Supplementary Slides

Kernel Methods & Kernel K-Means

Why Data Representation?

- Data representation is a mandatory pre-requisite for data mining tasks
- Complex data like text, sequences, images, etc. need a multivariate vectors representation
  - i.e., given a data instance x (e.g., a sequence), we need to find a mapping Φ so that Φ(x) is the vector representation of x
- For numeric data matrix too, a non-linear mapping Φ may be used so that Φ(x) is a vector in the corresponding high-dimensional space
  - Required to discover non-linear relationships among the attributes
- Input space: Refers to data space for the input data x
- Feature space: Refers to space of mapped vectors Φ(x)

Example

Example 5.1 (Sequence-based Features): Consider a dataset of DNA sequences over the alphabet $\Sigma = \{A, C, G, T\}$. One simple feature space is to represent each sequence in terms of the probability distribution over symbols in Σ. That is, given a sequence $\mathbf{x}$ with length $|\mathbf{x}| = m$, the mapping into feature space is given as

$$\phi(\mathbf{x}) = \{P(A), P(C), P(G), P(T)\}$$

where $P(s) = \frac{n_s}{m}$ is the probability of observing symbol $s \in \Sigma$, and n_s is the number of times s appears in sequence $\mathbf{x}$. Here the input space is the set of sequences Σ^*, and the feature space is $\mathbb{R}^4$. For example, if $\mathbf{x} = ACAGCAGTA$, with $m = |\mathbf{x}| = 9$, since A occurs four times, C and G occur twice, and T occurs once, we have

$$\phi(\mathbf{x}) = (4/9, 2/9, 2/9, 1/9) = (0.44, 0.22, 0.22, 0.11)$$

Example (cont…)

Likewise, for another sequence $\mathbf{y} = AGCAAGCGAG$, we have

$$\phi(\mathbf{y}) = (4/10, 2/10, 4/10, 0) = (0.4, 0.2, 0.4, 0)$$

The mapping ϕ now allows one to compute statistics over the data sample, and make inferences about the population. For example, we may compute the mean symbol composition. We can also define the distance between any two sequences, for example,

$$\begin{aligned}\delta(\mathbf{x}, \mathbf{y}) &= \|\phi(\mathbf{x}) - \phi(\mathbf{y})\| \\ &= \sqrt{(0.44 - 0.4)^2 + (0.22 - 0.2)^2 + (0.22 - 0.4)^2 + (0.11 - 0)^2} = 0.22\end{aligned}$$

We can compute larger feature spaces by considering, for example, the probability distribution over all substrings or words of size up to k over the alphabet Σ, and so on.

Example (cont…)

Example 5.2 (Non-linear Features): As an example of a non-linear mapping consider the mapping ϕ that takes as input a vector $\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2$ and maps it to a "quadratic" feature space via the non-linear mapping

$$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T \in \mathbb{R}^3$$

For example, the point $\mathbf{x} = (5.9, 3)^T$ is mapped to the vector

$$\phi(\mathbf{x}) = (5.9^2, 3^2, \sqrt{2} \cdot 5.9 \cdot 3)^T = (34.81, 9, 25.03)^T$$

The main benefit of this transformation is that we may apply well known linear analysis methods in the feature space. However, since the features are non-linear combinations of the original attributes, this allows us to mine non-linear patterns and relationships.

Kernel Method

- Kernel methods avoid explicitly transforming each point x in the input space into the mapped point $\phi(x)$ in the feature space.
- Instead, the input objects are represented via their n×n pair-wise similarity values.
- The similarity function, called a kernel, is chosen so that it represents a dot product in some high-dimensional feature space, yet it can be computed without directly constructing $\phi(x)$.

Kernel Method

- Let I denotes the input space, which can comprise any arbitrary set of objects, and let D = $\{x_i\} \subset I$ be a dataset comprising n objects in the input space.
- We can represent the pair-wise similarity values between points in D via the n × n kernel matrix, defined as

$$\mathbf{K} = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

Where K : I × I → R is a kernel function satisfying the condition

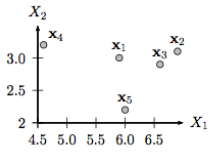
$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

Example 5.3 (Linear and Quadratic Kernels): Consider the identity mapping, $\phi(\mathbf{x}) \rightarrow \mathbf{x}$. This naturally leads to the *linear kernel*, which is simply the dot product between two input vectors, and thus satisfies (5.1)

$$\phi(\mathbf{x})^T \phi(\mathbf{y}) = \mathbf{x}^T \mathbf{y} = K(\mathbf{x}, \mathbf{y})$$

