Birch: An efficient data clustering method for very large databases

Tian Zhang, Raghu Ramakrishnan, Miron Livny

CPSC 504
Presenter: Ashique
Discussion: April

Outline
- What is data clustering
- Data clustering applications
- Previous Approaches
- Birch's Goal
- Clustering Feature
- Birch clustering algorithm
- Clustering example

What is Data Clustering?
A cluster is a closely-packed group.
A collection of data objects that are similar to one another and treated collectively as a group.

Data Clustering is the partitioning of a dataset into clusters

Why Clustering?
- Helps understand the natural grouping or structure in a dataset
- Large set of multidimensional data
- Data space is usually not uniformly occupied
- Identify the sparse and crowded places
- Helps visualization

Example

Data Clustering - previous approaches
- Probability based (Machine learning): make wrong assumption that distributions on attributes are independent on each other
- Probability representations of clusters is expensive
- Distance based approach assumes DB scanning is not costly
Requirements for large datasets

- Not more than one scan of the database
- Should be online
- Should be suspendable, stoppable, resumable
- Can work with limited memory

Birch’s goals:

- Minimize running time and data scans, thus formulating the problem for large databases
- Clustering decisions made without scanning the whole data
- Exploit the non uniformity of data – treat dense areas as one, and remove outliers (noise)

Discussion #1

- In what applications could you see data clustering being useful? In which of these applications can you imagine that it would be important that a clustering be found in a certain # of seconds? Minutes? Hours?

- Do you think the authors made the right choice in focusing their design on minimizing I/O? Why or why not? If not, do you think that some either criteria, such as efficiency, stability or immunity to abnormal data, might be a more appropriate criteria for determining if a data mining algorithm (such as BIRCH or APRIORI) is “good?”

Clustering Feature (CF)

- CF is a compact storage for data on points in a cluster
- Has enough information to calculate the intra-cluster distances
- Additivity theorem allows us to merge sub-clusters

Given N d-dimensional data points in a cluster: \( \{X\} \) where \( i = 1, 2, ..., N \),

\[
CF = (N, LS, SS)
\]

\( N \) is the number of data points in the cluster,

\( LS \) is the linear sum of the N data points,

\( SS \) is the square sum of the N data points.

CF Tree

- \( R \) = Max. no. of CF in a non-leaf node
- \( L \) = Max. no. of CF in a leaf node

Non-leaf node

Leaf node

\( T \) = Max. radius of a sub-cluster
**CF TREE**
- T is the threshold for the diameter or radius of the leaf nodes.
- The tree size is a function of T. The bigger T is, the smaller the tree will be.
- The CF tree is built dynamically as data is scanned.

**CF Tree Insertion**
- Identifying the appropriate leaf: recursively descending the CF tree and choosing the closest child node according to a chosen distance metric.
- Modifying the leaf: test whether the leaf can absorb the node without violating the threshold. If there is no room, split the node.
- Modifying the path: update CF information up the path.

**Birch Clustering Algorithm**
- Phase 1: Scan all data and build an initial in-memory CF tree.
- Phase 2: condense into desirable length by building a smaller CF tree.
- Phase 3: Global clustering.
- Phase 4: Cluster refining - this is optional, and requires more passes over the data to refine the results.

**Birch - Phase 1**
- Start with initial threshold and insert points into the tree.
- If run out of memory, increase threshold value, and rebuild a smaller tree by reinserting values from older tree and then other values.
- Good initial threshold is important but hard to figure out.
- Outlier removal - when rebuilding tree remove outliers.

**Birch - Phase 2**
- Optional.
- Phase 3 sometime have minimum size which performs well, so phase 2 prepares the tree for phase 3.
- Removes outliers, and grouping clusters.

**Birch - Phase 3**
- Problems after phase 1:
  - Input order affects results.
  - Splitting triggered by node size.
- Phase 3:
  - Cluster all leaf nodes on the CF values according to an existing algorithm.
  - Algorithm used here: agglomerative hierarchical clustering.
Birch - Phase 4

- Optional
- Additional scan/s of the dataset, attaching each item to the centroids found.
- Recalculating the centroids and redistributing the items.
- Always converges

Clustering example

Pixel classification in images
From top to bottom:
- BIRCH classification
- Visible wavelength band
- Near-infrared band

Conclusions

- Birch performs faster than then existing algorithms on large datasets
- Scans whole data only once
- Handles outliers

So far so good

- The CF tree has to reside in the memory
- Performs poorly when clusters don’t take shape of a circle
- Can handle only numeric data
- Sensitive to the order of data records

Discussion #2

- The BIRCH algorithm requires the user to specify a number of parameters (e.g., the page size, the initial threshold for cluster radius, a definition of outliers, etc).
  - Is it reasonable to expect users to specify and tune these parameters?
  - Is it possible for these decisions to be incorporated into the algorithm itself (i.e., automate parameter specification and tuning)?
  - And, would this be desirable?

Discussion #3 (time permitting)

- Both the BIRCH and APRIORI papers used synthetic data, instead of actual data, to evaluate their algorithms. Many members of the class expressed concern over this choice.
  - Why do you think the authors chose to use synthetic data?
  - Do you think that the results of their analysis would change if actual data was used instead?