Unsupervised Learning
Hierarchical Methods
Road Map

1. Basic Concepts
2. BIRCH
3. ROCK
The Principle

- Group data objects into a tree of clusters

- Hierarchical methods can be
  - *Agglomerative*: bottom-up approach
  - *Divisive*: top-down approach

- Hierarchical clustering has **no backtracking**

- If a particular merge or split turns out to be poor choice, the methods **cannot correct** it
Agglomerative & Divisive Clustering

Agglomerative Hierarchical Clustering

- Bottom-up strategy
- Each cluster starts with only one object
- Clusters are merged into larger and larger clusters until
  - All the objects are in a single cluster
  - Certain termination conditions are satisfied

Divisive Hierarchical Clustering

- Top-down strategy
- Start with all objects in one cluster
- Clusters are subdivided into smaller and smaller clusters until
  - Each object forms a cluster on its own
  - Certain termination conditions are satisfied
Agglomerative and divisive algorithms on a dataset of five objects \{a, b, c, d, e\}
Example

- **AGNES**
  - Clusters C1 and C2 may be merged if an object in C1 and an object in C2 form the minimum Euclidean distance between any two objects from different clusters.

- **DIANA**
  - A cluster is split according to some principle, e.g., the maximum Euclidian distance between the closest neighboring objects in the cluster.
Distance Between Clusters

- **First measure:** Minimum distance

\[ d_{\text{min}}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'| \]

- **Use cases**
  - An algorithm that uses the minimum distance to measure the distance between clusters is called sometimes **nearest-neighbor clustering algorithm**
  - If the clustering process terminates when the minimum distance between nearest clusters exceeds an arbitrary threshold, it is called **single-linkage algorithm**
  - An agglomerative algorithm that uses the minimum distance measure is also called **minimal spanning tree algorithm**
Distance Between Clusters

- Second measure: **Maximum distance**

\[ d_{\text{min}}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'| \]

| | is the distance between two objects \(p\) and \(p'\)

- **Use cases**
  - An algorithm that uses the maximum distance to measure the distance between clusters is called sometimes **farthest-neighbor clustering algorithm**
  - If the clustering process terminates when the maximum distance between nearest clusters exceeds an arbitrary threshold, it is called **complete-linkage algorithm**
Distance Between Clusters

- Minimum and maximum distances are extreme implying that they are overly sensitive to outliers or noisy data.

- Third measure: **Mean distance**
  
  \[ d_{\text{mean}}(C_i, C_j) = |m_i - m_j| \]

- Fourth measure: **Average distance**

\[
d_{\text{avg}}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'| \]

- Mean is difficult to compute for categorical data.
Challenges & Solutions

- It is **difficult** to select merge, or split points
- No **backtracking**
- Hierarchical clustering **does not scale** well: examines a good number of objects before any decision of split or merge
- One promising direction to solve these problems is to combine hierarchical clustering with other clustering techniques: **multiple-phase clustering**
1. Basic Concepts
2. BIRCH
3. ROCK
BIRCH: Balanced Iterative Reducing and Clustering Using Hierarchies

Agglomerative Clustering designed for clustering a large amount of numerical data

What Birch algorithm tries to solve?

- Most of the existing algorithms DO NOT consider the case that datasets can be too large to fit in main memory
- They DO NOT concentrate on minimizing the number of scans of the dataset
- I/O costs are very high
- The complexity of BIRCH is $O(n)$ where $n$ is the number of objects to be clustered.
If cluster 1 becomes too large (not compact) by adding object 2, then split the cluster.
BIRCH: The Idea by Example

Data Objects

1
2
3
4
5
6

Clustering Process (build a tree)

Leaf node

Cluster 1

Leaf node with two entries

Cluster 2
entry 1 is the closest to object 3

If cluster 1 becomes too large by adding object 3, then split the cluster
BIRCH: The Idea by Example

Data Objects

1
2
3
4
5
6

Clustering Process (build a tree)

