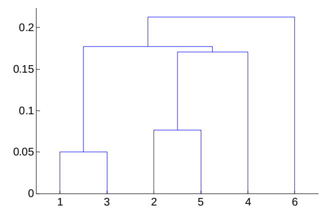
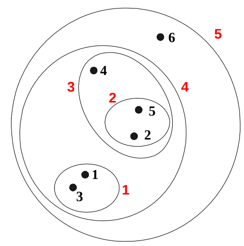
# Hierarchical Clustering

- 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, phylogeny reconstruction, …)

## 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 (or k clusters) 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 or distance matrix
  - Merge or split one cluster at a time

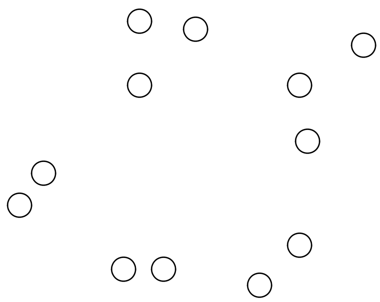


## Agglomerative Clustering Algorithm

- More popular hierarchical clustering 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 two 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



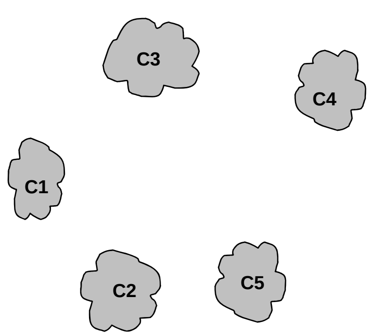
| | p1 | p2 | p3 | p4 | p5 | . . . |
|---|---|---|---|---|---|---|
| p1 | | | | | | |
| p2 | | | | | | |
| p3 | | | | | | |
| p4 | | | | | | |
| p5 | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |

**Proximity Matrix**



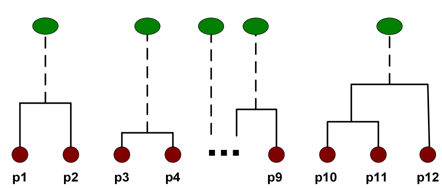

# Intermediate Situation

- After some merging steps, we have some clusters



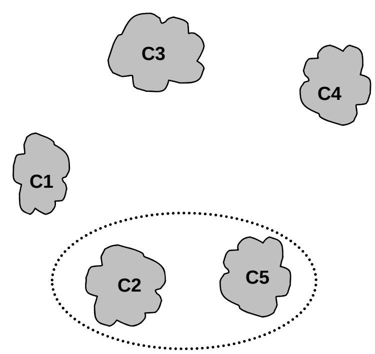

| | C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|---|
| C1 | | | | | |
| C2 | | | | | |
| C3 | | | | | |
| C4 | | | | | |
| C5 | | | | | |

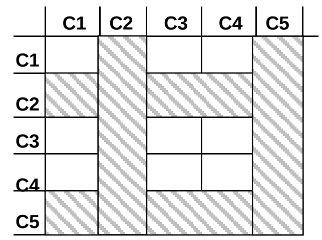
**Proximity Matrix**



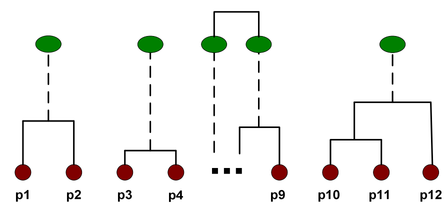
# Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



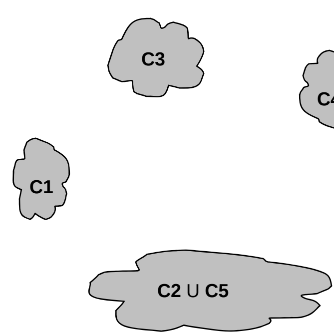

**Proximity Matrix**





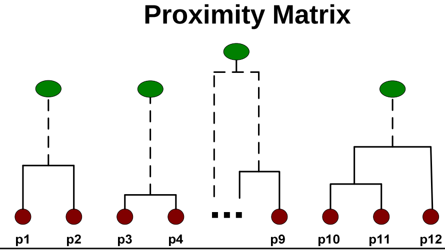

# After Merging

- The question is "How do we update the proximity matrix?"

