Cluster Analysis: Basic Concepts and Algorithms (cont.)

- **(Dis)Similarity measures** — Section 2.4 of course book
  - `Euclidian` distance
  - Simple matching coefficient, `Jaccard` coefficient
  - `Cosine` and `edit` similarity measures
- **Cluster validation** — Section 8.5 of course book
- **Hierarchical clustering** — Sections 8.3 and 8.4 of course book
  - Single link
  - Complete link
  - Average link
  - Cobweb algorithm

Dissimilarity Measure: Euclidean Distance

- Euclidean Distance

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Where n is the number of dimensions (attributes) and p_k and q_k are, respectively, the kth attributes (components) of records p and q.

- Standardization is necessary, if scales differ
- What if there are nominal (e.g. `i`th attribute) attributes?

$$d(p_i, q_i) = \begin{cases} 0 & \text{if } p_i = q_i \\ 1 & \text{if } p_i \neq q_i \end{cases}$$

Example: Customer Segmentation

- **Problem**: to develop meaningful customer groups that are similar based on individual behaviors
- **Goal:** to know your customer better and to apply that knowledge to increase profitability reduce operational cost, and enhance customer service
  - *Why are my customers leaving?*
  - *What do my best customers look like?*
- **Aproach**: Clustering
  - Low correlation between input variables
    - produces more stable clusters
  - Class attribute tends to dominate cluster formation
  - Low skewness reduces the chance of creating small outlier clusters

Example: transaction data

Products

Customers

	P1	P2	P3	P4	…
C1	0	1	1	0	
C2	1	1	0	0	
C3	1	1	0	1	
…					

Data representation: boolean matrix

- Which distance measure to use?

Similarity Between Binary Vectors

- Common situation is that objects, p and q, have only binary attributes
- Compute **similarities** using the following quantities

  M_{01} = the number of attributes where p was 0 and q was 1
  M_{10} = the number of attributes where p was 1 and q was 0
  M_{00} = the number of attributes where p was 0 and q was 0
  M_{11} = the number of attributes where p was 1 and q was 1

- **`Simple Matching Coefficient`** (symmetric attributes)

  SMC = number of matches / number of attributes
  $= (M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$

- **`Jaccard Coefficient`** (asymmetric attributes)

  J = n. of 11 matches / n. of not-both-zero attributes values
  $= (M_{11}) / (M_{01} + M_{10} + M_{11})$

Example : `SMC` versus `Jaccard`

$p =$ 1 0 0 0 0 0 0 0 0 0

$q =$ 0 0 0 0 0 0 1 0 0 1

$M_{01} = 2$ (the number of attributes where p was 0 and q was 1)
$M_{10} = 1$ (the number of attributes where p was 1 and q was 0)
$M_{00} = 7$ (the number of attributes where p was 0 and q was 0)
$M_{11} = 0$ (the number of attributes where p was 1 and q was 1)

SMC $= (M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00}) = (0+7) / (2+1+0+7) = 0.7$

J $= (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$

Example: *Which measure to use to cluster exams with similar answers to false/true questions?*

Example: Clustering CDs

- **Problem**: To divide (music) products into categories
  - What are these categories really?
  - Customers who prefer the same category (e.g. hard-rock) tend to buy the same CDs
  - Similar categories of CD's have similar sets of customers, and vice-versa
- **Goal:** To make shopping suggestions to customers
- **Aproach**: Represent a CD by the customers who bought it

Customers

	C1	C2	C3	C4	…
CD1	0	1	1	0	
CD2	1	1	0	0	
CD3	1	1	0	1	
…					

Data representation: boolean matrix

- Which distance measure to use?

Example: Clustering CDs

- Which similarity measure can be used?
  - **Answer**: **Jaccard**
  - If `0-0` matches were counted , most of the CDs would be highly similar to most of other CDs
- For Amazon, there can be millions of costumers
  - Each CD is represented by a huge binary vector
  - Few data mining algorithms can cope with huge vectors
    - **Compress the vectors**: use an hash function such as min hashing
- Use association rule mining for each category (cluster) of CDs
  - Find rules that describe the clusters

