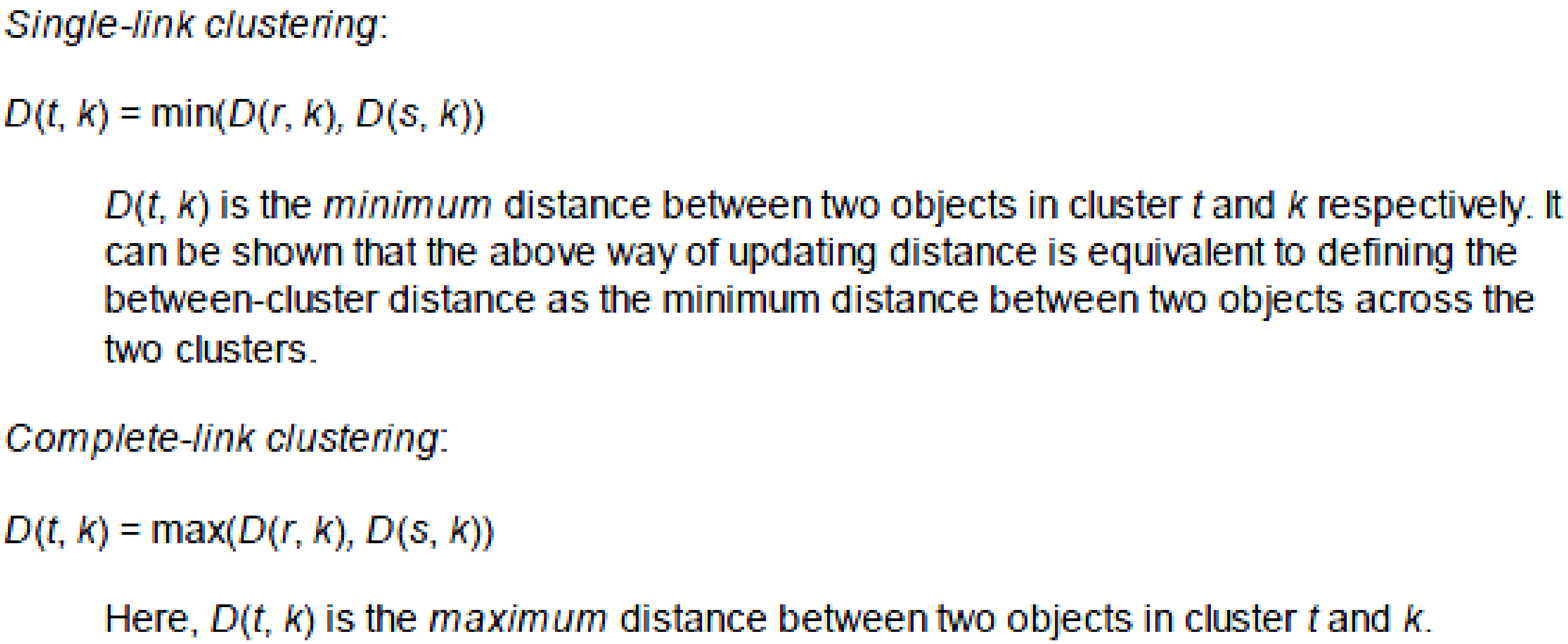
Hierarchical Clustering (cont'd)

- **Types of Linkage (Dissimilarity):**



Hierarchical Clustering (cont'd)

- **Types of Linkage (Dissimilarity):**



Single-link clustering:

$D(t, k) = \min(D(r, k), D(s, k))$

$D(t, k)$ is the *minimum* distance between two objects in cluster *t* and *k* respectively. It can be shown that the above way of updating distance is equivalent to defining the between-cluster distance as the minimum distance between two objects across the two clusters.