For example, consider the first five points from the two-dimensional Iris dataset shown in Figure 5.1a

$$\mathbf{x}_1 = \begin{pmatrix} 5.9 \\ 3 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} 6.9 \\ 3.1 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} 6.6 \\ 2.9 \end{pmatrix} \quad \mathbf{x}_4 = \begin{pmatrix} 4.6 \\ 3.2 \end{pmatrix} \quad \mathbf{x}_5 = \begin{pmatrix} 6 \\ 2.2 \end{pmatrix}$$



(a)

K	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$
$\mathbf{x}_1$	43.81	50.01	47.64	36.74	42.00
$\mathbf{x}_2$	50.01	57.22	54.53	41.66	48.22
$\mathbf{x}_3$	47.64	54.53	51.97	39.64	45.98
$\mathbf{x}_4$	36.74	41.66	39.64	31.40	34.64
$\mathbf{x}_5$	42.00	48.22	45.98	34.64	40.84

(b)

Consider the quadratic mapping $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ from Example 5.2, that maps $\mathbf{x} = (x_1, x_2)^T$ as follows

$$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2)^T$$

The dot product between the mapping for two input points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ is given as

$$\phi(\mathbf{x})^T \phi(\mathbf{y}) = x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 x_2 y_2$$

We can rearrange the above to obtain the (homogeneous) *quadratic kernel* function as follows

$$\begin{aligned} \phi(\mathbf{x})^T \phi(\mathbf{y}) &= x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 x_2 y_2 \\ &= (x_1 y_1 + x_2 y_2)^2 \\ &= (\mathbf{x}^T \mathbf{y})^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

We can thus see that the dot product in feature space can be computed by evaluating the kernel in input space, without explicitly mapping the points into feature space. For example, we have

$$\phi(\mathbf{x}_1) = (5.9^2, 3^2, \sqrt{2} \cdot 5.9 \cdot 3)^T = (34.81, 9, 25.03)^T$$
$$\phi(\mathbf{x}_2) = (6.9^2, 3.1^2, \sqrt{2} \cdot 6.9 \cdot 3.1)^T = (47.61, 9.61, 30.25)^T$$
$$\phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) = 34.81 \times 47.61 + 9 \times 9.61 + 25.03 \times 30.25 = 2501$$

We can verify that the homogeneous quadratic kernel gives the same value

$$K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^T \mathbf{x}_2)^2 = (50.01)^2 = 2501$$

Example 5.4: Consider the five points from Example 5.3 along with the linear kernel matrix shown in Figure 5.1. The mean of the five points in feature space is simply the mean in input space, since ϕ is the identity function for the linear kernel

$$\boldsymbol{\mu}_\phi = \sum_{i=1}^{5} \phi(\mathbf{x}_i) = \sum_{i=1}^{5} \mathbf{x}_i = (6.00, 2.88)^T$$

Now consider the squared magnitude of the mean in feature space

$$\|\boldsymbol{\mu}_\phi\|^2 = \boldsymbol{\mu}_\phi^T \boldsymbol{\mu}_\phi = (6.0^2 + 2.88^2) = 44.29$$

Since this involves only a dot product in feature space, the squared magnitude can be computed directly from $\mathbf{K}$. As we shall see later (see (5.12)) the squared norm of the mean vector in feature space is equivalent to the average value of the kernel matrix $\mathbf{K}$. For the kernel matrix in Figure 5.1b we have

$$\frac{1}{5^2} \sum_{i=1}^{5} \sum_{j=1}^{5} K(\mathbf{x}_i, \mathbf{x}_j) = \frac{1107.36}{25} = 44.29$$

which matches the $\|\boldsymbol{\mu}_\phi\|^2$ value computed above. This example illustrates that operations involving dot products in feature space can be cast as operations over the kernel matrix $\mathbf{K}$.

Kernel K-Means

- In K-means, the separating boundary between clusters is linear.
- Kernel K-means allows one to extract non-linear boundaries between clusters via the use of the kernel trick. This way the method can be used to detect non-convex clusters.
- The main idea is to conceptually map a data point x_i in input space to a point $\varphi(x_i)$ in some high dimensional feature space, via an appropriate non-linear mapping φ.
- However, the kernel trick allows us to carry out the clustering in feature space purely in terms of the kernel function $K(x_i, x_j)$, which can be computed in input space, but corresponds to a dot (or inner) product $\varphi(x_i)^T \varphi(x_j)$ in feature space.