Example: Clustering Documents

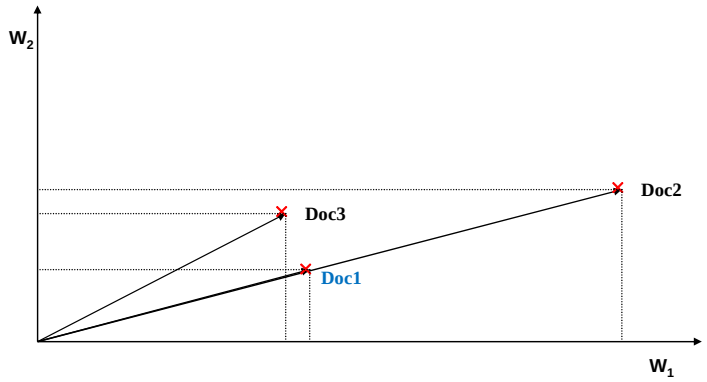
- **Problem**: documents with similar sets of keywords may be about the same topic
  - How to find groups of documents about same topic?
- **Approach**: represent a document by a vector

$$(x_1, x_2, \ldots, x_k)$$

$x_i \geq 0$ is n° of times the ith keyword appears in the document

  - Choose similarity (distance) measure and cluster the documents. Which similarity measure to use?
    - Most of the documents are likely to <u>not contain</u> many of the keywords
    - `0-0` matches should be ignore by the distance measure

Similarity of Two Document Vectors



- Which document is more similar to *Doc1*, based on the number of occurrences of keywords W_1 and W_2?

Cosine Similarity

- If d_1 and d_2 are two document vectors, then

$$\cos(X,Y) = \frac{X.Y}{||X||\ ||Y||}$$

where • indicates vector dot product and || *d* || is the length of vector *d*.

- Example:

d_1 = **3 2 0 5 0 0 0 2 0 0**
d_2 = **1 0 0 0 0 0 0 1 0 2**

cos(*d1*, *d2*) = 0.3150

$d_1 \bullet d_2$= 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5

$||d_1||$ = $(3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5}$ = $(42)^{0.5}$ = 6.481

$||d_2||$ = $(1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5}$ = $(6)^{0.5}$ = 2.245

Example: DNA Sequences

- **Problem**: Similar DNA sequences have similar functions or indicate family relationships
  - Cluster DNA sequences
- **Approach**: DNA are sequences of {C,A,T,G}
  - Which similarity (distance) function to choose?
  - Use **Edit distance**: measures the minimum number of "character edit operations" (*insert*, *delete*, *substitute*) needed to turn one sequence into another

Andrew

Amdrewz

1. Substitute "m" by "n"
2. Delete "z"

Distance = 2

Convergence of *K-Means*

- K-means finds (a local) minimum for some distance measures
  - **Manhattan distance** $d(X,Y) = \sum_{k=1}^{n} \text{abs}(x_k - y_k)$
    - If the records are binary vectors then Manhattan distance is the number of bits that are different between both records
    - Manhattan distance could be used for clustering exams with false/true answers
  - **Euclidean distance**
  - **Cosine**
- ***K-means*** can be parameterized by any distance function
  - **K-means** stops when the clusters become stable, or
  - Maximum number of iterations has been reached

Cluster Validity

- For supervised classification we have a variety of measures to evaluate how good our model is
  - *Accuracy, precision, recall*
  - *Cross-validation can be used to improve estimations*
- For cluster analysis, the analogous question is how to evaluate the "goodness" of the resulting clusters?
- Methods to evaluate clustering
  - Unsupervised
  - Supervised
    - To compare the results of clustering to externally known results (e.g. externally provided class labels)

Unsupervised Cluster Validity: Cohesion and Separation

- Cluster Cohesion: Measures how closely related are objects in a cluster

$$SSE = \sum_i \sum_{x \in C_i} dist^2(x, m_i)$$

  - small ***SSE*** are preferred

SSE is available in Weka

- Cluster Separation: Measures how distinct or well-separated the clusters are from one another

$$SSB = \sum_i |C_i| dist^2(m, m_i)$$

  - higher ***SSB*** are preferred

Where $|C_i|$ is the size of cluster i
m_i is the centroid of cluster i
m is the centroid of the overall data