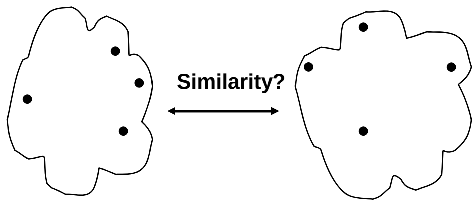


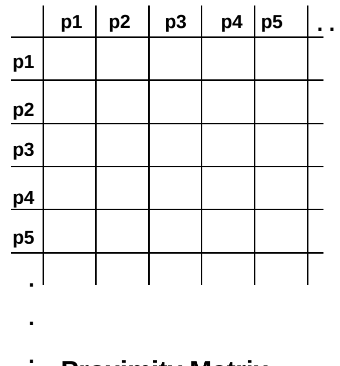

| | C1 | C2 U C5 | C3 | C4 |
|---|---|---|---|---|
| C1 | | ? | | |
| C2 U C5 | ? | ? | ? | ? |
| C3 | | ? | | |
| C4 | | ? | | |

**Proximity Matrix**



## How to Define Inter-Cluster Similarity



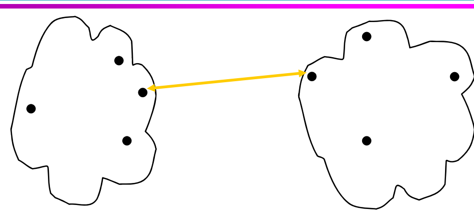



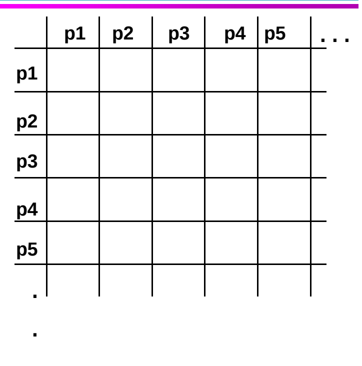

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  – Ward's Method uses squared error



## How to Define Inter-Cluster Similarity

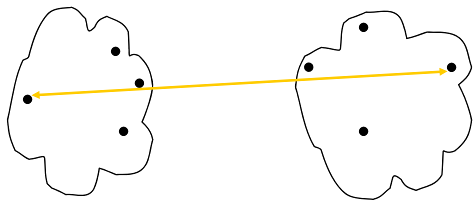





Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  – Ward's Method uses squared error

## How to Define Inter-Cluster Similarity



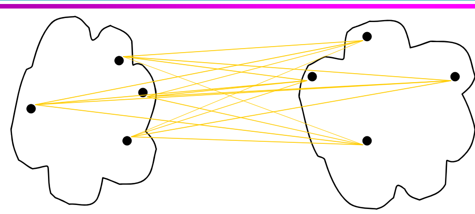
| | p1 | p2 | p3 | p4 | p5 | . . . |
|---|---|---|---|---|---|---|
| p1 | | | | | | |
| p2 | | | | | | |
| p3 | | | | | | |
| p4 | | | | | | |
| p5 | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |

**Proximity Matrix**

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  – Ward's Method uses squared error



## How to Define Inter-Cluster Similarity

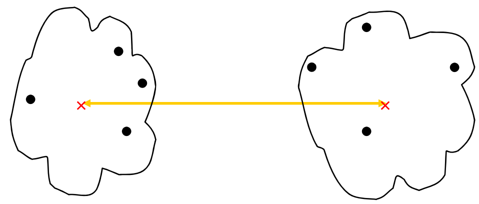


| | p1 | p2 | p3 | p4 | p5 | . . . |
|---|---|---|---|---|---|---|
| p1 | | | | | | |
| p2 | | | | | | |
| p3 | | | | | | |
| p4 | | | | | | |
| p5 | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |

**Proximity Matrix**

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  – Ward's Method uses squared error

## How to Define Inter-Cluster Similarity



|  | p1 | p2 | p3 | p4 | p5 | . . . |
|---|---|---|---|---|---|---|
| p1 |  |  |  |  |  |  |
| p2 |  |  |  |  |  |  |
| p3 |  |  |  |  |  |  |
| p4 |  |  |  |  |  |  |
| p5 |  |  |  |  |  |  |
| . |  |  |  |  |  |  |
| . |  |  |  |  |  |  |
| . |  |  |  |  |  |  |

**Proximity Matrix**

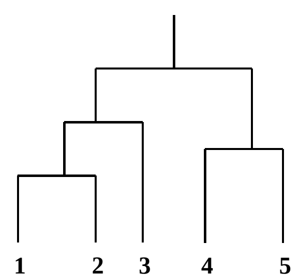
- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error