Kernel K-Means

- Let $\{C_1, \cdots, C_k\}$ specify the partitioning of the n points into k clusters, and let the corresponding cluster means in feature space be given as $\{\mu_1^\varphi, \cdots, \mu_k^\varphi\}$, where

$$\boldsymbol{\mu}_i^\phi = \frac{1}{n_i} \sum_{\mathbf{x}_j \in C_i} \phi(\mathbf{x}_j)$$

is the mean of cluster C_i in feature space, with $n_i = |C_i|$.

- In feature space, the kernel K-means sum of squared errors objective can be written as

$$\min_{\mathcal{C}} \; SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \left\| \phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^\phi \right\|^2$$

Expanding the kernel SSE objective in terms of the kernel function, we ge

$$
\begin{aligned}
SSE(\mathcal{C}) &= \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \left\| \phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^{\phi} \right\|^2 \\
&= \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \|\phi(\mathbf{x}_j)\|^2 - 2\phi(\mathbf{x}_j)^T \boldsymbol{\mu}_i^{\phi} + \left\| \boldsymbol{\mu}_i^{\phi} \right\|^2 \\
&= \sum_{i=1}^{k} \left(\Big(\sum_{\mathbf{x}_j \in C_i} \|\phi(\mathbf{x}_j)\|^2 \Big) - 2n_i \Big(\frac{1}{n_i} \sum_{\mathbf{x}_j \in C_i} \phi(\mathbf{x}_j) \Big)^T \boldsymbol{\mu}_i^{\phi} + n_i \left\| \boldsymbol{\mu}_i^{\phi} \right\|^2 \right) \\
&= \Big(\sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_j) \Big) - \Big(\sum_{i=1}^{k} n_i \left\| \boldsymbol{\mu}_i^{\phi} \right\|^2 \Big) \\
&= \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} K(\mathbf{x}_j, \mathbf{x}_j) - \sum_{i=1}^{k} \frac{1}{n_i} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b) \\
&= \sum_{j=1}^{n} K(\mathbf{x}_j, \mathbf{x}_j) - \sum_{i=1}^{k} \frac{1}{n_i} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b)
\end{aligned}
$$

Kernel Method

- However, the main difficulty is that we cannot explicitly compute the mean of each cluster in feature space.
- Fortunately, explicitly obtaining the cluster means is not required; all operations can be carried out in terms of the kernel function $K(x_i,x_j) = \phi(x_i)^T \phi(x_j)$.

14

Consider the distance of a point $\phi(\mathbf{x}_j)$ to the mean $\boldsymbol{\mu}_i^{\phi}$ in feature space, which can be computed as

$$
\begin{aligned}
\left\| \phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^{\phi} \right\|^2 &= \|\phi(\mathbf{x}_j)\|^2 - 2\phi(\mathbf{x}_j)^T \boldsymbol{\mu}_i^{\phi} + \left\| \boldsymbol{\mu}_i^{\phi} \right\|^2 \\
&= \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_j) - \frac{2}{n_i} \sum_{\mathbf{x}_a \in C_i} \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_a) + \frac{1}{n_i^2} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} \phi(\mathbf{x}_a)^T \phi(\mathbf{x}_b) \\
&= K(\mathbf{x}_j, \mathbf{x}_j) - \frac{2}{n_i} \sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j) + \frac{1}{n_i^2} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b) \quad (13.4)
\end{aligned}
$$

15

Thus, the distance of a point to a cluster mean in feature space can be computed using only kernel operations. In the cluster assignment step of kernel K-means, we assign a point to the closest cluster mean as follows

$$
\begin{aligned}
C^*(\mathbf{x}_j) &= \arg\min_i \left\{ \left\| \phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^{\phi} \right\|^2 \right\} \\
&= \arg\min_i \left\{ K(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{n_i} \sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j) + \frac{1}{n_i^2} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b) \right\} \\
&= \arg\min_i \left\{ \frac{1}{n_i^2} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b) - \frac{2}{n_i} \sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j) \right\} \quad (13.5)
\end{aligned}
$$