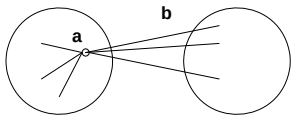
Silhouette Coefficient

- Silhouette Coefficient combine ideas of both cohesion and separation, but for individual points
- For an individual point, ***i***
  - Calculate ***a*** = average distance of *i* to the points in its cluster
  - Calculate ***b*** = min (average distance of *i* to points in another cluster)
  - The silhouette coefficient for a point is then given by

s = 1 – a/b, if a < b, (or s = b/a - 1 if a ≥ b, not the usual case)

$$-1 \le s = \frac{b-a}{\max(a,b)} \le 1$$

  - The closer to 1 the better



Goodness of clustering can be measured by the average silhouette coefficient of all points in the cluster

Supervised Cluster Validity

- Measure the degree of correspondence between the cluster labels and the class labels
  - Entropy of cluster **i** $$E_i = -\sum_{j=1}^{n} p_{ij} \log p_{ij}$$
  - Precision: the fraction of a cluster **i** that consists of objects of a class **j** $$\text{precision}(i, j) = p_{ij}$$
  - Recall: the extent to which a cluster **i** contains all objects of a class **j** $$\text{recall}(i, j) = \frac{m_{ij}}{m_j}$$

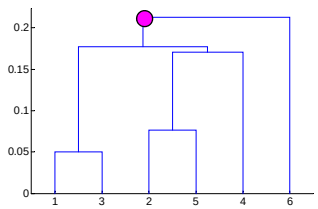
$p_{ij} = m_{ij}/m_i$

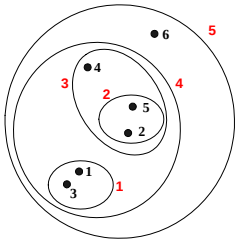
m_{ij} – number of records in cluster i that belong to class j

m_j – number of records in class j

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, genes with similar functions, …)

Hierarchical Clustering

- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity function between clusters or a distance matrix
  - Merge or split one cluster at a time

Agglomerative Clustering Algorithm

- More popular hierarchical clustering (agglomerative) technique

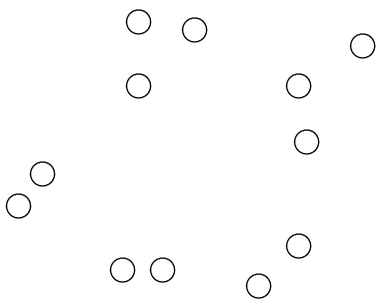
Basic algorithm is straightforward

1. Compute the proximity matrix
2. Let each data point be a cluster
3. **Repeat**
4. Merge the *two closest* clusters
5. Update the proximity matrix
6. **Until** only a single cluster remains

- Key operation is the computation of the proximity of clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Starting Situation

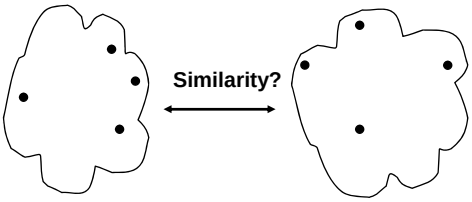
- Start with clusters of individual points and a proximity matrix



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

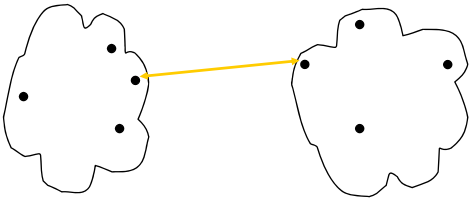


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN, single-linkage
- MAX, complete-linkage
- Group Average, average-linkage

How to Define Inter-Cluster Similarity

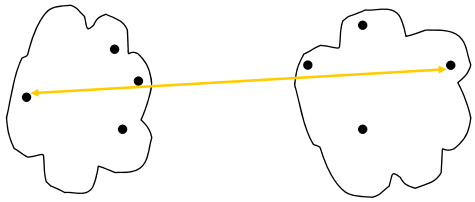


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN: cluster distance as the distance between the closest two points that are in different clusters

How to Define Inter-Cluster Similarity

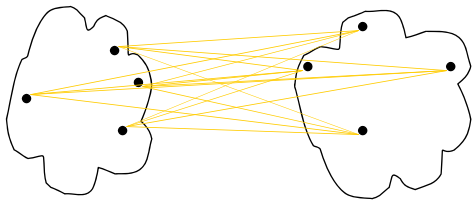


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MAX: cluster distance as the distance between the farthest two points that are in different clusters