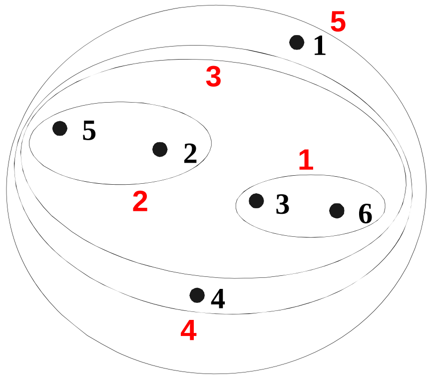
## Cluster Similarity: MIN or Single Link

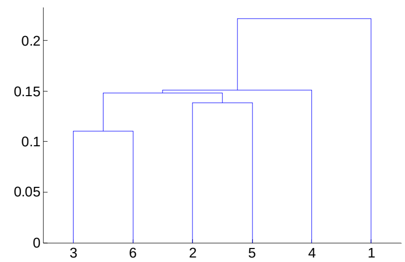
- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph.

|  | I1 | I2 | I3 | I4 | I5 |
|---|---|---|---|---|---|
| I1 | 1.00 | 0.90 | 0.10 | 0.65 | 0.20 |
| I2 | 0.90 | 1.00 | 0.70 | 0.60 | 0.50 |
| I3 | 0.10 | 0.70 | 1.00 | 0.40 | 0.30 |
| I4 | 0.65 | 0.60 | 0.40 | 1.00 | 0.80 |
| I5 | 0.20 | 0.50 | 0.30 | 0.80 | 1.00 |



# Hierarchical Clustering: MIN

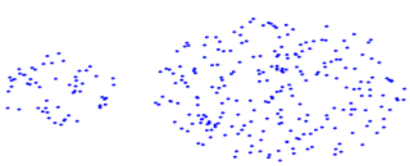



**Nested Clusters** **Dendrogram**



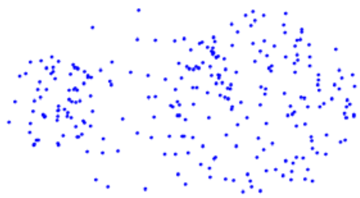
# Strength of MIN



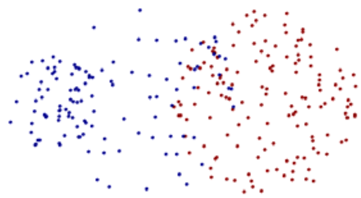
**Original Points** **Two Clusters**

• **Can handle non-elliptical shapes**

## Limitations of MIN



Original Points



Two Clusters

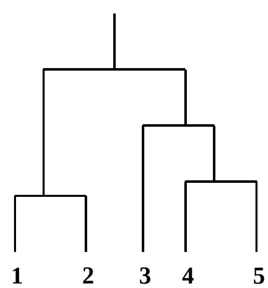
• **Sensitive to noise and outliers**



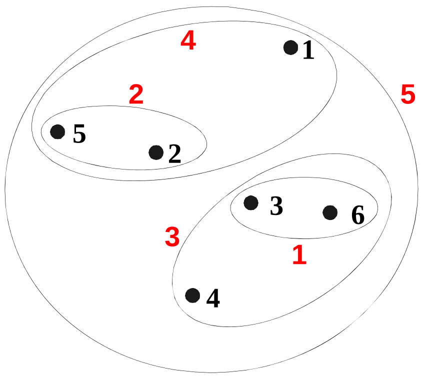
## Cluster Similarity: MAX or Complete Linkage

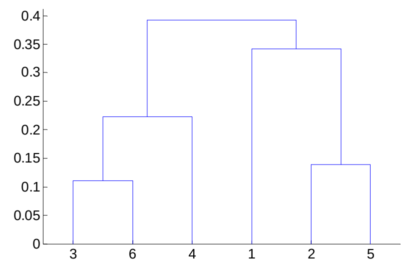
- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  – Determined by all pairs of points in the two clusters

| | I1 | I2 | I3 | I4 | I5 |
|---|---|---|---|---|---|
| I1 | 1.00 | 0.90 | 0.10 | 0.65 | 0.20 |
| I2 | 0.90 | 1.00 | 0.70 | 0.60 | 0.50 |
| I3 | 0.10 | 0.70 | 1.00 | 0.40 | 0.30 |
| I4 | 0.65 | 0.60 | 0.40 | 1.00 | 0.80 |
| I5 | 0.20 | 0.50 | 0.30 | 0.80 | 1.00 |