16

Algorithm 13.2: Kernel K-means Algorithm

```
KERNEL-KMEANS(K, k, ε):
t ← 0
C^t ← {C_1^t, ··· , C_k^t} // Randomly partition points into k clusters
repeat
    t ← t + 1
    foreach C_i ∈ C^{t−1} do // Compute squared norm of cluster means
        sqnorm_i ← (1/n_i^2) Σ_{x_a ∈ C_i} Σ_{x_b ∈ C_i} K(x_a, x_b)
    foreach x_j ∈ D do // Average kernel value for x_j and C_i
        foreach C_i ∈ C^{t−1} do
            avg_ji ← (1/n_i) Σ_{x_a ∈ C_i} K(x_a, x_j)
    // Find closest cluster for each point
    foreach x_j ∈ D do
        foreach C_i ∈ C^{t−1} do
            d(x_j, C_i) ← sqnorm_i − 2 · avg_ji
        j* ← arg min_{C_i} {d(x_j, C_i)}
        C_{j*}^t ← C_{j*}^t ∪ {x_j} // Cluster reassignment
    C^t ← {C_1^t, ..., C_k^t}
until 1 − (1/n) Σ_{i=1}^{k} |C_i^t ∩ C_i^{t−1}| ≤ ε
```

Hierarchical Clustering

- Given *n* points in a *d*-dimensional space, hierarchical clustering aims to create a sequence of nested partitions, which can be conveniently visualized via a tree or hierarchy of clusters, also called the cluster dendrogram.
- The clusters in the hierarchy range from the fine-grained to the coarse-grained
  - The lowest level of the tree (the leaves) consists of each point in its own cluster
  - The highest level (the root) consists of all points in one cluster
  - Both of these may be considered to be trivial clusters

18

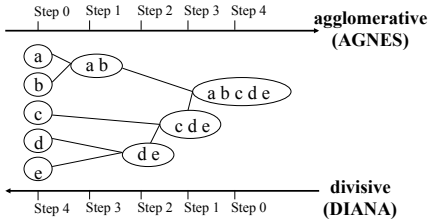
Hierarchical Clustering (cont…)

- Two main algorithmic approaches to mine hierarchical clusters:
  - Agglomerative: Work in a bottom-up manner. Starting with each of the *n* points in a separate cluster, repeatedly merges the most similar pair of clusters until all points are members of a single cluster.
  - Divisive: Do just the opposite; working in a top-down manner. Starting with all the points in the same cluster, recursively splits the clusters until all points are in separate clusters.

19

Hierarchical Clustering (cont…)

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input. However, it may be provided as an input.



20

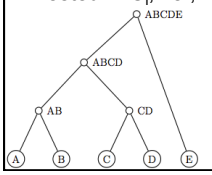
Preliminaries

- Clustering: Given a dataset D = {$x_1,\cdots, x_n$}, where $x_i \in R^d$, a clustering C = {$C_1,\cdots, C_k$} is a partition of D, i.e., each cluster is a set of points $C_i \subseteq D$, such that the clusters are pairwise disjoint, i.e., $C_i \cap C_j = \varnothing$ (for all i<>j), and $\cup C_i = D$
- Cluster Nesting: A clustering A = {$A_1,\cdots, A_r$} is said to be nested in another clustering B = {$B_1,\cdots, B_s$} if and only if r > s, and for each cluster $A_i \in A$, there exists a cluster $B_j \in B$, such that $A_i \subseteq B_j$.
- Hierarchical clustering yields a sequence of *n* nested partitions $C_1, \cdots, C_n$, ranging from the trivial clustering C_1 = {{x_1}, $\cdots$, {x_n}} where each point is in a separate cluster, to the other trivial clustering C_n = {{$x_1, \cdots, x_n$}}, where all points are in a single cluster.

21

Preliminaries (cont…)

- In general, the clustering C_{t-1} is nested in the clustering C_t.
- The cluster dendrogram is a rooted binary tree that captures the cluster nesting structure, with edges between cluster $C_i \in C_{t-1}$ and cluster $C_j \in C_t$ if C_i is nested in C_j, i.e., if $C_i \subset C_j$.



clustering	clusters
C_1	{A}, {B}, {C}, {D}, {E}
C_2	{AB}, {C}, {D}, {E}
C_3	{AB}, {CD}, {E}
C_4	{ABCD}, {E}
C_5	{ABCDE}

Agglomerative Clustering

- Begins with each of the n points in a separate cluster, and repeatedly merges the two closest clusters until all points are members of the same cluster
- Formally, given a set of clusters C = {C_1, C_2, .., C_m}, we find the closest pair of clusters C_i and C_j and merge them into a new cluster $C_{ij} = C_i \cup C_j$.
- Next, we update the set of clusters by removing C_i and C_j and adding C_{ij}, as follows C = C \{{C_i}∪{C_j} } ∪ {C_{ij} }, and repeat the process until C contains only one cluster.
- Since the number of clusters decreases by one in each step, this process results in a sequence of n nested clusterings.
- If specified, we can stop the merging process when there are exactly k clusters remaining.