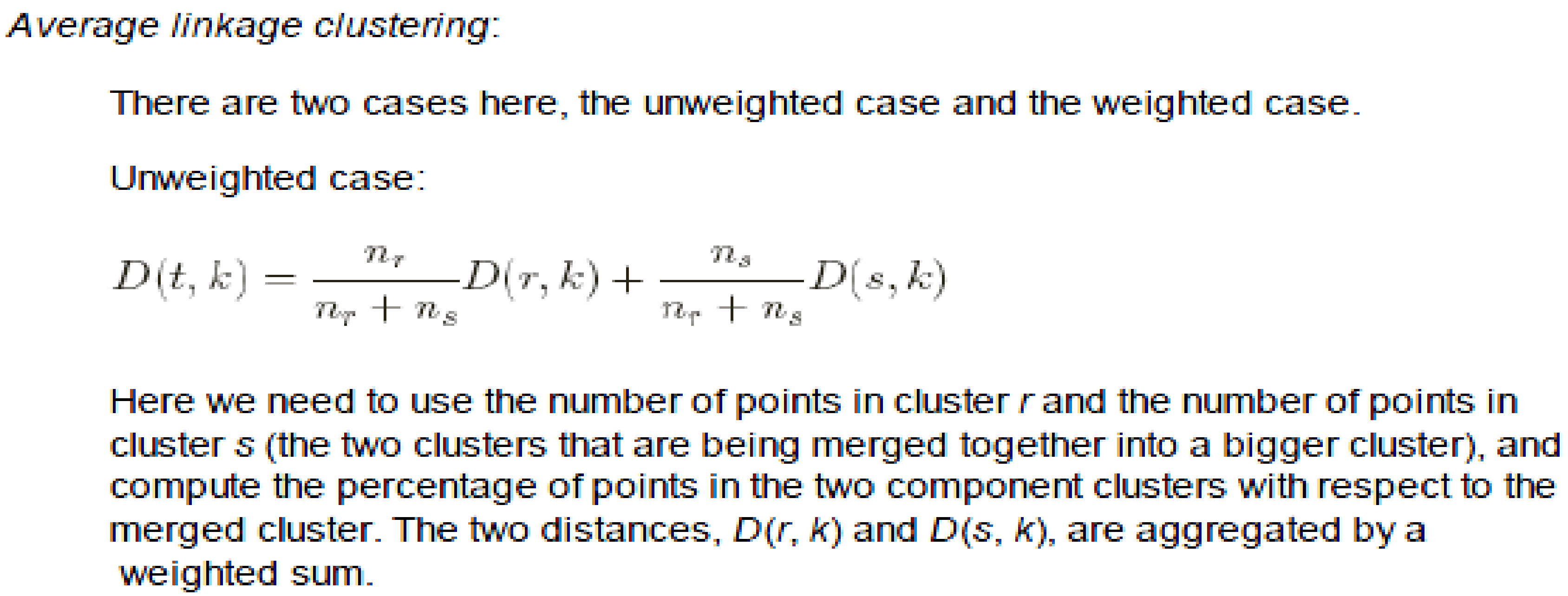
Complete-link clustering:

$D(t, k) = \max(D(r, k), D(s, k))$

Here, $D(t, k)$ is the *maximum* distance between two objects in cluster *t* and *k*.

Hierarchical Clustering (cont’d)

- **Types of Linkage (Dissimilarity):**



Average linkage clustering:

There are two cases here, the unweighted case and the weighted case.

