Clustering techniques are generally classified into these classes:

- **PARTITIONING ALGORITHMS**
  Directly divides data points into some prespecified number of clusters without a hierarchical structure

- **HIERARCHICAL ALGORITHMS**
  Groups data with a sequence of nested partitions, either from singleton clusters to a cluster containing all elements, or vice versa

- **DENSITY–BASED**
  Divides the feature space in order to find dense areas separated by empty space

- **GRID–BASED**
  Based on a multiple-level granularity structure

- **MODEL–BASED**
  Based on a model hypothesis in which the clusters can fit
Hierarchical clustering is usually represented by a tree-like structure called dendrogram.
The **root node** of the dendrogram represents the **whole data set**, each **leaf node** is regarded as a **data point**

- The intermediate nodes describe the extent to which the objects are **proximal to each other**

- The **height** of the dendrogram **expresses the distance** between each pair of data points or clusters, or a data point and a cluster.

- The clustering results can be obtained by **cutting the dendrogram** at different levels
There are two general approaches to hierarchical clustering:

- **AGGLOMERATIVE APPROACH**

  Starts from a set of singleton clusters (where each cluster contains a single element), and progressively aggregate them into bigger clusters until the desired number of cluster is reached or there is a single cluster containing all the elements.

- **DIVISIVE APPROACH**

  Starts from a single cluster containing all the elements, and progressively splits it into smaller clusters, until the desired number of clusters is reached or there are only singleton clusters.
Agglomerative Approach

1. Start
2. Represent each point as a cluster
3. Calculate the proximity matrix
4. Merge a pair of clusters with the minimal distance
5. One cluster left?
   - Yes: Generate the clusters by cutting the dendrogram at an appropriate level
   - No: Go back to step 4

End
Put every point in a cluster by itself.
For $i=1$ to $N-1$ do{
    let $C_1$ and $C_2$ be the most mergeable pair of clusters
    Create $C_{1,2}$ as parent of $C_1$ and $C_2$}

Example: For simplicity, we still use 1-dimensional objects.
- Numerical difference is used as the distance

Objects: 1, 2, 5, 6, 7

agglomerative clustering:
- find two closest objects and merge;
- => {1,2}, so we have now {1.5, 5, 6, 7};
- => {1,2}, {5,6}, so {1.5, 5.5, 7};
- => {1,2}, {{5,6}, 7}. 

```
1 2 5 6 7
```
Distance between clusters

- The merge of a pair of clusters or the formation of a new cluster is dependent on the definition of the distance function between two clusters

- Which proximity measure we use?
Distance between clusters

- The merge of a pair of clusters or the formation of a new cluster is dependent on the definition of the distance function between two clusters.

- Which proximity measure we use?

  - Euclidean, Mahalanobis, Pearson, Cosine, etc.
The merge of a pair of clusters or the formation of a new cluster is dependent on the definition of the distance function between two clusters.

Which proximity measure we use?

But....

Proximity measure can compute distance between two elements; how do we compute distance between two groups of elements?
The merge of a pair of clusters or the formation of a new cluster is dependent on the definition of the distance function between two clusters.

Which proximity measure we use?

But....

Proximity measure can compute distance between two elements; how do we compute distance between two groups of elements?

LINKAGE

Euclidean, Mahalanobis, Pearson, Cosine, etc.
Linkage defines how we measure the proximity of two groups of elements.

There are several choices of linkages:

- Single Linkage
- Complete Linkage
- Group Average Linkage
- Weighted Average Linkage
- Centroid Linkage
- Median Linkage
Single Linkage

- The distance between a pair of clusters is determined by the two closest objects to the different clusters.
Single Linkage

- The distance between a pair of clusters is determined by the two closest objects to the different clusters.

- When two clusters $C_i$ and $C_j$ are merged, the distance to a third cluster $C_l$ can be recomputed as:

$$D(C_l, (C_i, C_j)) = \min(D(C_l, C_i), D(C_l, C_j))$$
Single Link Example
Single Link Example
Single Link Example
Single Link Example
Single Link Example
Single Link Example
Single Link Example
Single Link Example
Single Linkage

- Single linkage clustering tends to generate elongated clusters

- As a result, two clusters with quite different properties may be connected due to the existence of noise

- However, if the clusters are separated far from each other, the single linkage method works well
Complete Linkage

- This method uses the farthest distance of a pair of objects to define inter-cluster distance.
Complete Linkage

- This method uses the farthest distance of a pair of objects to define inter-cluster distance.