Cluster1
Cluster2
Cluster3

entry 1
entry 2
entry 3

Leaf node

Leaf node with three entries
BIRCH: The Idea by Example

Data Objects

1  2  3  4  5  6

Clustering Process (build a tree)

entry 1  entry 2  entry 3

Leaf node

Cluster 1  Cluster 3  Cluster 2

entry 3 is the closest to object 4

Cluster 2 remains compact when adding object 4

then add object 4 to cluster 2
**BIRCH: The Idea by Example**

### Data Objects

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6

### Clustering Process (build a tree)

- **Leaf node**
  - entry 1
  - entry 2
  - entry 3

- **Cluster 1**
  - object 1

- **Cluster 3**
  - object 3
  - object 5

- **Cluster 2**
  - object 2
  - object 4

**entry 2 is the closest to object 5**

Cluster 3 becomes too large by adding object 5 then split cluster 3?

**BUT** there is a limit to the number of entries a node can have

Thus, split the node
BIRCH: The Idea by Example

Data Objects

1 2 3 4

Cluster1 Cluster3 Cluster4 Cluster2

Clustering Process (build a tree)

entry 1 entry 2

entry 1.1 entry 1.2 entry 2.1 entry 2.2

Leaf node Leaf node

Non-Leaf node
entry 1.2 is the closest to object 6

Cluster 3 remains compact when adding object 6 then add object 6 to cluster 3
BIRCH: Key Components

- **Clustering Feature (CF)**
  - Summary of the statistics for a given cluster: the 0-th, 1st and 2nd moments of the cluster from the statistical point of view
  - Used to compute centroids, and measures the compactness and distance of clusters

- **CF-Tree**
  - Height-balanced tree
  - Two parameters:
    - Number of entries in each node
    - The diameter of all entries in a leaf node
  - Leaf nodes are connected via prev and next pointers
Clustering Feature (CF): $\text{CF} = (N, LS, SS)$

- $N$: Number of data points

- $LS$: linear sum of $N$ points:
  $$\sum_{i=1}^{N} X_i$$

- $SS$: square sum of $N$ points:
  $$\sum_{i=1}^{N} X_i^2$$

$\text{CF}_1 = (3, (2+3+4, 5+2+3), (2^2+3^2+4^2, 5^2+2^2+3^2)) = (3, (9,10), (29,38))$

$\text{CF}_2 = (3, (35,36), (417,440))$

$\text{CF}_3 = \text{CF}_1 + \text{CF}_2 = (3+3, (9+35, 10+36), (29+417, 38+440)) = (6, (44,46), (446,478))$
Properties of Clustering Feature

- CF entry is a **summary** of statistics of the cluster
- A **representation** of the cluster
- A CF entry has **sufficient information** to calculate the centroid, radius, diameter and many other distance measures
- **Additively** theorem allows us to **merge sub-clusters incrementally**
Distance Measures

- Given a cluster with data points

**Centroid:**

\[ x_0 = \frac{\sum_{i=1}^{n} X_i}{n} \]

**Radius:** average distance from any point of the cluster to its centroid

\[ R = \sqrt{\frac{\sum_{i=1}^{n} (x_i - x_0)^2}{n}} \]

**Diameter:** square root of average mean squared distance between all pairs of points in the cluster

\[ D = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (x_i - x_j)^2}{n}} \]
**CF Tree**

- **B** = Branching Factor, maximum children in a non-leaf node
- **T** = Threshold for diameter or radius of the cluster in a leaf
- **L** = number of entries in a leaf
- CF entry in parent = sum of CF entries of a child of that entry
- In-memory, height-balanced tree
CF Tree Insertion

- Start with the root

- Find the CF entry in the root closest to the data point, move to that child and repeat the process until a closest leaf entry is found

- At the leaf
  - If the point can be accommodated in the cluster, update the entry
  - If this addition violates the threshold T, split the entry, if this violates the limit imposed by L, split the leaf. If its parent node too is full, split that and so on

- Update the CF entries from the root to the leaf to accommodate this point
BIRCH Algorithm

1. Phase 1: Load into memory by building a CF tree
2. Phase 2 (optional): Condense tree into desirable range by building a smaller CF tree
3. Phase 3: Global Clustering
4. Phase 4: (optional and offline): Cluster Refining

Data → Initial CF tree → Smaller CF tree → Good Clusters → Better Clusters
Choose an initial value for threshold, start inserting the data points one by one into the tree as per the insertion algorithm.