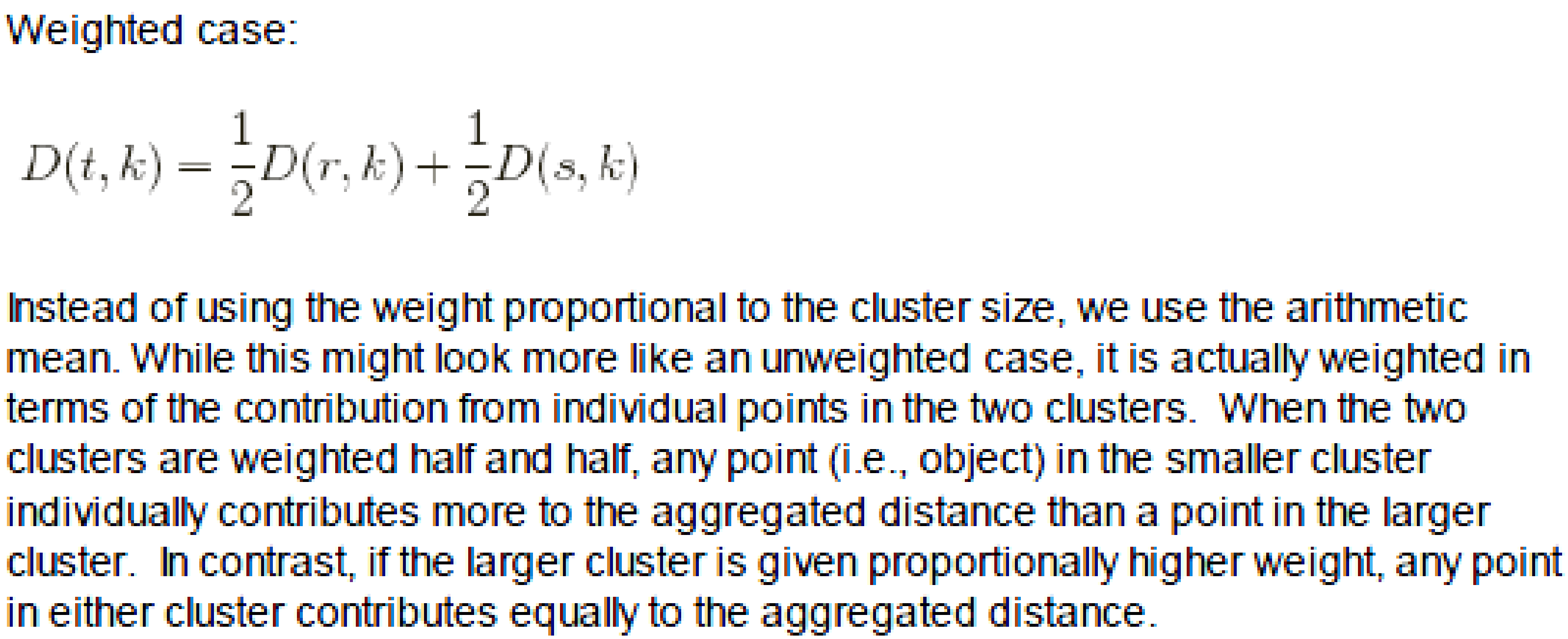
Unweighted case:

$$D(t,k) = \frac{n_r}{n_r + n_s} D(r,k) + \frac{n_s}{n_r + n_s} D(s,k)$$

Here we need to use the number of points in cluster *r* and the number of points in cluster *s* (the two clusters that are being merged together into a bigger cluster), and compute the percentage of points in the two component clusters with respect to the merged cluster. The two distances, *D*(*r*, *k*) and *D*(*s*, *k*), are aggregated by a weighted sum.

Hierarchical Clustering (cont'd)