- When two clusters $C_i$ and $C_j$ are merged, the distance to a third cluster $C_l$ can be recomputed as:

  $$D(C_l, (C_i, C_j)) = \max(D(C_l, C_i), D(C_l, C_j))$$

- It is effective in uncovering small and compact clusters.
Complete Link Example
Complete Link Example
Complete Link Example
Complete Link Example
Complete Link Example
Complete Link Example
Complete Link Example
Complete Link Example
Group Average Linkage

- This method is also known as the unweighted pair group method average (UPGMA)

- The distance between two clusters is defined as the average of the distance between all pairs of data points, each of which comes from a different group.

- When two clusters $C_i$ and $C_j$ are merged, the distance to a third cluster $C_l$ can be recomputed as:

$$D (C_l, (C_i, C_j)) = \frac{1}{2} (D (C_l, C_i) + D (C_l, C_j))$$
Weighted Group Average Linkage

- This method is also known as the weighted pair group method average (WPGMA)
Weighted Group Average Linkage

- This method is also known as the weighted pair group method average (WPGMA)

- The distance between two clusters is defined as the average of the distance between all pairs of data points, each of which comes from a different group

- The difference is that the distances between the newly formed cluster and the rest are weighted based on the number of data points in each cluster
Weighted Group Average Linkage

- This method is also known as the weighted pair group method average (WPGMA)

- When two clusters \( C_i \) and \( C_j \) are merged, the distance to a third cluster \( C_l \) can be recomputed as:

\[
D \left( C_l, (C_i, C_j) \right) = \frac{n_i}{n_i + n_j} D \left( C_l, C_i \right) + \frac{n_j}{n_i + n_j} D \left( C_l, C_j \right)
\]
Centroid Linkage

- This method is also known as the unweighted pair group method centroid (UPGMC)
- The centroid is the mean of all points in a cluster
Centroid Linkage

- This method is also known as the unweighted pair group method centroid (UPGMC)

- When two clusters \( C_i \) and \( C_j \) are merged, the distance to a third cluster \( C_l \) can be recomputed as:

\[
D (C_l, (C_i, C_j)) = \frac{n_i}{n_i + n_j} D (C_l, C_i) + \frac{n_j}{n_i + n_j} D (C_l, C_j) - \frac{n_in_j}{(n_i + n_j)^2} D (C_i, C_j)
\]
Median Linkage

- Similar to the centroid linkage, except that equal weight is given to the clusters to be merged.

- When two clusters $C_i$ and $C_j$ are merged, the distance to a third cluster $C_l$ can be recomputed as:

$$D(C_l, (C_i, C_j)) = \frac{1}{2} D(C_l, C_i) + \frac{1}{2} D(C_l, C_j) - \frac{1}{4} D(C_i, C_j)$$
**Linkage**

- Single linkage, complete linkage, and average linkage consider all points of a pair of clusters when calculating their inter-cluster distance, so they are also called **graph methods**

- The others are called **geometric methods** because they use geometric centers to represent clusters and determine their distances

**Equivalence to graph theory**

- If the dissimilarity matrix is defined as

  \[ D_{ij} = \begin{cases} 
  1 & \text{if } d(x_i, x_j) < d_0 \\
  0 & \text{otherwise} 
  \end{cases} \]

  Both the single linkage method and the complete linkage method can be described on the basis of the threshold graph:

  - Single linkage clustering is equivalent to seeking maximally connected sub-graphs (components), while complete linkage clustering corresponds to finding maximally complete sub-graphs (cliques).
Example

Example of agglomerative approach

```
1  2  3  4
1  0.0 0.20 0.15 0.30
2  0.20 0.0 0.40 0.50
3  0.15 0.40 0.0 0.10
4  0.30 0.50 0.10 0.0
```

(a) Dissimilarity Matrix

Try to cluster this dataset with single and complete linkage
Example

- Problem: clustering analysis with agglomerative algorithm

- Euclidean distance
  \[
  d_{AB} = \sqrt{(1-1.5)^2 + (1-1.5)^2} = 0.7071
  \]
  \[
  d_{DF} = \sqrt{(3-3)^2 + (4-3.5)^2} = 0.5
  \]