# Hierarchical Clustering: MAX

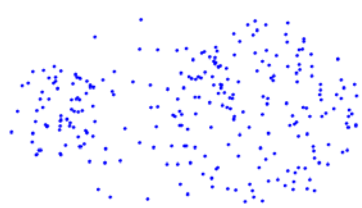


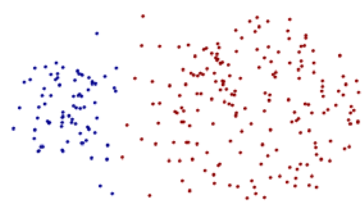




**Nested Clusters** **Dendrogram**



# Strength of MAX

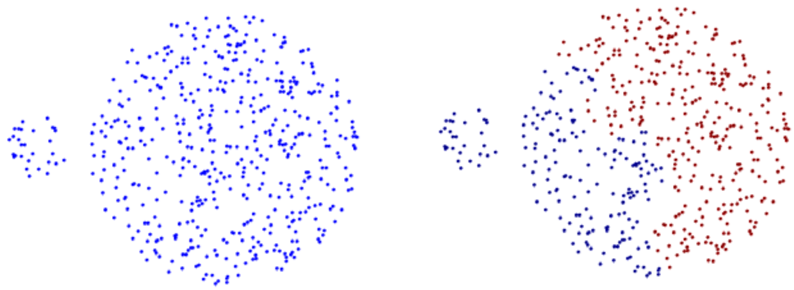




**Original Points** **Two Clusters**

- **Less susceptible to noise and outliers**

# Limitations of MAX



Original Points | Two Clusters

•Tends to break large clusters

•Biased towards globular clusters



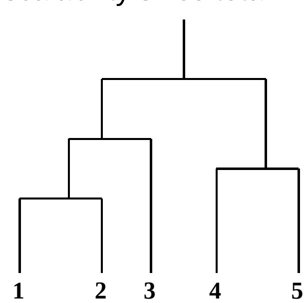
# Cluster Similarity: Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

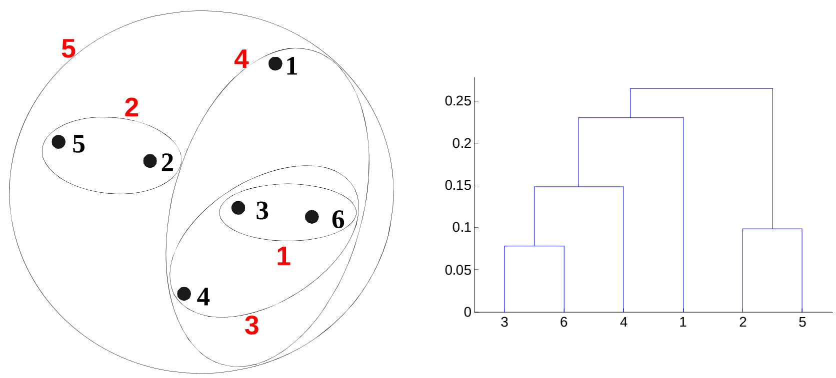
$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum\limits_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters

| | I1 | I2 | I3 | I4 | I5 |
|---|---|---|---|---|---|
| I1 | 1.00 | 0.90 | 0.10 | 0.65 | 0.20 |
| I2 | 0.90 | 1.00 | 0.70 | 0.60 | 0.50 |
| I3 | 0.10 | 0.70 | 1.00 | 0.40 | 0.30 |
| I4 | 0.65 | 0.60 | 0.40 | 1.00 | 0.80 |
| I5 | 0.20 | 0.50 | 0.30 | 0.80 | 1.00 |