- **Types of Linkage (Dissimilarity):**



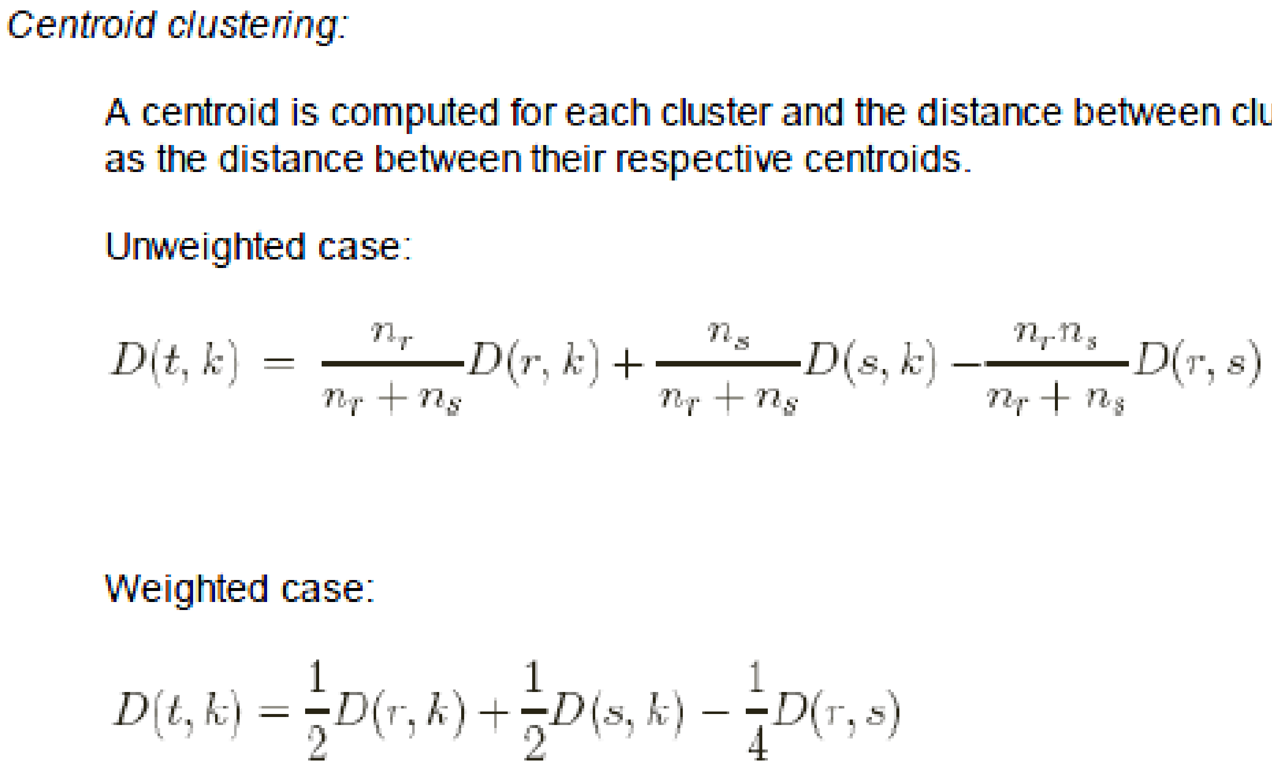
Weighted case:

$$D(t,k) = \frac{1}{2}D(r,k) + \frac{1}{2}D(s,k)$$

Instead of using the weight proportional to the cluster size, we use the arithmetic mean. While this might look more like an unweighted case, it is actually weighted in terms of the contribution from individual points in the two clusters. When the two clusters are weighted half and half, any point (i.e., object) in the smaller cluster individually contributes more to the aggregated distance than a point in the larger cluster. In contrast, if the larger cluster is given proportionally higher weight, any point in either cluster contributes equally to the aggregated distance.

Hierarchical Clustering (cont’d)

- **Types of Linkage (Dissimilarity):**



Centroid clustering:

A centroid is computed for each cluster and the distance between clusters is defined as the distance between their respective centroids.

Unweighted case:

$$D(t,k) = \frac{n_r}{n_r + n_s} D(r,k) + \frac{n_s}{n_r + n_s} D(s,k) - \frac{n_r n_s}{n_r + n_s} D(r,s)$$

Weighted case:

$$D(t,k) = \frac{1}{2} D(r,k) + \frac{1}{2} D(s,k) - \frac{1}{4} D(r,s)$$

Summary

- PCA for data visualization and dimension reduction.
- K-means clustering algorithm.
- Hierarchical clustering algorithm.