- Data matrix
  \[
  \begin{array}{cccccc}
  & A & B & C & D & E & F \\
  A & 1 & 7.1 & 5.66 & 3.61 & 4.24 & 3.20 \\
  B & 0.71 & 0.00 & 4.95 & 2.92 & 3.54 & 2.50 \\
  C & 5.66 & 4.95 & 0.00 & 2.24 & 1.41 & 2.50 \\
  D & 3.61 & 2.92 & 2.24 & 0.00 & 1.00 & 0.50 \\
  E & 4.24 & 3.54 & 1.41 & 1.00 & 0.00 & 1.12 \\
  F & 3.20 & 2.50 & 2.50 & 0.50 & 1.12 & 0.00 \\
  \end{array}
  \]
Example

- Merge two closest clusters (iteration 1)
Example

Update distance matrix (iteration 1)

\[
\begin{array}{cccccc}
& A & B & C & D & E & F \\
A & 0.00 & 0.71 & 5.66 & 3.61 & 4.24 & 3.20 \\
B & 0.71 & 0.00 & 4.95 & 2.92 & 3.54 & 2.50 \\
C & 5.66 & 4.95 & 0.00 & 2.24 & 1.41 & 2.50 \\
D & 3.61 & 2.92 & 2.24 & 0.00 & 1.00 & 0.50 \\
E & 4.24 & 3.54 & 1.41 & 1.00 & 0.00 & 1.12 \\
F & 3.20 & 2.50 & 2.50 & 0.50 & 1.12 & 0.00 \\
\end{array}
\]

\[
d_{(D,F)\to A} = \min(d_{DA}, d_{FA}) = \min(3.61, 3.20) = 3.20 \\
d_{(D,F)\to B} = \min(d_{DB}, d_{FB}) = \min(2.92, 2.50) = 2.50 \\
d_{(D,F)\to C} = \min(d_{DC}, d_{FC}) = \min(2.24, 2.50) = 2.24 \\
d_{E\to(D,F)} = \min(d_{ED}, d_{EF}) = \min(1.00, 1.12) = 1.00
\]

Min Distance (Single Linkage)
Example

- Merge two closest clusters (iteration 2)
Example

- Update distance matrix (iteration 2)

### Min Distance (Single Linkage)

<table>
<thead>
<tr>
<th>Dist</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D, F</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.00</td>
<td>0.71</td>
<td>5.66</td>
<td>3.20</td>
<td>4.24</td>
</tr>
<tr>
<td>B</td>
<td>0.71</td>
<td>0.00</td>
<td>4.95</td>
<td>2.50</td>
<td>3.54</td>
</tr>
<tr>
<td>C</td>
<td>5.66</td>
<td>4.95</td>
<td>0.00</td>
<td>2.24</td>
<td>1.41</td>
</tr>
<tr>
<td>D, F</td>
<td>3.20</td>
<td>2.50</td>
<td>2.24</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>E</td>
<td>4.24</td>
<td>3.54</td>
<td>1.41</td>
<td>1.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Calculations

- $d_{C \rightarrow (A, B)} = \min (d_{CA}, d_{CB}) = \min (5.66, 4.95) = 4.95$
- $d_{(D, F) \rightarrow (A, B)} = \min (d_{DA}, d_{DB}, d_{FA}, d_{FB}) = \min (3.61, 2.92, 3.20, 2.50) = 2.50$
- $d_{E \rightarrow (A, B)} = \min (d_{EA}, d_{EB}) = \min (4.24, 3.54) = 3.54$

### Updated Distance Matrix

<table>
<thead>
<tr>
<th>Dist</th>
<th>A, B</th>
<th>C</th>
<th>(D, F)</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>C</td>
<td>?</td>
<td>0</td>
<td>2.24</td>
<td>1.41</td>
</tr>
<tr>
<td>(D, F)</td>
<td>?</td>
<td>2.24</td>
<td>0</td>
<td>1.00</td>
</tr>
<tr>
<td>E</td>
<td>?</td>
<td>1.41</td>
<td>1.00</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

- Merge two closest clusters/update distance matrix (iteration 3)

### Min Distance (Single Linkage)

<table>
<thead>
<tr>
<th>Dist</th>
<th>(A,B)</th>
<th>C</th>
<th>(D, F)</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B</td>
<td>0</td>
<td>4.95</td>
<td>2.50</td>
<td>3.54</td>
</tr>
<tr>
<td>C</td>
<td>4.95</td>
<td>0</td>
<td>2.24</td>
<td>1.41</td>
</tr>
<tr>
<td>(D, F)</td>
<td>2.50</td>
<td>2.24</td>
<td>0</td>
<td>1.00</td>
</tr>
<tr>
<td>E</td>
<td>3.54</td>
<td>1.41</td>
<td>1.00</td>
<td>0</td>
</tr>
</tbody>
</table>