23

Aggl. Clustering (Pseudo Code

Algorithm 14.1: Agglomerative Hierarchical Clustering Algorithm

```
AGGLOMERATIVECLUSTERING(D, k):
C ← {C_i = {x_i} | x_i ∈ D} // Each point in separate cluster
Δ ← {δ(x_i, x_j): x_i, x_j ∈ D} // Compute distance matrix
repeat
    Find the closest pair of clusters C_i, C_j ∈ C
    C_ij ← C_i ∪ C_j // Merge the clusters
    C ← C \ {{C_i} ∪ {C_j}} ∪ {C_ij} // Update the clustering
    Update distance matrix Δ to reflect new clustering
until |C| = k
```

24

Distance Between Clusters

- The main step in hierarchical clustering algorithm is to determine the closest pair of clusters, which can be computed using various distance measures, such as single link, complete link, group average, and so on.
- The between cluster distances are ultimately based on the distance between two points, which is typically computed using the Euclidean distance or L_2-norm, defined as

$$\delta(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2 = \left(\sum_{i=1}^{d}(x_i-y_i)^2\right)^{1/2}$$

25

Dist. Between Clusters (cont...)

- Single link: Given two clusters C_i and C_j , the distance between them, denoted $\delta(C_i, C_j)$ is defined as the minimum distance between a point in C_i and a point in C_j

$$\delta(C_i, C_j) = \min\{\delta(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in C_i, \mathbf{y} \in C_j\}$$

- Complete link: The distance between two clusters is defined as the maximum distance between a point in C_i and a point in C_j

$$\delta(C_i, C_j) = \max\{\delta(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in C_i, \mathbf{y} \in C_j\}$$

26

Dist. Between Clusters (cont...)

- Group Average: The distance between two clusters is defined as the average pairwise distance between points in C_i and C_j

$$\delta(C_i, C_j) = \frac{\sum_{\mathbf{x}\in C_i}\sum_{\mathbf{y}\in C_j}\delta(\mathbf{x},\mathbf{y})}{n_i \cdot n_j}$$

- Mean Distance: The distance between two clusters is defined as the distance between the means or centroids of the two clusters

$$\delta(C_i, C_j) = \delta(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$$

where $\boldsymbol{\mu}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in C_i}\mathbf{x}$.

27

Dist. Between Clusters (cont...)

- Minimum Variance: Ward's Method: The distance between two clusters is defined as the increase in the sum of squared errors (SSE) when the two clusters are merged. The SSE for a given cluster C_i is given as

$$SSE_i = \sum_{\mathbf{x}\in C_i}\|\mathbf{x}-\boldsymbol{\mu}_i\|^2$$

which can also be written as

$$\begin{aligned} SSE_i &= \sum_{\mathbf{x}\in C_i}\|\mathbf{x}-\boldsymbol{\mu}_i\|^2 \\ &= \sum_{\mathbf{x}\in C_i}\mathbf{x}^T\mathbf{x} - 2\sum_{\mathbf{x}\in C_i}\mathbf{x}^T\boldsymbol{\mu}_i + \sum_{\mathbf{x}\in C_i}\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i \\ &= \left(\sum_{\mathbf{x}\in C_i}\mathbf{x}^T\mathbf{x}\right) - n_i\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i \end{aligned}$$

28

Ward's Method (cont...)

- The SSE for a clustering C = {C_1,⋯ ,C_m}, is given as

$$SSE = \sum_{i=1}^{m} SSE_i = \sum_{i=1}^{m}\sum_{\mathbf{x}\in C_i}\|\mathbf{x}-\boldsymbol{\mu}_i\|^2$$

- Ward's measure defines the distance between two clusters C_i and C_j as the net change in the SSE value when we merge C_i and C_j into C_{ij}, given as

$$\delta(C_i, C_j) = \Delta SSE_{ij} = SSE_{ij} - SSE_i - SSE_j$$

29

Ward's Method (cont...)