# Hierarchical Clustering: Group Average




**Nested Clusters** **Dendrogram**



# Hierarchical Clustering: Group Average

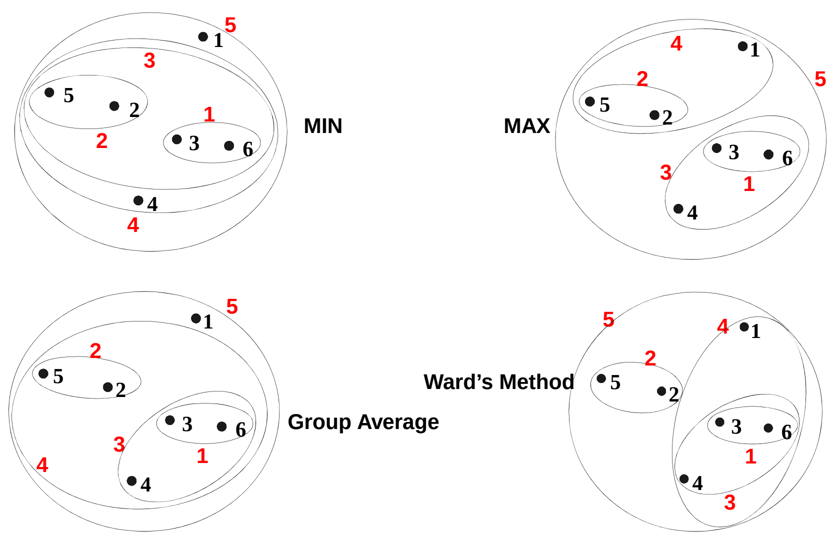
- Compromise between Single and Complete Link
- Strengths
  – Less susceptible to noise and outliers
- Limitations
  – Biased towards globular clusters

## Cluster Similarity: Ward's Method

- Similarity of two clusters is based on the increase in squared error when two clusters are merged
  - Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
  - Can be used to initialize K-means



## Hierarchical Clustering: Comparison



## 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



## Hierarchical Clustering: Problems and Limitations

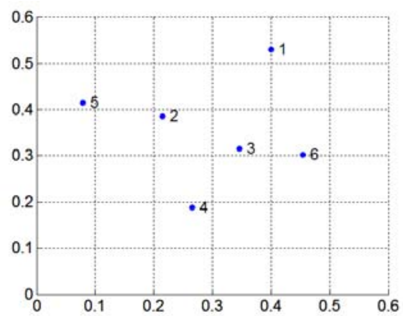
- Once a decision is made to combine two clusters, it cannot be undone

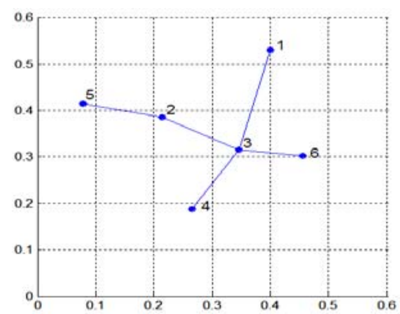
- No objective function is directly minimized

- Different schemes have problems with one or more of the following:
  – Sensitivity to noise and outliers
  – Difficulty handling different sized clusters and convex shapes
  – Breaking large clusters

# MST: Divisive Hierarchical Clustering

- Build MST (Minimum Spanning Tree)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points (p, q) such that one point (p) is in the current tree but the other (q) is not
  - Add q to the tree and put an edge between p and q






# MST: Divisive Hierarchical Clustering

- Use MST for constructing hierarchy of clusters

**Algorithm 7.5** MST Divisive Hierarchical Clustering Algorithm

1: Compute a minimum spanning tree for the proximity graph.
2: **repeat**
3: Create a new cluster by breaking the link corresponding to the largest distance (smallest similarity).
4: **until** Only singleton clusters remain

## More on hierarchical clustering methods

- Major weakness of agglomerative clustering methods
  - Do not scale well: time complexity of at least $O(n^2)$, where $n$ is the number of total objects
  - Cannot undo what was done previously
- Integration of hierarchical with distance-based clustering
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CURE (1998): selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling



## BIRCH (1996)

- Birch: Balanced Iterative Reducing and Clustering using Hierarchies, by Zhang, Ramakrishnan, Livny (SIGMOD'96)
- Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
  - Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
  - Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- Scales linearly: finds a good clustering with a single scan and improves the quality with a few additional scans
- Weakness: handles only numeric data, and sensitive to the order of the data record.

# Clustering Feature Vector

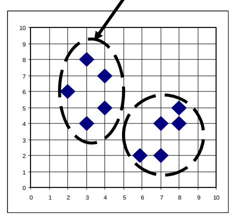
**Clustering Feature:** $CF = (N, \vec{LS}, \vec{SS})$

$N$: **Number of data points**

$\vec{LS}: \sum_{i=1}^{N} = \vec{X_i}$

$\vec{SS}: \sum_{i=1}^{N} = \vec{X_i}^2$

CF = (5, (16,30),(54,190))



(3,4)
(2,6)
(4,5)
(4,7)
(3,8)



# CF Tree