### Min Distance (Single Linkage)

<table>
<thead>
<tr>
<th>Dist</th>
<th>(A,B)</th>
<th>C</th>
<th>(D, F)</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A,B)</td>
<td>0.00</td>
<td>4.95</td>
<td>2.50</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>4.95</td>
<td>0.00</td>
<td>1.41</td>
<td></td>
</tr>
<tr>
<td>(D, F)</td>
<td>2.50</td>
<td>1.41</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>
Example

- Merge two closest clusters/update distance matrix (iteration 4)
Final result (meeting termination condition)
Example

Example of agglomerative approach

Try to cluster this dataset with **single** and **complete** linkage

(a) Dissimilarity Matrix
Example of agglomerative approach

(a) Dissimilarity Matrix

(b) Single Link

(c) Complete Link
Combines HAC and K-Means clustering.

First randomly take a sample of instances of size $N$

Run group-averaging HAC on this sample, which takes only $O(N)$ time.

Use the results of HAC as initial seeds for K-means.

Overall algorithm is $O(N)$ and avoids problems of bad seed selection.

Uses HAC to bootstrap K-means
Compared to agglomerative hierarchical clustering, divisive clustering proceeds in the opposite way.

In the beginning, the entire data set belongs to a cluster, and a procedure successively divides it until all clusters are singletons.

For a data set with N objects, a divisive hierarchical algorithm would start by considering $2^{N-1} - 1$ possible divisions of the data into two nonempty subsets, which is computationally expensive even for small-scale data sets.

Therefore, divisive clustering is not a common choice in practice.
However, the divisive clustering algorithms do provide clearer insights of the main structure of the data.

The larger clusters are generated at the early stage of the clustering process.

It is less likely to suffer from the accumulated erroneous decisions, which cannot be corrected by the successive process.

Given the computational burden of this approach, we rely on heuristic methods.
DIANA

DIANA (Divisive Analysis) considers only a part of all the possible divisions.

DIANA consists of a series of iterative steps in order to move the closer objects into the splinter group, which is seeded with the object that is farthest from the others in the cluster to be divided.

The cluster with the largest diameter, defined as the largest distance between any pair of objects, is selected for further division.
Suppose that cluster $C_l$ is going to be split into clusters $C_i$ and $C_j$:

1. Start with $C_i = C_l$ and $C_j$ empty
2. For each data object $x_m$ in $C_i$:
   a) For the first iteration, compute its average distance to all the other objects:
      
      $$d (x_m, C_i \setminus \{x_m\}) = \frac{1}{N_{C_i} - 1} \sum_{p \neq m, x_p \in C_i} d (x_m, x_p)$$

      b) For the remaining iterations, compute the difference between the average distance to $C_i$ and the average distance to $C_j$:

      $$d (x_m, C_i \setminus \{x_m\}) - d (x_m, C_j) = \frac{1}{N_{C_i} - 1} \sum_{x_p \in C_i, p \neq m} d (x_m, x_p) - \frac{1}{N_{C_j} - 1} \sum_{x_q \in C_j} d (x_m, x_q)$$
3. Decide whether to move element $x_m$ to cluster $C_j$ or keep it in cluster $C_i$:

a) For the first iteration, move the object with the maximum value to $C_j$ (that is, the element farther from every other element is separated from the others)

b) For the remaining iterations, if the maximum difference value is greater than zero, move the data object with the maximum difference into $C_j$, then repeat steps 2b and 3b. If the maximum value is less than zero, stop.
During each division of DIANA, all features are used; hence the divisive algorithm is called polythetic.

On the other hand, if the division is on a one-feature-one-step basis, the corresponding algorithm is said to be monothetic.

One such algorithm is called MONA (Monothetic Analysis) and is used for data objects with binary features.

The criterion for selecting a feature to divide the data is based on its similarity with other features, through measures of association.
Recent Advances

Criticisms of classical hierarchical clustering algorithms:

- lack of robustness (sensitivity to noise and outliers)
- not capable of correcting possible previous misclassification
- computational complexity, which is at least $O(N^2)$
Recent Advances

New clustering methods, with hierarchical cluster results, have appeared and have greatly improved the clustering performance.

- BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) (Zhang, et al., 1996)
- CURE (Clustering Using REpresentatives) (Guha et al., 1998)
- ROCK (RObust Clustering using linKs) (Guha et al., 2000)
- Chameleon (Karypis et al., 1999)
BIRCH

Designed for very large data sets

- Time and memory are limited
- Incremental and dynamic clustering of incoming objects
- Only one scan of data is necessary
- Does not need the whole data set in advance

Two key phases:

- Scans the database to build an in-memory tree
- Applies clustering algorithm to cluster the leaf nodes
Given a cluster of instances $\{\vec{X}_i\}$ we define:

**Centroid:**

$$\vec{X}_0 = \frac{\sum_{i=1}^{N} \vec{X}_i}{N}$$

**Radius:** average distance from member points to centroid

$$R = \left( \frac{\sum_{i=1}^{N} (\vec{X}_i - \vec{X}_0)^2}{N} \right)^{\frac{1}{2}}$$

**Diameter:** average pair-wise distance within a cluster

$$D = \left( \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X}_i - \vec{X}_j)^2}{N(N-1)} \right)^{\frac{1}{2}}$$
BIRCH

centroid Euclidean distance:

$$D_0 = \left( \sum_{i=1}^{d} (X_{01}^{(i)} - X_{02}^{(i)})^2 \right)^{\frac{1}{2}}$$

centroid Manhattan distance:

$$D_1 = |X_{01} - X_{02}| = \sum_{i=1}^{d} |X_{01}^{(i)} - X_{02}^{(i)}|$$

average inter-cluster:

$$D_2 = \left( \frac{\sum_{i=1}^{N_1} \sum_{j=N_1+1}^{N_1+N_2} (\bar{X}_i - \bar{X}_j)^2}{N_1N_2} \right)^{\frac{1}{2}}$$

average intra-cluster:

$$D_3 = \left( \frac{\sum_{i=1}^{N_1+N_2} \sum_{j=1}^{N_1+N_2} (\bar{X}_i - \bar{X}_j)^2}{(N_1 + N_2)(N_1 + N_2 - 1)} \right)^{\frac{1}{2}}$$

variance increase:

$$D_4 = \left( \frac{\sum_{k=1}^{N_1+N_2} (\bar{X}_k - \sum_{i=1}^{N_1+N_2} \frac{X_i}{N_1+N_2})^2}{\sum_{i=1}^{N_1} (\bar{X}_i - \sum_{i=1}^{N_1} \frac{X_i}{N_1})^2} - \frac{\sum_{i=1}^{N_1+N_2} \sum_{j=N_1+1}^{N_1+N_2} \frac{X_i}{N_1+N_2})^2}{\sum_{j=N_1+1}^{N_1+N_2} (\bar{X}_j - \sum_{j=N_1+1}^{N_1+N_2} \frac{X_i}{N_1+N_2})^2} \right)^{\frac{1}{2}}$$
To calculate centroid, radius, diameter, $D_0$, $D_1$, $D_2$, $D_3$ and $D_4$, NOT ALL POINTS ARE NEEDED!
The Birch algorithm builds a dendrogram called clustering feature tree (CF tree) while scanning the data set.

Each entry in the CF tree represents a cluster of objects and is characterized by a 3-tuple: \((N_i, LS_i, SS_i)\), where \(N_i\) is the number of objects in the \(i\)-th cluster and

\[
LS_i = \sum_{j=1}^{N_i} x_i \quad \text{linear sum of the data points}
\]

\[
SS_i = \sum_{j=1}^{N_i} x_i^2 \quad \text{square sum of the data points}
\]
Properties of Clustering Features

- CF entry is more compact
  - Stores significantly less than all of the data points in the sub-cluster

- A CF entry has sufficient information to calculate D0–D4

- Additivity theorem allows us to merge sub-clusters incrementally & consistently
Properties of Clustering Features

- CF entry is more compact
  - Stores significantly less than all of the data points in the sub-cluster

- A CF entry has sufficient information to calculate D0–D4

- Additivity theorem allows us to merge sub-clusters incrementally & consistently

\[
\text{CF}_1 + \text{CF}_2 = (N_1 + N_2, \tilde{L}\tilde{S}_1 + \tilde{L}\tilde{S}_2, SS_1 + SS_2)
\]
CF Additivity

Example:

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

(3,4)
(2,6)
(4,5)
(4,7)
(3,8)
Each non-leaf node has at most $B$ entries.

Each leaf node has at most $L$ CF entries, each of which satisfies threshold $T$.