- We can obtain a simpler expression for the Ward's measure by substituting values, and noting that since $C_{ij} = C_i \cup C_j$ and $C_i \cap C_j = \varnothing$, we have $|C_{ij}| = n_{ij} = n_i + n_j$, and therefore

$$\begin{aligned} \delta(C_i, C_j) &= \Delta SSE_{ij} \\ &= \sum_{\mathbf{z}\in C_{ij}}\|\mathbf{z}-\boldsymbol{\mu}_{ij}\|^2 - \sum_{\mathbf{x}\in C_i}\|\mathbf{x}-\boldsymbol{\mu}_i\|^2 - \sum_{\mathbf{y}\in C_j}\|\mathbf{y}-\boldsymbol{\mu}_j\|^2 \\ &= \sum_{\mathbf{z}\in C_{ij}}\mathbf{z}^T\mathbf{z} - n_{ij}\boldsymbol{\mu}_{ij}^T\boldsymbol{\mu}_{ij} - \sum_{\mathbf{x}\in C_i}\mathbf{x}^T\mathbf{x} + n_i\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i - \sum_{\mathbf{y}\in C_j}\mathbf{y}^T\mathbf{y} + n_j\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j \\ &= n_i\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i + n_j\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j - (n_i+n_j)\boldsymbol{\mu}_{ij}^T\boldsymbol{\mu}_{ij} \end{aligned}$$

Since $\sum_{\mathbf{z}\in C_{ij}}\mathbf{z}^T\mathbf{z} = \sum_{\mathbf{x}\in C_i}\mathbf{x}^T\mathbf{x} + \sum_{\mathbf{y}\in C_j}\mathbf{y}^T\mathbf{y}$.

30

Ward's Method (cont...)

Since $\boldsymbol{\mu}_{ij} = \frac{n_i\boldsymbol{\mu}_i + n_j\boldsymbol{\mu}_j}{n_i + n_j}$

We obtain

$$\boldsymbol{\mu}_{ij}^T\boldsymbol{\mu}_{ij} = \frac{1}{(n_i+n_j)^2}\left(n_i^2\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i + 2n_in_j\boldsymbol{\mu}_i^T\boldsymbol{\mu}_j + n_j^2\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j\right)$$

$$\begin{aligned}\delta(C_i, C_j) &= \Delta SSE_{ij} \\ &= n_i\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i + n_j\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j - \frac{1}{(n_i+n_j)}\left(n_i^2\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i + 2n_in_j\boldsymbol{\mu}_i^T\boldsymbol{\mu}_j + n_j^2\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j\right) \\ &= \frac{n_i(n_i+n_j)\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i + n_j(n_i+n_j)\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j - n_i^2\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i - 2n_in_j\boldsymbol{\mu}_i^T\boldsymbol{\mu}_j - n_j^2\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j}{n_i+n_j} \\ &= \frac{n_in_j\left(\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i - 2\boldsymbol{\mu}_i^T\boldsymbol{\mu}_j + \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j\right)}{n_i+n_j} \\ &= \left(\frac{n_in_j}{n_i+n_j}\right)\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2\end{aligned}$$

Ward's Method (cont...)

- Ward's measure is therefore a weighted version of the mean distance measure
- We can see that the only difference is that Ward's measure weights the distance between the means by half of the harmonic mean of the cluster sizes, where the harmonic mean of two numbers n1 and n2 is given as

$$\frac{2}{\frac{1}{n_1}+\frac{1}{n_2}} = \frac{2n_1n_2}{n_1+n_2}$$

32

Updating Distance Matrix

- The Lance-Williams formula provides a general equation to re-compute the distances for all of the cluster proximity measures we considered earlier; it is given as

$$\delta(C_{ij}, C_r) = \alpha_i \cdot \delta(C_i, C_r) + \alpha_j \cdot \delta(C_j, C_r) + \beta \cdot \delta(C_i, C_j) + \gamma \cdot \left|\delta(C_i, C_r) - \delta(C_j, C_r)\right|$$

Measure	α_i	α_j	β	γ
Single Link	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete Link	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
Group Average	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Mean Distance	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$\frac{-n_i \cdot n_j}{(n_i+n_j)^2}$	0
Ward's	$\frac{n_i+n_r}{n_i+n_j+n_r}$	$\frac{n_j+n_r}{n_i+n_j+n_r}$	$\frac{-n_k}{n_i+n_j+n_r}$	0

33

Computational Complexity

- Initially it takes $O(n^2)$ time to create the pairwise distance matrix, unless it is specified as an input to the algorithm.
- Merge step:
  - In step t, we compute $O(n - t)$ distances.
  - Closest pair finding: Keep the n^2 distances in a Heap, which allows to find the minimum distance in O(1) time; creating the heap takes $O(n^2)$ time.
- Deleting/updating distances from the merged cluster takes O(log n) time for each operation, for a total time across all merge steps of $O(n^2 \log n)$.

34