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- Group Average: average distance from any member of one cluster to any member of the other cluster

Hierarchical Clustering: Time and Space requirements

- $O(N^2)$ space since it uses the proximity matrix
  - N is the number of points
- $O(N^3)$ time in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches
- Experiment results shows that the complete-link (MAX) algorithms generally yield better clustering quality than the single-link algorithms

Hierarchical clustering: COBWEB algorithm

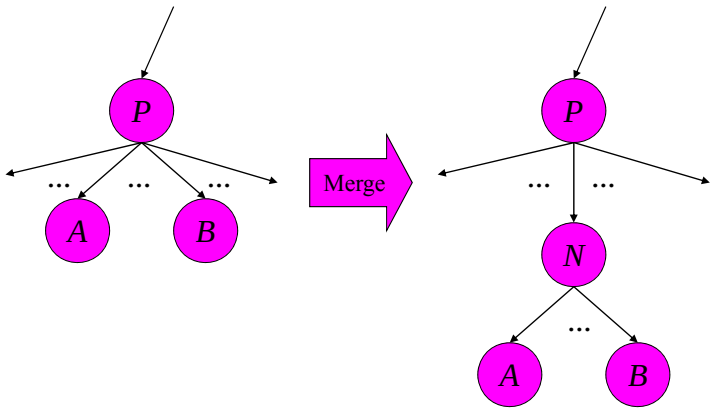
- The COBWEB (Fisher 1987) algorithm constructs a classification tree incrementally by inserting the objects into the classification tree one by one
- When inserting an object into the classification tree, the COBWEB algorithm traverses the tree top-down starting from the root node
  - Decision of how and where to insert a new object is guided by a function that measures the overall partition of records into clusters
    - Category utility (CU) function

COBWEB operations

- At each node, the COBWEB algorithm considers 4 possible operations and selects the one that yields the highest CU function value
  - Operations:
    - **insert** an instance in an existing node
    - **create** new node
    - **merge** two nodes (clusters)
    - **split** a node
- COBWEB incrementally organizes records into a tree

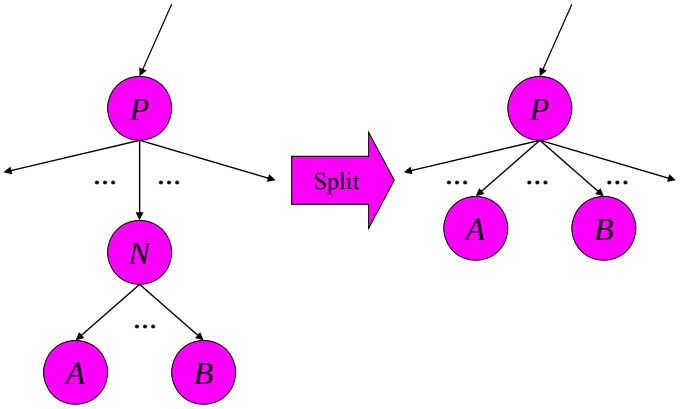
COBWEB operations: merge

- The COBWEB algorithm considers merging the two existing child nodes with the highest and second highest scores



COBWEB operations: split

- The COBWEB algorithm considers splitting the existing child node with the highest score



Category Utility

- Was developed in research of human categorization (Gluck and Corter 1985)
- Category utility attempts to maximize both the probability that two objects in the same category have values in common and the probability that objects in different categories will have different property values
  - For a cluster $\mathbf{C_\ell}$, this idea can be expressed as

$$\sum_i \sum_j (P(a_i = v_{ij}|C_l)^2 - P(a_i = v_{ij})^2)$$

COBWEB

- COBWEB forms a taxonomy (tree, hierarchy) of categories
  - Available in WEKA
- Works with both categorical and numeric attributes
  - For numeric attributes, a normal distribution is assumed
    - Mean (μ) and standard deviation (σ) parameters are estimated from the data (in the cluster)
    - If a cluster has only one record then estimated standard deviation becomes zero
    - **Acuity parameter**: minimum variance (measurement error)
    - **Cutoff parameter**: new nodes are added, if increase in CU function is above the cutoff
- COBWEB can help to find the **k** number of clusters for K-Means