Node size is determined by dimensionality of data space and input parameter $P$ (page size).
Each non-leaf node has at most $B$ entries

Each leaf node has at most $L$ CF entries, each of which satisfies threshold $T$

Node size is determined by dimensionality of data space and input parameter $P$ (page size)
Each non-leaf node has at most $B$ entries

Each leaf node has at most $L$ CF entries, each of which satisfies threshold $T$

Node size is determined by dimensionality of data space and input parameter $P$ (page size)
CF–Tree

- The tree size is a function of $T$ (the larger the $T$ is, the smaller the tree is)

- We require a node to fit in a page of size of $P$

- $B$ and $L$ are determined by $P$ ($P$ can be varied for performance tuning)

- Very compact representation of the dataset because each entry in a leaf node is not a single data point but a subcluster.
Phase 1: Scan dataset once, build a CF tree in memory

Phase 2: (Optional) Condense the CF tree to a smaller CF tree

Phase 3: Global Clustering

Phase 4: (Optional) Clustering Refining (require scan of dataset)
BIRCH Overview

Data

Phase 1: Load into memory by building a CF tree

Initial CF tree

Phase 2 (optional): Condense into desirable range by building a smaller CF tree

smaller CF tree

Phase 3: Global Clustering

Good Clusters

Phase 4: (optional and off line) : Cluster Refining

Better Clusters
Building CF Tree (Phase 1)

- CF of a data point (3,4) is (1,(3,4),25)

- Insert a point to the tree
  - Find the path (based on $D_0$, $D_1$, $D_2$, $D_3$, $D_4$ between CF of children in a non-leaf node)
  - Modify the leaf
    - Find closest leaf node entry (based on $D_0$, $D_1$, $D_2$, $D_3$, $D_4$ of CF in leaf node)
    - Check if it can “absorb” the new data point
  - Modify the path to the leaf
  - Splitting – if leaf node is full, split into two leaf node, add one more entry in parent
CF–Tree Insertion

➢ Recurse down from root, find the appropriate leaf
  ◦ Follow the "closest"–CF path, w.r.t. D0 / … / D4

➢ Modify the leaf
  ◦ If the closest–CF leaf cannot absorb, make a new CF entry.
    If there is no room for new leaf, split the parent node

➢ Traverse back
  ◦ Updating CFs on the path or splitting nodes
Building CF Tree (Phase 1)

CF(N,LS,SS) – under condition D<T or R<T
- If we run out of space, increase threshold $T$
  - By increasing the threshold, CFs absorb more data
- Rebuilding "pushes" CFs over
  - The larger $T$ allows different CFs to group together
- Reducibility theorem
  - Increasing $T$ will result in a CF–tree smaller than the original
  - Rebuilding needs at most $h$ extra pages of memory
Example of BIRCH

New subcluster

LN1

LN2

LN3

Root
If the branching factor of a leaf node can not exceed 3, then LN1 is split.
If the branching factor of a non-leaf node cannot exceed 3, then the root is split and the height of the CF Tree increases by one.
Merge Operation in BIRCH

Assume that the subclusters are numbered according to the order of formation.
If the branching factor of a leaf node cannot exceed 3, then LN2 is split.
LN2’ and LN1 will be merged, and the newly formed node will be split immediately.
Condensing CF Tree (Phase 2)

- Chose a larger T (threshold)
- Consider entries in leaf nodes
- Reinsert CF entries in the new tree
  - If new “path” is “before” original “path”, move it to new “path”
  - If new “path” is the same as original “path”, leave it unchanged
Global Clustering (Phase 3)

- Consider CF entries in leaf nodes only
- Use centroid as the representative of a cluster
- Perform traditional clustering (e.g. agglomerative hierarchy (complete link \( \equiv D_2 \)) or K-mean or CL...)
- Cluster CF instead of data points
Cluster Refining (Phase 4)

- Require scan of dataset one more time
- Use clusters found in phase 3 as seeds
- Redistribute data points to their closest seeds and form new clusters
- Removal of outliers
- Acquisition of membership information
Conclusions

- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering.
- Given a limited amount of main memory, BIRCH can minimize the time required for I/O.
- BIRCH is a scalable clustering algorithm with respect to the number of objects, and good quality of clustering of the data.
- A clustering algorithm taking consideration of I/O costs, memory limitation
- Utilize local information (each clustering decision is made without scanning all data points)
- Not every data point is equally important for clustering purpose