If, in the middle of the above step, the size of the CF tree exceeds the size of the available memory, increase the value of threshold.

Convert the partially built tree into a new tree.

Repeat the above steps until the entire dataset is scanned and a full tree is built.

Outlier Handling.
BIRCH Algorithm: Phase 2, 3, 4

- **Phase 2**
  - A bridge between phase 1 and phase 3
  - Builds a smaller CF tree by increasing the threshold

- **Phase 3**
  - Apply global clustering algorithm to the sub-clusters given by leaf entries of the CF tree
  - Improves clustering quality

- **Phase 4**
  - Scan the entire dataset to label the data points
  - Outlier handling
Road Map

1. Basic Concepts
2. BIRCH
3. ROCK
ROCK: For Categorical Data

- Experiments show that distance functions do not lead to high quality clusters when clustering categorical data.

- Most clustering techniques assess the similarity between points to create clusters.

- At each step, points that are similar are merged into a single cluster.

- Localized approach prone to errors.

- ROCK: used links instead of distances.
Example: Compute Jaccard Coefficient

**Transaction items:** a, b, c, d, e, f, g

Compute Jaccard coefficient between transactions

\[
sim(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}
\]

Sim({a, b, c}, {b, d, e}) = 1/5 = 0.2

Jaccard coefficient between transactions of Cluster 1 ranges from 0.2 to 0.5

Jaccard coefficient between transactions belonging to different clusters can also reach 0.5

Sim({a, b, c}, {a, b, f}) = 2/4 = 0.5

**Two clusters of transactions**

Cluster 1. <a, b, c, d, e>

{a, b, c}
{a, b, d}
{a, b, e}
{a, c, d}
{a, c, e}
{a, d, e}
{b, c, d}
{b, c, e}
{b, d, e}
{c, d, e}

Cluster 2. <a, b, f, g>

{a, b, f}
{a, b, g}
{a, f, g}
{b, f, g}
Example: Using Links

**Transaction items**: a, b, c, d, e, f, g

The number of links between $T_i$ and $T_j$ is the number of common neighbors.

$T_i$ and $T_j$ are neighbors if $\text{Sim}(T_i, T_j) > \theta$.

Consider $\theta = 0.5$.

**Link**({a,b,f}, {a,b,g}) = 5
(common neighbors)

**Link**({a,b,f}, {a,b,c})=3
(common neighbors)

Link is a better measure than Jaccard coefficient.

**Two clusters of transactions**

**Cluster1**: <a, b, c, d, e>
- {a, b, c}
- {a, b, d}
- {a, b, e}
- {a, c, d}
- {a, c, e}
- {a, d, e}
- {b, c, d}
- {b, c, e}
- {b, d, e}
- {c, d, e}

**Cluster2**: <a, b, f, g>
- {a, b, f}
- {a, b, g}
- {a, f, g}
- {b, f, g}
ROCK

- ROCK: Robust Clustering using linKs

- Major Ideas
  - Use links to measure similarity/proximity
  - Not distance-based
  - Computational complexity
    - $m_a$: average number of neighbors
    - $m_m$: maximum number of neighbors
    - $n$: number of objects
    
      $O(n^2 + nm_m m_a + n^2 \log n)$

- Algorithm
  - Sampling-based clustering
  - Draw random sample
  - Cluster with links
  - Label data in disk